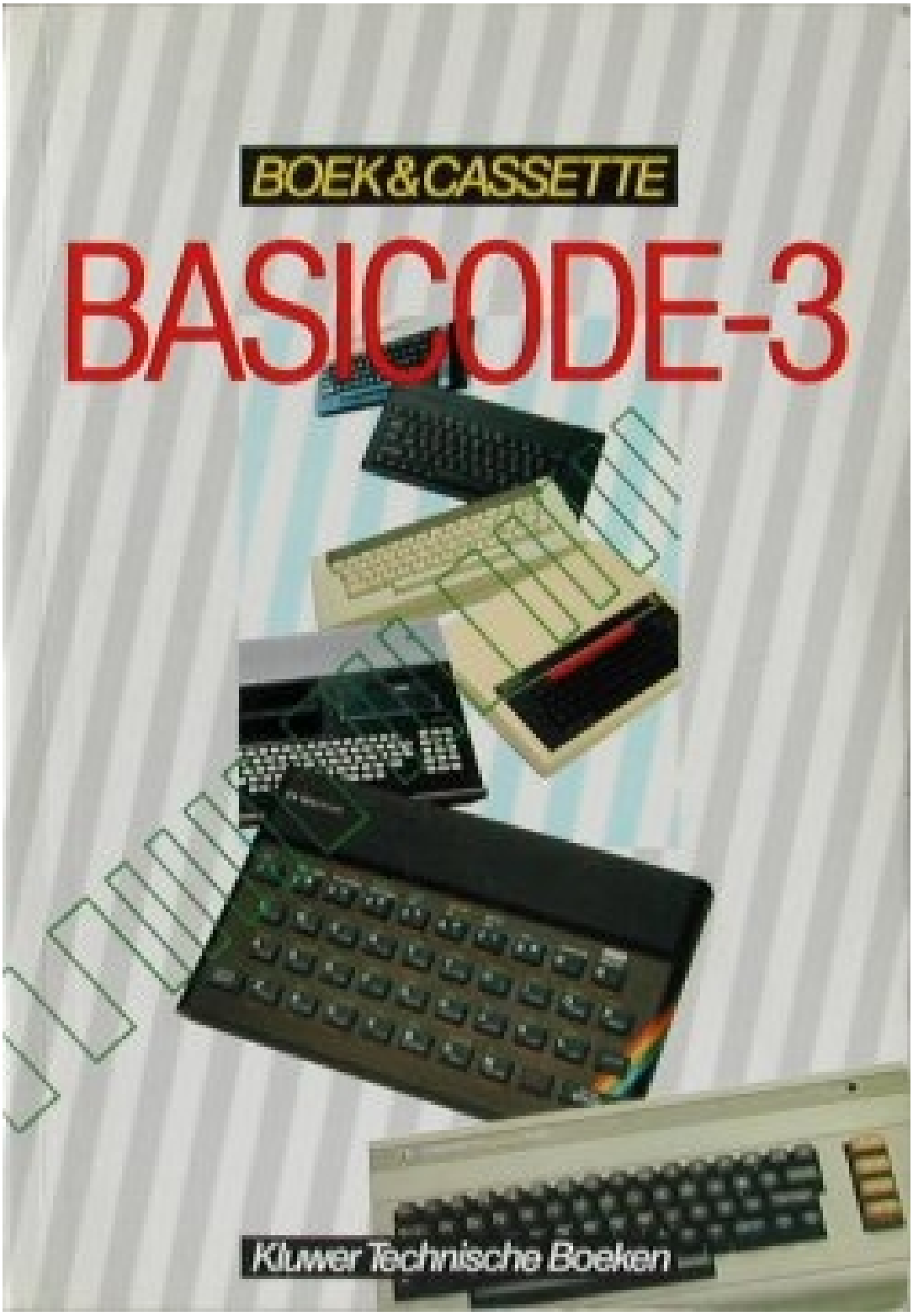


BUCH & KASSETTE

BASICODE-3

(Auszugsweise Grob-Übersetzung aus dem Niederländischen)

Kluwer Technische Boeken

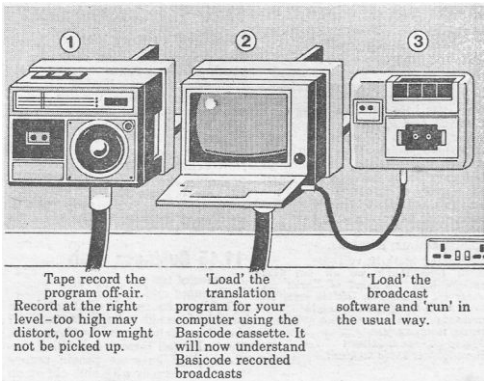
1 BASICODE, was ist das?

(…) BASICODE löst eine Reihe dieser Probleme für Sie. Dafür wurden drei Schritte getan.

1. In vielen Programmen ist es nicht erforderlich, die besonderen Möglichkeiten einer bestimmten Art Heim-Computer zu nutzen. BASICODE gibt eine Art 'größten gemeinsamen Nenner' für BASIC-Instruktionen. Ein Verzeichnis von Anweisungen, an das sich wohl jeder BASIC-Interpreter/Übersetzer heranwagen kann. Ein Programm, das nur dieses verwendet, kann mithin in ziemlich jedem Computer arbeiten.
2. Danach erweist sich noch, daß es eine Anzahl von Handlungen gibt, die man in einem Programm fast nicht entbehren kann. Ein Beispiel, das jeden ansprechen wird, ist so etwas Einfaches wie das Löschen des Bildschirmes. Das geht in fast jedem Computer anders vor sich. BASICODE gibt einen einfachen Umweg an, für Kenner: eine vereinbarte Subroutine, mit der die Handlung für alle Computer auf die gleiche Weise im Programm verarbeitet werden kann.
3. Das Wegschreiben auf eine Kassette und das Zurücklesen der Programme geschieht in ziemlich jedem Computer durch ein Stückchen Programm in Maschinensprache. Das Anschließen des Kassettenrecorders an den Mikroprozessor, das Herz des Computers, ist so einfach und billig wie nur möglich gehalten. Alle Schlaubergereien sitzen im Maschinensprachprogramm. Das sitzt immer schon im Computer, für die Kenner: im ROM. Hierin werden auch die Tönchen und Rasselgeräusche bestimmt, mit denen die BASIC-Programme auf die Kassette gesetzt werden. Ein anderes Programm für die Untersuchung des Kassettenanschlusses, und es wird ein anderes Geräusch aufgenommen.
   Für BASICODE ist erstmalig genau vereinbart, was für Tönchen da benutzt werden müssen. Danach wurde für jeden Computer ein neues Stückchen Maschinensprache gemacht, so daß mit diesen vereinbarten Tönen gelesen und geschrieben werden kann. Dieses Programm ist ein Bestandteil des BASICODE-Übersetzungsprogramms. Dieses ist unterschiedlich für jede Art Heim-Computer. Es muß eingelesen werden, ebenso wie ein gewöhnliches Programm dieses Computers. Doch danach ist es möglich, ein Programm in BASICODE von einer Kassette einzulesen. Das kann mithin aus einem ganz anderen Computer kommen, denn was da auf er Kassette steht, ist zuvor ganz genau vereinbart worden. So ist es meistens auch gleichzeitig möglich, ein eigenes BASIC-Programm nach diesem BASICODE-Verfahren auf die Kassette zu setzen.

2 Die Entwicklung von BASICODE

BASICODE ist eigentlich durch ein Rundfunkprogramm entstanden. Im Jahre 1979 hat NOS (Nederlands Omroep Stichting = Niederländische Rundfunkstiftung) in der



Radiosendung 'Hobbyscoop' damit begonnen, Computerprogramme auszustrahlen. Es hat als ein Versuch begonnen, doch schon bald ging Hans Janssen, der Betreuer dieses Programms für Hobbyisten, dazu über, allwöchentlich ein Computerprogramm auszustrahlen. Das war abwechselnd jedesmal für einen anderen der vier zu jener Zeit wirklich verbreiteten Computer: Apple II, Exidy, PET (Commodore) und TRS 80 Model-1 (tandy). Es wurde jede Woche ein von Hobbyisten eingesandtes Programm auf 'Studiotape' 'gesaved'. Dieses wurde am Ende des Funkprogramms in die Sendung eingefügt. Den Hörern wurde empfohlen, alles auf den Kassettenrecorder aufzunehmen und danach das Computerprogramm auf die übliche Weise in den eigenen Computer einzulesen. Die Sendungen wurden betreut durch das qualitativ ausgezeichnete FM-Sendernetz. Bei einem störungsfreien Empfang verlief das sehr vernünftig. Und so hörten die ahnungslosen Hörer jede Woche ein paar Minuten lang unverständliche und gräßliche Geräusche aus ihrem Rundfunkgerät kommem.

Hans Janssen war durch seine Sendungen mit einer Reihe sehr aktiver Computer-Hobbyisten in Kontakt gekommen. Von ihnen bekam er auch viele der Programme, die ausgestrahlt wurden. Die Probleme, die dieser Art des Rundfunksendens anhaften, waren natürlich allen bekannt. Es war Anfang 1981, als einer von ihnen, Klaas Robers, Ingenieur im Physikalischen Institut von Philips, mit einer neuen Idee kam. Dadurch sollte es möglich werden, die Computerprogramme so auszustrahlen, daß alle Typen diese einlesen können. Es war der Grundgedanke von BASICODE: ein genau vereinbarter Geräusch-Code, um Programme auf eine Kassette zu bringen, und ein dazugehöriges Hilfsprogramm je Computer-Typ. Durch eine geschickt ausgewählte Codierung würde ein Programm, das mit etwas Störung empfangen wird, nur geringfügig verstümmelt sein. Jeder Computer-Hobbyist in jenen Tagen hatte eine ausreichende Kenntnis des Programmierens, um diesen Verstümmelungen abhelfen zu können.

▲ De ploeg van (Hobby)Scoop voor de radio- en tv-toren in Hilversum. Zittend in het midden eindredacteur Hans G. Janssen, links van hem Ingrid Drissen die het programma de laatste twaalf jaar heeft gepresenteerd. Foto: NOS

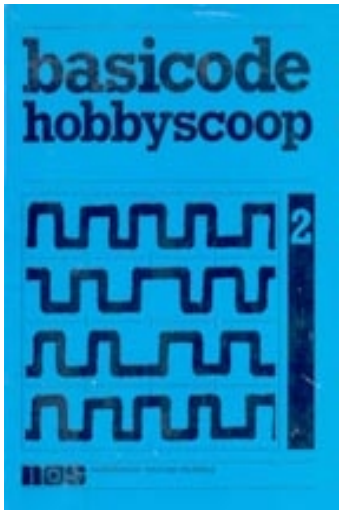
Am Abend des 14. Mai 1981, einem Donnerstag, wurde eine Anzahl aktiver Hobbyisten zusammengetrommelt, um die Idee zu erörtern. Jeder von ihnen kannte seinen Computer in- und auswendig. Das war nötig, denn niemand hatte ganz allein das Wissen, um für eine andere als seine eigene Maschine die ziemlich komplizierten Maschinensprachroutinen zu erstellen.

Am 11. August hatte NOS eine Pressekonferenz geplant, um BASICODE weltweit bekannt zu machen. Das war das erste Mal, daß man wieder zusammenkam, und jeder hatte seinen eigenen Computer mitgebracht. Und seht mal an, das Wunder geschah. Eine Kassette, geschrieben für eine Marke, wurde einer anderen gegeben und konnte so ohne weiteres eingelesen werden. Bei jedem hörte man das gleiche Rasselgeräusch, ungeachtet der Form, der Farbe und des Namens des Computers. Das wars, wofür gearbeitet worden war. Doch wenn man es geschehen sah und hörte, war es phantastischer als man es sich vorgestellt hatte. Durch eine gemeinsame Bemühung war die Sprachbarriere überwunden worden.

