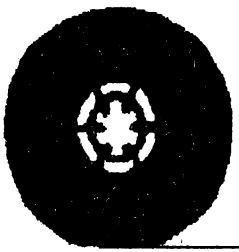
COMPUTER CLUB

Jugend DT64 radio

D RADIO R

RE MAGAZIN

Ein Esperanto fuer Computer



BASICODE

Aus Holland im Rundfunk der DDR

1. Das Grundprinzip

BASICODE wurde von holländischen Computerspezialisten um 1979 entwickelt, wird in Holland regelmäßig im Rundfunk verwendet und wurde fortlaufend verbessert. BASICODE ermöglicht, daß unabhängig vom verwendeten Rechnertyp Programme ausgetauscht und im Rundfunk übertragen werden können. Heute existiert in den Niederlanden eine gewinnfrei arbeitende Stiftung unter dem Vorsitz von Klaas Robers, der auch entscheidender Initiator des BASICODE ist. Von dieser Stiftung hat der Rundfunk der DDR die Rechte zur Anwendung erworben. Mit der Entwicklung der Computertechnik wurde auch BASICODE verbessert. Aus diesem Grunde existiert jetzt die dritte Version BASICODE-3, welche wir verwenden. BASICODE-4 ist in Vorbereitung und besitzt dann zusätzliche Möglichkeiten.
BASICODE kann als eine spezielle Variante von BASIC angesehen werden. In reichlich 50 Befehlen ist BASICODE identisch mit BASIC (siehe Anhang 1). Weitere 30 BASIC-Befehle werden indirekt über spezielle GOSUB-Routinen realisiert (Anhang 2). Dies sind Befehle, die auf verschiedenen Rechnern unterschiedlich realisiert oder z.T. nicht vorhanden sind. So existiert nur bei wenigen Rechnern die Möglichkeit: LOCATE <Spalte, Zeile>, welche den Cursor an eine bestimmte Bildschirm-Position setzt. Mit BASICODE existiert hierzu GOSUB 110, wobei vorher auf die Variablen HO und VE die horizontale bzw. vertikale Koordinate zu übergeben ist. Da nicht alle Rechner die gleiche Anzahl Zeilen und Spalten besitzen, erfolgt durch BASICODE auch eine Oberprüfung auf die Zulässigkeit der Werte.
In BASICODE-3 existieren weiter einheitliche Grafik- und Ton-Routinen. Sie ermöglichen, daß sich Bild und Ton automatisch den Gegebenheiten des Rechners anpassen. Eine Grafik erscheint so bei allen Rechnern in gleicher Form und am gleichen Ort auf dem Bildschirm. Lediglich die Qualität des Bildes hängt von den Möglichkeiten des Rechners ab. Höchste Qualität kann so bei IBM-kompatiblen Rechnern mit Hercules- oder EGA-Karte erreicht werden. Der KC 85/3 schneidet auch recht gut ab. Aber selbst der KC 87, der eigentlich keine echte Grafik besitzt, liefert noch brauchbare Bilder. Beim Ton ist für praktisch alle Rechner eine brauchbare Melodie in richtiger Tonhöhe erreichbar.
Doch damit sind noch keineswegs die Vorteile von BASICODE erschöpft. Ein besonders wichtiger ist dadurch gegeben, daß alle BASICODE-Programme einheitlich auf Kassette aufgezeichnet werden. Eine Kassette, die vom BASICODE von irgendeinem Rechner, z.B. einem Spectrum aufgezeichnet wurde, ist auf jeden anderen Rechner mittels BASICODE wieder lesbar. Dem Austausch von BASICODE-Programmen sind so keine Grenzen gesetzt. Deshalb brauchen wir im Rundfunk ein BASICODE-Programm auch nur in dieser einen Version zu senden. Stellen Sie sich nur einmal vor, daß auf diese Weise alle Rechnerfreunde problemlos miteinander in Verbindung treten können. Die Vorteile dieses Verfahrens sind also überhaupt nicht abzusehen.
Weiter sei erwähnt, daß viele Bascoder so geschrieben sind, daß die Programme zuweilen durchaus schneller als normale BASIC-Programme ablaufen.
Natürlich gibt es auch Nachteile. Sie sind jedoch vergleichsweise gering und betreffen folgende Punkte:
Erstens ist ein zusätzliches Programm, der "Bascoder" (siehe unten), vorher zu laden. Dieses Programm ist rechnerspezifisch und muß damit für jeden Rechner zur Verfügung stehen. Dank der holländischen Spezialisten stehen uns die Bascoder für die meisten Importrechner bereits zur Verfügung. Für die DDR-Rechner konnten inzwischen von DDR-Computerfreunden ebenfalls Bascoder entwickelt werden. Weitere werden gewiß noch entstehen. Die Bascoder

für die wichtigsten Rechner stehen, somit von geringen Unkosten abgesehen, unentgeltlich zur Verfügung.
Zweitens ist das Programmieren in BASICODE etwas ungewohnt, und es müssen mehrere Vorschriften streng eingehalten werden. Denn der Rechner selbst kann auch mit dem Bascoder nicht feststellen, ob unerlaubte Befehle verwendet werden, die zwar auf diesem Rechner, jedoch nicht auf anderen laufen.
Drittens benötigen einige Rechner eine geringe zusätzliche Hardware.

2. Betrieb in BASICODE

Für jeden Rechner existiert ein besonderer Bascoder, der jeweils von einem anderen Experten entwickelt wurde. Hierdurch und durch die Spezifika der Hardware ist es nicht möglich, eine allgemein gültige Anleitung für den Umgang mit BASICODE zu geben. Dennoch gibt es weitgehend gültige Grundsätze. Sie seien im folgenden dargestellt.
Bei den normalen BASIC existieren drei Schichten des Betriebes:

1. Das Betriebssystem.
   Bei den KC-Rechnern CAOS bzw. OS. Hier können z.B. Maschinenprogramme geladen und abgearbeitet werden.
2. Der BASIC-Betrieb (Direkt-Mode).
   Er bestätigt Eingaben wie CLS, LIST usw. nach der Abarbeit mit OK oder bei anderen Rechnern mit READY.
3. Die Abarbeitung eines BASIC-Programms.
   Sie wird eingeleitet durch den Befehl RUN.

Beim BASICODE wird in der Regel in 2. der Bascoder geladen. Dies ist ein spezielles BASIC-Programm, welches meist Maschinenroutinen enthält und es ermöglicht, BASICODE-Programme unter Zuhilfenahme des existierenden BASIC-Interpreters abzuarbeiten. Der Bascoder lagert sich also nach dem Punkt 2. an und es entstehen in etwa vier Schichten:

1. Betriebssystem
2. BASIC-Betrieb
3. Bascoder
4. BASICODE-Mode

Erst nachdem der Bascoder (und meist auch das BASICODE-Programm) geladen ist, kann der BASICODE-Mode erreicht werden. Hier kann die Abarbeitung eines ebenfalls geladenen BASICODE-Programms beginnen. Sein Start erfolgt mit

```
RUN
```

Der BASICODE-Mode wird also mittelbar über den Bascoder bewirkt und ist ein besonderer Zustand Ihres Rechners. Dieser Zustand ist zur Beendigung des BASICODE-Mode folglich auch wieder zu verlassen. Aus diesem Grunde enden BASICODE-Programme auch nicht wie übliche BASIC-Programme, z.B. durch ein END bzw. STOP im Programm oder ohne besondere Hinweise, indem das Programm eben einfach aufhört. Ein BASICODE-Programm muß mit GOTO 950 enden.
Diese Grundsätze wollen wir mit unserem ersten Programm "HYDRA" demonstrieren: Es beginnt mit der Zeile 1000. Sie entspricht der CLEAR-Anweisung im normalen BASIC. A=100 bedeutet, daß ein Stringraum der Größe hundert Byte reserviert wird. GOTO 20 führt zum Bascoder und initialisiert den BASICODE-

Mode. Dabei werden die für den Rechner notwendigen Spezifika ausgelöst, die Variablen gelöscht und weitere notwendige Aktionen bewirkt. Vom Bascoder erfolgt dann ein Rücksprung zur Zeile 1010 und die Abarbeitung des BASICODE-Programms beginnt. Nach seiner Abarbeitung - sie ist bei diesem speziellen Programm für Z=0 in Zeile 1030 erreicht - erfolgt der Sprung mit GOTO 950 in den Bascoder. Er hebt dort den BASICODE-Mode auf und führt den Rechner in den normalen BASIC-Zustand zurück. Es erscheint wieder das OK bzw. das READY. Aus diesen Ausführungen wird deutlich, warum innerhalb eines BASICODE-Prgramms \

