***** BASICODE-3 *****

Hallo Computerfreunde!

Endlich ist es da: BASICODE im Rundfunk der DDR. Wir hoffen, daß BASICODE vielen Spaß machen wird und einen umfangreichen Softwaretausch ermöglicht.

Für die hilfreiche Unterstützung möchten wir allen Mitgliedern der niederländischen Stichting BASICODE, insbesondere aber den Herren Klaas Robers und Jacob Haubrich recht herzlich danken.

In diesem Computer-Journal geben wir einige Hinweise zur Funktion der Bascoder für die KC-Rechner und den Z-1013.

***** Bascoder für KC – Serie *****

Die Bascoder für die KC-Typen sind BASIC-Programme mit eingebauten Maschinenroutinen. Sie werden mit:

CLOAD"Name"

eingelesen. Der Name für die verschiedenen Typen ist

BAC853 für KC 85/2,3
BAC854 für KC 85/4
BAC87 für KC 85/1 und KC 87

Der Start des Bascoders erfolgt mit RUN. Die Auswahl der Funktionen des Bascoders erfolgt durch Eingabe eines * und eines Buchstabens. Für die KC 85/2-4 können aber auch die Funktionstasten benutzt werden.

Folgende Funktionen werden realisiert:

| Eingabe | Funktion |
|---|---|
| * | Es erscheint das Hilfsmenu mit allen Funktionen |
| *L (F1) | So wird ein BASICODE-Programm von Kassette eingelesen, automatisch in das KC-BASIC übersetzt und gestartet |
| *S (F2) | Speichern eines BASICODE-Programms im KC-Format |
| *A (F3) | Einlesen eines Programms oder ASCII-files im BASICODE-Format. Es wird nicht übersetzt und gestartet und ist daher zum Kopieren von BASICODE-files geeignet. |
| *W (F4) | Mit dieser Funktion werden BASICODE-Programme auf Kassette gespeichert. |
| *T (F5) | Ein im Speicher stehendes Programm im BASICODE-Format wird in das KC-BASIC transformiert. |
| *C (F6) | Ein im Speicher stehendes BASIC-Programm im KC-Format wird in ein ASCII-file (BASICODE-Format) konvertiert. |
| *K (F7) | Listen eines BASICODE-Programms |

Hinweis: Der Start eines BASICODE-Programms erfolgt mit RUN, fortsetzen mit CONT nach BRK, listen im KC

Format mit LIST und BASICODE-Restart nach RESET mit WBASIC bzw. REBASIC und CALL*40D

Die Bascoder für die KC-Rechner wurden von Uwe Zierott (geb.1954) und Andreas Zierott (geb.1969) aus Lehnin entwickelt. Uwe ist Facharbeiter für Datenverarbeitung und arbeitet in einer KC-Vertragswerkstatt. Andreas hat den gleichen Beruf und arbeitet im Rechenzentrum des Meteorologischen Diensts / Potsdam. Beide computern seit ca. 4 Jahren.

***** BASCODER für Z-1013 *****

Da viele Z-1013 Besitzer noch die 16 k Grundversion haben, mußte für den Bascoder eine spezielle Lösung gefunden werden. Sie besteht darin, daß zur Arbeit mit BASICODE zwei Teilprogramme benötigt werden: BASCON und der eigentliche BASCODER
Beide Programme können mit HEADERSAVE, MAINTAPE oder ab 2.Vorton eingelesen werden.

<u>BASCON</u>

L 100 EFF  Start mit J 100

BASCON ist ein Konvertierungsprogramm und stellt nach seinem Start folgende Funktionen bereit:

Einlesen (BC) von BASICODE-Programmen oder ASCII-files im BASICODE-Format

Save (KC) bedeutet Abspeichern im Arbeitsformat des Bascoders (KC-Format)

Load (KC) bedeutet einlesen von Basicode-Programmen im KC-Format

Abspeichern (BC) von BASICODE-Programmen im BC-Format

Textkorrektur erlaubt in einigen Fällen die Korrektur von Einlesefehlern der BASICODE-Programme

Beenden = Rückkehr in den Monitor

Hinweis: Die Funktionsauswahl erfolgt durch Eingabe des jeweiligen Anfangsbuchstabens. Bei SAVE (KC) ist VERIFY mit Y zu quittieren. Ein Warmstart von BASCON kann mit J 102 erfolgen.

<u>Z-1013 BASCODER</u>

L 100 2AFF  Start mit J 300

Der Bascoder ist ein speziell entwickelter Z-1013-BASIC-Interpreter. Er benötigt ein eigenes Kassettenformat, das mit BASCON bereitgestellt werden kann. Das Einlesen von BASICODE-Programmen (KC-Format) erfolgt mit:

LOAD#1"Name"

Der Name muß mit dem übereinstimmen, wie er unter BASCON mit SAVE (KC) abgespeichert wurde. Will man bereits existierende BASIC-Programme an BASICODE anpassen, so sind sie mit dem normalen 10k BASIC-Interpreter mit LIST#1"Name" zu saven und können wie beschrieben in den Bascoder eingelesen werden. Das anzupassende Programm kann in Probeläufen

mit RUN auf unzulässige Befehle überprüft werden.

Ein Warmstart des Bascoders erfolgt mit J 302.

Der Bascoder für den Z-1013 wurde von Martin Duchrow (geb.1954) aus Berlin entwickelt. Er ist Dipl.-Ing.-Ökonom und beschäftigt sich seit 1986 mit dem Z 1013.

Bemerkung

Die hier beschriebenen Hinweise zur Benutzung der Bascoder stellen nur eine Minimalvariante dar. Eine ausführliche Dokumentation wird in einem umfangreichen BASICODE-Handbuch enthalten sein, das voraussichtlich Anfang nächsten Jahres vom Verlag Technik herausgegeben wird.

HINWEIS

Für die weitere Gestaltung unserer Sendungen sind wir an Ihren Erfahrungen (Einlesbarkeit der Programme), Hinweisen und Programmangeboten interessiert. Schreiben Sie uns bitte, welche Aufzeichnungstechnik und welchen Computer Sie verwenden.

Unsere Adresse:

```
**********************************
*       R E M          RADIO DDR II    *

*     Nalepastr.          BERLIN, 1160    *
**********************************
```

**Grundlagen zum Programmieren in BASICODE**

sind
- die Programmiersprache BASIC,
- der BASIC-Dialekt des Computers,
- die unter BASICODE erlaubten Anweisungen, Funktionen, Operatoren und Variablennamen,
- die BASICODE-Subroutinen (Unterprogramme) und
- die Regeln zum Aufbau eines Programms.

Ausgehend vom 'BASICODE-3-Protokoll' soll auf die Besonderheiten hingewiesen werden.

BASIC bzw. BASICODE schließen strukturiertes Programmieren nicht aus, erzwingen es aber auch nicht, wie z.B. Pascal. Die Gestaltung eines Programms liegt in der Hand des Programmierers. Vorhandene BASICODE-Programme zeigen, daß es möglich ist, Programme zu strukturieren; sie geben Anregungen und Beispiele.

<u>Der Programmaufbau</u>

Ein Programm besteht aus den Zeilen

```
    0 -   999 - BASICODE-Subroutinen,
 1000 - 32767 - BASICODE-Programm.
```

Der erste Teil (# 0 - 999) wird im Speicher als BASIC- oder Maschinensprache-Programm (durch Laden des Bascoders) abgelegt.
Der zweite Teil (# 1000 - 32767) - das

BASICODE-Programm - wird mit dem für jeden Rechner spezifischen Befehl nachgeladen oder kann frei programmiert werden.

Die Zeilen # 0 - 999 können bzw. dürfen nicht verändert werden!

Der Zeilenbereich # 0 - 999 hat folgenden Aufbau:

```
   10 GOTO 1000
   20  -  Überführen des Computers in
          die Betriebsart 'BASICODE'
   25 GOTO 1010
  100  \
    :    \  Computerspezifische
    :    /  BASICODE-Subroutinen
  650  /
  950  -  Zurücksetzen in die
          normale Betriebsart
  955 END
```

Ist dieser Teil als Maschinenspracheprogramm abgelegt, so bleibt er 'verdeckt'; er ist nicht listbar.

Das eigentliche Programm beginnt mit der Zeilenummer # 1000. Der Inhalt dieser Zeile ist verbindlich vorgeschrieben.

```
  1000 A=wert:GOTO 20:REM programmname
  1010 \
    :     \ Hauptprogramm
    :     / GOTO 950 (Ende)
 19999 /
 20000 \
    :     \  Computerspezifische
    :      \  Unterprogramme, die in
    :      /  BASICODE nicht erlaubte
    :     /   Anweisungen enthalten
 24999 /
 25000 \
    :     >DATA-Zeilen
 29999 /
 30000 \
    :     \  REM-Zeilen
    :     /  Hinweise, Bemerkungen
 31999 /
 32000 \
    :     \ REM-Zeilen
    :     / Autor, Computer, Datum
 32767 /
```

<u>Die Programmzeile</u>

Die Länge einer Programmzeile ist auf 60 Zeichen beschränkt - einschließlich

- Zeilennummer,
- Zwischenräume und
- Abschlußzeichen
  (Carriage return, CHR$(13)).

In einer Programmzeile können mehrere Anweisungen - getrennt durch ':' (Doppelpunkt) - aufgenommen werden. Es ist eine gute Gewohnheit,  die Zeilen in Schritten von '10' zu numerieren (z.B. ...

2000, 2010, 2020, ...); es bleibt Raum für Änderungen und Ergänzungen.

Die Programmzeile # 1000

Der Inhalt der Zeile # 1000 ist festgelegt:

A=wert : GOTO 20 : REM programmname

Mit dem Sprung zur Zeile # 20 wird der Computer in die Betriebsart 'BASICODE' versetzt. Dazu gehören:

- Löschen des Bildschirms (CLS),
- Löschen der Variablen (CLR),
- Wahl des Text-Modus,
- Setzen der Bildschirmfarben (Zeichen - weiß, Hintergrund - dunkel),
- Deklarieren und Initialisieren der Variablen HO, VE, HG, VG und SV.

Wichtig ist der Wert der Variablen A, der vom Programmierer festzulegen ist. Für Computer, die mit einer Z-80-ähnlichen CPU arbeiten, muß der String-Speicherraum festgelegt werden (z.B. CLEAR 500). Reicht der Speicherplatz nicht aus, wird ein 'Out of string space' (OS) – Fehler angezeigt.

Bei Rechnern mit dynamischer Speicherverwaltung (z.B. Commodore) spielt das keine Rolle, ist aber zu beachten, um die Portabilität der Programme zu gewährleisten. Für eine Abschätzung der Wertzuweisung bietet sich an, die Subroutine # 270 zu nutzen, die in der Variablen FR den freien Speicherplatz (in Byte) zurückgibt:

- Aufstellen des Programms mit A=500,
- Abspeichern des Programms,
- Aus/Einschalten bzw. Reset,
- Laden des Programms,
- GOSUB 270 : PRINT FR (RETURN),
- Lauf des Programms,
- GOSUB 270 : PRINT FR (RETURN).