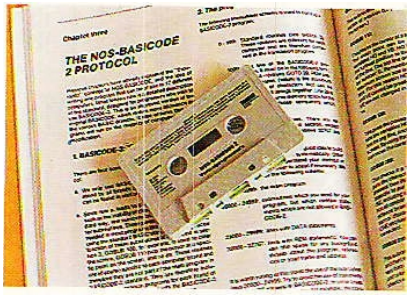
Ende September 1981 wurde BASICODE in der Presse vorgestellt und wurden die ersten Übersetzungsprogramme über den Rundfunk ausgestrahlt. Nach kurzer Zeit gab es über zehn Marken/Typen, für die in irgendeiner Form ein BASICODE-Übersetzungsprogramm erstellt worden war. Es wurden mitunter sehr schöpferisch gefertigte Programme in BASICODE ausgestrahlt. Es wurde inzwischen Zeit, einmal nachzusehen, was noch weiter zu tun war. In der Praxis arbeitete der verwendete Ton-Code problemlos. Die Sendungen ließen sich überall gut empfangen, und durch das Selbstrestaurierungsvermögen war eine kleine Störung kein Problem. Am schwierigsten war es wohl noch, die gesendeten/ausgestrahlten Programme der eigenen Marke anzupassen. Durch die Popularisierung des Heim-Computers war der Durchschnittsnutzer weniger gut mit dem Programmieren vertraut. Die Handreichungen für den Nutzer hätten eigentlich weiter gehen müssen. Gab es eine Möglichkeit, es so zu machen, daß die Programme ohne weiteres auf allen angeschlossenen Computern laufen konnten?

Am Abend des 30. September 1982 kam - wieder im Gebäude Santbergen in Hilversum - die inzwischen erweiterte und ein wenig veränderte BASICODE-Gruppe zusammen, um neue Pläne aufzustellen. Klaas Robers hatte zusammen mit Jochen Herrmann – damals noch Student an der Technischen Hochschule Eindhoven – Pläne ausgearbeitet, um eine Programmstruktur mit festen Subroutinen zu erreichen. Im ersten Jahr BASICODE hatten sich gute Erkenntnisse in Bezug auf die größten Engpässe ergeben. Bestimmte Handlungen mußten sich in den Programmen ausführen lassen. Außerdem hatte sich herausgestellt, daß einzelne Instruktionen in BASIC von den verschiedenen Computern etwas anders behandelt wurden, wie z. B. das Plazieren von Zahlen auf dem Schirm.

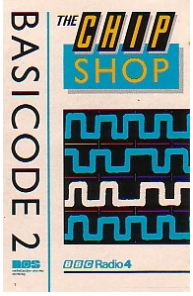
Die vorgeschlagene Methode mit genau definierten Subroutinen konnte die bestehenden Probleme lösen. Gemeinsam wurden die Zeilenzahlen für die Standard-Subroutinen ausgewählt. Auch schien es vernünftig, der Zeilennumerierung innerhalb des Programms etwas mehr Beständigkeit zu verleihen. Insbesondere vonseiten des Schul-/Bildungswesens wurde darum gebeten, so daß man etwas leichter den Weg in ein neu empfangenes Computerprogramm würde finden können. Man entschied sich dafür, fortan von BASICODE-2 zu sprechen, wenn die Subroutinen für das Laufen des Programms erforderlich sind. Die alte Version/Fassung - ohne diese Erweiterung - sollte unter dem Namen BASICODE-1 weiterbestehen. Dadurch blieb es möglich, Programme unverändert von einer Marke auf die andere überzuspielen.



Unterdessen wurde auch an einem neuen BASICODE-Buch gearbeitet. Anstelle einer Neuauflage des ausverkauften BASICODE-Buches erschien im Juli 1983 sofort ein BASICODE-2-Buch, natürlich wieder mit einer Kassette. Das Protokoll stand vollständig in diesem Buch. Ungefähr gleichzeitig mit dem Erscheinen des neuen Buches wurde BASICODE-2 auf dem Wege über Hobbyscoop eingeführt. Das Interesse war sehr groß. Zum ersten Mal war es wirklich möglich, Programme zu fertigen, die ohne weiteres auf unterschiedlichen Computer-Marken/Typen liefen. Das Unterrichts- und Bildungswesen war hell begeistert. Die Einschränkungen, die BASICODE-2 auferlegte, wogen viel weniger schwer als der Vorteil der Austauschbarkeit des Programms.



Auch im Ausland begann das Interesse zuzunehmen. Angeregt durch NOS begannen WDR und BBC mit Sendungen von BASICODE. In Deutschland ging das auf den Tonkanal des Fernsehens. Die BBC begann auf der Langwelle mit Sendungen mitten in der Nacht auf Radio-4. Später zog BASICODE in das englische Radio-1 FM-Sendernetz um. In der Praxis erwies sich die Übertragung über Mittelwelle und Langwelle am Abend und in der Nacht als unbrauchbar für die schnellen BASICODE-Signale. Auch die Kurzwelle



läßt sich dafür nicht verwenden. Jonathan nahm Proben über den 'Weredomroep' vor, die dieses bestätigten. Rundfunk-Sende-Amateure bemerkten, daß die VHF- und UHF-Bänder für wechselseitigen Austausch digitaler Information über BASICODE zugeschnitten waren.

Am 11. Mai 1985 kam die BASICODE-Gruppe wieder in einem Sälchen in einem der Philips-Komplexe in Eindhoven zusammen. Jedem war folgendes klar: BASICODE-3 mußte kommen. Eine Anzahl schon lange gewünschter Dinge mußte durch diese Erweiterung möglich gemacht werden. Jochen Herrmann und Jacques Haubrich, Mathematiklehrer in Eindhoven, hatten einen umfangreichen Vorschlag gemacht. Musik, graphische Möglichkeiten, inverses Video, Bestände: es war ganz schön was hinzugekommen. Es würde nicht leicht sein, aber wir waren im Laufe der Zeit auch viel weiser geworden.

Seit Mitte 1986 wurden wöchentlich von TROS Programme in BASICODE-3 ausgestrahlt. Das geschieht in Zusammenarbeit mit der 'Stichtung (Stiftung) BASICODE'. Bis Juli 1987 geschieht das am Mittwoch von 17.41 bis 17.46 auf Radio 5. Jeder/jede kann seine/ihre selbstgemachten Programme dem TROS-BASICODE-3, Postbus (Postschließfach) 450, 1200 AL HILVERSUM, anbieten.

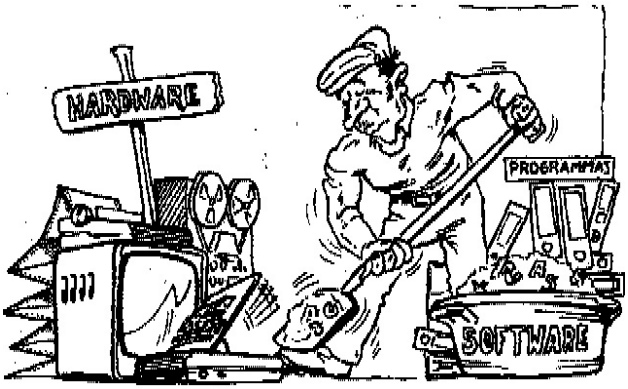
(...)

4 Das BASICODE-3-Protokoll

4.1 Einleitung

Dieses Kapitel enthält das offizielle BASICODE-3- Protokoll. Dieses Protokoll enthält alle Regeln, an die man sich halten muß. Ein Programm, das diese Regeln nicht erfüllt, ist somit kein gutes BASICODE-3-Programm. Wer ein Programm veröffentlicht, das dieses Protokoll nicht erfüllt, veröffentlicht somit eigentlich kein BASICODE-3-Programm. Wer es dann doch BASICODE-3 nennt, macht einen Witz. Das darf nicht sein. Wenn jemand so etwas wiederholt und mit erkennbarer Absicht oder aus reiner Schludrigkeit tut, so riskiert er dieses und jenes. BASICODE ist ein gutes System, und niemanden ist damit genützt, daß andere mit schlechten oder unvollständigen Programmen den guten Namen von BASICODE verderben. Eventuelle Beschwerden in dieser Hinsicht bitten wir dem Verleger dieses Buches mitzuteilen.

4.2 Allgemeine Vereinbarungen



Wie wir bereits bemerkten, gibt es Unterschiede in den verschiedenen Ausführungen der BASIC-Programmiersprache. Um Programme nicht nur austauschen zu können, sondern auch problemlos bei einem anderen Computer-Typ arbeiten zu lassen, wurden im BASICODE-3-Protokoll mehrere Regeln festgelegt, die diese Programme erfüllen müssen:

a) Wir verwenden nur die BASIC-Befehle, die alle Computer kennen und auf die gleiche Weise ausführen. Die Befehle werden in Kapitel 4.6 eingehend beschrieben.

b) Wir reservieren die Zeilennummern bis Zeile 1000 für Subroutinen, die bestimmte wichtige Aufgaben erfüllen, die aufgrund von Regel a nicht programmierbar sind. Die Arbeitsweise dieser Subroutinen wird in Kapitel 4.3 festgelegt. So löscht die mit GOSUB 100 aufgerufene Subroutine den Bildschirm, setzt die Subroutine auf Zeile 110 den Cursor auf eine bestimmte Stelle am Schirm usw. Da diese Routinen für jeden Computer-Typ anders sind, werden sie nicht zusammen mit dem BASICODE-Programm auf die Kassette geschrieben. Es ist die Aufgabe des Übersetzungsprogramms, diese Routinen hinzuzufügen und so das Programm zu vervollständigen.

c) Die Abmessungen des Bildschirmes sind so ziemlich bei jeder Computer-Marke und bei jedem Computer-Typ unterschiedlich. Die Zahl der Zeilen zwischen 16 (TRS-80) und 32 oder mehr, und die Zahl der Zeichen je Zeile beträgt in einem VIC-20 nur 22 und in vielen anderen Computern 40, 80 oder mitunter noch mehr. So unterscheiden sich auch die graphischen Möglichkeiten stark voneinander, von grober Blockgraphik mit nur 80 Blöckchen je Zeile bis zu einer wirklich hohen Auflösung mit 300 oder mehr Bildpunkten je Zeile. Ein gutes BASICODE-Programm berücksichtigt derartige Unterschiede. Beim Start eines BASICODE-3-Programms steht daher in der Variablen HO, wieviel Zeichen je Zeile möglich sind, und in der Variablen VE, wieviel Zeilen auf den Schirm dürfen (eigentlich steht in beiden Variablen 1 weniger als soeben beschrieben; wir kommen darauf noch zurück). Ebenso wird in den Variablen HG und VG angegeben, wieviel graphische Punkte dort horizontal bzw. vertikal in dem jeweiligen Computer möglich sind. In Bezug auf Einzelangaben hierüber wird auf Anhang 1 verwiesen. Als vernünftigen Durchschnitt darf man einen Bildschirm von 24 Zeilen mit jeweils 40 Zeichen und eine grafische Auflösung von 180 Linien mit jeweils 250 Bildpunkten betrachten. Ein wirklich gutes BASICODE-Programm sorgt für eine Schirmeinteilung, die übereinstimmt mit den Möglichkeiten des Computers, in

dem es arbeitet. Man vermeide so viel wie nur möglich das Printen/Drucken von Zeichen auf die letzte Position einer Schirmzeile. Nicht alle Computer reagieren auf die gleiche Weise.

d) Jede Programmzeile darf einschließlich der Zeilennummer und der Zwischenräume höchstens 60 Zeichen lang sein.

e) Programme, die sich eines Printers bedienen, müssen mit einem Printer mit einer Zeilenlänge von 80 rechnen. Die Nutzung des Printers muß mithin so erfolgen, daß nie mehr als 80 Zeichen auf die gleiche Zeile gedruckt werden.

f) Der Aufbau des Programms.