RUN, END und STOP

verboten sind (verbotene Befehle faßt Tabelle 3 zusammen). Weiter muß immer je eine Zeile 1000 und 1010 existieren. In der Zeile 1000 sollte der notwendige Stringraum über die Variable A festgelegt werden und danach erfolgt ein Sprung zur Zeile 20. Dieser Aufbau der Zeile 1000 ist notwendig. Vorschriftsmäßig sollte die Zeile mit einem REM und den Namen des Programms fortgesetzt werden.

```
1000 A=100: GOTO 20: REM ### Hydra ###
1010 PRINT "Hydra-Problem": PRINT
1020 INPUT "Eingabe einer Zahl:  0 fuer Ende";Z
1030 IF Z=0 THEN 950: REM Ende
1040 IF(Z<>INT(Z))OR(Z<2)THEN PRINT"Fehler": GOTO 1020
1050 Y=Z/2: IF Y=INT(Y) THEN Z=Y: GOTO 1070
1060 Z=Z+Z+Z+1
1070 PRINT Z;: IF Z>1 THEN 1040
1080 PRINT: GOTO 1020
1090 REM ----Programm-Ende------
30000 REM Das Hydra-Problem stammt von McCarthy.
30010 REM Es wird vermutet, dass alle ganzzahligen
30020 REM Eingaben zu 1 und damit zum Ende fuehren.
30030 REM Ein Beweis steht bis heute aus.
30040 REM Infolge der endlichen Arithmetik kann
30050 REM ein Rechner-Programm nur fuer einige Zahlen
30060 REM diese Hypothese unterstuetzen.
30070 REM Beobachten Sie das Zu- und Abnehmen
30080 REM der Zahlenwerte und versuchen Sie
30090 REM das Programm zu verstehen.
32000 REM ------------------------
32010 REM H. Voelz; 9.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Die Funktion des Programms besteht darin, eine Zahl dann durch zwei zu teilen, wenn sie geradzahlig ist, andernfalls sie mit drei zu multiplizieren und anschließend eins zu addieren (Zeilen 1040 bis 1060). Dies ist fortlaufend zu wiederholen bis 1 erreicht wird. Andere Bemerkungen stehen in den Kommentarzeilen. Ihre Zeilennummern sind für BASICODE genau festgelegt (Anhang 4). Ab 30000 stehen Erklärungen zum Programm und ab 32000 formale Fakten wie Autor, Datum, Versionsnummer, verwendeter Rechner usw.

3. Einfache Befehle im Text-Mode

Nach der ersten Einführung in BASICODE werden nun schrittweise weitere GOSUB-Routinen erklärt. Die nächsten vier Programme betreffen den Text-Mode. Der zweite Mode - der Grafik-Mode - wird im Abschnitt 5 behandelt.

```
1000 A=100: GOTO 20: REM ### DATEN ###
1010 PRINT "Bildschirmdaten"
1020 PRINT"Text: Spalten * Zeilen =";HO;"*";VE
1030 PRINT"Pixel: X * Y ="; HG;"*";VG
1040 PRINT"Taste betaetigen": GOSUB 210: GOTO 950
1050 REM ----Programm-Ende------
30000 REM Anzeige der vorhandenen Bildschirmwerte.
30010 REM Es existieren zwei Modi:
30020 REM Zeilen/Spalten fuer Text
30030 REM Pixel fuer Grafik
32000 REM ----------------------
32010 REM H. Voelz; 10.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Das zuvor stehende Programm "Daten" demonstriert drei Fakten:

1) Anhalten des Programms bis eine Taste betätigt wird.
2) Anzeige der spezifischen Parameter des jeweiligen Rechners.
3) den Umgang mit BASICODE-spezifischen Variablen.

Mit GOTO 20 werden automatisch vier spezielle Variablen mit Daten belegt, die für den angewendeten Rechner typisch sind:

HO enthält die Anzahl der möglichen Spalten -1,
VE enthält die Anzahl der möglichen Zeilen -1,
HG enthält die Anzahl der möglichen Pixel in x-Richtung,
VG enthält die Anzahl der möglichen Pixel in y-Richtung.

Diese Werte werden in den Zeilen 1020 und 1030 angezeigt. Damit ist das Programm eigentlich bereits zu Ende. Jedoch, wenn darauf GOTO 950 unmittelbar folgen würde, ginge der Rechner sofort in den BASIC-Mode über und alles wäre gelöscht. Deshalb muß man die Anzeige solange sichtbar halten, bis eine Taste betätigt wird. Im normalen BASIC geschieht dies oft mit: INPUT A oder INPUT"";A. Bei beiden erscheint aber zumindest der Cursor eventuell auch noch ein Fragezeichen. BASICODE besitzt einen besseren Befehl mit GOSUB 210. Das Programm wird es Ihnen zeigen.
Neben den oben genannten 4 Variablen (HO, VE, HG, HV) und dem A in Zeile 1000 verwendet BASICODE weitere spezifische Variablen auf die beim Programmieren acht zu geben ist. Sie werden noch erläutert und sind im Anhang in Tabelle 5 zusammengestellt. Darüber hinaus sind auch einige Variablen - ähnlich wie im normalen BASIC verboten. Sie faßt Tabelle 6 zusammen.
Das nächste Programm "ZUTEXT" erzeugt Zufallstext und verteilt ihn unregelmäßig über den Bildschirm. Hierzu verwenden viele BASIC-Dialekte den Befehl RND. Da er aber nicht in allen Rechnern existiert, wird statt dessen in BASICODE die Anweisung GOSUB 260 benutzt. Mit ihr wird in die Variable RV ein Zufallswert gemäß 0<=RV<1 abgelegt. Ein "RANDOMIZE" ist ebenfalls bei der Initialisierung über GOTO 20 vorhanden. Dadurch ist die Pseudozufallsfolge in ihrem Start bei RUN unbestimmt.
Die zufällige Anordnung auf dem Bildschirm erfolgt mittels eines Befehls

der PRINT AT ähnelt und durch GOSUB 110 realisiert wird. Die Zeilen- und Spaltenpositionen werden hierbei mittels der Variablen HO und VE übergeben. Dies sind genau jene Variablen, die nach GOTO 20 die Spalten- bzw-. Zeilenzahl enthalten. Deshalb werden diese Werte in 1010 auf die Variablen HT und VT zur Normierung der Zufallszahlen übergeben. So kann erreicht werden, daß bei jedem Rechner der Bildschirm gleichmäßig beschrieben wird. Mit CHR$(32+RV*58) werden zufällig die ASCII-Zeichen von 32 bis 89 erzeugt. In dem Programm fällt das GOSUB 100 in Zeile 1080 auf. Es entspricht im wesentlichen dem Löschen des Bildschirmes und holt erneut die Bildschirmparameter auf die o.g. Variablen zurück. Es setzt auch den Text-Mode.

```
1000 A=100: GOTO 20: REM ### ZUTEXT ###
1010 HT=HO: VT=VE
1020 FOR X=1 TO 200
1030   GOSUB 260: HO=RV*HT: GOSUB 260: VE=RV*VT
1040   GOSUB 110: GOSUB 260: PRINT CHR$(32+RV*95);
1050 NEXT X
1060 HO=0: VE=0: GOSUB 110
1070 INPUT "0=Ende 1=weiter"; Z
1080 IF Z=1 THEN GOSUB 100: GOTO 1020
1090 GOTO 950
1100 REM ----Programm-Ende------
30000 REM Es werden Zufallszeichen an zufaellige
30010 REM Bildschirmpositionen gelegt. Beobachten
30020 REM Sie das entstehende Muster und
30030 REM eventuell sich ausbildende Texte.
32000 REM ----------------------
32010 REM H. Voelz; 9.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Das folgende Programm "TASTE" verwendet zwei weitere BASICODE-Befehle, von dem einer nur selten ein Äquivalent im normalen BASIC besitzt.

1) Mit GOSUB 150 wird eine in SR$ vorher abgelegte Zeichenkette auffällig auf den Bildschirm gebracht. Bei den meisten Bascodern ist dies eine inverse Zeichendarstellung. Zur besseren Hervorhebung werden an SR$ dabei noch je rechts und links drei Leerzeichen angefügt. In dem Programm wird aus dem Wert von HO nach GOSUB 100 so die Lage des auffälligen Textes gemäß HO=INT(HO/2)-7 berechnet, daß er bei jedem Rechner in der Mitte auf dem Bildschirm steht. Der Zahlenwert 7 ergibt sich aus der halben Länge von SR$ plus die drei Leerzeichen.
2) Mit GOSUB 250 wird ein kurzer Ton (BEEP) erzeugt, der als Hinweis für besondere Ereignisse verwendet werden kann, z.B. wenn ein länger rechnendes Programm zum Ende gekommen ist.

Hauptsächlich dient das Programm jedoch zur genaueren Demonstration der Eingaberoutine GOSUB 210. Sie übergibt nämlich den Tastencode an zwei Variablen:
IN erhält ASCII-Werte in leicht modifizierter Art. So werden z.B. meist nur die Zahlenwerte für Großbuchstaben übergeben, auch wenn Kleinbuchstaben betätigt wurden. IN$ enthält das Zeichen, also z.B. "A" bzw. "a" bei IN=65. Ferner werden folgende Steuerzeichen übergeben:

13 für ENTER, CR, 127 für DELETE
28 für Cursor nach links 29 für Cursor nach rechts
30 für Cursor nach unten 31 für Cursor nach oben

Es ist zu beachten, daß diese Obergaben sehr rechnerspezifisch sein können. Deshalb ist ein Experimentieren mit dem Programm sehr nützlich. Beachten Sie aber beim Schreiben von BASICODE-Programmen die o.g. Einschränkungen. Es sei noch erwähnt, daß die mit GOSUB 210 verwandte Routine GOSUB 200 bezüglich der Variablen sich gleichwertig verhält. Hierbei wird jedoch nicht die Tastenbetätigung abgewartet. Wenn beim Aufruf keine Taste berührt wurde, betragen IN=0 und IN$="".