Die Differenz beider Werte ergibt annähernd den Wert, der aufgerundet der Variablen A zuzuweisen ist.

Beenden des Programms

Das Programm ist ordnungsgemäß zu beenden; die Verwendung der Anweisungen END oder STOP ist verboten. Mit dem Sprung 'GOTO 950' wird das Programm beendet, der Bildschirm wird gelöscht, BASICODE-Programm und -Subroutinen bleiben erhalten.

## Ein BASICODE-Programm

Das folgende Programm ('programm') läuft schnell ab, hat keinen Effekt, ist aber ein BASICODE-Programm:

```
1000 A=100:GOTO 20:REM programm
1010 REM hauptprogramm
1020 GOTO 950
```

## Variable

Wie in allen BASIC-Dialekten sind numerische und String-Variable zugelassen.

Für numerische Variablen gilt:
- Typ 'real',
- einfache Genauigkeit (6 gültige Ziffern),
- Integer-Variable sind nicht erlaubt.

Eine Stringvariable wird durch den Zusatz '$' nach dem Namen gekennzeichnet; sie kann 255 Zeichen lang sein.

Logische Werte werden je nach Computer unterschiedlich repräsentiert (z.B. 'wahr' => '+1' oder '-1'). Eine logische Variable darf deshalb nicht Gegenstand arithmetischer Operationen sein.

Variablennamen sind maximal zwei Zeichen lang:
- Buchstabe, Buchstabe,
- Buchstabe, Ziffer.

Ausgeschlossen ist der Gebrauch von Variablennamen,
- die mit dem Buchstaben 'O' beginnen; diese sind den BASICODE-Standard-Routinen vorbehalten,
- die BASIC-Schlüsselwörter oder Systemvariable darstellen:
  AS, AT, DI, DI$, DO, DO$, DS, DS$, EI, EI$, EL, ER, FN, GO, GO$, GR, HC, IF, LN, MA, MP, PI, SQ, SQ$, ST, ST$, TI, TI$, TO, TO$,
- die Teil von BASICODE-Subroutinen sind.

Folgende Variablen werden über die BASICODE- Unterprogramme aufgerufen oder ihnen übergeben und haben eine festgelegte Bedeutung:

HO,VE - Cursor-Positionierung Textbetrieb (# 110, 120)
- Auslesen des Textschirms (# 220)
- Cursor-Positionierung Grafik-Betrieb (# 620 ... 650)

IN, IN$ - Tastaturabfrage (# 200, 210, 450)

IN - Auslesen Textschirm (# 220)
- Status-Variable externe Speicher (# 540, 550, 560)

IN$ - Lesen von externen Speichern
(# 550)

SR, SR$ - Bilden einer Stringvariablen
(# 300, 310)

SR$ - String-Ausgabe auf
Bildschirm (# 150, 650)
- Drucker (# 350)
- Externe Speicher (# 560)
- Klein- => Groß-Buchstaben
(# 330)

CN - Formatierung numerischer
Ausgaben (# 310)
- Zeichenfarbe bei grafischem
Betrieb (# 620, 630, 650)

CT - Formatierung numerischer
Ausgaben (# 310)

NF, NF$ - Externes Speichern von
Daten-Files (# 500 ... 580)

FR - Freier Speicherplatz (# 270)
- Wirksamkeit der Stop-Taste
(# 280)

SD - Warte-Routine (# 450)
- Musik (# 400)

SP, SV - Musik (# 400)

RV - Zufallsvariable (# 260)

HG, VG - Grafischer Cursor,
Bildpunkte
(# 620, 630, 650)

Abweichend vom Umgang mit Variablen, der auf manchem Computer möglich ist, sind in BASICODE Variable vor ihrem Aufruf im Programm zu deklarieren und zu initialisieren, d.h. es ist allen im Programm verwendeten Variablen ein Wert zuzuweisen. Die Variablen HO, VE, HG, VG und SV werden durch den Sprung nach Zeile # 20 initialisiert. Man kann nicht davon ausgehen, daß den anderen bei ihrem Aufruf implizit der Wert '0' oder 'leer' zugewiesen wird.

Feldvariable

Vor Gebrauch in einem Programm sind Felder (arrays) zu dimensionieren. Zugelassen sind ein- oder zwei-dimensionale Felder (Listen oder Tabellen). Mit der Dimensionierung werden gleichzeitig die Elemente auf '0' bzw. 'leer' gesetzt. Es kann nicht davon ausgegangen werden, daß der Aufruf einer Feldvariablen im Programm automatisch ein Feld mit 11 Elementen dimensioniert. Auch Felder mit weniger als 11 Elementen sind zu dimensionieren (z.B.DIM A(4)). Die Zählung der Feldelemente beginnt bei '0'.

## BASIC-Sprachumfang

Im Vergleich zu manchem BASIC-Dialekt ist der Sprachumfang reduziert. Um die mit BASICODE angestrebte Portabilität der Programme zu erreichen, mußte die Anzahl der zugelassenen BASIC-Anweisungen und -Operatoren auf den bei allen beteiligten Computern vorhandenen Umfang beschränkt werden. Erlaubt ist die Verwendung folgender Anweisungen:

| | | | | |
|---|---|---|---|---|
| DATA | DEF FN | DIM | FOR | GOSUB |
| GOTO | IF | INPUT | LET | NEXT |
| ON | PRINT | READ | REM | RESTORE |
| RETURN | STEP | TAB | THEN | TO |

## Funktionen und Operatoren:

| | | | | |
|---|---|---|---|---|
| ABS | ASC | ATN | CHR$ | COS |
| EXP | INT | LEFT$ | LEN | LOG |
| MID$ | RIGHT$ | SGN | SIN | SQR |
| TAN | VAL | | | |

AND OR NOT

^ Exponentation
* Multiplikation
/ Division
+ - Addition
  - Verkettung (concatenation) von Stringvariablen
- - Subtraktion

= <> \
< > > Vergleichsoperatoren
<= >= /

Die Reihenfolge der Zeichen der Vergleichsoperatoren '<=', '>=' und '<>' ist vorgeschrieben.

## Der Zeichensatz

Die in BASICODE verwendbaren Zeichen werden nach ASCII codiert. Zu beachten ist jedoch, daß für bestimmte Computer Abweichungen gelten (z.B. Commodore) und daß die Steuercodes nicht benutzt werden können.
Die ASC- und CHR$- Funktionen sind daher mit Vorsicht zu verwenden. Bei manchen Computern wird eine exakte Beziehung zwischen 'IN' und 'IN$' nicht hergestellt.

Bei Aufruf der BASICODE-Subroutinen # 200, # 210, # 220 und # 450 ist dies zu beachten.

Der ASCII gilt in folgenden Bereichen:

| ASCII | Zeichen |
|---|---|
| 32 | Space |
| 33 ... 47 | Sonderzeichen |
| 48 ... 57 | Ziffern |
| 58 ... 64 | Sonderzeichen |
| 65 ... 90 | Groß-Buchstaben |
| 91 ... 96 | Sonderzeichen |
| 97 ... 122 | Klein-Buchstaben |
| 123 ... 126 | Sonderzeichen |

Bei der Darstellung der Sonderzeichen
können bei einzelnen Computern Unterschiede
- je nach implementiertem Zeichensatz -
auftreten, (vergl. 'basicode-ascii').

Bei Lauf eines BASICODE-Programmes auf einem
Commodore wirkt sich der Unterschied der
Codierung zwischen 'echtem' ASCII und
CBM-ASCII deutlich aus, wenn es sich um die
Codierung von Groß- und Klein-Buchstaben
handelt ('cbm-ascii').

Außerdem können über die Variable 'IN'
folgende Steuercodes abgefragt werden:

| | |
|---|---|
| 13 | Carriage Return / Enter (CR) |
| 28 | Cursor nach links |
| 29 | Cursor nach rechts |
| 30 | Cursor nach unten |
| 31 | Cursor nach oben |
| 127 | Links ein Zeichen löschen (DEL) |

Diese Steuerfunktionen können aber in
BASICODE-Programmen nicht aufgerufen werden.

Mit den Unterprogrammen # 200, 210 kann nur
der Wert von 'IN' abgefragt werden, der zur
Steuerung des Programmlaufs verwendet werden
kann.

Zur Cursor-Positionierung dient allein die
Subroutine # 110 mit den Variablen HO und VE.

Funktionen

Funktionen geben in Abhängigkeit von ein,
zwei oder drei Argumenten einen Wert
zurück, der einer Variablen zugewiesen oder
ausgegeben werden kann oder Teil eines
Ausdrucks wird. Bei der Verwendung von
Funktionen in BASICODE-Programmen existieren
bestimmte Beschränkungen. Unterschieden
werden

- numerische Funktionen,
- Zeichenkettenfunktionen und
- Hybridfunktionen (Typwandlung).

Numerische Funktionen

Die Standard-Funktionen

- ABS, INT, SGN, EXP, SQR,
  COS, SIN, TAN, ATN

können wie üblich verwendet werden.

LOG(argument) gibt den natürlichen
Logarithmus zur Basis e (= 2.718...) zurück.
Einzelne BASIC-Dialekte geben über LOG als
Wert der Funktion den Logarithmus zur Basis
10. Es wird zwischen LOG und LN
unterschieden. Der Bascoder gleicht dies aus.

Für die Winkelfunktionen ist das Argument
im Bogenmaß anzugeben.

Die numerischen Funktionen
- MEM oder
- FRE(parameter)

zur Angabe des freien Speicherplatzes dürfen in BASICODE nicht verwendet werden. Dazu dient allein die Subroutine # 270 mit gleichzeitiger 'garbage collection' und die Abfrage des Wertes der Variablen FR.

Die Bildung von Pseudozufallszahlen geschieht in BASICODE-Programmen allein über die Subroutine # 260 und die Variable RV. Funktionsaufrufe oder Kommandos wie:
- RND(parameter),
- RANDOMIZE, RAND

sind nicht zugelassen.

Neben den eingebauten Standardfunktionen können auch vom Benutzer definierte Funktionen nach ihrer Definition mit DEF FNname verwendet werden (vergl. Teil 4, Vereinbarungen).

Zeichenkettenfunktionen

Sie geben als Wert eine Stringkonstante zurück; dazu gehören:

- LEFT$(A$, M), \
- RIGHT$(A$, M), \ vergleiche
- MID$(A$, M, N) > 'stringfunkt.'
  und /
- MID$(A$, N). /

Der numerische Ausdruck (M, N) kann einen Wert von 1 ... 255 annehmen; der Wert '0' führt zu einer Fehlermeldung.

Zeichenkettenfunktionen durch den Benutzer zu definieren, ist in BASICODE-Programmen nicht möglich.