Wenn das BASICODE-Programm erst einmal vollständig mit Subroutinen im Speicher steht, ist der Aufbau des Programms folgendermaßen:

0 - 999	Die Standardroutinen (siehe Kapitel 4.3). Diese Routinen sind je Computer unterschiedlich und werden durch das Übersetzungsprogramm bereitgestellt.
1000	Die erste Zeile des BASICODE-3-Programms. Diese hat obligatorisch die folgende Form: 1000 A=<wert>:GOTO 20:REM Programmname <wert> ist die maximale Anzahl von Schriftzeichen, die für alle 'strings' zusammen benötigt werden. Durch den Sprung nach Zeile 20 reservieren die Computer, für die das erforderlich ist, Speicherplatz für die 'strings'.
1010 - 19999	Das Hauptprogramm
20000 - 24999	Subroutinen, die Sie für das Programm benötigen, obwohl in ihnen Befehle stehen, die in BASICODE-3 nicht erlaubt sind und die nicht durch die Subroutinen des Übersetzungsprogramms umgesetzt werden können. Diese Befehle müssen in REM-Zeilen erläutert werden, damit andere sie an ihre Computer anpassen können.
25000 - 29999	Eventuell benötigte DATA-Zeilen
30000 - 31999	Ausschließlich REM-Zeilen. Hier kann eine kurze Beschreibung des Programms stehen (falls nötig) und eventuelle Literaturhinweise.
32000 - 32767	In REM-Zeilen der Name und die Anschrift des Programmautors.

Das Programm beginnt und endet immer in dieser Reihenfolge. Zeile 10 ist nur dazu bestimmt, über die Standard-Subroutinen hinwegzuspringen. Zeile 1000 gibt an, wieviel 'stringbytes' das Programm benötigt. Zeile 20 reserviert den benötigten Raum in Computern, wo das erforderlich ist, bringt den Computer

in den BASICODE-Modus, setzt den Schirm in die richtige Farbe und regelt noch das eine und das andere, das geregelt werden muß. Dannach wird auf Zeile 1010 übergesprungen, für das eigentliche BASICODE-Programm. Wenn das abgelaufen ist, springt es auf Zeile 950, wo der Computer wieder in den normalen Betriebsstand, in die normale Farbe und dergleichen zurückgebracht wird.

Über die Subroutinen von 20000-24999 bemerken wir noch das folgende: Versuchen Sie, so weit wie nur möglich zu vermeiden, daß Sie Befehle verwenden, die in BASICODE-3 nicht zugelassen sind. In einigen Fällen ist das jedoch nicht zu vermeiden (beispielsweise wenn Sie durchaus auf diese oder jene Weise mit Farbe arbeiten wollen). In diesem Falle setzen Sie diese Routinen auf die Zeilen 20000-24999. Es ist dann von großer Bedeutung, das Sie mit REM-Zeilen gut angeben, was in diesen Routinen genau geschehen soll.

Es ist des weiteren eine gute Gewohnheit, die Zeilenzahlen mit Abständen von 10 auflaufen zu lassen, so daß später noch Zeilen dazwischen eingefügt werden können.

4.3 Die BASICODE-3-Standardroutinen

In diesem Kapitel werden die Standard-Subroutinen beschrieben, so wie diese auf den Zeilen mit Zahlen unter 1000 zu finden sind. Diese Subroutinen sind für je-den Computer gesondert geschrieben, so daß sie für jedes BASIC anders aussehen. Sie tun jedoch in allen Computern genau das gleiche!

GOTO 20 (Programm-Anlauf)

Dieser Befehl (also eigentlich noch keine Subroutine!) darf und muß allein in Zeile 1000 stehen. Hiermit wird der Computer in den BASICODE-Zustand gebracht, wird der Schirm gelöscht, wird der nötige Speicherraum für 'strings' reserviert, werden alle Variablen gelöscht, werden die Variablen HO und VE mit den Maßen des Textschirmes versehen und werden in HG und VG die Maße des graphischen Schirmes gesetzt. Danach wird auf Zeile 1010 im Programm übergesprungen. Zeile 1010 ist eigentlich die erste Zeile des Programms. Die Werte, die in HO und VE gesetzt werden, sind beide um 1 niedriger als die Zahl der Zeichen je Zeile bzw. die Zahl der Textzeilen, die möglich ist. In einem Computer mit 40 Zeichen je Zeile wird somit in HO der Wert 39 stehen. In HG und VG wird angegeben, wieviel Bildpunkte dort im graphischen Betrieb in horizontaler und vertikaler Richtung gezeigt werden können. Die Be deutung dessen wird bei der Besprechung der graphischen Subroutinen klar werden.

GOSUB 100 (Textbetrieb und Schirm löschen)

Diese Subroutine schaltet (sofern noch nötig) um nach Textbetrieb, löscht den Schirm und plaziert den Cursor auf die Position 0,0 (das ist links oben auf dem Schirm). Es wird keine Variable verändert, also auch nicht HO und/oder VE.

GOSUB 110 (Cursor nach HO,VE)

Die Subroutine setzt den Cursor in der Position HO, VE auf den Schirm. HO ist die Position auf der Zeile; dabei stimmt der Wert 0 mit der am meisten links liegenden Position überein. VE ist die Zahl der Schirmzeile; die oberste Schirmzeile ist Zeile 0. In vielen Computern ist der Schirm 40 Zeichen breit und 24 Zeilen hoch. In diesen Fällen darf HO nicht größer sein als 39 und VE nicht größer als 23. Die Variablen HO und VE verändern sich wertmäßig nicht durch Aufruf dieser Subroutinen. Ein schlaues Programm bedient sich bei der Bestimmung der Schirmpositionen der Werte, die beim Start (nach GOTO 20) in HO und VE gesetzt werden.

GOSUB 120 (Cursor-Position in HO und VE)

Man ermittlee die Position, in der der Cursor auf dem Schirm steht, und setze diese in die Variablen HO und VE. Wenn somit nach dem Ablauf HO=0 ist, so steht der Cursor auf der linkesten Position auf einer Zeile, und wenn VE=0 ist, so steht der Cursor in der obersten Zeile des Schirms. Zusammen mit der vorigen Subroutine können Sie beispielsweise den Cursor eine Zeile oder mehrere Zeilen höher oder tiefer setzen:

```
2000 GOSUB 120 :REM FRAGE CURSORPOSITION AB
2010 HO=0:GOSUB 110:REM CURSOR AN ANFANG SETZEN
2020 PRINT"                    ":REM ZEILENANFANG LÖSCHEN
2030 GOSUB 110 :REM POSITION AKTUALISIEREN
2040 PRINT"Das steht links“
2050 VE=VE+1:GOSUB 110 :REM CURSOR POSITIONIEREN
2060 PRINT"und das steht darunter“
```

GOSUB 150 (man drucke auffallend)

Man drucke den Inhalt von 'string' SRS auf eine auffallende Weise auf den Schirm. In vielen Computern wird das dadurch verwirklicht, dass man den Text von SR$ im 'reverse-field' wiedergibt. Vor und nach den Zeichen von SR$ werden auf dem Schirm drei Zwischenräume gedruckt; einige davon können ebenfalls auf eine auffällige Weise wiedergegeben werden. Insgesamt werden somit sechs Zeichen mehr gedruckt als es der Länge von SR$ entspricht. Das Ganze darf nicht auf die folgende Schirmzeile 'überlaufen'. Ein Beispiel:

```
1010 SB=HO :REM SCHIRMBREITE SICHERN
.
.
2090 HO=INT(SB/2-10):IF HO<0 THEN HO=0
2100 VE=0:GOSUB 110
2200 SR$="KOPF DES PROGRAMMS":GOSUB 150
```

Nach Zeile 2200 wird der Text KOPF DES PROGRAMMS auf eine auffällige Weise in der Mitte der obersten Schirmzeile zu lesen sein. Warnung: Man benutze diese Subroutine mit Maßen! Wenn a l l e s auffällig ist, fällt gerade gar nichts mehr auf !

GOSUB 200 (man frage die Tastatur ab)

Man sehe nach, ob eine Taste gedrückt ist. Wenn ja, so setze man den 'string'-Wert dieser Taste in die Variable IN$. Wenn nicht, setze man in IN$ einen leeren 'string'. Gleichzeitig wird in der Variablen IN der 'wirkliche' ASCII-Wert der (eventuell) gedrückten Taste angegeben. Damit ist gemeint: eine Zahl entsprechend der Tabelle in Anhang 4, das heißt somit in normalen Fällen mindestens 32 und höchstens 95. Ungeachtet dessen, ob ein Klein- oder ein Großbuchstabe gedrückt ist, beispielsweise A oder a, wird in beiden Fällen der Code 65 abgegeben. Ziffer-/Zahlentasten geben Codes von 48 bis einschließlich 57. Wenn keine Taste gedrückt ist, wird in IN der Wert 0 stehen. Grundsätzlich können alle Tastenanschläge in IN$ stehen, also auch Kontrollzeichen wie RETURN oder ein Druck auf eine andere Steuerungstaste. Man achte jedoch auf folgendes: Die Kontrollzeichen haben bestimmt nicht bei allen Computern den gleichen Zahlenwert oder 'string'-Wert. Man gebrauche diese also lieber nicht in einem Programm. Ausnahmen sind dabei die folgenden:

Die RETURN- (oder ENTER-, NEW LINE- usw.) Tasten haben bei allen Computern den Code 13; wenn also diese Taste gedrückt ist, wird in IN=13 und IN$=CHR(13). Desweiteren wird beim Berühren bestimmter Steuerungstasten garantiert der folgende Code in IN (als Zahl) abgegeben (somit auch, wenn der Computer von Natur aus bei dieser Taste einen anderen Code abgeben müßte):

Cursor links	:	28 (dezimal)
Cursor rechts	:	29
Cursor nach unten	:	30
Cursor nach oben	:	31
Löschen/Delete	:	127

Nachbemerkung 1: Nach Ablauf dieser Routine kann sich in Abhängigkeit vom Computer-Typ ASC(IN$) von dem Wert in IN unterscheiden, und ebenso kann sich CHR$(IN) von IN$ unterscheiden. Dieser mögliche Unterschied kann grundsätzlich bei allen Zeichen auftreten.

Nachbemerkung 2: Wenn eine Cursor- oder Delete-Taste gedrückt war, ist der Wert in IN$ in Wirklichkeit unbestimmt. Testen auf IN$ oder ein Befehl als PRINT IN$ sind dann nicht sinnvoll.

Ein Beispiel:

```
1010  VT=VE:HT=HO
|
|
3000 GOSUB 110 :REM CURSOR POSITIONIEREN
3010 GOSUB 200 :REM TASTATUR ABFRAGEN
3020 IF IN<>127 THEN 3050
3030 HO=HO-1:GOSUB 110
3040 PRINT " ";:GOTO 3000
3050 IF IN=13 THEN HO=0:VE=VE+1
3060 IF IN=28 THEN HO=HO-1
3070 IF IN=29 THEN HO=HO+1
3080 IF IN=30 THEN VE=VE+1
3090 IF IN=31 THEN VE=VE-1
3095 IF HO<0 THEN HO=0:GOSUB 250
3100 IF IN>31 THEN PRINT IN$;:HO=HO+1
3110 IF HO>HT THEN HO=0:VE=VE+1
3115 IF VE<0 THEN VE=0
3120 IF VE>VT THEN VE=VT:GOSUB 250:REM PIEP
3130 GOTO 3000
```

GOSUB 210 (man warte auf Tastendruck)

Man warte, bis eine Taste gedrückt wird und setze den 'string'-Wert dieser Taste in die Variable IN$ und den Code in die Variable IN. Der Unterschied zur vorigen Routine besteht darin, daß diese Subroutine wartet, bis eine Taste eingedrückt wird, während die vorige eben nur danach schaute, ob in diesem Augenblick zufällig eine Taste gedrückt wurde.

Auch bei dieser Subroutine wird auf die gleiche Weise an die Variable IN der wirkliche ASCII-Wert abgegeben. Siehe dafür die oben bei Subroutine 200 gegebene nähere Erläuterung.

GOSUB 220 (man lese vom Textschirm)

Diese Subroutine liefert in IN den Code des Zeichens, das in der Position HO,VE auf dem Schirm sichtbar ist. Wenn HO,VE sich außerhalb des Schirms befinden, liefert die Subroutine als Ergebnis IN=0. Im anderen Fall erfüllt der Wert in IN die bei der Subroutine 200 genannte Regel, daß der Wert in IN immer mindestens 32 beträgt und höchstens 95 (vorbehaltlich einer gedrückten Steuerungstatste). Wenn also auf dem

Schirm ein Kleinbuchstabe a sichtbar ist, dann wird in IN der Wert 65 (von 'A') geliefert. Der Wert von IN$ wird durch diese Subroutine nicht geändert.