```
1000 A=100: GOTO 20: REM ### TASTE ###
1010 SR$="Tastentest": HO=INT(HO/2)-7
1020 VE=0: GOSUB 110: GOSUB 150: PRINT
1030 PRINT"Tasteneingabe; 0 fuer Ende": GOSUB 250
1040 GOSUB 210: PRINT IN$; IN: IF IN$<>"0" THEN 1040
1050 GOSUB 250: GOTO 950
1060 REM ----Programm-Ende------
30000 REM Test der Tastatureingabe gemaess GOSUB 210
30010 REM und Demonstration von GOSUB 150, 250
32000 REM ----------------------
32010 REM H. Voelz; 9.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Zuweilen ist es notwendig die Tastatur bezüglich Eingaben zu sperren, damit ein Programm ohne äußere Störung, z.B. auch in Schulen oder Arbeitsgemeinschaften ablaufen kann. Im Gegensatz zu den meisten BASIC-Dialekten bietet auch hier BASICODE mit GOSUB 280 eine Lösung an, die das folgende Programm demonstriert. Ausgewertet wird hierbei die Variable FR. Sie muß vor dem Aufruf belegt werden:

FR=0 STOP/BRK-Taste ist wirksam
FR=1 STOP/BRK-Taste ist unwirksam

```
1000 A=100: GOTO 20: REM ### STOP ###
1010 SR$="Stoptest": HO=INT(HO/2)-6
1020 VE=0: GOSUB 110: GOSUB 150: PRINT: FR=1
1030 PRINT "Stop: FR =";FR: GOSUB 280
1040 PRINT "Tasteneingabe; 0 fuer aendern": GOSUB 250
1050 GOSUB 210: PRINT IN$; IN: IF IN$<>"0" THEN 1050
1060 INPUT"0: stop moeglich; 1: nicht; 2: Ende"; FR
1070 GOSUB 250: IF FR<2 THEN 1030
1080 GOTO 950
1090 REM ----Programm-Ende------
30000 REM Test auf Moeglichkeit der Stoptaste
30010 REM also CTRL C, BRK, Stop oder aehnlich
30020 REM erfolgt mit FR und GOSUB 270
30030 REM Wenn Sie das Programm mit BRK usw.
30040 REM unterbrechen, denken Sie bitte daran,
30050 REM mit GOTO 950 Ihren Rechner wieder in
30060 REM in den Normalzustand zu versetzen.
32000 REM ----------------------
32010 REM H. Voelz; 9.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

4. Weiterhin TEXT-Mode

In diesem Abschnitt werden die restlichen GOSUB-Routinen des Text-Mode behandelt.
Das Programm "TEXTUM" liest insbesondere über GOSUB 220 Zeichen, die auf dem Bildschirm stehen, in die Variable IN zurück. Es besteht also eine gewisse Verwandtschaft mit dem Befehl VPEEK im KC-BASIC. Außerdem werden wieder die Besonderheiten der Routinen GOSUB 200 bzw. 210 wirksam. In IN$ werden werden zwar Groß- und Kleinbuchstaben zurückgemeldet, jedoch in IN nur die Werte der Großbuchstaben. Für "A" und "a" gilt also IN=65. Solche Routinen sind u.a. günstig, wenn man menugesteuert wie im CAOS-Mode des KC 85/3 oder 4 arbeiten will. Das Programm "TEXTUM" macht allerdings einen anderen Gebrauch von dieser Routine. Zunächst verwendet es den Kernteil vom Programm "ZUTEXT". Danach werden die Zufallszeichen wieder vom Bildschirm gelesen und dicht aneinandergefügt. Hier wurden auch - trotz der Probleme, die sich beim Commodore infolge seiner unüblichen Zeichenkodierung ergeben - die Kleinbuchstaben verwendet.

```
1000 A=100: GOTO 20: REM ### TEXTUM ###
1010 HT=HO: VT=VE: RESTORE
1020 FOR I=1 TO 4: READ A$: PRINT A$: NEXT I
1030 INPUT"Taste druecken"; A: GOSUB 100
1040 FOR X=1 TO 200
1050   GOSUB 260: HO=RV*HT: GOSUB 260: VE=RV*VT
1060   GOSUB 110: GOSUB 260: PRINT CHR$(32+RV*95);
1070 NEXT X
1080 HO=0: VE=0: GOSUB 110
1090 GOSUB 250: PRINT "Text zusammenfassen": BH=0: BE=1
1100 FOR VE=1 TO VT
1110   FOR HO=0 TO HT
1120      GOSUB 220: IF IN=32 THEN 1170
1130      BO=HO: HO=BH: BV=VE: VE=BE: GOSUB 110
1140      PRINT CHR$(IN);
1150      HO=BO: VE=BV: GOSUB 110
1160      BH=BH+1: IF BH=HT THEN BH=0: BE=BE+1
1170    NEXT HO
1180 NEXT VE
1190 GOSUB 250: HO=0: VE=BV+1
1200 GOSUB 110: PRINT "Ich bin fertig";
1210 GOSUB 210: GOTO 950
1220 REM ----Programm-Ende------
25000 DATA "Der Commodore verfuegt ueber einen"
25010 DATA "vom ASCII-Code abweichenden Zeichen-"
25020 DATA "satz. Daher funktioniert dieses "
25030 DATA "Programm bei ihm abweichend !"
30000 REM ------------------------
30010 REM Es werden Zufallsbuchstaben an zufaellige
30020 REM Bildschirmpositionen gelegt. Dann werden
30030 REM sie vom Bildschirm zurueckgelesen und
30040 REM hintereinander ausgegeben. Dabei werden
30050 REM Klein- in Grossbuchstaben gewandelt.
32000 REM ------------------------
32010 REM H. Voelz; 9.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Das Programm beginnt mit einer Besonderheit, welche die Möglichkeiten von DATA-Zeilen demonstriert. Für sie sind im BASICODE Zeilen von 25000 an reserviert. In diesem Fall enthalten sie einen Kommentar bezüglich der Probleme beim Commodore. Die DATA-Zeilen werden mit READ A$ gelesen und anschließend mit PRINT A$ angezeigt. Nach diesem Prinzip ist es z. B. auch möglich, ein Programm sich selbst dokumentieren zu lassen.
Im normalen BASIC hat man bei Sortierprogrammen ein Problem mit den Klein- und Großbuchstaben. Durch ihren ASCII-Code werden sie nicht gleichwertig geordnet. BASICODE stellt u.a. zu diesem Zweck eine Routine GOSUB 330 bereit. Sie verwandelt alle Kleinbuchstaben - die in SR$ enthalten sind - in Großbuchstaben. Dann ist ein exaktes Sortieren möglich. Genau dies nutzt das Programm "BUBBLE" aus.

```
1000 A=5000: GOTO 20: REM ### BUBBLE ###
1010 SR$="Einfacher Bubble-Sort": VE=0: M=21
1020 HO=INT(HO/2)-14: GOSUB 110: GOSUB 150
1030 PRINT: PRINT"bitte etwas warten"
1040 DIM A$(M), B$(M), A(M)
1050 FOR I=0 TO M
1060   GOSUB 260: N=INT(10*RV)+3: SR$=""
1070     FOR J=0 TO N
1080       GOSUB 260: IF RV<.5 THEN X=ASC("A")+RV*52
1090       IF RV)=.5 THEN X=ASC("a")+(RV-.5)*52
1100       SR$=SR$+CHR$(X)
1110     NEXT J
1120   A(I)=I: A$(I)=SR$: GOSUB 330: B$(I)=SR$
1130 NEXT I
1140 GOSUB 250: PRINT "Dies sind die Woerter:"
1150 FOR I=0 TO M: PRINT A$(I),B$(I): NEXT I
1160 PRINT "Nun wird sortiert :";
1170 FOR I=0 TO M: F=0
1180   FOR J=0 TO M-1-I
1190     IF B$(A(J))<=B$(A(J+1)) THEN 1210
1200     B=A(J): A(J)=A(J+1): A(J+1)=B: F=1
1210   NEXT J
1220   PRINT I;: IF F=0 THEN I=M
1230 NEXT I
1240 PRINT: GOSUB 250: PRINT "sortierte Woerter"
1250 FOR I=0 TO M: PRINT A$(A(I)): NEXT I
1260 GOSUB 210: GOTO 950
30000 REM ----------------------
30010 REM Es wird der indirekte Bubble-Sort verwendet.
30020 REM Zur Vermeidung von garbage collection wird
30030 REM der Vergleich und Austausch ueber das Feld
30040 REM A(I) realisiert. Weiter werden die Woerter
30050 REM mit Grossbuchstaben in B$(I) abgelegt.
30060 REM Nach der Erzeugung werden beide Worttypen
30070 REM nebeneinander angezeigt.
32000 REM ----------------------
32010 REM H. Voelz; 11.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

In den Zeilen von Zeile 1070 bis 1110 wird ein Zufallswort erzeugt. Durch eine spezielle Erzeugung der Zufallsdaten werden hier sowohl die Probleme des Zeichensatzes des Commodore als auch die Auswahl von nur Buchstaben

gleichermaßen gelöst. Der Bereich der Zufallszahlen wird mit der Entscheidung IF RV<.5 geteilt und es werden getrennt große und kleine Buchstaben erzeugt. Dabei wird gerade wegen des Commodore von ASC("a") bzw. ASC("A") ausgegangen. Jedes so erzeugte Wort wird anschließend als zweites Wort mittels GOSUB 330 in ein Wort geändert, das nur Großbuchstaben enthält. Beide Varianten werden dann für die M=21 Wörter (Zeile 1150) zur Kontrolle angezeigt.
Das Sortieren erfolgt nur bei den Wörtern mit Großbuchstaben und zur Vermeidung der garbage collection indirekt über die Pointer in A(J). Verwendet wird ein verbesserter Bubble-Sort, welcher die bereits sortierten Wörter mit FOR J=0 TO M-1-I (Zeile 1180) ausblendet und über das Flag F entscheidet ob noch weiter sortiert werden muß. Die Anzahl der Sortierläufe wird angezeigt und anschließend das sortierte Feld.
Das Programm "FORMAT" arbeitet ebenfalls im Text-Mode und betrifft die richtige Formatierung von Zahlen. (In einigen Dialekten existiert PRINT USING.) Sie sollen auf der Variablen SR stehen und werden dann mit GOSUB 310 unter Berücksichtigung von zwei Parametern formatiert dargestellt:

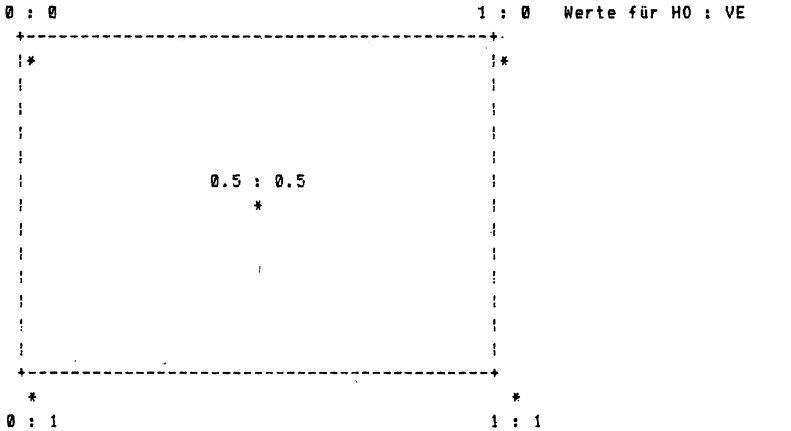
CT enthält die Anzahl der darzustellenden Ziffern und Hilfszeichen (Gesamtlänge von SR$),
CN benennt die davon hinter dem Dezimalpunkt stehenden Ziffern.

Je nach dem Rechnertyp sind hier beachtliche Leistungen, aber auch spezifische Rundungsfehler möglich. Deshalb wurde als Ausgangswert in SR$ eine sehr lange Ziffernfolge abgelegt. Sie wird schrittweise 14-mal durch 9 dividiert und je als formatierte Zahl und für den Rechner übliche Zahl dargestellt. So wird diese Routine einschließlich ihrer Grenzen demonstriert.