Hybridfunktionen

Damit sollen hier Funktionen bezeichnet werden, die einen numerischen in einen alphanumerischen Wert und umgekehrt umwandeln. Diese Funktionen können auch in Ausdrücken des entsprechenden Typs verwendet werden.

In BASICODE-Programmen sind erlaubt:

- LEN(A$) gibt die Länge der Zeichenkette (Zeichenzahl) zurück.

- ASC(A$) ergibt eine Zahl gleich dem ASCII-Wert des ersten Zeichens von A$.

- CHR$(A) das Zeichen, entsprechend dem ASCII-Wert von A wird angezeigt.

Bei der Verwendung der Funktionen ASC und CHR$ (der Umkehrfunktion von ASC) ist

Vorsicht geboten, da manche Computer den Zeichensatz abweichend vom ASCII kodieren (z.B. C64).

Die in BASIC-Dialekten üblichen Funktionen VAL(A$) und STR$(A) führen eine Typwandlung aus, d.h. es werden Zeichenketten mit ihrem numerischen Wert und numerische Werte als Zeichenketten dargestellt.

Da die VAL(A$)-Funktion in unterschiedlichen Dialekten unterschiedliche Wirkungen hat, ist sicherzustellen, das A$ rein numerisch ist.

Die Funktion STR$(A) ist in BASICODE verboten; an ihrer Stelle ist die Subroutine # 300 mit den Variablen SR bzw. SR$ aufzurufen.

Logische Ausdrücke

Die Bildung logischer Werte ('wahr' oder 'falsch') geschieht bei verschiedenen Computern unterschiedlich. So wird 'wahr' durch '+1' oder '-1' dargestellt, 'falsch' meistens durch '0'.
Um die Portabilität der Programme zu sichern, gelten für BASICODE-Programme hinsichtlich der Anwendung einige Begrenzungen:

- Logische Werte dürfen nicht Teil arithmetischer Operationen sein, z.B. A=3*(C=B),

- Vergleiche, die 'wahr' oder 'falsch' ergeben, sind in Klammern zu setzen, um die Abarbeitungsfolge zu gewährleisten:
  richtig: A=(B=C)
  falsch: A=B=C

- Die logischen Operatoren AND und OR dürfen in BASICODE nicht zur bitweisen Verknüpfung von Zahlen, z.B. A=5 AND 7 verwendet werden.

Es gilt, daß logische Ausdrücke nur in der Form

IF logischer ausdruck THEN anweisung,
IF logischer ausdruck THEN zeilennr.

verwendet werden dürfen (vergl. Teil 5).

BASICODE baut auf BASIC auf. Bei der zugrunde liegenden Konzeption - Portabilität der Programme - mußte ein Kompromiß über den in BASIC nutzbaren Sprachumfang gefunden werden.

Dies umso mehr als zwischen den BASIC-Dialekten der in das System eingebundenen Computer erhebliche Unterschiede bestehen. Das Resultat war eine Art Minimal-BASIC (vergl. Teil (2)), das zwar die üblichen BASIC-Anweisungen, -Funktionen und -Operatoren einschließt, aber für sie eine u.U. vom Gewohnten abweichende, bestimmte Syntax vorschreibt.

Vereinbarungen

Unmittelbar am Anfang des Programms sind benötigte Variablen-Felder zu dimensionieren und Funktionen zu definieren (DIM, DEF FNname).

DIM Variable (ganze Zahl)
DIM Variable (ganze Zahl, ganze Zahl)

Damit wird eine Feldvariable dimensioniert:
- eindimensional (Liste) oder
- zweidimensional (Tabelle).

Variablenfelder mit mehr als zwei Dimensionen sind nicht erlaubt. Felder sind auch dann zu dimensionieren, wenn sie nur bis zu 10 oder 11 Elemente enthalten. Man kann nicht voraussetzen, daß automatisch beim erstmaligen Aufruf einer Feldvariablen ein Feld dimensioniert wird. Der Name kennzeichnet das Feld; endet er mit '$' so werden Zeichenketten erwartet. Mit der DIM-Anweisung wird für die Feldvariable Speicherplatz reserviert, u.U. ist die Dimensionierung zu überprüfen und der Größe des Arbeitsspeichers anzupassen.

Die Zählung der Elemente beginnt mit '0' (Null).

Mit DIM A$(3) bzw. DIM B(4,4) werden Felder mit 4 bzw. 25 Elementen dimensioniert.

Mit einer DIM-Anweisung können auch mehrere Felder dimensioniert werden, z.B.

```
DIM A(10), B$(5,5), ...
```

Im Programm ist zu vermeiden, daß Felder ein zweites Mal dimensioniert werden. Dies kann bei Rücksprüngen auftreten und führt zu einer Fehlermeldung bzw. zum Abbruch des Programms.

DEF FN(name) = arithm. Ausdruck

Hiermit können arithmetische Funktionen für die Verwendung im Programm definiert werden. Parameter- und Ergebnisbereich umfassen Gleitkomma- (real-) Zahlen.

Für BASICODE-3-Programme gelten folgende Beschränkungen:

- Als Name der Funktion darf nur ein für eine numerische Variable erlaubter Name gebraucht werden.
- Die Funktionsdefinition darf nur eine Programmzeile (max. 60 Zeichen) ausfüllen.
- Erlaubt sind nur numerische oder logische Funktionen. (Vorsicht bei Verwendung logischer Werte in arithmetischen Operationen!)
- Die Funktion muß vor ihrem ersten Aufruf definiert worden sein.

Daten im Programm

Es ist selbstverständlich, daß zu einem Programm auch die zu verarbeitenden Daten gehören.

Für BASICODE-3-Programme gibt es folgende Möglichkeiten:

- als Teil des Programms:
  * Wertzuweisung (LET),
  * Datenliste (DATA , READ, RESTORE),
- interaktive Eingabe (INPUT),
- Zugriff auf externe Speicher,
  (Subroutinen # 500, 540, 580)

LET Variable = Wert
(Gleitkommazahl oder Zeichenkette)

Im allgemeinen erlauben es die BASIC-Dialekte LET wegzulassen; die Verwendung ist optional. Für das Sinclair-Spectrum-BASIC wird die Wertzuweisung mit diesem Schlüsselwort vom Übersetzungsprogramm ergänzt. Allerdings macht LET in einer Programm-Zeile den Unterschied zwischen Wertzuweisung und logischem Vergleich deutlich.

DATA Konstante, Konstante, ...
READ Variable, Variable, ...
RESTORE

Mit den Anweisungen DATA und READ werden numerischen oder String-Variablen Werte aus einer Konstanten-Liste zugewiesen. Für DATA-Zeilen gilt:

- Konstanten sind mit einem Komma zu trennen.
- Zeichenketten (strings) sind in Anführungszeichen zu setzen.
- Die Numerierung der DATA -Zeilen im Programm beginnt bei # 25000.
- In der DATA-Zeile dürfen nach der Datenliste keine weiteren Anweisungen (auch kein REM) folgen!

Die READ-Anweisung enthält die Variablen, denen die Daten zugewiesen werden sollen. Auf die Gleichheit des Typs von Variablen und Daten ist zu achten.

Beispiel:

```
 nnnnn READ A,B,C$,D$
 25000 DATA 100,200,"Hallo","BASICODE"
```

Hinter READ können mehrere Variablen - getrennt durch Kommata - aufgeführt werden.

Nach jedem Lesen (READ) eines Datenelementes stellt der Computer intern einen Zeiger auf das nächste Element weiter. Mit der Anweisung RESTORE wird der Zeiger auf das erste Element der Datenliste zurückgesetzt. Der in den DATA-Zeilen erfaßte Bestand wird von Beginn an erneut gelesen.

Die Form 'RESTORE zeilennummer', mit der der Zeiger auf einen bestimmten Teil der Datenliste gesetzt wird, ist in BASICODE-3 verboten.

INPUT Variable
INPUT "Zeichenkette" ; Variable

Nach der INPUT-Anweisung wartet der Computer auf über die Tastatur einzugebende Daten, die der bezeichneten Variablen zugewiesen werden. Die Eingabe muß dem Typ der Variablen entsprechen (numerisch oder Zeichenkette).

Die nach INPUT angezeigte Zeichenkette kann einen Benutzerhinweis (prompt) enthalten. Nach INPUT darf nur eine Variable folgen; für mehrere Variablen ist die INPUT-Anweisung zu wiederholen. 'Komma' oder 'Doppelpunkt' dürfen nicht eingegeben werden.

Mit der Subroutine # 110 kann die Eingabe auf dem Bildschirm plaziert werden.

BASICODE-Subroutinen # 500, 540, 580

Über die Unterprogamme # 500, 540, 580 ist es möglich, Daten aus sequentiellen Files zu lesen, die auf Kassette oder Diskette extern gespeichert sind (siehe weiter unten).

Ausgabe der Daten

Daten als Ergebnis des Laufs eines Programms können mit folgenden Anweisungen bzw. Unterprogrammen ausgegeben werden:
- Bildschirm (PRINT),
- Drucker
  (Subroutinen # 350, 360),
- Speicherung auf Kassette oder
  Diskette
  (Subroutinen # 500, 560, 580).

PRINT Konstante
PRINT Variable
PRINT Ausdruck

Konstanten, Variablen und Ausdrücke können numerisch (Zahlen, arithmetische Ausdrücke) oder Zeichenketten sein. Zeichenkonstante sind in Anführungszeichen zu setzen.

Steuerbefehle (Kontrollzeichen) dürfen nicht über eine PRINT-Anweisung übertragen werden. Eine 'leere' PRINT-Anweisung bewirkt die Ausgabe einer Leerzeile.

Sollen Anführungszeichen in einer Zeichenkette ausgegeben werden, so geht dies nur über den Ausdruck des ASCII-Zeichencodes (# 34), d.h. PRINT CHR$(34).

BASIC-Dialekte erlauben in Verbindung mit der PRINT-Anweisung die
- Positionierung (LOCATE, PRINT AT),
- Tabellierung (SPC(), TAB()),
  Komma als Trennzeichen (delimiter)
- und die Formatierung (PRINT USING)

der Ausgabe. In BASICODE-3-Programmen muß die Positionierung und Tabellierung der Ausgabe über die Subroutinen # 110, die Formatierung über die Subroutine # 310 durchgeführt werden.

Obwohl die Anweisung PRINT TAB() zugelassen ist, muß von ihrer Anwendung abgeraten werden, da zwischen BASIC-Dialekten Unterschiede bei der Verwendung des Argumentes zu TAB bestehen (0 ... oder 1 ...).

Zugelassen und vorgeschrieben ist als Trennzeichen nach einer PRINT-Anweisung das Semikolon (;), das 'Wagenrücklauf' (CR) und Linefeed (LF) unterdrückt. Außerdem ist die Verwendung als Trennzeichen der Elemente einer Ausgabeliste vorgeschrieben.