GOSUB 250 (Aufmerksamkeitssignal)

Diese Subroutine bringt einen kleinen Pfiff hervor, um den Nutzer ein Aufmerksamkeitssignal zu geben. Die Tonhöhe und Dauer sind nicht genau festgelegt, so daß die Routine sich nicht für das Musikmachen eignet. Man gebrauche diese Subroutine vernünftig. Viele Nutzer des Programms erbosen sich durch zuvieles Pfeifen eher als das sie besonders aufmerksam werden.

GOSUB 260 (man gebe eine Random-Zahl)

Diese Subroutine gibt eine zufällige Zahl in der Variablen RV. Diese Zahl ist immer kleiner als 1 und beträgt mindestens 0. So etwas ist nützlich in statistischen Programmen und bei Spielen.

GOSUB 270 (man melde freien Raum)

Diese Subroutine säubert den Variablenraum und bestimmt, wieviel Speicherraum noch frei ist (die Variablen werden nicht gelöscht!). Die Anzahl noch freier 'bytes' wird in der Variablen FR aufgestellt. Programme, die viel Speicherraum benötigen, können auf dem Wege über diese Subroutine erfahren, ob der benötigte Platz zur Verfügung steht. Beispielsweise:

```
4000 GOSUB 270
4010 l=INT(SQR((FR-500)/6))
4020 DIM MA(L,L)
```

Bei der Annahme, daß für jede Zahl sechs Speicherplätze benötigt werden, dimensioniert man auf diese Weise eine möglichst große quadratische Matrix, wobei für andere Zwecke noch mindestens 500 'bytes' Gedächtnisraum reserviert bleiben.

GOSUB 280 (STOP-Taste schalten)

Wenn beim Anrufen dieser Subroutine der Wert in FR gleich 1 ist, wird die Stoptaste ausgeschaltet. Wenn FR=0 ist, so wird diese Taste wieder in den normalen Betriebszustand gebracht. Das ist insbesondere von Bedeutung für Programme, die im Unterricht verwendet werden und bei denen nicht beabsichtigt ist, daß ein Schüler das Programm unterbrechen kann.

GOSUB 300 (Umsetzung in 'string')

Man mache einen 'string' SR$, der den Wert der Variablen SR angibt. Der 'string' enthält keine Zwischenräume am Anfang oder am Ende der Zahl, und zwar im Ge-

gensatz zur BASIC-Funktion STR$(), die das bei einigen BASIC-Dialekten eben gerade tut. Die Verwendung der Funktion STR$() ist in BASICODE-3 nicht gestattet.

Ein Beispiel (man achte auf die Leerzeichen in Zeile 5010):

```
5000 SR=VR: GOSUB 300
5010 PRINT"Ihr habt jetzt ";SR$;" Fragen gehabt"
```

GOSUB 310 (Zahl formatieren)

Diese Subroutine macht in SR$ einen 'string', der bestimmt wird durch die Variablen CT, CN und SR. SR$ wird ebenso wie in der vorigen Routine den Wert von SR angeben. Die Gesamtlänge von SR$ beträgt nun genau CT Zeichen, davon CN Zeichen nach dem Dezimalpunkt, besteht SR$ aus CT Sternen. Falls erforderlich, wird SR korrekt abgerundet. Die Variablen CT, CN und SR werden nicht verändert durch Aufruf der Routine. Einige Beispiele:

CT=7:CN=3:SR=2/3:GOSUB 310	ergibt SR$="0.667"
CT=9:CN=5:SR=-1.1E-3:GOSUB 310	ergibt SR$="- 0.00110"
CT=8:CN=2:SR=-1.1E-3:GOSUB 310	ergibt SR$="0.00"
CT=3:CN=0:SR=23.6:GOSUB 310	ergibt SR$=" 24"
CT=3:CN=1:SR=23.6:GOSUB 310	ergibt SR$="***"

GOSUB 330 (Großbuchstaben machen)

Diese Subroutine verwendet alle eventuell in SR$ vorhandenen Buchstaben in Großbuchstaben. Nach Ablauf kann beispielsweise MID$ (SR$, 3, 1) wohl 'A', aber nie 'a' sein. In Programmen, wo Daten in alphabetischer Reihenfolge sortiert werden, kann das sehr nützlich sein.

GOSUB 350 (Text nach Drucker)

Man drucke SR$ auf dem Drucker ab, schließe jedoch die Zeile noch nicht ab. Sie können somit noch mehr mittels dieser Routine auf die gleiche Zeile drucken. Auf eine andere Weise etwas zum Drucker zu schicken, ist in BASICODE-3 nicht gestattet. Man beachte: Lange nicht jeder hat einen Drucker zur Verfügung! Man bitte daher im Programm den Nutzer, eine Wahl zu treffen zwischen dem Abdrucken auf dem Drukker und dem Abdrucken auf dem Schirm.

GOSUB 360 (Zeilenschaltung für den Drucker)

Man schließe die Zeile auf dem Drucker ab und beginne eine neue Druckzeile. Wenn auf einem neuen Blatt Papier angefangen werden muß, wird es nötig sein, daß das Programm selbst behält, wieviele Zeilen schon gedruckt sind, und aufgrund dessen die Subroutine in einer richtigen Zahl von Malen aufruft.

GOSUB 400 (Musik)

Diese Subroutiene produziert einen Ton. Der gewünschte Ton wird in drei Variablen angegeben:

SP (Sound Pitch) für die Tonhöhe, 0 < = SP < = 127
SD (Sound Duration) für die Dauer, 1 < = SD < = 255
SV (Sound Volume) für die Stärke, 0 < = SV < = 15

Die Subroutine wird erst beendet, wenn der Ton abgelaufen ist.

Ein paar Beispiele:

SV = 0	absolute Stille
SV = 7	durchschnittliche, normale Lautstärke
SV = 15	maximale Lautstärke

SD	die Zeitdauer in Einheiten von 100 Millisekunden. SD = 10 stimmt somit überein mit einer Dauer von 1 Sekunde.

SP = 1	der niedrigste Ton, ein abscheuliches Gebrumm
SP = 60	das 'zentrale C'
SP = 69	das 'Standard'-A, meistens etwa 440 Hz
SP = 127	der höchste Ton, für viele Menschen unhörbar

Für jeden halben Ton höher oder tiefer ist SP 1 höher bzw. tiefer. Jede Oktave umfaßt 12 Töne. Es ist nicht sinnvoll, alle möglichen Werte von SP zu verwenden. Erstens weil die sehr niedrigen und sehr hohen Werte kein angenehmes Geräusch erbringen, und zweitens weil nicht jeder Computer in der Lage ist, den vollen Bereich von 127 Tönen hören zu lassen. In Bezug auf dieses Letztgenannte sei auch auf Anhang 1 verwiesen.

GOSUB 450 (abbrechbare Warteroutine)

Beim Aufrufen dieser Subroutine muß in SD eine Zahl stehen, die eine Wartezeit in Einheiten von 100 Millisekunden angibt. SD = 10 stimmt somit genau mit einer Sekunde überein. Die Subroutine wird automatisch verlassen, wenn die angegebene Zeit um ist. In diesem Fall ist beim Verlassen der Routine SD = 0, IN$= "" und IN=0.

Wenn jedoch während der Wartezeit eine Taste gedrückt wird, wird die Routine sofort beendet. In SD steht dann noch die übrige Zeit in Einheiten von 100 Millisekunden, in IN$ die gedrückte Taste und in IN der Code dieser Taste. Man beachte für die Codes in IN die Bemerkungen bei Subroutine 200.

Bemerkung: Für eine nicht unterbrechbare Warteroutine können Sie GOSUB 400 mit SV=0 benutzen.

Ein Beispiel:

```
6000 PRINT"Drücken Sie eine Taste..."
6010 SD=50:GOSUB 450
6020 IF SD>0 THEN 6050
6030 PRINT"Nach 5 Sekunden haben Sie noch immer nichts gedrückt"
6040 GOTO 6080
6050 SR=(50-SD)/10:CN=1:CT=4:GOSUB 310
6060 PRINT"Es dauerte ";SR$;" Sekunden"
6070 PRINT"bis Sie das ";IN$;" gedrückt hatten"
6080 RETURN
```

GOSUB 500 (Bestand öffnen)

Beim Aufrufen dieser Subroutine muß in NF$ der Name des zu öffnenden Bestandes stehen (maximal 7 Zeichen) und in NF ein Code:
NF = 0 : Öffnen für eine Eingabe von der BASICODE-Kassette aus
NF = 1 : Öffnen für die Ausgabe zur BASICODE-Kassette
NF = 2 : Öffnen für eine Eingabe aus dem eigenen Hintergrundspeicher
NF = 3 : Öffnen für die Ausgabe an den eigene Hintergrundspeicher
NF = 4 : Öffnen für die Eingabe von der Diskette
NF = 5 : Öffnen für die Ausgabe auf die Diskette
NF = 6 : Öffnen für die Eingabe von der Diskette
NF = 7 : Öffnen für die Ausgabe auf die Diskette

Kurzum:

gerade Zahlen	= Eingabe
ungerade Zahlen	= Ausgabe
Nummer 0 und 1	: Kassette in BASICODE-Format (siehe weiter unten); NF$ nicht verwendet;
Nummer 2 und 3	: Diskette oder Kassette auf die Art, gemäß der es der Computer normalerweise selbst tut;
Nummer 4 und 5	: ein zweiter Bestand, der nur möglich ist nach der Diskette;
Nummer 6 und 7	: ein dritter Bestand, auch wieder nur auf Diskette.

Kassetten, die mit Beständen in BASICODE-Format (NF=1) beschrieben sind, können allen BASICODE-Computern eingelesen (NF=0) und verarbeitet werden. Auf diese Weise können Computer verschiedener Typen somit Daten austauschen.

Während des Arbeitens dieser Bestandsroutine und der folgenden Bestandsroutinen können auf der untersten Schirmzeile bestimmte Meldungen erscheinen (beispielsweise: 'Stelle den Recorder auf Wiedergabe'). Diese Meldungen müssen natürlich

vom Nutzer befolgt werden. Während des Lesens und Schreibens von der Kassette wird auf dem Schirm 'unterste Zeile‘) einige Aktivität sichtbar sein. Diese dient als Kontrolle über die Wirkung der Routinen.

Nach Ablauf der Routine werden diese Meldungen wieder gelöscht. Dabei kann somit die unterste Schirmzeile gelöscht werden. Diese Zeile wird somit durch ein gutes BASICODE-3-Programm nicht genutzt werden. Bemerkung: Zur Diskette können somit maximal drei Bestände zugleich geöffnet werden. Es gibt keine Garantie (im Gegenteil), daß jeder BASICODE-Computer dazu imstande ist.

GOSUB 540 (Eingabe aus Bestand)

Bei Aufruf dieser Subroutine steht in NF der Bestandscode liefert in IN$ den ersten 'string' aus dem zuvor geöffneten Bestand und zugleich in IN den Wert 0. Wenn jedoch in IN$ der letzte 'string' aus dem Bestand geliefert wird, wird in IN der Wert 1 abgegeben. Wenn die Subroutine aufgerufen wird, wenn der letzte 'string' bereits abgeliefert ist, wird IN$ = "" und IN = 1. Wenn die Subroutine einen Fehler entdeckt, beispielsweise einen Lesefehler von der Kassette her, so wird IN = -1 gesetzt. Wenn keine Antwort in IN$ möglich ist, beispielsweise wenn der Bestand nicht geöffnet ist, wird IN$ = "" und IN = -1 abgeliefert.

GOSUB 560 (Ausgabe zum Bestand)

Der Inhalt von SR$ wird in den Bestand mit Codenummer NF geschrieben. In der Variablen IN werden eventuelle Fehler gemeldet: IN=0, wenn alles gut ist, IN = -1, wenn die erbetene Ausgabe unmöglich ist.

GOSUB 580 (Bestand schließen)

Der Bestand mit Code NF wird geschlossen. In der Variablen IN werden eventuelle Fehler gemeldet: IN = 0, wenn alles gut ist, IN = -1, wenn die erbetenen Aktion unmöglich ist.