```
1000 A=100: GOTO 20: REM ### FORMAT ###
1010 SR$="Formatieren": HO=INT(HO/2)-9
1020 VE=0: GOSUB 110: GOSUB 150: PRINT
1030 SR$="123456789123": INPUT "Zeichenzahl ="; CT
1040 SR=VAL(SR$): INPUT "Nachkommastellen ="; CN
1050 PRINT: PRINT "SR$";TAB(CT+5);"SR"
1060 FOR I=1 TO 14
1070   GOSUB 310: PRINT SR$; " : "; SR: SR=SR/9
1080 NEXT I
1090 INPUT "0=Ende; 1=weiter";B
1100 IF B=1 THEN 1030
1110 GOTO 950
1120 REM ----Programm-Ende------
30000 REM Zeigt die Moeglichkeiten der Zahlen-
30010 REM darstellung. Der String ist bewusst
30020 REM so gross gewaehlt, dass auch die
30030 REM die Genauigkeit der Arithmetik und
30040 REM die Rundung getestet werden kann.
32000 REM ----------------------
32010 REM H. Voelz; 10.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

5. Der Grafik-Mode

Der Grafik-Mode ist jene Betriebsart bei der kleine Punkte (Pixel), Linien, Kurven usw. mit hoher Auflösung gezeichnet werden. Im Gegensatz zum BASIC bei den KC- und einigen anderen Rechnern ist dies im BASICODE eine spezielle Betriebsart. Sie muß eingeschaltet werden. Dies erfolgt durch den Befehl GOSUB 600. In diesem Zustand sind die meisten Befehle des Text-Mode nicht mehr zulässig. Ganz besonders sind dies GOSUB 110, 120, 150 und 220, aber auch die BASIC-Befehle PRINT und INPUT. Für Textdarstellungen ist wieder zum Text-Mode mit GOSUB 100 zurückzugehen. Dabei wird aber automatisch der Bildschirm gelöscht. Wenn Text im Graphik-Mode Text zum Bildschirm gebracht werden soll, ist daher ein spezieller Befehl (siehe übernächstes Programm) notwendig.
Im Graphik-Mode existieren für die einzelnen Rechner sehr unterschiedliche Pixel-Anzahlen. Sie wurden ja schon mit dem Programm DATEN für ihren Rechner angezeigt. Damit nun ein Grafik-Programm auf allen Rechnern ein gleichartiges Bild erzeugt, müssen die Rechnerunterschiede vom BASICODE automatisch berücksichtigt werden. Deshalb wurden spezielle Festlegungen für die Koordinaten des Bildschirm getroffen. Seine Breite und Höhe sind auf 1 normiert. Der Punkt 0, 0 liegt links oben. Der Mitte des Bildschirms entspricht 0.5, 0.5. Diese Koordinatenwerte werden mit den Variablen HG und VG übergeben. Im Text-Mode erhalten sie die ganzzahligen (integer) Schreibpositionen, im Grafik-Mode dagegen Werte von 0 bis 1; wobei die 1 nicht mehr zugelassen ist. Das Längen-Höhen-Verhältnis ist auf 4:3 festgelegt. Für ein Quadrat muß sich also z.B. HO um 0.3 und VE um 0.4 ändern. Anschaulich zeigt sich also für den Bildschirm das folgende Bild.



Im folgenden Programm wird nun mit diesen Mitteln ein kleines Bild gezeichnet. Die entsprechenden Punkt-Koordinaten sind in der DATA-Zeile 25001 abgelegt. Sie werden in die Felder X(I) und Y(I) übernommen. Mit den Daten in Zeile 25002 wird die Reihenfolge angegeben, mit der die Punkte "angelaufen" werden. Damit dies deutlich erkennbar ist, wurde eine Zeitverzögerung über GOSUB 450 eingefügt. Der vorher an die Variable SD übergebene Wert bewirkt eine Pause von SD*0.1 Sekunden. Diese Pause kann durch Betätigen einer Taste vorzeitig beendet werden.

Mit dem Befehl GOSUB 620 wird ein Punkt entsprechend den Werten in HO und VE gesetzt. Gleichzeitig liegt dann an dieser Stelle der unsichtbare Grafik-Cursor.
Mit dem Befehl GOSUB 640 wird vom Punkt des Grafik-Cursors eine Linie zum durch HO und VE bestimmten Punkt gezogen und der Grafik-Cursor auf diesen Endpunkt gesetzt.
Für die Grafik-Befehle kann über CN auch ein "Farbwert" übergeben werden. CN=0 bedeutet die Vordergrundfarbe (immer weiß) und CN=1 die Hintergrundfarbe (stets schwarz). Mehr als diese beiden "Farbwerte" sind im BASICODE-3 nicht vorhanden. Für echte Farbdarstellungen muß also auf die Spezifika des jeweiligen Rechners zurückgegriffen werden. Dies sollte aber nur ausnahmsweise und dann in den Zeilen 20000 bis 24999 mit ausführlichen Kommentaren erfolgen (Zeilen ab 30000).

```
1000 A=100: GOTO 20: REM ### GRAFIK ###
1010 DIM X(4), Y(4): RESTORE
1020 FOR I=0 TO 4 :READ X(I), Y(I): NEXT I
1030 GOSUB 600: CN=0: GOSUB 250
1040 HO=X(0): VE=Y(0): GOSUB 620
1050 FOR I=0 TO 7
1060   READ X: SD=5: GOSUB 450
1070   HO=X(X): VE=Y(X): GOSUB 630
1080 NEXT I
1090 GOSUB 250: GOSUB 210: GOTO 950
25000 REM---- Data-Zeilen ------
25010 DATA .25, .75, .25, .3, .75, .3, .75, .75, .5, .1
25020 DATA 1, 2, 0, 3, 1, 4, 2, 3
30000 REM ---- Programm-Ende-----
30010 REM Es wird ein Bild gezeichnet.
32000 REM ----------------------
32010 REM H. Voelz; 11.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Ein zweites Grafik-Programm zeigt bereits etwas genauer die Leistungsfähigkeit der Grafik-Routinen im BASICODE. Es wird das ineinander geschachtelte N-Eck gezeichnet. Mit den Eingabeparametern können Sie versuchen, die grafischen Grenzen Ihres Rechners zu erproben. Vielleicht gehen Sie dazu vom Dreieck mit einem Dreh-Winkel von 20 Grad aus und verkleinern den Winkel und/oder erhöhen die Eckenzahl. Bei der Wahl eines zu großen Drehwehwinkels vergrößert sich das N-Eck und es muß ein spezieller Abbruch erfolgen. Deshalb ist das AND in Zeile 1210 notwendig. Zur Unterstützung Ihres Gedächtnisses wird am Ende jedes Bildes der alte Winkel angezeigt. Dieser und der folgende Text bezüglich der Fortsetzung des Programms sind Texteingaben, die im Grafik-Mode mittels SR$ an die Routine GOSUB 650 übergeben werden.
Schließlich sei noch auf die Klammern im Zusammenhang mit den logischen Befehlen hingewiesen. Gemäß Zeile 1040 und 1210 sind sie um die logischen Ausdrücke im Zusammenhang mit OR, AND und NOT in BASICODE (wiederum zur Kompatibilität bezüglich aller Rechner) notwendig. Generell sind deshalb die logischen Operatoren nur bezüglich Vergleichen zulässig. Unter anderem wird "wahr" nämlich in den verschiedenen Rechnern mit +1 oder -1 dargestellt. Verboten ist folglich in BASICODE so etwas wie 7 AND 45.