Beispiele:

```
PRINT A;B;C$;D$
PRINT"Hallo ";"Otto"
```

Im ersten Beispiel dienen die Semikolons als Trennzeichen, im zweiten als Trennzeichen und zur Unterdrückung des Rücklaufs (CR und LF) am Zeilenende.

BASICODE-Subroutinen # 350, 360

Über diese Unterprogramme können Daten und Konstanten (Zahlen und Zeichenketten) nach Zuweisung zur Variablen SR$ über einen Drucker ausgegeben werden.

BASICODE-Subroutinen # 500, 560, 580

Hiermit ist eine Speicherung von Daten auf externen Medien (Kassette oder Diskette) möglich (siehe weiter unten).

Programmsteuerungsanweisungen

Mit diesen Anweisungen ist es möglich, ein Programm nicht in der Folge der Zeilennummern abzuarbeiten, sondern den Programmlauf in einer bestimmten Zeile fortzusetzen oder zu beenden.

Dazu gehören:

- GOTO, ON ... GOTO,
- IF ... THEN,
- FOR ... NEXT,
- Aufruf von Unterprogrammen (GOSUB, ON ... GOSUB),
- unterbrechen bzw. beenden des Programms (STOP, END).

GOTO Zeilennummer
ON Variable GOTO Zn1,Zn2,...
ON Ausdruck GOTO Zn1,Zn2,...

Damit wird das Programm mit den Anweisungen der angegebenen Zeilennummer fortgesetzt. Das Sprungziel darf nur eine im Programm vorhandene Zeilennummer sein. Verboten sind

- Sprünge zu BASICODE-Subroutinen,
  Ausnahme: GOTO 20 - Zeile # 1000,
  GOTO 950 - Programmende,
  GOTO 1000 - RUN im Progr.
- Sprunganweisungen folgender Form:
  A=2000 : GOTO A

Sorgfalt ist geboten, wenn der Sprung aufgrund des Wertes einer Variablen oder eines Ausdrucks ausgeführt werden soll (ON ... GOTO ..., computed GOTO).

Der Wert der Variablen bzw. des Ausdrucks muß ganzzahlig sein. Sofern der BASIC-Interpreter dies nicht überwacht, ist die INT-Funktion zu verwenden. Ist der ganzzahlige Wert kleiner als 1 oder größer als die Anzahl der Zeilennumern (Sprungziele), so wird die nächstfolgende Anweisung ausgeführt.

So darf 'K' nur folgende Werte annehmen:

```
ON K GOTO 2000,3000,4000
K=1, 2, 3

ON K-10 GOTO 5000,6000,7000
K=11, 12, 13
```

Es ist deshalb sinnvoll, den Wert von 'K' vor einer solchen Anweisung durch das Programm prüfen zu lassen.

```
IF log. Ausdruck THEN Zeilennummer
IF log. Ausdruck THEN Anweisung
```

Die in BASICODE zugelassene einfache IF-Anweisung bewirkt, daß das Programm mit der folgenden Anweisung (Zeile) fortgesetzt wird, falls der logische Ausdruck 'falsch' ist; ist er 'wahr', wird die nach THEN folgende Anweisung ausgeführt.

Die zweiseitig bedingte IF-Anweisung der Form

```
IF log.Ausdruck THEN Anweisung ELSE Anweisung
```

ist in BASICODE nicht zugelassen.

An Stelle des Ausdrucks kann auch eine logische Variable eingesetzt werden, womit folgende Konstruktion möglich wird:

```
A=(B=3)
IF A THEN ...
```

Für die Lesbarkeit eines Programms ist es jedoch besser, wenn der logische Ausdruck hinter IF explizit angegeben wird.

Soll aufgrund des Wertes eines logischen Ausdrucks ein Sprung (GOTO) zu einer anderen Programmzeile ausgeführt werden, muß die Anweisung folgende Form haben:

```
IF log. Ausdruck THEN Zeilennummer
```

Nicht zugelassen sind:

```
IF log. Ausdruck THEN GOTO Zeilennummer
IF log. Ausdruck GOTO Zeilennummer
```

Der Aufruf eines Unterprogramms mit GOSUB wird wie eine Anweisung behandelt:

```
IF log. Ausdruck THEN GOSUB Zeilennummer
IF log. Ausdruck THEN PRINT ...
IF log. Ausdruck THEN Variable=Wert
```

FOR ... TO ... NEXT

Die aus BASIC bekannte Schleifenkonstruktion

```
FOR Laufvariable = Anfang TO Grenze
(STEP Schrittweite)

    Schleifeninhalt

NEXT Laufvariable
```

kann in BASICODE uneingeschränkt verwendet werden.

Zu beachten ist, daß hinter NEXT die zugehörige Laufvariable angegeben wird; dies gilt vor allem für verschachtelte Schleifen. Die Angabe von zwei oder mehr Laufvariablen ist unzulässig:

```
FOR A=1 TO 10
FOR B=1 TO 10

   Schleifeninhalt

NEXT B
NEXT A  (nicht NEXT B,A !)
```

Unterprogramme (Subroutinen)

Unterprogramme haben in BASICODE-Programmen eine besondere Bedeutung:

- sie stellen die BASICODE-Routinen dar, die die Portabilität der Programme möglich machen
- in BASICODE-Programmen sind sie die Programm-Module, die für ein strukturiertes Programmieren notwendig sind.

Der Aufruf eines Unterprogramms erfolgt mit

```
GOSUB Zeilennummer
```

Zum berechneten GOSUB (ON ... GOSUB ...) s.o. die Ausführungen zu ON ... GOTO ... !

Jedes Unterprogramm ist mit RETURN abzuschließen.

Beenden des Programms

Zum Unterbrechen bzw. zum Beenden des Programms dienen in den BASIC-Dialekten die Anweisungen STOP und END; in BASICODE-Programmen ist deren Verwendung verboten. Das Programm ist mit dem unbedingten Sprung

```
'GOTO 950'     zu beenden.
```

REM-Anweisungen

Die Funktion der REM-Anweisung ist bekannt; mit ihr können Anmerkungen, Hinweise, Kennzeichnung von Abschnitten in einem Programm untergebracht werden. Sie

erscheinen jedoch nur im Listing des Programms. Beim Programmlauf werden sie vom Interpreter überlesen.

Das BASICODE-System geht in seiner Konzeption von einer Nutzung der Programme durch andere aus, so daß sich der Zweck von Erläuterungen allein schon daraus ergibt.

Außerdem wird der Autor des Programms nach einiger Zeit für Erläuterungen selbst dankbar sein!

Gegen die Verwendung von REM-Anweisungen wird
- der Bedarf an Speicherplatz und
- eine Verlängerung der Laufzeit

angeführt. Beide Gründe gelten aber nur in besonderen Fällen.

Der zentrale Teil im BASICODE-System - um die Portabilität der Programme zu gewährleisten - sind die Subroutinen (Unterprogramme). Mit ihnen wird erreicht, daß trotz eines unterschiedlichen Sprachumfangs und unterschiedlicher Anweisungen, z.B. zur Steuerung der Ausgabe über den Bildschirm, zum Ansprechen eines Druckers oder zur Verwaltung peripherer Speicher (Files), die Aufgabe auf allen beteiligten Computern in gleicher Art ausgeführt wird.

Subroutinen - Übersicht

| GOSUB | Wirkung | Variable |
|---|---|---|
| # 100 | Textbetrieb, Schirm löschen | -- |
| # 110 | Positionieren des Cursors | HO, VE |
| # 120 | Registrieren der Cursorposition | HO, VE |
| # 150 | Bildschirm - reverse Darstellung | SR$ |
| # 200 | Tastaturabfrage | IN$, IN |
| # 210 | Warten auf Tasten-druck | IN$, IN |
| # 220 | Zeichen auf Bild-schirmposition | HO, VE, IN |
| # 250 | Akustisches Signal | -- |
| # 260 | Pseudozufallszahl | RV |
| # 270 | Freier Speicher-platz | FR |
| # 280 | Abschalten der Stoptaste | FR |

| | | |
|---|---|---|
| # 300 | Wandeln in String-variable | SR, SR$ |
| # 310 | Formatieren numer. Variablen | SR, SR$, CT, CN |
| # 330 | Wandeln von Klein- in Großbuchstabe | SR$ |
| # 350 | Druckerausgabe | SR$ |
| # 360 | Drucker - Wagen-rücklauf, Zeilen-vorschub | -- |
| # 400 | Tonerzeugung | SP, SD, SV |
| # 450 | Warteroutine | SD, IN, IN$ |
| # 500 | Peripherer Speicher, Öffnen des Kanals | NF$, NF |
| # 540 | desgl., Lesen | IN$, IN |
| # 550 | desgl., Schreiben | SR$, IN |
| # 580 | desgl., Abschluß | NF |
| # 600 | Grafischer Betrieb, Schirm löschen | -- |
| # 620 | Punkt (Pixel) setzen bzw. löschen | HO, VE, CN |
| # 640 | Linie zeichnen bzw. löschen | HO, VE, CN |
| # 650 | Text im Grafikbe-trieb ausgeben | SR$, HO, VE |

Textbetrieb

Beim Start eines BASICODE-Programms wird der Computer in den Textbetrieb gesetzt (Initialisierung mit dem unbedingten Sprung 'GOTO 20'). In dieser Betriebsart sind auf dem Bildschirm im Regelfall 24 Zeilen mit 40 Zeichen/Zeile darstellbar. Da manche Computer bis zu 80 Zeichen/Zeile abbilden können, ist diesem Umstand evtl. durch das Programm Rechnung zu tragen.
Die Belegung der Zeile 25 und die der letzten Position einer Bildschirmzeile ist zu vermeiden. U.U. kann dies zu einem Scrollen des Bildschirms und zu einem Versetzen des Cursors in die nächste Bildschirmzeile führen.

Die Position des Cursors wird durch die Variablen HO (0 ... 39) und VE (0 ... 24) und die Subroutine # 110 bestimmt.

Mit der Initialisierung, d.h. vor Beginn des eigentlichen Programms, werden den Variablen

HO und VE die je nach Computermodell möglichen max. Werte zugewiesen (z.B. HO=39 und VE=24). Die Zählung auf dem Bildschirm beginnt links oben (HO=0, VE=0).

In BASICODE-3 ist nur eine monochrome Zeichendarstellung - je nach TV-Gerät bzw. Monitor - möglich (Zeichen - weiß, grün, bernsteinfarben; Hintergrund - dunkel).

Subroutine # 100

Die Subroutine # 100 löscht den Bildschirm und setzt den Computer in den Textbetrieb, sofern vorher über die Subroutine # 600 der Grafik-Betrieb eingeschaltet war (siehe w.u.). Mit dem Start eines Programms (Initialisierung über den Sprung GOTO 20) wird das Unterprogramm # 100 automatisch aufgerufen.