Ein Programmbeispiel für die Bestandsroutinen:

```
8000 NF=1:NF$="TEST":GOSUB 500 :REM ÖFFNEN FÜR AUSGABE
8010 FOR I=1 TO 10
8020 INPUT"Geben Sie einen Namen ein ";SR$
8030 GOSUB 560 :REM NF=1
8040 NEXT I
8050 GOSUB 580 :REM BESTAND SCHLIESSEN
8060 PRINT"Spulen Sie die Kassette zurück und"
8070 PRINT"drücken Sie dann eine Taste"
8080 GOSUB 210
8090 NF=0:GOSUB 500 :REM ÖFFNEN FÜR EINGABE
```

```
8100 PRINT"Die Namen waren:"
8110 GOSUB 540:PRINT IN$
8120 IF IN=0 THEN 8110
8130 GOSUB 580
8140 ....
```

GOSUB 600 (graphischer Betrieb und Löschen des Schirms)

Diese Subroutine schaltet auf graphischen Betrieb um und löscht/reinigt den Schirm. Der graphische Cursor wird am Punkt 0,0 aufgestellt, was übereinstimmt mit der linken oberen Ecke des graphischen Schirms. Die Einteilung des graphischen Schirms ist folgendermaßen:

0,0 ist der Punkt genau links oben auf dem graphischen Schirm,
1,0 ist der Punkt genau außerhalb der rechten oberen Ecke,
0,1 ist der Punkt genau unter der linken unteren Ecke und
1,1 liegt schräg unter der rechten unteren Ecke des graphischen Schirms.

Nachbemerkung 1: Im graphischen Betrieb haben der übliche PRINT-Auftrag und die Subroutinen 100, 110, 120, 150, und 220 keine Bedeutung. Diese dürfen nicht verwendet werden, bevor mit GOSUB 100 wieder auf den textlichen Betrieb zurückgeschaltet worden ist. Behalten Sie das im Auge, wenn Sie mit Beständen arbeiten!

Nachbemerkung 2: Gemessen auf dem Bildschirm ist der Abstand von Punkt (0,0) zum Punkt (0,1) genau ¾ (75%) des Abstandes von (0,0) zu (0,1). Der Punkt (1/2, 1/2) kommt in die Mitte des Bildschirms.

GOSUB 620 (Plot und Punkt)

Diese Subroutine setzt den graphische Cursor auf den Punkt HO, VE und 'plottet' diesen Punkt. Sowohl HO als VE müssen mindestens gleich 0 und kleiner als 1 sein. Wenn CN=0 ist, wird der Schirmpunkt gesetzt (Vordergrundfarbe), wenn CN=1 ist, wird gerade dieser Punkt gelöscht (Hintergrundfarbe).

GOSUB 630 (Zug und Linienstück)

Diese Subroutine tut fast das gleiche wie Subroutine 620. Jetzt jedoch nicht nur der angegebene Punkt geplottet, sondern ein vollständiges Linienstück von dort an, wo der graphische Cursor stand, zu dem in HO und VE angegebenen Punkt. Das gesamte Linienstück wird in der mit CN bezeichneten Farbe gezogen. Nach Ablauf steht der graphische Cursor in dem von HO und VE angegebenen Endpunkt.

GOSUB 650 (Text auf dem graphischen Schirm)

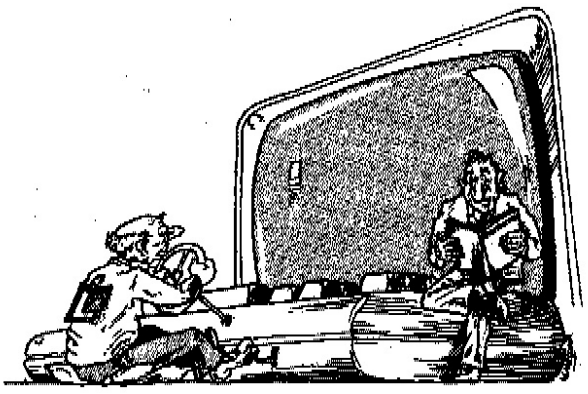
Diese Subroutine printet den Text in SR$ auf den graphischen Schirm. HO und VE müssen die (graphische) Position der linken oberen Ecke des ersten Zeichens angeben. Die Zeichen werden so weit wie nur möglich auf dem Schirm im 'normalen' Format abgebildet, wobei somit prinzipiell 40 auf den Schirm passen. Das ist jedoch nicht in allen Computern gewährleistet (vgl. Anhang 1). Es wird daher empfohlen, in SR$ nicht zu viele Zeichen einzugeben. Der Text wird wiedergegeben in Vordergrundfarbe (wie CN=0) oder in Hintergrundfarbe (wie CN=1). Nach Beendigung steht der graphische Cursor in HO, VE.
Ein einfaches Beispiel:

```
9000 GOSUB 600 :REM GRAFIKBETRIEB
9010 CN=0:HO=0.2:VE=0.1:GOSUB 620
9020 HO=0.8:GOSUB 630
9030 VE=0.9:GOSUB 630
9040 HO=0.2:GOSUB 630
9050 VE=0.1:GOSUB 630
9060 SR$="LINKS OBEN":GOSUB 650
9070 HO=0:VE=23.1/24:SR$="unterste Zeile":GOSUB 650
```

Dieses Stückchen Programm zeichnet auf den Schirm ein großes Quadrat, wobei zur linken oberen Ecke hin die Worte stehen 'LINKS OBEN'.

GOTO 950 (Programm-Beendigung)

Die Aufträge END und STOP sind in BASICODE-3 nicht zulässig. Stattdessen muß man verwenden: GOTO 950. Dann wird die Maschine wieder in den Zustand gebracht, in dem sie sich auch normalerweise auch nach dem Einschalten befindet, also Cursor an, Stopptaste in Betrieb, normaler Schirm, normaler 'string'-Raum usw. Danach wird der Schirm gelöscht, und das Programm ist zu Ende. Jedoch das Programm wird nicht angetastet, und die BASICODE-Routinen werden nicht vernichtet.
BASICODE-2-Programme müssen - um in BASICODE-3 arbeiten zu können - auf die gleiche Weise angepaßt werden.



GOTO 1000 (Programmstart)

Die Instruktion RUN ist in einem BASICODE-3-Programm untersagt. Wenn es erforderlich ist, vom Programm aus neu zu beginnen (beispielsweise um 'arrays' zu

löschen und neu zu dimensionieren), so muß man (obligatorisch) GOTO 1000 verwenden.

4.4 Übersicht der BASICODE-3-Subroutinen

```
 10  erste Zeile des Subroutineblocks
 20  Programmstart, System-Reset, Variable löschen usw.
100  man schalte um auf textlichen Betrieb und lösche den Schirm
110  man versetze den Cursor in die Position HO, VE
120  man registriere die Cursor-Position in HO, VE
150  man drucke auf auffällige Weise 3 Zwischenräume;SR$; 3 Zwischenräume
200  man gebe eine eventuell eingedrückte Taste in IN$ und IN
210  man warte bis zum Tastendruck und gebe diesen in IN$ und IN
220  man gebe in IN den Code dessen, was in Schirmposition HO, VE zu sehen ist
250  man gebe einen kleinen Pfiff als Aufmerksamkeitszeichen
260  man gebe eine Zufallszahl in RV, so daß 0 < = RV < 1
270  man mache 'garbage collect' und gebe in FR die Gesamtzahl freier 'bytes'
280  man schalte die Stopp-Taste ein (FR = 0) oder aus (FR = 1)
300  man gebe in SR$ die Textform von SR, ohne Leerzeichen
310  man gebe in SR$ die Textform von SR, formatiert gemäß CT und CN
330  man verändere alle Kleinbuchstaben in SR$ in Großbuchstaben
350  man schicke SR$ an den Printer
360  man schließe die Printzeile ab mit 'Wagenrücklauf' und 'Zeilensprung'
400  man mache einen Ton gemäß SV, SD und SP
450  man warte maximal SD * 100 Millisekunden auf einen Tastendruck
500  man öffne den Bestand NF$ gemäß Code NF
540  man lese IN$ aus geöffneten Bestand NF und in IN den Status
560  man bringe SR$ in den geöffneten Bestand mit Code NF
580  man schließe den Bestand mit Code NF ab
600  man schalte um auf graphischen Betrieb und lösche den  Schirm
620  man plotte den Punkt in Posi-
     tion HO, VE in Farbe CN
630  man ziehe eine Linie nach
     Punkt HO, VE in Farbe CN
650  man drucke SR$ als Text aus
     der Position HO; VE
950  man schließe das Programm ab
     und schalte die Maschine auf
     normalen Betrieb
```

4.5 Variable und BASIC-Befehle

Die Variablen, die im Programm verwendet werden, sind an einige Einschränkungen gebunden. Das ist nötig, um die Austauschbarkeit des Programms zu gewährleisten.

a) Numerische Variable sind "real" und haben einfache Genauigkeit. Die Genauigkeit beträgt in Abhängigkeit vom Computer mitunter nicht mehr als 6 Dezimalstellen; man rechne auf keinen Fall mit einer größeren Genauigkeit.

b) Die Namen der Variablen sind maximal zwei Zeichen lang, wobei das erste Zeichen ein Großbuchstabe sein muß. Wenn noch ein weiteres Zeichen folgt, darf es ein Großbuchstabe oder eine Zahl sein. In Namen von Variablen dürfen somit keine Kleinbuchstaben gebraucht werden. Für "String"-Variable folgt dem Namen ein $. Alle anderen Zusätze (wie %,# und !) sind in BASICODE-3 untersagt.

c) Logische Variable sind Variable, die "wahr" oder "unwahr" sind. Es darf kein Gebrauch gemacht werden, von dem evtl. numerischen Wert logischer Variablen. In einigen Computern wird "wahr" nämlich durch +1 dargestellt, in anderen durch -1. Das Ergebnis kann nur verwendet werden in einer IF...THEN-Konstruktion (so etwas wie A=3*(B=1) ist also absolut nicht erlaubt! Man gebrauche in einem solchen Falle beispielsweise A=0: IF B=1 THEN A=3.

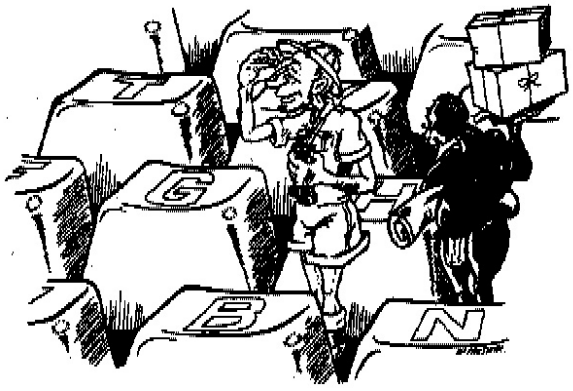
d) Bevor eine Variable verwendet wird, muß sie stets einen Wert bekommen. Sie dürfen mithin nicht annehmen, daß die Variablen beim Start eines Programmes automatisch auf Null gesetzt werden.

e) “String"-Variable dürfen maximal 255 Zeichen lang sein.

f) Namen von Variablen dürfen nicht mit dem Buchstaben O beginnen. Solche Namen sind dem Gebrauch innerhalb der BASICODE-3-Standardroutinen vorbehalten.

g) Ausgeschlossen sind gleichzeitig folgende Variablen:
A$, AT, DI, EI, FN, GO, GR, IF, LN, PI, SQ, ST, TI, TI$, TO.

h) Beim Aufrufen von BASICODE-3-Subroutinen können die folgenden Variablen verwendet werden: CN, CT, FR, HG, HO, IN, IN$, NF, NF$, RV, SD, SP, SR, SR$, SV, VE, VG.



4.6 Erlaubte BASIC-Befehle

Dieses Kapitel gibt eine erschöpfende Darstellung der Programme in BASIC. Von jedem Befehl wird eine kurze, sachliche Beschreibung gegeben, weil das BASICODE-3-Protokoll der Verwendung mancher Befehle einige Einschränkungen auferlegt. Ausschließlich die nachstehenden BASIC-Befehle und Funktionen dürfen in den Programmen verwendet werden:

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
^	*	/	+	-			
=	<	>	<=	>=	<>		

Nachstehend finden Sie in alphabetischer Reihenfolge die Beschreibungen der Arbeitsweise der erlaubten BASIC-Befehle.

ABS Gibt den ganzzahligen Wert der mitgegebenen Variablen.