```
1000 A=100: GOTO 20: REM ### N-ECK ###
1010 HO=INT(HO/2)-6: VE=0
1020 GOSUB 110: SR$="N-ECK": GOSUB 150: PRINT
1030 INPUT "Anzahl der Ecken"; N: N=INT(N)
1040 IF (N>12) OR (N<3) THEN PRINT "Fehler": GOTO 1020
1050 Q=6.283185/N: DIM X(N), Y(N): XX=4/3: YY=.5
1060 R=1/3: P2=1/15: INPUT"Drehwinkel =";W
1070 D=W*.0174533: CN=0: REM pi/180
1080 S=SIN(D): C=COS(D): P=COS(Q): T=SIN(Q)
1090 P4=T/(C*T+S-P*S): X(1)=P: Y(1)=T: GOSUB 600
1100 FOR I=1 TO N-1
1110    P=I*Q: X(I)=COS(P): Y(I)=SIN(P)
1120 NEXT I
1130 X(0)=1: Y(0)=0: X(N)=1: Y(N)=0
1140 HO=YY+R*X(0): RV=XX*R: VE=YY-RV*Y(0): GOSUB 620
1150 FOR I=1 TO N
1160    HO=YY+R*X(I): VE=YY-RV*Y(I): GOSUB 630
1170 NEXT I
1180 FOR I=0 TO N
1190    P=X(I)*C-Y(I)*S: Y(I)=Y(I)*C+X(I)*S: X(I)=P
1200 NEXT I
1210 R=R*P4: IF (R>P2) AND (R<.5) THEN 1140
1220 HO=0: VE=.1: SR=W: GOSUB 300: SR$="Winkel = "+SR$
1230 GOSUB 650: VE=.2: SR$="1:weiter  0=Ende"
1240 GOSUB 650: GOSUB 210: IF IN=49 THEN 1000
1250 GOTO 950
30000 REM Es wird ein geschachteltes N-Eck gezeichnet
30010 REM Durch den Winkel wird die Dichte bestimmt
32000 REM ----------------------
32010 REM H. Voelz; 11.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

6. Die Tonausgabe

Eine Tonausgabe - der Ton zur Aufmerksamkeit - wurde bereits mehrfach verwendet. Natürlich kann BASICODE etwas mehr obwohl auch hier wieder auf die Grenzen der "schwächsten" Rechner Rücksicht genommen werden mußte. Die Tonausgabe ist daher nur auf einen Kanal beschränkt. Es kann also nicht mehrstimmig musiziert werden. Für den einen Ton kann aber über drei Parameter alles wesentliche erreicht werden:

SP die Tonhöhe in Halbtonabstufung mit 69 als Kammerton a (etwa 440 Hz).
SD die Tondauer in Stufen zu 0.1 Sekunden.
SV die Lautstärke mit 0 = Null und 15 = maximal

Die Grenzen für die Tonhöhe hängen vom Rechnertyp ab. Aber in der Regel sind mehrere Oktaven möglich. Dennoch ist es wegen des Klanges sinnvoll, sich auf den mittleren Bereich zu beschränken. Aufgerufen wird die Tonausgabe mit GOSUB 400. Mit dem folgenden Programm wird ein einfaches Kinderlied gespielt und der zugehörige Text parallel auf dem Bildschirm angezeigt.

```
1000 A=100: GOTO 20: REM ### LIED ###
1010 RESTORE: SR$="Alle meine Entchen": VE=0
1020 HO=INT(HO/2)-13: GOSUB 110: GOSUB 150: PRINT: PRINT
1030 FOR I=1 TO 27
1040     SV=0: SD=2: GOSUB 400
1050     SV=15: READ A$, SP, SD
1060     IF SD<0 THEN SD=-SD: PRINT
1070     GOSUB 400: PRINT A$;: SD=1: SV=0: GOSUB 400
1080 NEXT I
1090 GOSUB 210: GOTO 950
25000 REM---- Data-Zeilen ------
25010 DATA "Al",67,3,"le",69,3," mei",71,3,"ne",72,3
25020 DATA " Ent",74,6,"chen",74,6,"schwim",76,-3
25030 DATA "men",76,3," auf",76,3," dem",76,3," See",74,9
25040 DATA "schwim",76,-3,"men",76,3
25050 DATA " auf",76,3," dem",76,3," See",74,9
25060 DATA "Koepf",72,-3,"chen",72,3," in",72,3," das",72,3
25070 DATA " Was",71,6,"ser",71,6,"Schwaenz",74,-3
25080 DATA "chen",74,3," in",74,3," die",74,3," Hoeh",67,10
30000 REM ---- Programm-Ende-----
30010 REM Ein Kinderlied wird gespielt
32000 REM ------------------------
32010 REM H. Voelz; 11.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

Auffällig sind in diesem Programm noch die Zeilen 25060 und 25080. Sie erreichen nämlich die in BASICODE maximal zugelassene Länge von 60 Zeichen. Bis auf das Leerzeichen nach DATA sind hier auch keine redundanten Leerzeichen enthalten. Einige Rechner besitzen keinen längeren Eingabepuffer und hierauf muß beim Einlesen des BASICODE-Programms Rücksicht genommen werden. Da Leerzeichen ohnehin von BASICODE weitgehend beim Abspeichern und beim Einlesen entfernt werden, stehen sie hier in allen Programmen vor allem zur besseren Obersicht. Notwendig sind Leerzeichen nur vor GOSUB, GOTO und THEN und dabei auch nur dann, wenn das vorangehende Zeichen kein ":" oder ")" ist.

7. Umgang mit Daten-Files

Neben Programmen sind auch oft Daten zu speichern. Ihre Zusammenfassung wird oft als File bezeichnet. In vielen BASIC-Dialekten erfolgt dies bezüglich Felder mit den Befehlen CSAVE* bzw. CLOAD*. In anderen Fällen werden Dateien geöffnet (OPEN), die Daten in der Datei abgelegt oder aus der Datei geholt. Danach ist die Datei wieder zu schließen (CLOSE). Dieses Prinzip erscheint dem Unerfahrenen oft recht kompliziert, dennoch ist es effektiv und insbesondere durch den Umgang mit Disketten und Festplattenspeichern bestimmt. Hier sucht man nämlich nicht nach dem Ort, wo die Daten stehen oder abgelegt werden sollen. Das ist nur typisch für die Kassettenrecorder, bei denen das Zählwerk genutzt wird. Die Buchführung muß man dann selber auf einem "Zettel" übernehmen.

Für BASICODE wurde die sequentielle Datei ausgewählt. Das Öffnen der Datei bewirkt die Bereitstellung eines entsprechendes Speicherplatzes, der durch den Namen der Datei beschrieben wird. Deshalb muß der Name hierfür auf der Variablen NF$ übergeben werden. Er darf maximal 7 Zeichen enthalten. Ferner muß mitgeteilt werden, ob in die Datei geschrieben oder aus ihr gelesen werden soll. Dies erfolgt durch den Wert der Variablen NF. Es bedeuten

NF gerade - (0, 2, 4, 6) Lesen vom Speicher,
NF ungerade (1, 3, 5, 7) Schreiben in den Speicher.

Die verschiedenen Möglichkeiten wurden deshalb gewählt, weil an einem Rechner mehrere Speicher angeschlossen sein können, z.B. mehrere Kassettenrecorder und/oder Diskettenlaufwerke usw. Für die einfachen Rechner mit nur einem Speicher z.B. Kassettenrecorder genügen also die beiden Werte 0/1.
Zunächst sei die Speicherung von Daten auf einem externen Medium , z.B. auf Kassette betrachtet. Mit NF$ und NF=1 kann nun der Speichervorgang eingeleitet, also die Datei eröffnet werden. (Im folgenden Programm erfolgt dies ab Zeile 1120.) Hierzu ist lediglich GOSUB 500 aufzurufen. Dadurch wird der Rechner in den Mode zur Dateiarbeit versetzt. Anschließend werden Daten aus dem internen RAM des Rechners auf Kassette geschrieben. Hierzu werden die Daten als SR$ an die Routine GOSUB 560 übergeben. Dieser Schritt kann beliebig oft wiederholt werden. Echt gespeichert sind diese Daten aber erst dann, wenn die Datei (das File) abgeschlossen wird. Dies erfolgt mit dem Befehl GOSUB 580. Dadurch wird auch wieder der normale Zustand des Rechners erreicht.
Aus dem Speicher werden die Daten in analoger Weise geholt. Für die Routine GOSUB 500 ist dazu lediglich NF=0 (oder 2, 4, 6) zu wählen. Jetzt wird aber geprüft ob der Name NF$ im File existiert. Wenn dies der Fall ist, wird von der Routine in IN der Wert 0 übergeben. Bei anderen Werten existiert das File nicht. Das Lesen erfolgt mit der Routine GOSUB 540 und es werden die String immer auf IN$ übergeben. Abschließend muß das File wieder mit GOSUB 580 geschlossen werden.
Generell nach jeder File-Routine wird an IN ein Code übergeben, der darauf hinweist, ob der Befehl erfolgreich, d.h. fehlerfrei realisiert werden konnte. Es ist daher sinnvoll immer IN abzufragen (dies wurde zur Vereinfachung des Programms im folgenden Beispiel weggelassen). Ist IN<>-1 so war alles in Ordnung.
Mit dem folgenden Programm können Sie nun Zeichenketten, die z.B. einen Namen und die dazugehörende Telefon-Nr. enthalten, abspeichern. Im Gegensatz zu CSAVE* und CLOAD* brauchen dabei nur die wirklich belegten Feldelemente gespeichert zu werden. Das ist doch sehr vorteilhaft! Die Abfrage erfolgt später nach dem Anfang der Zeichenkette, wobei die Zeichenzahl frei wählbar ist.
Sie rufen also zunächst das Programm auf und starten es. Dann geben Sie die Namen und Nummern gemeinsam ein. Wenn Sie diese Arbeit abgeschlossen haben, geben Sie END ein. Dadurch geht der Rechner in die Dateiarbeit über und verlangt den Namen für Ihre Datei. Vielleicht wählen Sie TELEFON. Nun werden die Daten abgelegt (vorher Kassettenrecorder starten!). Wenn dies erfolgt ist, kehrt der Rechner in den normalen Mode zurück. Er verlangt "1:File erzeugen 2:File lesen", und Sie können entsprechend fortfahren. Dieser Zustand wird aber auch beim Start des Programm erreicht und so ist es möglich ein vorhandenes File einzulesen.