Werte, die Variablen vor Aufruf des Unterprogramms zugewiesen wurden, werden nicht verändert. Dies gilt auch für HO und VE.

Subroutine # 110

Die Subroutine # 110 positioniert den Cursor; sie entspricht damit den Anweisungen 'LOCATE (X,Y)' oder 'PRINT AT X,Y'. Sinnvoll ist dieses Unterprogramm in Verbindung mit PRINT- oder INPUT-Anweisungen oder mit der Aufforderung zur Tastatur-Betätigung über die Subroutine # 210.

Werden den Variablen HO und VE unzulässige Werte zugewiesen (z.B. HO>39 oder VE>24), so wird der Cursor willkürlich gesetzt, z.B. in die Mitte des Bildschirms !

Subroutine # 120

Über diese Subroutine kann die momentane Cursorposition abgefragt werden. Deren Stellung ergibt sich aus den Variablen HO und VE. In Verbindung mit der Subroutine # 110 kann die Bildschirmausgabe gesteuert werden

Subroutine # 150

Das Unterprogramm # 150 erlaubt die reverse ('auffallende') Darstellung eines Text-Strings.

Der String (A$='Titel') wird mit PRINT A$ in üblicher Form dargestellt. Nach Zuweisung an SR$ werden am Beginn und Ende je drei Leerzeichen zugefügt, die Darstellung erfolgt revers. Die Stringlänge vergrößert sich um sechs Zeichen. Der Cursor steht nach dem dritten Leerzeichen nach A$ in der gleichen Zeile; um ihn in die nächste Zeile zu setzen, muß eine PRINT-Anweisung folgen. Ggf. kann der Cursor mit GOSUB 110 neu positioniert werden.

Beispiel:

```
nnnn A$="Titel"
nnnn HO=10:VE=3:GOSUB 110
nnnn SR$=A$:GOSUB 150:PRINT
```

Subroutine # 220

Mit der Subroutine # 220 wird der ASCII-Wert eines in der Position HO, VE dargestellten Zeichens an die Variable IN zurückgegeben. IN nimmt nur Werte von 32 - 95 an, es sei denn, es wurden Steuertasten betätigt. Das bedeutet, daß zwischen Klein- und Großbuchstaben nicht unterschieden wird. 'a' und 'A' ergeben IN=65. Zu berücksichtigen bleibt auch die für einzelne Computer unterschiedliche Codierung der Zeichen (z.B. Commodore).

Eine mögliche Anwendung ist eine Hardcopy des Textbildschirms.

Tastatur

Subroutine # 200, Subroutine # 210

Zur Abfrage der Tastatur stellt BASICODE-3 zwei Routinen zur Verfügung, die sich in ihrer Wirkung auf den Lauf des Programmes unterscheiden. Sie entsprechen den BASIC-Anweisungen GET und INKEY$ o.ä.

Die Subroutine # 200 registriert, ob während des Programmlaufs - d.h. während der Wirksamkeit dieses Unterprogramms - eine Taste gedrückt wurde; das Programm läuft weiter. Im Gegensatz dazu hält die Subroutine # 210 den Programmlauf bis Tastaturbetätigung an.

In beiden Fällen wird den Variablen IN und IN$ ein Wert zugewiesen:

```
IN$ - das der Taste entsprechende
      Zeichen (als String) und
IN  - 'echter' ASCII-Wert dieses Zeichens.
```

Wurde keine Taste betätigt (Subroutinen # 200, # 450), so haben die Variablen die Werte: IN$=leer; IN=0.

IN kann Werte von 32 ... 95 annehmen; Groß- und Kleinbuchstaben werden durch die gleichen ASCII-Werte repräsentiert (ASCII 65 => 'A' oder 'a').

Außerdem gelten:

```
ASCII 13     - Return / Enter,
 29 ... 31   - Cursor-Steuerung,
 127         - Delete.
```

(Vergl. auch 'Programmieren (2) - Zeichensatz')

Wurde eine Nicht-ASCII-Taste gedrückt, wird der Variablen IN eine negative Zahl zugewiesen.

Häufig wird die Subroutine # 210 angewendet, um durch Betätigung einer Taste den Programmlauf zu steuern (Menu-Auswahl, Ende des Programms usw.). Als Vorteil stellt sich dar, daß die ASCII-Werte der Variablen IN keinen Unterschied zwischen Klein- und Großbuchstaben machen. Es genügt die Abfrage der Variablen IN (I); das durch Tastendruck dargestellte Zeichen muß nicht über IN$ abgefragt werden (II).

In den folgenden Beispielen soll IN bzw. IN$ auf 'J/N' (ja/nein) abgefragt werden.

```
I  nn10 GOSUB 210
   nn20 IF IN=74 THEN...
   nn30 IF IN=78 THEN...
   nn40 GOTO nn10

II nn10 GOSUB 210
   nn20 IF (IN$='J') OR (IN$='j') THEN...
   nn30 IF (IN$='N') OR (IN$='n') THEN...
   nn40 GOTO nn10
```

Die Verringerung des Aufwandes ist deutlich.

Eine andere Verwendung der Subroutine # 210 ist die Nachbildung der INPUT-Anweisung in der Form, daß auch Anführungszeichen, Doppelpunkt und Komma direkt eingegeben und einer String- Variablen zugewiesen werden können.

Warteroutine - Subroutine # 450

Diese Subroutine unterbricht den Programmlauf für eine vorgegebene Zeitspanne. Durch Drücken einer Taste kann sie abgebrochen werden. Sie entspricht der PAUSE- bzw. SLEEP-Anweisung mancher BASIC-Dialekte.

Die Wartezeit ist vor dem Aufruf der Variablen SD zuzuweisen:

$$SD = \frac{\text{Wartezeit (Sekunden)}}{0.1}$$

Wird eine Taste gedrückt, so wird die Routine abgebrochen, den Variablen IN und IN$ werden der ASCII-Wert und das Zeichen übergeben, die Variable SD enthält die Restzeit (in Einheiten von 0.1 Sekunden).

STOP-Taste - Subroutine # 280

Diese Routine schaltet die Wirksamkeit einer STOP-, BREAK-, ESCAPE-Taste aus, wenn vor ihrem Aufruf der Variablen FR der Wert '1' zugewiesen wurde. Unterbrechen eines laufenden Programms mit Tastendruck ist nicht mehr möglich.

Wird FR=0 gesetzt, wird nach GOSUB 280 die STOP-Taste wieder aktiviert.

Signalton - Subroutine # 250

Diese Subroutine gibt ein akustisches Signal (entspricht ASCII 7 - BEL). Damit kann - sofern erforderlich - die Aufmerksamkeit auf einen bestimmten Schritt im Programm gelenkt werden.

Allerdings ist das Signal nur von kurzer Dauer. Oft ist es notwendig, das Signal andauern zu lassen, um dann den Programmlauf nach Betätigen einer Taste verzweigen oder enden zu lassen.

Zufallszahlen - Subroutine # 260

Hiermit wird der Zufallszahlengenerator des Computers aufgerufen; in RV werden dann Pseudozufallszahlen im Bereich

0 <= RV < 1

zurückgegeben. In der Regel werden nur ganzzahlige Werte benötigt, die durch Rechnung gewonnen werden können.

Freier Arbeitsspeicher - Subroutine 270

Man kann davon ausgehen, daß BASICODE-Programme eine Länge von max. 18 KByte haben können, in Einzelfällen auch mehr. Infolge der Übertragung der einzelnen Zeichen und der rechnerinternen Umwandlung in 'Token' ist die im Computer gespeicherte Programmlänge kürzer. Im Computer ist ein freier Arbeitsspeicher von mind. 16 KByte erforderlich.

Der nach Laden des Bascoders freie Speicherplatz (in Bytes) kann im Direkt-Modus mit

GOSUB 270:PRINT FR

abgefragt werden. Die Wiederholung nach dem Laden des Programms und die Bildung der Differenz ergibt die computer-spezifische Programmlänge (in Bytes).

Variablen SR und SR$

In BASICODE-3-Programmen und -Subroutinen kommen den Variablen SR und SR$ besondere Aufgaben zu:

- Wandeln einer numerischen in eine Stringvariable,
- Formatieren der Ausgabe numerischer Daten,
- Wandeln von Klein- in Großbuchstaben,
- Ausgabe über den Drucker,
- Schreiben eines sequentiellen Files auf Kassette oder Diskette,
- Textausgabe im grafischen Betrieb.

Die freie Verwendung dieser Namen in einem Programm ist nicht erlaubt.

## Typwandlung - Subroutine # 300

Die Subroutine # 300 entspricht der in BASIC vorhandenen Funktion STR$(x), mit der numerische in Stringvariable gewandelt werden können. Damit werden die Leerräume vor und hinter numerischen Daten unterdrückt:

A$=STR$(A) wird ersetzt durch:

```
SR=A:GOSUB  300:A$=SR$
```

## Formatieren numerischer Daten - Subroutine # 310

Zahlen werden in BASICODE - je nach Größe und Computer - mit 6 bzw. 9 Stellen oder in wissenschaftlicher Notation ('E-Format') angezeigt. Manche Computer (PC's) erlauben die Darstellung in 'doppelter' Genauigkeit (bis zu 18 gültige Ziffern).

Die Subroutine # 310 ähnelt der 'PRINT USING' - Anweisung mancher BASIC-Dialekte, die jedoch vielfältiger eingesetzt werden kann als die Subroutine 310. In BASICODE-3 ist nur das Formatieren der Ausgabe numerischer Werte - über Bildschirm oder Drucker - möglich.

Die formatierte Ausgabe des Wertes der numerischen Variablen A erfolgt über die Variablen SR$, CT und CN. Dabei bedeuten:

SR numerische Variable, deren Wert in SR$ formatiert dargestellt werden soll,

CT Anzahl der Zeichen, die in SR$ enthalten sind (einschl. Dezimalpunkt und Vorzeichen),

CN Anzahl der Nachkommastellen.

Diese Variablen sind vor dem Aufruf der Subroutine # 310 zu belegen.

Die Zeichenkette SR$ kann maximal nur neun Ziffern enthalten, d.h. daß CT begrenzt ist:

```
    1   - Vorzeichen,
+ vk    - Anzahl der Vorkommastellen,
+  1    - Dezimalpunkt,
+ CN - Anzahl der Nachkommastellen,
  --------------------------------
   CT  - Anzahl der Zeichen
```

In Abhängigkeit von der Größe der darzustellenden Zahl gilt:

vk + CN <= 9 (ohne führende Null falls SR < 1)

Mit der Subroutine # 310 ist es nicht möglich, Zahlen im wissenschaftlichen Format darzustellen.

Kann die Zahl nicht im vorausbestimmten Format angezeigt werden, enthält SR$ Sterne ('*').