Beispiele:

```
A= 10 :B=ABS(A)     nach Ablauf ist B=10
A=-20 :B=ABS(A)     nach Ablauf ist B=20
A=-1   :B=ABS(A-5)  nach Ablauf ist B=6
```

AND Logisches AND. Darf nur verwendet werden für logische Variable. Das Ergebnis ist ein logischer Wert. Die Verwendung von Klammern, um die Bearbeitungsfolge klar anzugeben, ist in BASICODE-3 obligatorisch.

Beispiele:

```
IF (A=5) AND (B=0) THEN ...
Q=(A=5) AND (B=0) : IF Q THEN ...
```

ASC Gibt den ASCII-Wert des ersten Zeichens des mitgegebenen 'string'.

Beispiele:

```
A$="A" :B=ASC(A$)          meistens gilt jetzt: B=65
AS="BEER" :B=ASC(AS)    meistens gilt jetzt: B=66
```

Nachbemerkung: Bei einigen Computer-Typen wird intern ein anderer Code für die verschiedenen Zeichen verwendet. Es kann dann beispielsweise geschehen, daß in den obigen Beispielen in B die Zahlen 193 bzw. 194 stehen. Was die Buchstaben oder Zahlen betrifft, gilt wohl immer, daß sich die abweichenden Werte um ein Vielfaches von 32 vom "normalen" Wert unterscheiden. Die Verwendung der ASC-Funktion muß daher am besten so weit wie nur möglich vermieden werden.

ATN Gibt den Arctangens in Radialen der beigegebenen Variablen.

Beispiele:

PRINT ATN(1) gibt das Ergebnis .785398
PRINT ATN(-1.5) gibt das Ergebnis -0.982794

CHR$ Gibt ein Zeichen, das im Computer übereinstimmt mit dem Wert der mitgegebenen Variablen. Die Variable darf einen Wert von 32 bis einschließlich 127 annehmen. Bei Werten, die kleiner sind als 32, ist Vorsicht am Platze, da sich die Kontrollzeichen bei verschiedenen Computern stark unterscheiden; lediglich die RETURN-Taste hat immer ASCII-Code 13.

Beispiel:

A$=CHRS(66) A$ enthält jetzt (meistens) den Buchstaben B.

Wie bereits bei ASC erwähnt, gibt es Computer, die intern abweichende Codes verwenden. Man gebrauche daher auch die CHR$-Funktion mit der größtmöglichen Vorsicht.

COS Gibt den Cosinus des mitgegebenen Winkels. Die Variable muß den Winkel in Radialen enthalten.

Beispiel:

PRINT COS(1) gibt das Ergebnis .540302

DATA Nach diesem Kommando folgen bis zum Ende des Zeile Werte für Variable, die mit READ gelesen werden können. Auf der Zeile dürfen nach mit READ zu lesenden Variablen keine anderen Kommandos (beispielsweise REM) auftreten.

Beispiel:

DATA 100, 200, "HALLO", "BASICODE", 4.6.89

DIM muß immer verwendet werden, um 'arrays' zu dimensionieren. Ein 'array' darf nur einmal in einem Programm dimensioniert werden, und zwar, ehe es im Programm verwendet wird. Die Zahl der Dimensionen beträgt im BASICODE-3 maximal zwei. Die Anzahl der Elemente wird begrenzt durch die Größe des Speichers. Verschiedene 'arrays' dürfen mit dem gleichen DIM-Auftrag dimensioniert werden. Man vergleiche das Beispiel.

Nachbemerkung 1: Bei einigen Computern brauchen 'arrays' bis zu zehn Elementen nicht dimensioniert zu werden. In BASICODE-Programmen ist das an sich immer nötig!

Nachbemerkung 2: Auch das Element mit der Nummer zählt mit, also A(0), und AD$ (0,0) gibt es auch.

Beispiel: DIM A$(12), HD(100,100), MP(1000).

EXP Erhebt die Zahl e (2.71828...) in eine bestimmte Potenz.

Beispiel:

PRINT EXP(2) erbringt als Ergebnis 7.38906

FOR FOR ... TO ... STEP ... NEXT ... Schleifenkonstruktion; die Schleife wird mindestens einmal durchlaufen. STEP und der Wert dürfen weggelassen werden, die Schrittgröße ist dann 1. Nach NEXT muß immer die Variable mitgegeben werden! Mehr als eine Variable nach NEXT ist nicht zulässig. Beispiele:

```
FOR X=10 TO 100
PRINT X,X+34
NEXT X
```

```
FOR C=A TO B STEP -3
PRINT C,C-B
NEXT C
```

```
FOR I=1 TO 10; FOR J=1 TO 5
PRINT I; " "; J; "="; I; J
NEXT J : NEXT I
```

Nachbemerkung: Man vermeide das Springen aus einer FOR-NEXT-Schleife, ohne daß diese beendet ist! Man verlasse eine FOR-NEXT-Schleife also nur über NEXT, nötigenfalls durch die Schleifenvariable auf den Endwert zu setzen. Verwenden Sie in Fällen, bei denen Sie aus der Schleife springen wollen, kein FOR-NEXT für die Schleifenstruktur.

GOSUB Man rufe eine Subroutine auf, beginnend bei der Zeilennummer, die hinter diesem Kommando erwähnt wird. Beispiel:

```
GOSUB 100
```

Nachbemerkung: Verboten ist A=100: GOSUB A

GOTO Man springe auf die Zeilennummer, die hinter diesem Kommando erwähnt wird.

Beispiel:

GOTO 1500

Nachbemerkung 1: Verboten ist A=1500. GOTO A
Nachbemerkung 2: Ein GOTO-Auftrag darf nie auf eine nicht vorhandene Zeilennummer oder auf eine Zeilennummer in einer BASICODE - Subroutine springen. Für das letztgenannte gibt es zwei Ausnahmen:
-1- In Zeile 1000 muß GOTO 20 stehen und
-2- GOTO 950 muß mindestens einmal im Programm als Abschluß des Programmes vorkommen (an Stelle der in den meisten BASIC-Versionen gebräuchlichen END- oder STOP-Befehle, die in BASICODE nicht erlaubt sind).

IF ... THEN Bedingte Verzweigung. Zwischen IF und THEN steht eine logische Variable oder ein logischer Vergleich. Wenn dieser 'wahr' ist, geht die Ausführung nach THEN weiter, wenn nicht, dann geht die Ausführung weiter auf die nächste BASIC- Zeile. Auch kann nach THEN eine Zeilennummer folgen, wo die Ausführung des Programmes weitergehen soll.

Nachbemerkung: ELSE ist in BASICODE-3 nicht zulässig.

Beispiele:

```
IF A=3 THEN B=0:C=5
IF A>3 THEN 1500
C=(A>3):IF C THEN GOSUB 100
```

NICHT: IF ... GOTO 2000 sondern IF ... THEN 2000
NICHT: IF ... GOSUB 2000 sondern IF ... THEN GOSUB 2000

INPUT Ersucht den Nutzer um Eingabe, die der Variablen nach INPUT zugewiesen wird. Diese Variable darf eine numerische Variable oder eine 'string'-Variable sein. Ein eingegebener 'string' darf keine Kommas oder 'Doppelpunkte' enthalten. Wenn man das doch wünscht, sollte man lieber die Subroutine auf Zeile 210 verwenden. Ein 'prompt-string' ist nicht erlaubt, ebensowenig wie mehr als eine Variable nach einem INPUT. Die meisten BASICs drucken ein Fragezeichen, wenn dieser Befehl erteilt wird. Nach dem Drücken der Return-Taste wird bei einigen Computern die übrige Zeile von der Cursorposition bis zum Ende gelöscht. Beispiele:

```
INPUT"Wie ist Ihr Name";N$
PRINT"Tippen Sie die Werte ein";: INPUT A: INPUT B
```

Nachbemerkung: INPUT"Ihr Name"; A$ ist somit nicht zulässig!

INT — Geben Sie die größte ganze Zahl, die höchstens der mitgegebenen Variablen gleich ist. Beispiele:

```
A=2.1 : B=INT(A)  nach Ablauf ist B=2
B=INT(-2.5)       nach Ablauf ist B=3
```

LEFT$ — Geben Sie einen 'string', der aus mehreren Zeichen des mitgegebenen 'string' besteht, beginnend mit dem am meisten links stehenden Zeichen. Die Zahl der Zeichen, die verlangt wird, muß minimal 1 sein und darf maximal die Länge des 'string' betragen.

Beispiel:

```
A$=LEFT$ ("BASICODE",5)   nach Ablauf ist A$=BASIC
```

Nachbemerkung: C$=LEFT$ (BASICODE, 0) ist also nicht erlaubt!

LEN — Gibt die Länge des mitgegebenen 'string'. Beispiele:

```
A$=BASICODE-3 : A=LEN(A$)  nach Ablauf ist A=10
A$="" : A=LEN(A$)          nach Ablauf ist A=0
```

LET — darf verwendet werden, wenn einer Variablen ein Wert zugewiesen wird, ist jedoch nicht erforderlich. Beispiel:

```
LET A=5 ist dasselbe wie A=5
LET wird daher fast immer weggelassen.
```

LOG — Berechnet den natürlichen Logarithmus der mitgegebenen Variablen oder des mitgegebenen Ausdruckes. Beispiele:

```
PRINT LOG(1)     bringt das Ergebnis  0
PRINT LOG(10)    bringt das Ergebnis  2.302585
```

Nachbemerkung: In einigen Computern wird der e-Logarithmus nicht mit LOG, sondern mit LN angegeben; in diesem Falle wird also LOG für den 10-Logarithmus gebraucht. Das Übersetzungsprogramm sorgt in diesen Fällen für eine richtige Übersetzung.

MID$ Holt eine Anzahl von Zeichen aus einem 'string'. MID$(A$,X,Y) gibt Y Zeichen von A$ beginnend mit dem X-ten Zeichen (das erste Zeichen hat die Nummer 1, X=0 oder Y=0 ist nicht erlaubt). Beispiel:

A$="BASICODE IST HÜBSCH": BS=MID$ (A$,10,2)
B$ enthält nun IS

Nachbemerkung: Nicht erlaubt ist der Befehl A$=MID$ (C$,5)

NEXT Abschließender Befehl einer Wiederholungsschleife (vgl. auch FOR). Nach NEXT muß immer die dazugehörige Variable stehen.

Beispiele: vgl. FOR...

NOT Logische Verneinung, nur anwendbar auf logische Variable (vgl. auch AND). Beispiele:

A=5; B=NOT(A=6) nach Ablauf ist B= 'wahr'
A=(5=5): B = NOT A nach Ablauf ist B = 'unwahr'

ON ON ... GOSUB ... oder
ON ... GOTO ...
Macht einen Sprung zu einer Subroutine oder zu einer Programmzeile. Nach ON folgt ein Ausdruck oder eine Variable, nach GOSUB oder GOTO eine Reihe von Zeilennummern, untereinander durch Kommas getrennt. Der Wert der Variablen oder des Ausdrucks muß eine ganze Zahl sein und bestimmt, welche Zeilennummer gewählt wird. Dazu können Sie sich die Zeilennummern nummeriert denken: Wenn die Variable 1 ist, wird die erste Zeilennummer gewählt, wenn die Variable 2 ist, die zweite Zeilennummer usw. Die Variable darf nicht größer werden können als die Anzahl der angegebenen Zeilennummern.

Beispiele:

ON K GOTO 1100,3400,1500 K muß nun 1, 2, oder 3 sein!
ON K-5 GOSUB 6000, 7000, 3000 K muß nun 6, 7, oder 8 sein!

OR Logisches OR, darf nur verwendet werden bei logischen Variablen (vgl. auch AND). Beispiele:

IF (A=5) OR (B=3) THEN ...
C = (A=5) OR (B 3): IF C THEN …

Nachbemerkung: Ebenso wie bei AND ist die Verwendung von Klammern obligatorisch.