```
1000 A=5000: GOTO 20: REM ### FILE ###
1010 M=99: SR$="File-Arbeit": VE=0: DIM A$(M)
1020 HO=INT(HO/2)-9: GOSUB 110: GOSUB 150: PRINT
1030 INPUT"1:File erzeugen 2:File lesen"; B
1040 IF B=1 THEN 1070
1050 IF B=2 THEN 1170
1060 PRINT"falsche Eingabe": GOTO 1030
1070 PRINT "Eingabe von Namen: Telefon-Nr."
1080 PRINT"Beenden mit: END"
1090 FOR I=0 TO M
1100 PRINT "Nummer:";I+1;" ";
1110   INPUT A$(I): IF A$(I)="END" THEN N=I: I=M
1120 NEXT I
1130 GOSUB 250: PRINT "speichern des File"
1140 INPUT"Name =";NF$: NF=1: GOSUB 500
1150 FOR I=0 TO N: SR$=A$(I): GOSUB 560: NEXT I
1160 GOSUB 580: GOTO 1210
1170 PRINT"Name =";: NF=0: INPUT NF$: GOSUB 500
1180 FOR I=0 TO M
1190   GOSUB 540: A$(I)=IN$: IF IN$="END" THEN I=M
1200 NEXT I
1210 GOSUB 580: PRINT "Das File kann genutzt werden"
1220 PRINT "gesuchter Anfang des Namens; 0=Ende"
1230 INPUT A$: L=LEN(A$): N=-1: IF A$="0" THEN GOTO 950
1240 FOR I=0 TO M
1250   IF LEFT$(A$(I),L)=A$ THEN N=I: I=M
1260 NEXT I
1270 IF N<0 THEN PRINT "nicht vorhanden": GOTO 1220
1280 PRINT A$(N): GOTO 1220
30000 REM ---- Programm-Ende-----
30010 REM Es wird ein Datenfile erzeugt, in welchem
30020 REM gesucht werden kann. Es kann gerettet und
30030 REM geladen werden. Als Beispiel koennen hier
30040 REM Namen und Telefon-Nr. eingegeben werden.
30050 REM Beispiel: "Meyer 27 508 35"
30060 REM Gesucht wird nach dem Anfang der Zeichenkette.
30070 REM z.B. "Mey" oder "M" usw.
32000 REM ----------------------
32010 REM H. Voelz; 11.5.89; fuer Rundfunk
32020 REM XT-compatibler Rechner
```

8. Ergänzungen zum Bascoder

Der Bascoder ist das entscheidende Programm, welche die Möglichkeiten von BASICODE zu realisieren gestattet. Er berücksichtigt also die Besonderheiten des jeweiligen Rechners. Deshalb müssen soviel unterschiedliche Bascoder existieren, wie Rechnertypen für BASICODE verwendet werden. Dieses Programm wird mit der üblichen rechnerspezifischen Art - meist über einen Kassettenrecorder - eingelesen.
Neben den verschieden zuvor besprochenen Routinen stellt der Bascoder vor allen die neuen Schreib- und Lese-Routinen für das einheitliche BASICODE-Kassetteninterface bereit. Nur über dieses Interface ist ja der einheitliche Datenaustausch möglich. Hierbei wird die 0 durch eine 1200-Hz-Welle und die 1 durch zwei 2400 Hz-Wellen dargestellt. Die Datenübertragung erfolgt

für jedes Byte asynchron mit einem Start- und zwei Stoppbit. Zusätzlich wird das achte Datenbit, also das höchstwertigste invertiert. Dadurch stehen für die meisten Byte "eigentlich" drei Stoppbit bereit. Dies erhöht wesentlich die Übertragungssicherheit und ermöglicht insbesondere eines schnelles Einsynchronisieren nach einem Fehler. Durch diese Betriebsart ist BASICODE sehr zuverlässig. Auf Mittelwelle können so Entfernungen von mehr als tausend km überbrückt werden.
Die BASICODE-Programme werden im ASCII-Format, also nicht mit Token, gespeichert. Für die KC-Rechner entspricht dies etwa der Speicherung mit LIST#1 und dem Laden mit LOAD#1. Genau wie in dieser Betriebsart müssen die Programme dann zunächst die Programme in die interne Darstellung umgewandelt werden. Für alle in BASICODE gültigen BASIC-Befehle (siehe Tabelle 1) realisiert dies der Bascoder. Hierfür werden u.a. Begriffe wie Obersetzen oder translate verwendet. Für das Schreiben von BASICODE-Programmen auf Kassette erfolgt dann wieder die Rückwandlung in das ASCII-Format. Für die Speicherung sind weiter zwei Fälle zu unterscheiden. Normalerweise sollte nur das eigentliche BASICODE-Programm ab Zeile 1000 gespeichert werden. Es ist aber bei einigen Bascodern auch möglich, den Bascoder kombiniert mit dem Programm zu speichern. Dies sind dann alle BASIC-Zeilen - beginnend bei Zeile 10 bis zum zum Ende des BASICODE-Programms.
Der Befehl LIST hat meist im BASICODE die gleichen Eigenschaften wie im BASIC Ihres Rechners. Dies bedeutet, daß in der Regel mit LIST auch die BASIC-Zeilen Ihres Bascoders angezeigt werden. Für die Anzeige des BASICODE-Programm empfiehlt sich daher ein LIST 1000- . Diese Anzeige ist auch deshalb wichtig, damit Sie nicht eventuell versehentlich in Ihrem Bascoder editieren.

9. Ergänzungen zu den GOSUB-Routinen

Hier werden nur jene GOSUB-Routinen berücksichtigt, für die entsprechend den vorangegangenen Beschreibungen noch Ergänzungen notwendig sind bzw. dort nicht behandelt wurden. Eine vollständige Zusammenstellung enthält Tabelle 2 im Anhang. Zu Anfang stehen im folgenden lediglich die ausgewählten Zeilennummern

20 Ist wegen GOTO eigentlich keine Subroutine. Dieser Befehl darf und muß einzig in Zeile 1000 stehen. Die übergebenen Werte für HO, VE sind wegen des Zählbeginns bei Null um eins kleiner als die entsprechenden Zeilen-, und Spaltenzahlen. Die Rückkehr von dieser Initialisierungsroutine führt zur Zeile 1010. Während die Pixelzahlen weitgehend vom Rechnertyp abhängen, werden bei vielen Rechnern 24 Zeilen und 40 Zeichen je Zeile verwendet. In jedem Fall ist es günstig die Werte des jeweiligen Rechners abzufragen und dann im Programm zu benutzen.

110 Hier sollten immer die Maximalwerte von HO und VE berücksichtigt werden, um eine optimale Bildschirmdarstellung zu erreichen. Beispiele enthalten die Programme ZUTEXT in Zeile 1030 und TASTE in Zeile 1010. Die Werte von HO und VE werden durch den Aufruf nicht verändert.

150 Hierbei ist zu beachten, daß die Länge des String einschließlich der 2 mal 3 Leerzeichen nicht über das Zeilenende hinausgeht.

200 Es sei noch einmal auf die rechnerspezifische Ausgabe in die Variablen IN und IN$ sowie auf den Unterschied für beide und die zulässigen Steuerzeichen hingewiesen. PRINT CHR$(IN) ist nicht sinnvoll.

220 Diese Routine liefert nur einen Wert in IN. IN$ wird durch sie also nicht verändert. In IN werden aber nur die ASCII-Werte für Großbuchstaben übergeben. Wenn auf der Position des Bildschirmes ein Kleinbuchstabe steht, wird der Wert in IN den des zugehörigen Großbuchstabens annehmen. Werden Koordinaten außerhalb des Schirmes angesprochen, so ist IN=0.

270 Sofern der Stringraum entsprechend organisiert ist, wird zunächst eine garbage collection ausgeführt. Dann werden die freien Speicherplätze (Byte) berechnet. Im Gegensatz zu FRE werden der Speicherplatz von FRE (X) und FRE (X$) addiert auf die Variable FR übergeben. Die kann z.B. dazu benutzt werden, ein Array maximal zu dimensionieren.

300 Im Gegensatz zur vergleichbaren, in BASICODE nicht zugelassenen Anweisung STR$ werden hier alle Leerzeichen (Spaces) entfernt.