Ggf. wird die Zahl auf CN Stellen gerundet. Die Werte der Variablen CT, CN und SR werden mit dem Aufruf der Subroutine nicht verändert.

Beim Programmieren sind die Werte für CT und CN sorgfältig zu bestimmen, z.B.:

- Ganze Zahlen (-1E8 ... +1E8):
  CT=11

- SR < 1 (Vorzeichen, führende Null, Dezimalpunkt, neun Nachkommastellen):
  CT=12, CN=9

Der Variablen CT kann ein Wert bis zu 20 zugewiesen werden; dies führt zu einer Positionierung der Ausgabe in der Zeile. Besser ist es, die Subroutine # 110 zu benutzen.

Klein- -> Großbuchstaben - Subroutine # 330

Dieses Unterprogramm ändert alle in der Zeichenkette SR$ vorhandenen Kleinbuchstaben in Großbuchstaben, indem der ASCII-Wert der Zeichen im Bereich 96 ... 128 um 32 vermindert wird. Aus ASCII 97 ('a') wird ASCII 65 ('A'). Da das Alphabet die ASCII-Werte 97 ... 122 umfaßt, werden auch die Sonderzeichen im Bereich 123 ... 126 gewandelt.

Die Zeichen der ursprünglichen Zeichenkette werden nicht geändert.

Ausgabe über den Drucker - Subroutinen # 350, 360

Grundsätzlich kann davon ausgegangen werden, daß neben der Ausgabe über den Bildschirm auch ein Ausdruck auf Papier (Erläuterungen,Tabellen usw.) sinnvoll ist. Das Programm soll also die Wahl unter beiden Möglichkeiten lassen.

Der Drucker wird über die Subroutinen # 350 bzw. # 360 - und nur über diese - angesprochen. Die Anweisung 'GOSUB 350' entspricht der Anweisung 'PRINT SR$;' - bei einer Ausgabe über den Bildschirm.
Vorher ist der Inhalt der auszugebenden Variablen der Variablen SR$ zuzuweisen. Dies geschieht für

Zeichenkettenvariable (z.B. A$):

```
SR$=A$:GOSUB  350 oder
SR$='abcde':GOSUB  350
```

numerische Variable (z.B. A):

```
SR=A:GOSUB  300 (oder GOSUB  310)
(Wandlung in Stringvariable SR$)
GOSUB  350
```

Der Ausdruck erfolgt ohne Wagenrücklauf und Zeilenvorschub; die Druckzeile wird nicht

abgeschlossen. Die bewirkt die Subroutine # 360; mit dieser Anweisung ist jede Anweisungsfolge zum Ausdruck einer Zeile zu beenden.

Wird die Anweisung 'GOSUB 360' allein benutzt, erfolgt der Ausdruck einer Leerzeile (dies entspricht 'PRINT' bei einer Ausgabe über den Bildschirm).

BASICODE-3 erlaubt es, Datenfiles auf externen Speichern (Kassette oder Diskette) anzulegen, zu schreiben und zu lesen. Das beschränkt sich jedoch auf sequentielle Files. Relative Files werden von den einzelnen Computern in zu unterschiedlicher Form verwaltet.

Damit ist es möglich, Datenfiles zwischen verschiedenen Computern - im BASICODE-Format - auszutauschen.

Die File-Verwaltung sieht vor:

- Eröffnen eines Files,
- Schreiben bzw. Lesen,
- Schließen des Files,
- Fehlerabfrage.

<u>Eröffnen eines Files - Subroutine # 500</u>

Mit der Eröffnung eines Files ist dessen Name in der Variablen NF$ und das angesprochene Speichermedium (Kassette, Diskette oder Microdrive) in NF festzulegen. Der Filename (NF$) kann maximal sieben Zeichen umfassen. Die Variable NF enthält den Code, der das Speichermedium festlegt und bestimmt ob das File zum Schreiben oder Lesen eröffnet wird.

Die Wirkung des der Variablen NF zuzuweisenden Codes wird im Einzelnen durch das Übersetzungsprogramm bestimmt. Grundsätzlich gilt für die Zuweisung an NF:

| Lesen | Schreiben | Speichermedium |
|---|---|---|
| 0 | 1 | BASICODE-Kassette |
| 2 | 3 | Computerspezifisch, Kassette / Diskette |
| 4,6 | 5,7 | Diskette |

Um die Austauschbarkeit sicherzustellen, empfiehlt es sich, Datenbestände im BASICODE- Format abzulegen.

Bei Verwendung der Codeziffern 2, 4, 6 bzw. 3, 5, 7 ist zu beachten, daß es computertypische Unterschiede gibt.

Zuordnung für C64 und ZX Spectrum:

| n | ZX Spectrum | CBM C64 |
|---|---|---|
| 0,1 | BASICODE-Kassette | |
| 2,3 \ | | CBM-Kassette |
| 4,5 > | Microdrive/Disk | Diskette |
| 6,7 / | | Diskette |

Mit der folgenden Programmzeile wird ein Datenfile (zum Schreiben oder Lesen) eröffnet:

```
nnnn NF=n:NF$='name':GOSUB  500
```

Mit Abfrage der Variablen IN kann der Status, d.h. das Auftreten eines Fehlers (vergl. weiter unten) festgestellt werden.

Schließen eines Files - Subroutine # 580

Mit diesem Unterprogramm wird das mit NF=n geöffnete File geschlossen. Es genuegt

```
nnnn GOSUB  580
```

ohne daß NF=n vorher explizit angegeben wird.

Sollen zur Anlage eines Files mehrere Speicher angesprochen werden (z.B. BASICODE-Kassette und Diskette), so ist das erste File zu schließen bevor das zweite eröffnet wird. (Zur Fehlerabfrage siehe w.u.)

Schreiben und Lesen eines Files - Subroutinen # 560 und # 540

Zum Schreiben eines Datenfiles dient die Subroutine # 560. Der Inhalt von SR$ wird in das File - gekennzeichnet durch NF – geschrieben. Numerische Werte sind über 'GOSUB  300' oder 'GOSUB  310' in die Stringvariable SR$ zu wandeln. Strings sind an SR$ zu übergeben.

Die Schreibroutine hat folgende Form:

```
nnnn SR=A:GOSUB  300 (oder 310)
nnnn GOSUB  560
```

oder

```
nnnn SR$=A$:GOSUB  560
```

Das Lesen eines Files (gekennzeichnet durch NF) erfolgt über die Subroutine # 540. Der gelesene Wert wird der Variablen IN$ zugewiesen, auch hier ist bei numerischen Werten und anschließenden arithmetischen Operationen eine Typwandlung über VAL(IN$) durchzuführen.

So gilt für das Lesen eines Files:

```
nnnn GOSUB  540:A$=IN$
```

oder

```
nnnn GOSUB  540:A=VAL(IN$)
```

Die Variablen A$ bzw. A können dann im Programm weiter verwendet werden. Auch hier ist die Fehlerabfrage sinnvoll.

Status- bzw. Fehlerabfrage

Bei jedem Zugriff auf externe Speicher wird der Variablen IN ein Wert zugewiesen, der zeigt, ob er fehlerfrei ablief. Vom Inhalt der Variablen IN können dann weitere Handlungen abhängig gemacht werden.

| IN | Bedeutung |
|---|---|
| 0 | Operation fehlerfrei (o.k) |
| -1 | Operation nicht ausgeführt (Fehler !) |
| 1 | EOF (End of File), nach dem Lesen des letzten Datenelements (IN$='leer') |

Anmerkungen:

1. In Verbindung mit dem Ansprechen externer Speicher werden u.U. die untersten Zeilen des Bildschirms für Benutzerhinweise (prompts) benötigt; diese sind deshalb freizuhalten.

2. Im grafischen Betrieb sollen die Routinen zur Fileverwaltung nicht angesprochen werden. Der Computer ist über 'GOSUB 100' in den Textbetrieb zu setzen.

**BASICODE-3C**

BASICODE - in den Niederlanden geboren und seit 1981 zur Programmübertragung benutzt - wurde 1986 zur Version -3 weiterentwickelt. Seit 1989 wird dieses System in Deutschland zur Übertragung von Computerprogrammen in der Rundfunksendung "REM" des Deutschlandsenders Kultur verwendet. In diesem Jahr (1991) wurde die Version BASICODE-3C vorgestellt; möglich wurde dies wiederum durch das Engagement vieler Hobbyisten und der Stichting BASICODE, Eindhoven.

Diese Erweiterung ist durch folgende Merkmale gekennzeichnet:

1) Darstellung von Text oder Grafik auf dem Bildschirm in wahlweise acht Farben - nur in Verbindung mit einem Farbmonitor/TV !

2) Herstellung eines Screendumps (Textbetrieb)

Neue Übersetzungsprogramme bzw. Erweiterungen liegen z.Zt. für folgende Computer vor:

Commodore C-64,
MSX-1 und MSX-2
Schneider CPC
Philips P 2000

Ein für die praktische Anwendung wichtiges Merkmal ist die absolute Kompatibilität von BASICODE-3 und BASICODE-3C:

| Aufgestellt: | > | Ablauf: |
|---|---|---|
| BC-3 | s/w > | BC-3C s/w |
| BC-3C | Farbe > | BC-3 s/w |

(s/w) = schwarz/weiße
Bildschirmdarstellung (monochrom)

BASICODE-3C-Programme werden, falls nur ein monochromer Monitor (TV) vorhanden ist, mit schwarz/weißem Text bzw. Grafik dargestellt.

Programmieren mit Farbe

Wie schon in der Version -3 werden die hier notwendigen Anweisungen durch Subroutinen ersetzt.
Es sind folgende Farben zugelassen:

| Code | Farbe |
|---|---|
| 0 | Schwarz |
| 1 | Blau |
| 2 | Rot |
| 3 | Violett (Magenta) |
| 4 | Grün |
| 5 | Hellblau (Cyan) |
| 6 | Gelb |
| 7 | Weiß |

Die Code-Ziffern werden den Variablen CC(0) und CC(1) zugewiesen; als Default-Einstellung gilt:

CC(0)=7 > Zeichenfarbe - Weiß
CC(1)=0 > Hintergrund - Schwarz

Bei Start des Programms werden diese Werte mit der Subroutine 100 übernommen und die Farben entsprechend gesetzt.
Sie gelten im Ablauf des Programms bis zu einer Änderung und der nächstfolgenden GOSUB 100 Anweisung. Die Programmzeile

```
CC(0)=2:CC(1)=6:GOSUB 100
```

liefert nach Löschen des Schirmes rote Zeichen auf gelbem Hintergrund.

Um einen hinreichenden Kontrast der Darstellung zu erreichen, empfiehlt sich ein Unterschied von '4' zwischen den Variablen in CC(0) und CC(1).

Der Variablenname 'CC' gilt im Sinne des BASICODE-Protokolls als verboten!