PRINT Druckt eine Variable oder 'string' auf dem Schirm ab. Verschiedene Variable in einem PRINT-Befehl müssen durch ein Semikolongetrennt werden. Wenn kein automatischer Übergang zur nächsten Zeile gewünscht wird, muß am Ende des Befehls ein Semikolon stehen. Bei einigen Computern werden beim Drucken einer Zahl ein Zwischenraum oder mehrere Zwischenräume vor und/oder nach den Zahlen gedruckt. Wenn Sie das nicht möchten, können Sie die Subroutine auf Zeile 300 oder Zeile 310 verwenden. Beispiele:

```
A=5:A$="HALLO":PRINT A;" ";A$
5 HALLO

PRINT "HALLO";:PRINT " DAAR"
HALLO DAAR
CN=3:CT=5:SR=5:GOSUB 310:PRINT "VIJF=";SR$
VIJF=5.000
```

READ Liest die Angaben in den DATA-Befehlen und erkennt sie der/den Variablen nach READ zu. Mehrere Variable nach einem READ müssen durch ein Komma getrennt werden. Nach dem Starten eines Programms beginnt das Lesen beim DATA-Befehl mit der niedrigsten Zeilenzahl, bis alle Angaben auf dieser Zeile gelesen sind. Danach ist die nächste DATA-Zeile an der Reihe. Man beachte: Eine numerische Variable muß Zahlen lesen, eine 'string'- Variable muß 'strings' lesen!
Beispiele:

```
DATA 1, "COMPUTER" , 3
READ A: READ A$: READ B  oder
READ A, A$: READ B       oder
READ A, A$, B
```

REM Mit diesem Befehl können Sie einen Kommentar in ein Programm bringen, um es für andere verständlicher zu machen. Alles, was nach REM steht bis zum Ende der Zeile, wird von BASIC übersprungen. Es darf in der Zeile kein Doppelpunkt vorkommen. Das bringt mitunter Probleme mit sich.

RESTORE Lassen Sie das READ-Kommando wieder von der ersten DATA Zeile an im Programm lesen. Beachten Sie: Nach RESTORE darf keine Zeilennummer stehen!

RETURN Gibt das Ende einer Subroutine an. Nach diesem Befehl geht die Ausführung des Programms weiter bem ersten Befehl, der dem dazugehörigen GOSUB folgt. Eine Subroutine muß immer mit RETURN abgeschlossen werden!

RIGHT$ Gibt eine Anzahl von Zeichen eines bestimmten 'string', der beim letzten Zeichen endet. Die Mindestzahl der geforderten Zeichen muß 1 sein, die Höchstzahl die Länge des 'string'.
Beispiel:

A$=BASICODE: B$=RIGHT$(A$, 4) nach Ablauf ist B$ = "CODE"

Man beachte: A$="Protokoll": A=0: B$=RIGHTS (A$,A) ist nicht erlaubt, weil A=0

SIN Bestimmt den Sinus eines in Radialen angegebenen Winkels. Man vergleiche des weiteren bei COS.

SGN Gibt -1, wenn die Variable (oder der Ausdruck) negativ ist, 0, wenn die Variable gleich 0 ist und +1, wenn die Variable positiv ist.
Beispiele:

A=5: B=SGN(A) nach Ablauf ist B=1
A=-.001: B=SGN(A) nach Ablauf ist B=- 1

SQR Bestimmt die Wurzel einer Variablen oder eines Ausdrucks, die nicht negativ sein dürfen. Beispiel:

A=SQR(2*32) nach Ablauf ist A = 8

STEP bestimmt die Schrittgröße in elner Wiederholungsschleife. Vgl. FOR …

TAB Wird in den Print-Befehlen verwendet, um den Cursor auf eine bestimmte Stelle auf der Zeile zu setzen. Der Cursor kann weiter in die Zeile gesetzt werden, und in Abhängigkeit vom Computer werden Leerzeichen gedruckt, oder es bleibt stehen, was schon auf der Zeile stand. TAB (0) ist nicht gestattet. Die meisten Computer beginnen bei 0 zu zählen. Es gibt jedoch auch Computer, die bei 1 zu zählen beginnen. Daher ist es besser, die Subroutine auf Zeile 110 zu verwenden!

Beispiel:

PRINT "A"; TAB (5); "B"; TAB (10); "C"
A B C aber bei einem anderen Computer:
A B C

TAN Berechnet den Tangens eines in Radialen angegebenen Winkels. Vgl. des weiteren bei COS

THEN Vgl. IF

TO Vgl. FOR

VAL Bestimmt den numerischen Wert eines 'string'. Wenn der 'string' nicht rein numerisch ist, ist das Ergebnis nicht bei jedem Computer das gleiche. Beispiele:

A$= "1.4E6": A=VAL (A$) nach Ablauf ist A=1,4E6
A$= 12D : A=VAL(A$) nach Ablauf ist A unbestimmt.

4.7 Die zugelassenen Operatoren

4.7.1 Die Rechenoperatoren

Auf die Rechenoperatoren findet folgendes Anwendung: erst Potenzieren, dann von links nach rechts Multiplizieren und Dividieren, dann von links nach rechts Addieren und Subtrahieren. Darüber hinaus gilt folgendes: Was zwischen Klammern steht, wird zuerst berechnet, und Funktionswerte werden bestimmt, ehe das Rechnen mit den Rechenoperatoren beginnt.

= Ordnet den Wert des Ausdrucks rechts vom =-Zeichen den Variablen links vom =-Zeichen zu. Beispiel:

A=4+6

^ Erhebt eine Zahl oder Variable in eine bestimmte Potenz. Beispiel:

A=2:B=16:C=A^B nach Ablauf ist C=65536

* Multipliziert zwei Zahlen oder Variable miteinander. Beispiel:

A=5:B=3*2*A nach Ablauf ist B=30

/ Dividiert zwei Zahlen oder Variable miteinander. Beispiel:

A=5:B=100/A/2 nach Ablauf ist B=10

+ Addiert zwei Zahlen oder Variable miteinander. Beispiel:

B=1:A=B+9 nach Ablauf ist B=10

- Subtrahiert zwei Zahlen oder Variable voneinander. Beispiel:

  A=10-3-4 nach Ablauf ist A=3

4.7.2 Die 'string'-Operatoren

+ Koppelt 'strings' miteinander. Beispiele:

  A$="BAS": B$ = "ICO" : C$="DE-3": D$=A$+B$+C$
  nach Ablauf ist D$="BASICODE-3"
  Im Hinblick auf 'string'-Operatoren vergleiche man auch mit LEFT$, MID$ und RIGHT$.

4.7.3 Die logischen Operatoren

Unter logischen Operatoren verstehen wir die Operatoren, die als Ergebnis der Operation einen logischen Wert, nämlich 'wahr' oder 'unwahr', liefern. Dieser logische Wert kann sofort genutzt werden, beispielsweise nach IF, kann aber auch in einer numerischen Variablen gespeichert/aufbewahrt werden. Man beachte jedoch folgendes: Eine numerische Variable, in der ein logischer Wert gespeichert/aufbewahrt wird, darf nicht für rechenkundige Bearbeitungen genutzt werden.

= Vergleicht die zwei Variablen oder Ausdrücke links und rechts vom Gleichheitszeichen/=-Zeichen miteinander. Das Ergebnis ist entweder 'wahr' oder aber 'unwahr'. (Gleichzeitig: Ordnet den Wert des Ausdrucks rechts vom =-Zeichen der Variablen links vom =-Zeichen zu). Beispiele:

  A=(5=6) nach Ablauf ist A = 'unwahr'
  IF A$=B$ THEN ...

< Vergleicht zwei Variable oder Ausdrücke miteinander und betrachtet, ob der linke kleiner ist als der rechte. Das Ergebnis ist eine logische Variable. Wenn 'strings' verglichen werden, wird darauf gesehen, ob der linke 'string' eher als der rechte in einer alphanumerisch geordneten Reihe vorkommt (so können Sie also alphabetisch ordnen). Beispiele:

  A=5:B(A<7) B ist nun 'wahr'
  A$="HO":B$="HA":A=(A$<B$) A ist nun 'unwahr'

> Ebenso wie bei <, nur wird jetzt geprüft, ob größer als bzw. später in der Reihe.

<> Schaut, ob zwei Variable oder Ausdrücke ungleich zueinander sind.
Das Ergebnis ist wieder ein logischer Wert.

Beispiele:

A=(6<>7)	A ist nun 'wahr'
A$="HO":B$ ="H": A=(A$<>B$)	A ist nun 'wahr'
IF A<>5 THEN ...	

<= Kleiner als oder gleich mit. In bezug auf die Wirkungsweise vgl. <, doch man lese anstelle von 'kleiner als' 'kleiner als oder gleich mit'.

>= Größer als oder gleich mit.

Für logische Operatoren vergleiche man des weiteren mit AND, OR und NOT.

Bei den letzten drei Operatoren ist ausschließlich die oben angegebene Reihenfolge der Zeichen gestattet. Somit ist IF A=<5 beispielsweise FALSCH.

Commodore 64 und Commodore 128

Einlesen des Übersetzungsprogramms

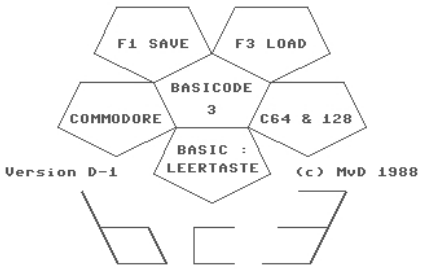
Das Übersetzungsprogramm ist völlig in Maschinensprache geschrieben, und es kann somit nicht mit einem LIST-Auftrag ausgelesen werden. Es wird nach dem 'resetten' (aus- und einschalten) des Computers von der Kassette aus mit dem Befehl LOAD oder - falls gewünscht - mit LOAD"C-64 BASICODE-3 eingelesen. Während des Einlesens ist leider nichts auf dem Schirm zu sehen. Das Einlesen von der Kassette dauert fast drei Minuten. Nach dem Einlesen kann das Programm mit RUN gestartet werden.

Nach diesem RUN-Befehl bringt das Programm den Computer in den BASICODE-3-Zustand. Dabei wird der Speicherbereich von $8C00 bis einschl. $CFFF in Gebrauch genommen und für BASIC abgeschirmt. Das dauert nur eine Sekunde. Sodann erscheint die Meldung, daß die Maschine in den BASICODE-Zustand gebracht worden ist, und im Speicher stehen nur die festen BASICODE-3-Subroutinezeilen.

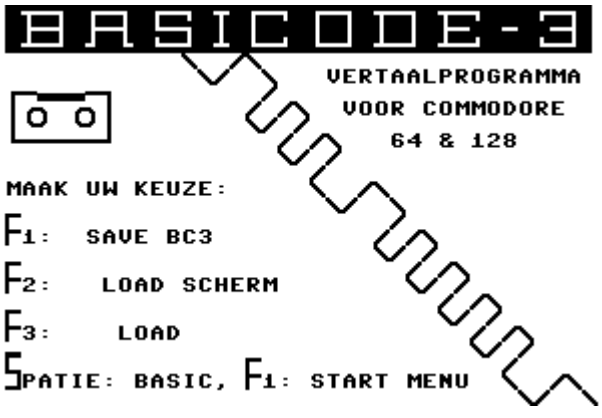
Es ist höchstwarscheinlich, daß das Übersetzungsprogramm nicht arbeiten wird in einem C-64 mit anderem Zubehör als einem ursprünglichen Commodore-Diskdrive und/oder Printer. Insbesondere wird das Programm nicht arbeiten, wenn ein Cartridge eingebracht worden ist. Wird das Programm in einen anderen Commodore-Computer als den C-64 geladen, so wird es ebenfalls nicht arbeiten, sondern ziemlich sicher den ganzen Computer auf 'tilt' bringen, und warscheinlich wird nur ein 'reset' diesen Computer wieder zu einer normalen Verhaltensweise bringen.

Gemäß Angabe vieler Nutzer arbeitet das Programm zugleich auf einem Commodore 128, der in '64 mode' gestartet wurde. Leider war es dem Verfasser nicht möglich, diese Angaben ordentlich zu überprüfen. Es ist mithin nicht ganz ausgeschlossen, daß bestimmte Programm-Bestandteile in einem Commodore 128 doch Probleme mit sich bringen.