310 Wie das Beispielprogramm FORMAT demonstriert kann der String durchaus mehr Stellen als die eigentliche Zahl enthalten. Bei einigen Rechnern werden dadurch die Eigenschaften der implementierten Arithmetik sichtbar. Bei der meist gebräuchlichen Binärarithmetik zeigen die zusätzlichen Stellen dann die Konvertierungsfehler beim Obergang von der Dezimalzahl (im ASCII-Code) an. Diese Fehler werden sonst meist nur bei der Subtraktion in der Umgebung von Null sichtbar.

350 Diese Routine wurde nicht besprochen. Sie wird nur im Zusammenhang mit Druckern verwendet und sendet den String SR$ an den Drucker. Initialisierungsroutinen für Drucker enthält BASICODE in der Regel nicht. Es besteht eine gewisse Analogie zum unerlaubten BASIC-Befehl LPRINT bzw. PRINT#2.

360 Sendet Newline, also 0D, 0A (CRLF) an den Drucker.

400 Die Subroutine wird erst beendet, wenn der Ton abgelaufen ist. Es gelten folgende Datengrenzen, die aber nicht von jedem Rechner voll erfüllt werden:

SP (Sound Pitch) für die Tonhöhe, 0 < = SP < = 127
Kammerton a bei SP=69, zentrales C bei SP=60
Eine ganze Zahl bedeutet einen Halbtonschritt
SD (Sound Duration) für die Dauer, 1 < = SD < = 255
SV (Sound Volume) für die Stärke, 0 < = SV < = 15
= 0 bedeutet kein hörbarer Ton, Pause.

450 Der Abbruch dieser Routine kann auf zwei Arten erfolgen. Durch Ablauf der Zeit oder vorher durch Betätigen einer beliebigen Taste. Im ersten Fall gilt SD=0, IN$="" und IN=0. Im zweiten Fall enthält SD die noch verbliebene Restzeit und IN, IN$ die Werte gemäß GOSUB 200, 210. Für eine nicht unterbrechbare Warteroutine kann GOSUB 400 mit SV = 0 benutzt werden.

500 Die Routinen GOSUB 500, 540, 560 und 580 verwenden teilweise die unterste Bildschirmzeile für Anweisungen und Meldungen. Sie sind nur im Text-Mode nutzbar. Als Statusmeldung fungiert die Variable IN wie folgt:

IN=0 : Fehlerfreier Verlauf.
IN=1 : Es wurde jetzt oder früher der letzte String gelesen.
IN=-1: Ein nicht behebbarer Fehler ist aufgetreten.

Mit dem Code in NF werden vorbereitet:

NF = 0 : Lesen von einer BASICODE-Kassette.
NF = 1 : Schreiben auf eine BASICODE-Kassette
NF = 2 : Lesen von einem rechnerspezifischen externen Speicher
NF = 3 : Schreiben auf einen rechnerspezifischen externen Speicher
NF = 4 : Lesen von Diskette
NF = 5 : Schreiben auf Diskette
NF = 6 : Lesen von einer zweiten Diskette
NF = 7 : Schreiben auf einer zweiten Diskette.

Nur für 0 und 1 wird universelle Lesbarkeit gemäß BASICODE bewirkt. Auf der untersten Bildschirmzeile können bei der File-Arbeit nützliche Hinweise erscheinen. Bei der Diskettenarbeit stellen die meisten Basco-der die Möglichkeit von bis zu drei geöffnete Files bereit.

540 Im Normalfall ist IN=0 und IN$ enthält den gelesenen String. Bei Lesefehlern wird IN=-1. Der letzte String wird mit IN=1 an IN$ übergeben. Bei weiteren Zugriffen bleibt IN=1 und IN$ wird ein Leerstring. Wenn auf eine nicht geöffnete Datei zugegriffen wird, erfolgt eine Fehlermeldung.

600 Die Subroutinen GOSUB 110, 120, 150 und 220 sind verboten. Im Grafik-Mode ist auch keine File-Arbeit möglich. Der Grafik-Mode muß mit GOSUB 100 beendet werden.

620 Beachten Sie die unterschiedlichen Maßstäbe in X- und Y-Richtung von 4:3 und die Belegung mit dem Farbwert CN.

630 Die Linie geht von der aktuellen (unsichtbaren) Cursor-Position zu den Pixel-Koordinaten HO, VE.

650 Der unsichtbare Cursor ist die linke obere Ecke des Schriftbeginns. Es ist darauf zu achten, daß die Schrift nicht den rechten Bildrand überschreitet. Die Schrift erscheint meist im normalen Textformat. Nach der Schriftausgabe ist der Ort des grafischen Cursors in HO und VE enthalten.

10. Hinweise zu den Variablen

Für die Variablen gibt es im BASICODE Einschränkungen, die in den Tabellen 5 und 6 zusammengefaßt sind. Variablen in BASICODE bestehen aus ein oder zwei Zeichen, wovon das erste ein Großbuchstabe sein muß. Anfügen von %, ! oder # ist nicht erlaubt. Alle Variablen besitzen das Format real und je nach Rechnertyp zumindest 6 gültige Mantissenziffern. Stringvariablen sind durch Anhängen von $ gekennzeichnet.
Logische Variable werden indirekt realisiert und können nur 'wahr' oder 'falsch" sein. Mit logischen Variablen darf nicht numerisch gerechnet werden, denn "wahr" wird in Rechnern unterschiedlich durch +/- 1 oder andere Werte realisiert. Deshalb ist etwa A=3*(B=1) nicht erlaubt. Statt dessen ist beispielsweise zu verwenden A=0: IF B=1 THEN A=3.
Bevor eine Variable verwendet wird, muß ihr ein Wert zugewiesen werden. Nicht jeder Rechner setzt die Variablen am Programmbeginn zu Null.
Für ein schnelles Programm ist
auch die Reihenfolge dieser Initialisierung zu beachten. Oft gebrauchte Variablen sollten zuerst - eventuell auch mit Dummy-Werten - belegt werden. Arrays sollten ebenfalls vor jeder Anwendung dimensioniert werden. Nicht alle Rechner wählen sonst automatisch den Wert 10.

11. Zu den erlaubten BASIC-Befehlen

Hier werden nur eventuelle Abweichungen vom üblichen Gebrauch notiert. Eine Liste der erlaubten Befehle enthält Tabelle 1 im Anhang.

AND Es darf nur auf logische Ausdrücke (Vergleiche) bezogen werden. Die zugehörigen Ausdrücke sollten immer geklammert werden. Also z.B. (X>45) AND (X<70). Zuweisungen von logischen Ergebnissen an eine Variable sind zur besseren Obersicht und Verkürzung zulässig. Also z.B.: Q = (A=5) AND (B=0): IF Q THEN ...

ASC Nicht alle Computer liefern in allen Fällen exakt den gleichen Wert. Das ist jedoch für die Buchstaben weitgehend erfüllt. Die Nutzung der Funktion sollte daher mit Umsicht und für nur ein Zeichen erfolgen. (Besonderheiten beim Commodore beachten!)

CHR$ Bei Werten, die kleiner als 32 sind (Steuerzeichen) empfiehlt sich Vorsicht.

DIM hat immer vor der Nutzung von Feldern zu erfolgen. In BASICODE sind maximal 2 Dimensionen, also z.B. DIM A(3,15) zugelassen. Das Feld beginnt stets mit dem nullten Element, also A(0,0).

FOR...TO...STEP...NEXT: Die Schleife wird mindestens einmal durchlaufen. STEP und der Wert danach darf weggelassen werden, dann beträgt die Schrittweite 1. Nach NEXT muß immer die zugehörige Variable stehen. Zu einem FOR ist auch nur ein NEXT zulässig. D.h. die Schleife kann nur an einer einzigen Stelle verlassen werden. Aus der Schleife darf nicht herausgesprungen werden. Vorzeitiges Verlassen über eine Bedingung erfolgt, durch Setzen des Laufparameters auf seinen Endwert und Sprung zum NEXT.

GOSUB Es muß eine Zahl und keine Variable als Zeilennummer folgen. Eine nicht existierende Zeilenzahl darf nicht verwendet werden. IF ... GOSUB ist nicht zulässig, richtig ist IF ... THEN GOSUB.