Die Subroutine 150

Die Subroutine 150 bewirkt eine inverse Bildschirmdarstellung, d.h.

dem obigen Beispiel folgend werden schwarze Zeichen auf einem hellen Hintergrund dargestellt. Es bestehen aber noch weitere Möglichkeiten. Ein Beispiel:

```
CC(0)=4:CC(1)=1:SR$='TEST':GOSUB 150
```

Damit wird das Wort TEST in Blau auf einem grünen Untergrund ausgegeben. Für den weiteren Ablauf des Programms bleibt es bei den vor der letzten GOSUB 100-Anweisung mit CC(0) und CC(1) eingestellten Farben. Änderungen haben bis zum nächsten GOSUB 100 keinen Einfluß auf die normale Zeichen-Darstellung.

<u>Farbe im grafischen Betrieb</u>

BASICODE-3 kennt drei Subroutinen zur grafischen Darstellung:

- GOSUB 600
  Löschen des Schirms
  Einleiten des grafischen Betriebes

- CN=0:GOSUB 620
  CN=1:GOSUB 620
  Zeichnen/Löschen eines Punktes

- CN=0:GOSUB 630
  CN=1:GOSUB 630
  Zeichnen/Löschen einer Linie

- CN=0:GOSUB 650
  CN=1:GOSUB 650
  Schreiben/Löschen von Text

Für das Programmieren in BASICODE-3C gilt:

- Subroutine 600 löscht den Schirm und zeigt die Farbe, die in CC(1) codiert ist.

- Subroutinen 620, 630, 650:

  CN=0 - Grafik/Text wird in der Farbe dargestellt, die in CC(0) festgelegt ist.

  CN=1 - Grafik/Text wird gelöscht, d.h. in der Farbe dargestellt, die vor dem letzten GOSUB 600 in CC(1) codiert war.

<u>Screendump - Hardcopy</u>

Die schon aus BASICODE-3 bekannte Subroutine 220 lieferte beim Auslesen der Variablen IN mit PRINT CHR$(IN) nur Großbuchstaben ab.

Dies wurde jetzt verbessert. Durch Hinzufügen der Variablen CN (Wert computerabhängig, in den Subroutinen festgelegt) wird über einen Drucker das Zeichen ausgegeben, das auf dem Bildschirm gezeigt wird:

```
xxxx GOSUB 220:SR$=CHR$(IN+CN):GOSUB 350
```

Beim Lauf von BASICODE-3C-Programmen mit einem BASICODE-3(!)-Übersetzungsprogramm können allerdings Kompatibilitätsprobleme auftreten.

Als Beispiel einer Hardcopy-Routine für BC-3C- Programme bietet sich an:

```
  1010 HT=HO:VT=VE: REM schirmgröße
    :
 21000 CN=0 :                  REM wenn bc-3c-prog. mit
 21010 FOR VE=0 TO VT : REM bc-3(!) routinen laufen
 21020     SR$=""
 21030     FOR HO=0 TO HT
 21040         GOSUB 220
 21050         SR$=SR$+CHR$(IN+CN)
 21060     NEXT HO
 21070     GOSUB 350:GOSUB 360
 21080 NEXT VE
 21090 RETURN
```

Funktionstasten

In BASICODE-3(!)- Programmen sind über die Routinen 200, 210 und 450 neben den Zeichentasten (ASCII 32 ... 126) lediglich sechs Steuertasten ansprechbar (ASCII 13, 28 ... 31 und 127). In der Version -3C können auch die – sofern vorhandenen - Funktionstasten betätigt und zur Steuerung des Programmablaufs verwendet werden.

Über die o.a. Routinen wird

```
 IN$ = ""          und für
 F1 : IN = -1
 F2 : IN = -2
 F3 : IN = -3
 usw.              zurückgegeben.
```

Es bleibt zu hoffen, daß dieser knappe Überblick über die Möglichkeiten, die die neue Version - BASICODE-3C - eröffnet, einige Hörer des 'Computermagazins' neugierig macht und ihnen bald die erweiterten Übersetzungs- Programme für ihre Computer zur Verfügung stehen.

Verwendete Unterlagen:


Stichting BASICODE:

Definitie BASICODE-3C (April 1991)
Toelichting BASICODE-3C (April 1991)

BASICODE-3 Bulletin no. 74 und 75
10. und 17.7.91)



Friedrich Dormeier
Berlin 39 (Sept. 1991)

***** Bascoder für ATARI *****

Der Bascoder für diese Rechner besteht aus mehreren Teilroutinen, einer Schreib-Lese-Routine, dem BASICODE-Interpreter und den GOSUB-Routinen bis 1000. Letztere sind im BASICODE-Format codiert und können nur eingelesen werden, wenn die erforderliche Hardwareanpassung der Datasette durchgeführt wurde (siehe unten). Da das Atari-BASIC von *üblichen* BASIC-Dialekten stark abweicht, mußte für BASICODE ein eigener Interpreter geschrieben werden. Folgende Funktionen werden vom Bascoder bereitgestellt:

| | |
|---|---|
| LOAD filename | Laden eines Programms,wobei unter filename eine Zeichenkette verstanden wird, die der Atari-Norm entspricht (Gerätename, Doppelpunkt, Name, Typ...). |
| SAVE filename | Abspeichern eines Programms (bei Programmteilen wie bei LIST). Bedeutung von filename wie bei LOAD. |
| CLR | Löschen aller Variablen, Felder und selbstdefinierten Funktionen. |
| LIST | Ausgabe eines Programms auf Bildschirm. Es besteht die Möglichkeit der Anzeige von Programmteilen (z.B. LIST -250 oder LIST 100-700). |
| CL. | Laden eines BASICODE-Programms von Kassette. |
| CS. | Abspeichern eines Programms im BASICODE-Format (ab Zeile 1000). Beim Einlesen und Abspeichern von BASICODE-Programmen wird der Bildschirm abgeschaltet! |

Schriftliche Informationen zum Umbau der Datasette können Sie erhalten, wenn Sie einen an sich selbst adressierten Briefumschlag schicken an:

Radio DDR, REM, Kennwort: Atari-Bascoder, Berlin, 1160 schicken.

Der Bascoder für die Atari-Typen wurde von Andreas Graf (geb. 1958) aus Berlin entwickelt. Andreas ist wissenschaftlicher Mitarbeiter im ZKI der Akademie der Wissenschaften.

***** Bascoder für 64k C-16 und Plus/4 *****

Der Bascoder wird mit LOAD <RETURN> von Tonband eingelesen und mit RUN gestartet.

Kopieren des Programms ist möglich mit LOAD <RETURN> zum Einlesen und dann SAVE'name' <RETURN> zum Abspeichern.

Für Abspeichern auf Diskette sollte statt 'SAVE' 'DSAVE' benutzt werden.

Nach Betätigen der SPACE-Taste stehen alle Subroutines bereit.

Belegung der Funktionstasten:

F1,sys5056 : MENU
F2,sys4955 : LOAD BASICODE
F4,sys5421 : SUBROUTINES
F5,sys4629 : SAVE BASICODE

Nach F-Taste <RETURN> drücken!

Mit F2 wird ein BASICODE-Programm
von Kassette HINTER ein bereits
anwesendes Programm geladen.

Normalerweise sollte man also zuerst F4 <RETURN>
benutzen (Subroutinen bereitstellen)
und erst dann F2 <RETURN> drücken.

Beim Abspeichern eines BASICODE-Programms im
BASICODE-Format (F5) wird nach einer Anfangs-
und Endezeile gefragt. Wenn man dann nur
zweimal <RETURN> drückt, werden dafür Zeile
1000 und die letzte Programmzeile genommen.
Zweck dieser Option ist die Abspeicherung
eines Programmteils.

Der Bascoder funktioniert nur auf C-16 mit 64k RAM
und auf (jedem) Plus/4.

Ein 16k C-16 wird blockiert; ein 32k C-16
gibt eine Fehlermeldung.

Schirmgrenzen in Textmode: HO=39 VE=24
40*25 Buchstaben

Schirmauflösung im Grafikmode: HG=250 VG=200

Im Grafikmode existieren 25 Zeilen und 31 Spalten.

Ein Zeichen ist aus 8*8 Bildpunkten aufgebaut.

Die Maschine rechnet mit 9 Ziffern Genauigkeit.

Grenzen für SP: SP=45 bis zu SP=127

Accoladen existieren nicht.

ASCII-Verhalten und Zeichen sind wie beim
COMMODORE 64. Also Achtung!

Das Programm ändert die Funktion der
Funktionstasten F1, F2, F4 und F5.

Das Schreiben kann unterbrochen werden durch
Stoppen des Recorders. Das Lesen kann nur
unterbrochen werden mit <RUN/STOP>/RESET.

Über X <RETURN> wird der Maschinensprachemonitor
verlassen und der Computer geht in den Direkt-Mode über.

Mit RUN <RETURN> wird der Bascoder wieder gestartet.

Wenn von Routine 280 erlaubt, wird mit einem
Druck auf <RUN/STOP> das Programm über GOTO950
mit einer BREAK- Meldung in der entsprechenden Zeile
beendet.

In dieser Version kommen Kassettenmeldungen wie
PRESS PLAY ON TAPE,
PRESS PLAY AND RECORD ON TAPE
noch nicht auf die Bildschirmzeile 25.

Wenn ein Diskettenlaufwerk vom Programm benutzt
wird, aber nicht anwesend ist, erscheint nicht
ein ?DEVICE NOT PRESENT ERROR sondern es
blinkt unten auf dem Bildschirm der Text 'FILE OPEN
ERROR'.

Während Lesen und Schreiben von/auf ist
der Bildschirm ausgeschaltet.

Der Bascoder für den C+4 (64k C-16) wurde
von Robert Nico Mast aus Zwaag (Niederlande) entwickelt.

***** Bascoder für KC 85/2,3 *****

Der Bascoder für den KC 85/2,3 ist ein
BASIC-Programm und wird mit CLOAD'BAC853'
eingelesen und mit RUN gestartet. Die Auswahl
der Funktionen des Bascoders erfolgt durch die
Eingabe eines * und eines Buchstabeens (es
können aber auch die Funktionstasten benutzt
werden). Folgende Funktionen werden
realisiert:

| Eingabe | Funktion |
|---|---|
| * | Es erscheint das Hilfsmenu mit allen Funktionen |
| *L (F1) | So wird ein BASICODE-Programm von Kassette eingelesen, automatisch in das KC-BASIC übersetzt und gestartet. |
| *S (F2) | Speichern eines BASICODE-Programms im KC-Format |
| *A (F3) | Einlesen eines Programms oder anderen ASCII-files im BASICODE-Format. Es wird nicht übersetzt und gestartet und ist daher zum Kopieren von BASICODE-files geeignet. |
| *W (F4) | Mit dieser Funktion werden BASICODE-Programme auf Kassette gespeichert. |
| *T (F5) | Ein im Speicher stehendes Programm im BASICODE-Format wird in das KC-BASIC transformiert. |
| *C (F6) | Ein im Speicher stehendes BASIC-Programm im KC-Format wird in ein ASCII-file (BASICODE-Format) konvertiert. |
| *K (F7) | Listen eines BASICODE-Programms |

Hinweis: Der Start eines BASICODE-Programms
erfolgt mit RUN (oder F8),
Fortsetzen mit CONT (F9) nach BRK,
Listen im KC-Format mit LIST (FA) und
BASICODE-Restart nach RESET mit
REBASIC:CALL*40D (FB).