GOTO Wie GOSUB, mit Ausnahme von 20 und 950. Auch IF ... GOTO ist nicht zulässig. Auch wenn meist IF ... THEN Zeilennummer funktioniert sollte konsequenterweise IF ... THEN GOTO benutzt werden.

LOG Bezieht sich auf die Basis e. Es ist zu beachten, daß in einigen Rechnern hierfür auch LN existiert, was nicht verwendet werden darf. In diesen Fällen kann sich LOG fälschlicherweise auf den dekadischen Logarithmus beziehen.

MID$ Erfordert drei oder zwei Werte. MID$(A$,5) ist erlaubt.

NEXT Verlangt die Laufvariable, siehe FOR.

NOT Verlangt Klammern, siehe AND

ON Nach ON darf die Variable nur die zulässigen Werte annehmen, also von 1 bis zur Anzahl der nach GOTO bzw. GOSUB stehenden Adressen.

OR Siehe Bemerkungen unter AND.

PRINT Zur Formatierung existieren nur das Komma, Semikolon und TAB. Mehrere Druckaufträge hinter PRINT müssen generell durch ein Semikolon getrennt werden. Es wird empfohlen, TAB oder Komma durch GOSUB 110 zu ersetzen

REM Es gibt nur REM und keine alternativen Zeichen, wie ! oder . In einer REM-Zeile sollte kein Doppelpunkt verwendet werden.

RESTORE existiert nur ohne Zeilenzahl.

TAB TAB(0) ist nicht erlaubt. Vorsicht! Es gibt einige Computer die mit eins 1 zu zählen beginnen. Für die meisten ist die erste Position 0. Subroutine GOSUB 110 befreit von dieser Unsicherheit.

VAL nimmt bei nicht rein numerischen Argumenten in verschiedenen Rechnern unterschiedliche Werte an.

Anhang

Tabelle 1. Zusammenstellung der in BASICODE gültigen BASIC-Befehle

ABS	AND	ASC	ATN	CHR$	COS	DATA	DIM	EXP	FOR	GOSUB
GOTO	IF	INPUT	INT	LEFT$	LEN	LET	LOG	MID$	NEXT	NOT
ON	OR	PRINT	READ	REM	RESTORE	RETURN	RIGHT$	SGN	SIN	SQR
STEP	TAB	TAN	THEN	TO	VAL					
+	-	x	/	^	=	<	>	<=	>=	< >

Im begrenzten Umfang ist DEF FN möglich

Tabelle 2. Obersicht der GOSUB-Befehle mit Hinweis auf etwa äquivalente Befehle im KC-BASIC

20	Programmstart, System-Reset, Variable löschen usw.	CLEAR
100	Bildschirm löschen und Text-Modus einschalten	CLS
110	Cursor auf die Position HO, VE	LOCATE
120	Cursor-Position in HO, VE zurückholen	VGET, POS
150	Auffälliges Anzeigen von SR$; rechts und links 3 Spaces	
200	Daten einer eventuell gedrückten Taste in IN$ und IN	INKEY$
210	wie 200, jedoch mit warten auf Tastendruck	
220	Holen des Zeichens aus Schirmposition HO, VE auf IN	VGET$
250	Erzeugen eines kurzen Aufmerksamkeitstones	BEEP
260	Zufallsvariable in RV mit 0 <= RV < 1	RND
270	Ausführen von garbage collection und Speicherplatz in FR	FRE(X)
280	Aus- bzw. Einschalten der STOP/BRK-Taste FR=0 bzw. 1	
300	SR wird ohne Space in SR$ gewandelt	STR$
310	wie 300, jedoch als Zahl mit CT und CN formatiert	USING
330	Alle Kleinbuchstaben in SR$ in Großbuchstaben wandeln	
350	Obergabe von SR$ an den Drucker	PRINT#2
360	In neue Zeile mit Drucker (CRLF)	
400	Erzeugung eines Ton gemäß SV, SD und SP	SOUND
450	Warten von maximal SD*0.1 s auf einen Tasteneindruck	PAUSE
500	Eröffnen eines File mit Namen NF$ gemäß NF	OPEN
540	Aus dem File wird ein String an IN$ übergeben	LOAD*
560	SR$ wird in das File geschrieben	SAVE*
580	man schließe den Bestand mit Code NF ab	CLOSE
600	Rechner auf graphischen Betrieb umschalten	SCREEN
620	Setzen eines Punktes in die Position HO, VE mit Farbe CN	PSET
630	Zeichnen einer Linie zum Punkt HO, VE in Farbe CN	LINE
650	SR$ an der Position HO, VE anzeigen (Grafik-Mode)	
950	Beenden des BASICODE-Mode	END

Tabelle 3. Nicht erlaubte BASIC-Befehle der KC-Rechner im BASICODE-Mode

!	AT	BEEP	BLOAD	BORDER	BYE	CALL*	CALL	CIRCLE	COLOR
CLEAR	CLOAD*	CLOSE	CLS	CONT	CSAVE*	CSRLIN	DEEK	DOKE	END
FRE	INK	INKEY$	INP	INPUT#	INSTR	JOYST	KEY	KEYLIST	LINE
LINES	LIST#	LOAD#	LOAD*	LOCATE	NULL	OPEN	OUT	PAPER	PAUSE
PEEK	PI	POKE	POSE	PRESET	PRINT#	PSET	RND	RANDOMIZE	RENUMBER
RUN	SOUND	SPC	STOP	STRING$	STR$	SWITCH	TROFF	TRON	USR
VGET$	VPEEK	VPOKE	WAIT	WIDTH	WINDOW	(LN durch LOG ersetzen)			

Tabelle 4. Grundsätzlicher Zeilen-Aufbau beim BASICODE

0 - 999	Bascoder (für jeden Computer anders)
1000	Erste Zeile des BASICODE-Programms. Sie muß folgende Form haben: 1000 A = <Wert> : GOTO 20: REM <Programmname>
1010 - 19999	Eigentliches BASICODE-Programm
20000 - 24999	Subroutinen, welche unerlaubte Befehle verwenden (Tab. 4). Sie sollen möglichst vermieden, zumindest aber in REM-Zeilen genau erklärt werden. Ein mögliches Beispiel sind farbige Bilder, welche in BASICODE-3 noch nicht möglich sind.
25000 - 29999	Eventuell benötigte DATA-Zeilen.
30000 - 31999	REM-Zeilen, die z.B. eine kurze Beschreibung des Programms und Literaturhinweise enthalten.
32000 - 32767	REM-Zeilen für den Name und die Anschrift des Autors und andere formale Bemerkungen.

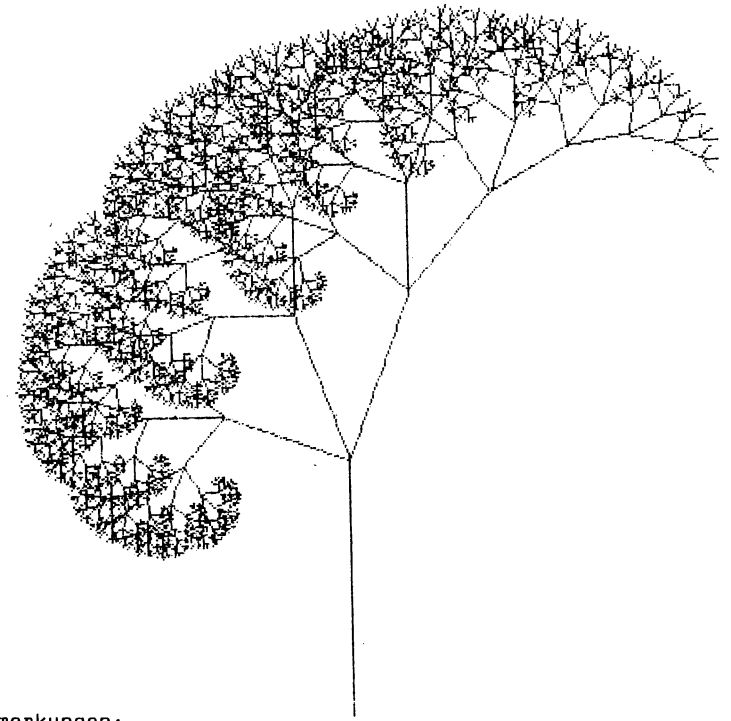
Der Zeilenabstand sollte im Programm möglichst in 10er Abständen gewählt werden.

Tabelle 5. In BASICODE verbotene Variablen

1. Alle Variablen, die mit dem Buchstaben O beginnen.
2. AS, AT, DI, EI, FN, GO, GR, IF, LN, SQ, ST, TI, TI$, TO, DI$, EI$, SQ$.
3. PI, es enthält aber auch nicht den Zahlenwert 3.14159.

Tabelle 6. Im Bascoder verwendete Variablen mit besonderer Bedeutung

A, CN, CT, FR, HG, HO, IN, IN$, NF, NF$, RV, SD, SP, SR, SR$, SV, VE, VG.



Bemerkungen:

Mit viel Liebe und Fleiß hat den Entwurf dieses Manuskriptes und auch die Programme Herr Jacques Haubrich(Niederlande) durchgesehen und viele wertvolle Vorschläge unterbreitet. Hierfür möchten wir ihm ganz herzlich danken. Dennoch können sich beim Editieren neue Fehler eingeschlichen haben. Deshalb gilt dieses Material nur als Arbeitsversion. Mit der geplanten Ausgabe eines weitaus ausführlicheren Handbuches sollen diese Fehler behoben und neue Erkenntnisse berücksichtigt werden. Wir bitten daher alle Hörer, uns umgehend Vorschläge, Mängel und Fehler mitzuteilen.

REM - das Computermagazin
Radio DDR
Nalepastraße
Berlin, 1160

Unsere Sendefrequenzen auf UKW

	Radio DDR II	DT 64 (MHz)
Frankfurt/O.	87,6	101,5
Putbus	88,6	91,5
Helpterberg	90,5	103,8
Marlow	91,0	100,8
Sonneberg	91,7	102,7
Dresden	92,2	102,4
Inselsberg	92,5	102,2
Schwerin	92,8	101,3
K.-M.-Stadt	92,8	100,0
Suhl	93,7	-
Leipzig	93,9	102,9
Brocken	94,6	101,4
Dequede	94,9	101,0
Löbau	98,2	91,8
Cottbus	98,6	103,2
Berlin	99,7	102,6
Hoyerswerda	100,4	-
Marlow	102,8	-
sowie auf MW:		
Burg	-	657 kHz
Neubrandenburg	-	657 kHz
Reichenbach	-	657 kHz

Dieses Heft entstand in Zusammenarbeit von Radio DDR II mit Prof. Dr. Horst Völz, der Direktion für Computerliteratur und Software beim Ministerium für Kultur und der Abteilung Öffentlichkeitsarbeit des Rundfunks der DDR. Ag 142/129/89