Der Entwickler des Bascoder für den KC 85/2,3
ist Uwe Zierott (geb. 1954) aus Lehnin. Er ist
FA für Datenverarbeitung und arbeitet im
Kundendienst einer KC-Vertragswerkstatt.
Beschäftigung mit Computern seit 1985.

***** Bascoder für AC 1 *****

Der Bascoder ist ein eigenständiger
BASIC-Interpreter und heißt BACOBAS. Er
benötigt ab 3000H-RAM-Speicher und ist je
nach Version maximal 12 kByte lang. Der
Kaltstart erfolgt auf Adresse 3000H und der
Warmstart auf 3003H. Wer sich das Programm
erst einmal kopieren möchte, sollte dazu das
Kopierprogramm 'Copy' benutzen, da sonst das
Titelbild und die Autostartfunktion

verlorengehen. BACOBAS wird wie jedes andere Maschinenprogramm mit L eingelesen. Zuerst erscheint das Titelbild, nach etwas mehr als einer Minute ist das Einlesen beendet. BACOBAS ist selbststartend. Der Bildschirm wird gelöscht, es werden die Überschrift und die Frage nach der Speichergröße ausgegeben (mit CR quittieren). Danach befindet sich BACOBAS im normalen BASIC-Mode, der immer durch einen vollausgemalten ganzen oder halben Cursor angezeigt wird. Für die Bearbeitung von BASICODE-files enthält BACOBAS folgende Steuerbefehle:

BACO BASICODE-Mode einstellen
CONV BASIC-Programm in BASICODE wandeln
BSAVE gewandeltes Programm in BASICODE-Codierung auf Kassette ausgeben
BLOAD BASICODE von Kassette einlesen
TRANS BASICODE-Programm in BASIC-Tokens wandeln
BLIST BASICODE-Puffer auf Bildschirm listen
BEDIT BASICODE-Puffer editieren

Der Bascoder für den AC 1 wurde von Frank Heyder (geb.1955) und Bodo Nickel (geb.1951) entwickelt. Frank ist Dipl.Ing.für Informationstechnik. Er entwickelte den AC 1 und ist begeisterter Funkamateur. Bodo ist Ingenieur für Nachrichtentechnik und beschäftigt sich seit 1986 mit dem AC 1.

***** Bascoder für Commodore 64 *****

Den Bascoder für den C 64 schrieb Melis van Deelen (geb. 1965) aus den Niederlanden. Für die Verwendung im Rundfunk der DDR wurde er ins Deutsche übertragen. Bei dem C 64 Bascoder handelt es sich um ein Maschinenprogramm mit eingebautem BASIC-Teil (Titelbild).

1. Einlesen und Starten des Bascoders: Der Bascoder für den C 64 kann mit dem Befehl LOAD oder mit LOAD"Programmname" von Kassette eingelesen werden. Das dauert ca. zweieinhalb Minuten. Der Start erfolgt mit dem Befehl RUN. Damit ist der C 64 für die Arbeit mit BASICODE bereit. Durch Drücken der RETURN-Taste und der Funktionstaste F1 kommt man ins Menü. Hier stehen folgende Funktionen zur Auswahl:

2. BASICODE-Menü:

| Taste | Funktion |
|---|---|
| F1 | Abspeichern eines BASICODE-Programms auf Kassette |
| F3 | Laden eines BASICODE-Programms von Kassette und übersetzen ins C 64 Basic |
| Space | Rückkehr nach C 64 Basic |

3. Abspeichern des Bascoders: Von jedem über den Rundfunk übertragenen Programm sollte eine Kopie angefertigt werden. So auch für den Bascoder. Das kann einfach mit einem Kopierprogramm, mit dem Befehl SAVE oder SAVE"Programmname" erfolgen. Es ist aber zu beachten, daß der Bascoder für den C 64 nach dem Laden ohne vorheriges RUN abgespeichert wird!

***** Bascoder für Schneider CPC*****

Der Bascoder für diese Rechner wird in der Kassettenbetriebsart mit dem Befehl: RUN"BASCODER" oder einfach RUN "" eingelesen. Das Programm besteht aus mehreren Teilen (vier BASIC- und zwei Maschinen-Teile), wobei das Einlesen nur der BASIC-Teile am Bildschirm verfolgt werden kann. Die Maschinen-Teile werden automatisch geladen. Die Kopie des Bascoder kann in zwei Schritten durchgeführt werden:

1. Aufruf des BASIC-Editors mit B und ENTER. Dann SAVE "BASCODER" und ENTER.
2. Abspeichern des Maschinen-Code mit SAVE"BC3BIN.BIN",B,&9E00,&0800 und ENTER.

Nach dem Start des Bascoder stehen folgende Funktionen bereit:

L Laden BASICODE Das Einlesen von BASICODE-Programmen kann optisch am Bildschirm verfolgt werden. Nach dem Einlesen erfolgt automatisch die Übersetzung des Programms ins eigene CPC-Format. Einlesefehler werden gemeldet und können nach Möglichkeit per Hand mit dem BASIC-Editor beseitigt werden.

W Sichern BASICODE Eingelesene oder selbst geschriebene BASICODE-Programme können mit diesem Kommando wieder abgespeichert werden. Während des Abspeicherns ist auf dem Bildschirm die Meldung:
Writing BASICODE - WAIT! zu sehen.

B BASIC-Editor

Nach der Eingabe des gewünschten Buchstabens ist die ENTER-Taste zu betätigen.

Der Entwickler des Bascoder für die Schneider-CPC ist Sjef Simons (geb. 1953) aus den Niederlanden. Eine deutsche Bearbeitung wurde von Marius van der Meer (geb. 1950) aus Berlin durchgeführt.

***** Bascoder für ZX-Spectrum *****

Der Bascoder des ZX-Spectrum besteht aus zwei Teilen: 1. BASIC-Lader + Subroutinen 2. Maschinenprogramm

Der Bascoder wird mit LOAD"BASICODE 3" geladen und meldet sich einmalig mit einem Titelbild und der Versionsangabe. Nun gelangt man durch Betätigen der 'ENTER'-Taste in das Hilfsmenu. Folgende Funktionen stehen hier zur Auswahl:

*L Einlesen eines BASICODE-Programms und sofern fehlerfrei automatisches Übersetzen ins ZX-BASIC sowie Starten (bei Fehlermeldung Listen mit *K möglich).

*T Übersetzen eines geladenen BASICODE-Programms ins ZX-BASIC (hilfreich bei *L mit Lesefehler, nach *T und RUN Fehlersuche möglich).

*C Konvertieren eines im Speicher vorhandenen ZX-BASIC-Programms nach BASICODE. Hierbei wird automatisch auf einige verbotene BASICODE-Befehle (POKE, BEEP, PLOT …) abgetestet.

*W Abspeichern eines BASICODE-Programms auf Kassette.

*K Listen eines Programms im BASICODE-Format (Listen eines Programms im ZX-Format mit üblichen Befehl LIST).

*S Abspeichern eines BASICODE-Programms im ZX-Format. Nach Aufruf erfolgt die Abfrage: 'K'assette oder 'D'iskette und die Eingabe eines Programmnamens.

*B Herstellung einer Sicherheitskopie (BACKUP) des Bascoders auf Kassette oder Diskette (BETADISC).

Bei der Arbeit mit BETADISC wird das fertige BASICODE-Programm im ZX-Format zusammen mit den Subroutinen auf Diskette abgelegt und kann bei Neuaufruf sofort gestartet werden, wenn sich der Maschinenteil des Bascoders ('CODE BASICODE-3') im Speicher befindet.

Der Bascoder für den ZX-Spectrum wurde von Jan Bredenbeek (geb.1964) aus den Niederlanden entwickelt.

***** Bascoder für Z-1013 *****

Da viele Z-1013 Besitzer noch die 16 k Grundversion haben, mußte für den Bascoder eine spezielle Lösung gefunden werden. Sie besteht darin, daß zur Arbeit mit BASICODE zwei Teilprogramme benötigt werden: BASCON und der eigentliche BASCODER. Beide Programme können mit HEADERSAVE, MAINTAPE oder ab 2.Vorton eingelesen werden.

1. BASCON L 100 EFF
   Start mit J 100

BASCON ist ein Konvertierungsprogramm und stellt nach seinem Start folgende Funktionen bereit:

Einlesen (BC) von BASICODE-Programmen oder ASCII-files im BASICODE-Format.
Save (KC) bedeutet Abspeichern im Arbeitsformat des Bascoders (KC-Format).
Load (KC) bedeutet Einlesen von Basicode-Programmen im KC-Format.
Abspeichern (BC) von BASICODE-Programmen im BC-Format.

Textkorrektur erlaubt in einigen Fällen die Korrektur von Einlesefehlern der BASICODE-Programme.
Beenden = Rückkehr in den Monitor.

Hinweis: Die Funktionsauswahl erfolgt durch Eingabe des jeweiligen Anfangsbuchstabens.

Bei SAVE (KC) ist VERIFY mit Y zu quittieren.

Ein Warmstart von BASCON kann mit J 102 erfolgen.

2. BASCODER L 100 2AFF
Start mit J 300

Der Bascoder ist ein speziell entwickelter Z-1013-BASIC-Interpreter. Er benötigt ein eigenes Kassettenformat, das mit BASCON bereitgestellt werden kann. Das Einlesen von BASICODE-Programmen (KC-Format) erfolgt mit:

LOAD#1"Name".

Der Name muß mit dem übereinstimmen, wie er unter BASCON mit SAVE (KC) abgespeichert wurde.

Will man bereits existierende BASIC-Programme an BASICODE anpassen, so sind sie mit dem normalen 10k BASIC-Interpreter mit LIST#1"Name" zu saven und können wie beschrieben in den Bascoder eingelesen werden. Das anzupassende Programm kann in Probeläufen mit RUN auf unzulässige Befehle überprüft werden. Ein Warmstart des Bascoders erfolgt mit J 302.

Der Bascoder für den Z-1013 wurde von Martin Duchrow (geb.1954) aus Berlin entwickelt. Er ist Dipl.-Ing.-Ökonom und beschäftigt sich seit 1986 mit dem Z-1013.

die hier dargestellten texte sind eine zusammenfassung von bascoder-kurzdokumentationen, die in der ff-dabei von sept.-dez. 1989 abgedruckt waren.
erstausstrahlung: radio ddr
REM-spezial 900501