Nutzung des Übersetzungsprogramms



Nachdem der Computer in den BASICODE-Zustand gekommen ist, kann man ein Wahl-Menu aufrufen, indem man auf eine (ganz gleich welche) der Funktionstasten drückt. In diesem Menu wird dann erwähnt, daß man mit einem Drücken auf die Funktionstaste f3 ein BASICODE-Programm einlesen kann (wobei - wenn nötig - die festen Subroutinen automatisch eingeregelt werden) und daß man mit einem Druck auf Funktionstaste f5 ein bereits vorhandenes Programm, das das BASI-CODE-3-Protokoll erfüllt, wegschreiben kann.
Mit einem Druck auf die Leertaste wird das Menu wieder verlassen, und man kehrt zurück zur normalen Funktionsweise des Computers. Wenn man - während das Menu auf dem Schirm steht - auf f2 drückt, erscheinen einige Extra-Zeilen im Bild, woraus hervorgeht, daß nächstens mit Funktionstaste f2 die Standard-Subroutinen bereitgestellt werden (nützlich, wenn man ein Programm in BASICODE schreiben will), daß mit Funktionstaste f4 BASICODE eingelesen und dem eventuell bereits vorhandenen Programm angekoppelt wird (man achte dann darauf, daß die Zeilennummern weiter laufen) und daß mit Funktionstaste f6 alle Programmzeilen (also ab Zeilennummer 0) in BASICODE-Format weggeschrieben werden. Die letzten beiden Möglichkeiten sind nur von Bedeutung, wenn man gemäß dem jetzt völlig veralteten BASICODE-1-Standard arbeiten möchte.



Entscheidet man sich für das Einlesen von BASICODE, so muß der Recorder angelassen werden. Dazu erscheint die übliche Meldung auf dem Schirm. Danach wird der ganze Schirm rot. Sobald das Leseprogramm einen BASICODE-Anlaufton liest, wird der Schirm grün. Sobald das eigentliche Programm eingelesen wird, wird der Bildschirm alle 1,2 Sekunden seine Farbe verändern, bis das ganze Programm eingelesen ist. Danach erscheint die normale READY-Meldung. Wenn ein Lesefehler festgestellt wird, wird das nach dem Einlesen gemeldet. Das Programm kann dann doch 'gelistet' werden, wodurch man möglicher-weise den Lesefehler finden und beheben kann. Wenn die Farbveränderungen aus-bleiben, wird offenbar nicht korrekt gelesen. Man kann das Einlesen dann abbrechen, indem man den Recorder stillsetzt,

das Band zurückspult und durch mehrmaliges Drücken der Funktionstasten das Einlesen erneut versucht.
Wenn man sich im Menu für Funktionstaste f5 (oder f6) entscheidet, also für das Wegschreiben, so wird das Übersetzungsprogramm erst kontrollieren, ob die wegzuschreibenden Programmzeilen die wichtigsten Forderungen erfüllen, nämlich ob die Zeilenlänge maximal 60 Zeichen beträgt und ob in den Programmzeilen verbotene Kommandos vorkommen (wie beispielsweise POKE und SYS). Diese Kontrolle geht sehr schnell und ist auf dem Schirm sichtbar. Wenn ein Fehler gefunden wird, stoppt das Programm mit einer Meldung, in welcher Zeile welcher Fehler ermittelt wurde. Man muß dann selbst diese Zeile 'listen' und verbessern. Danach kann wieder mit der richtigen Funktionstaste das Wegschreiben aufgerufen werden. Wenn keine Fehler gefunden werden, erscheint kurz nach der Kontrolle die normale Meldung, daß man den Recorder auf das Aufnehmen einstellen soll. Danach beginnt das Wegschreiben. Während ein BASICODE-Programm auf die Kassette weggeschrieben wird, wird - genau so wie während des Einlesens - der Bildschirm alle 1,2 Sekunden die Farbe wechseln. Nur während des Schreibens des 5 Sekunden wärenden Anlauftons ist der Schirm konstant grün gefärbt.

Das Übersetzungsprogramm kopieren

Eine Kopie des Übersetzungsprogramms kann dadurch gefertigt werden, daß das Übersetzungsprogramm erst mit LOAD eingelesen wird, wobei vor allem noch kein RUN-Befehl gegeben wird, sondern gerade ein passender SAVE-Auftrag, beispielsweise:
SAVE"C-64 BASICODE-3" oder
SAVE"0:C-64 BASICODE-3", 8 wenn man über ein 'disk-drive' verfügt.

Das Kopieren des Programms ist ausschließlich für den eigenen Gebrauch gestattet.
Man vergleiche in diesem Zusammenhang den Anhang über 'copyright'.

Spezielle Besonderheiten bei Commodore 64:

In BASICODE besteht der Textschirm aus 25 Zeilen mit je 40 Zeichen. Graphische Zeichen sind nicht gestattet. Unter BASICODE arbeitet die Maschine immerhin ausschließlich im Textbetrieb, wobei Großbuchstaben neben Kleinbuchstaben möglich sind.

Der graphische Schirm umfaßt 200 Zeilen/Linien von 288 Bildpunkten. Der graphische Schirm ist somit etwas schmaler als das, was maximal möglich wäre. Im graphischen Schirm sind 25 Zeilen Text mit maximal 36 Zeichen je Zeile möglich.

Die Musik umfaßt theoretisch fast neun Oktaven, nämlich von SP=0 bis einschließlich SP=106. Die niedrigsten beiden Oktaven produzieren jedoch eher Brummtöne als das, was man als angenehmes Geräusch empfindet.

Die in den meisten Computern verfügbaren 'accolades' sind im C-64 nicht möglich. An deren Stelle erscheinen Blockhaken auf dem Schirm.

Die Funktionstasten werden ausschließlich für das Aufrufen des BASICODE-Menu verwendet.

Die Farbtasten dürfen in BASICODE nicht verwendet werden.

Das Übersetzungsprogramm kann während des Lesens oder Schreibens von der Kassette getrennt werden, indem man den Recorder stillsetzt. In Notfällen kann man STOP + RESTORE verwenden.

Ein BASICODE-3-Programm kann mit der STOP-Taste abgebrochen werden. Es ist dann zu empfehlen, sofort das Kommando GOTO 950 einzutippen, um auf diese Weise den Computer in den normalen Zustand zu schalten. Wenn das Programm die STOP-Taste ausgeschaltet hat (Subroutine 280), wird in den meisten Fällen auch STOP + RESTORE nicht arbeiten, und das Abbrechen ist dann nur möglich durch das Ausschalten des Computers.

Im C-64 ist das Lesen und Schreiben von Beständen auf Kassette oder Diskette (anders als im BASICODE-Format, also mit NF=2 oder größer) nicht ganz gemäß dem Protokoll ausgeführt. In allen Fällen wird nämlich IN=0 abgeliefert. Der Urheber/Verfasser des Übersetzungsprogramms nimmt gern Vorschläge zur Verbesserung der jeweiligen Subroutinen entgegen.

Die BASICODE-3-Subroutinen enthalten viele SYS-Befehle nach Programmen in Maschinensprache im abgeschirmten Speicherbereich. Ein BASICODE-Programm wird also nur arbeiten können, wenn in diesem Speicherbereich die jeweilige Programmierung vorhanden ist. Man bringe somit erst den Computer in den BASICODE-Zustand, bevor man ein BASICODE-Programm verwendet. Man lese also erst das Übersetzungsprogramm ein und lasse dieses laufen, bevor man ein BASICODE-3-Programm aufnimmt und startet. Das ist vor allem von Bedeutung, wenn man eingelesene BASICODE-Programme auf eigener Diskette speichert und sie später wieder von der Scheibe/Platte aus wieder einliest.

SINCLAIR ZX-SPECTRUM

Einlesen

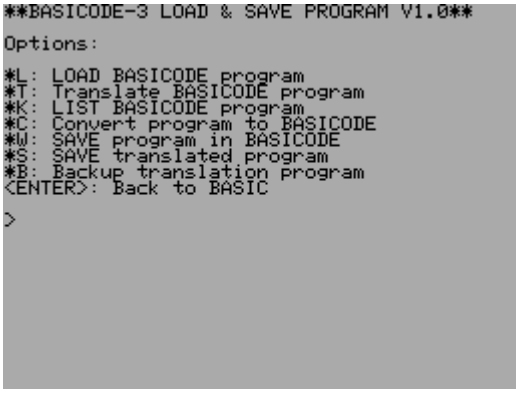
Man lese das Übersetzungsprogramm ein mit LOAD "". Das Übersetzungsprogramm besteht aus zwei Teilen, die nacheinander eingelesen werden: die BASIC-Subroutinen und das eigentliche Übersetzungsprogramm.

Sobald das Ganze eingelesen ist, startet das Programm von selbst und präsentiert sich. Danach gelangen Sie zu BASIC. Das Programm ist jetzt fertig zum Gebrauch.

Arbeitsweise und Wahlmöglichkeiten

Die verschiedenen Teile des Übersetzungsprogramms werden aktiviert duch das Eintippen eines Sternchens '*', dem ein Buchstabe folgt:

*L : Lesen Sie ein BASICODE-Programm ein.
*T : Übersetzen Sie das eingelesene Programm in Spectrum-BASIC.
*C : Verwandeln Sie ein Programm in Spectrum-BASIC zu BASICODE.
*W: Schreiben Sie das umgewandelte Programm in BASICODE um.
*K : 'LIST' das übersetzte Programm.
*S : 'SAVE' das übersetzte Programm in Spectrum-Format.
*B : 'Backup' des Übersetzungsprogramms.



Sie dürfen sowohl Großbuchstaben als auch Kleinbuchstaben eintippen. Wenn Sie nur ein Sternchen oder ein nicht bestehendes Kommando eingeben, wird ein Menue mit den obenstehenden Kommandos gezeigt. Sie können dann die gewünschte Wahl treffen, indem Sie auf die jeweilige Buchstabentaste drücken.

*L

Hiermit lesen Sie ein BASICODE-Programm ein. Setzen Sie den Recorder auf den Anfangston der Aufnahme und stellen Sie ihn auf das übliche Niveau ein. Sobald das Einleseprogramm den Anfangston gefunden hat, erscheinen Streifen auf dem Schirmrand. Wenn das wirkliche Programm eingegeben wird, sehen Sie die Programmzeilen unten auf dem Schirm vorbeihuschen. Solange das Einleseprogramm keinen BASICODE-Anfangston liest, wird der Schirmrand regelmäßig zwischen Rot und Zyan flackern. Das Einlesen hört auf, wenn Sie auf die Leertaste drücken oder wenn der Speicher voll ist oder wenn das Ende des BASICODE-Programms erreicht ist. Wenn das Einlesen erfolgreich verlaufen ist, macht das Programm sofort weiter mit dem Übersetzen (der nächste Schritt), sonst bekommen Sie eine Fehlermeldung 'BREAK into program', 'Out of memory' oder 'Tape loading error'. Sie können dann eventuell dennoch übersetzen, indem Sie die *T-Option wählen.

*T

Diese Option wird normalerweise nach dem fehlerlosen Einladen automatisch ausgeführt. Das in den freien Speicherbereich eingelesene BASICODE-Programm wird umgesetzt in ein Programm in Spectrum-BASIC. Jede übersetzte Programmzeile wird auf dem Schirm gezeigt.

*C

Das ist das entgegengesetzte der Option *T: Das im Spectrum vorhandene BASIC-Programm wird ab Zeile 1000 zu einem BASICODE-Programm im freien Speicherbereich umgesetzt. Zuallererst sieht sich das Übersetzungsprogramm um, ob Sie keine verbotenen Befehle verwendet haben, wie beispielsweise POKE, oder PLOT, und ob alle Funktionen gefolgt werden durch ein Argument zwischen Klammern (also nicht SIN X und TAB 10, sondern SIN (X) und TAB (10)). Sollte dennoch solch ein Fehler in Ihrem Programm sitzen, so bekommen Sie die Meldung 'ERROR', der die Zeile folgt, in der der Fehler steckt, mit einem Fragezeichen hinter dem falschen Wort. Sie können dann diese Zeile verbessern und die *C-Option aufs neue beginnen.
Wenn das Programm keinen der oben genannten Fehler enthält, so wird automatisch weitergemacht mit dem eigentlichen Übersetzen. Es ist hierbei möglich, daß Sie eine Meldung 'Line too long at line', gefolgt von der Zeilennummer, bekommen. Da ist dann eine Zeile entstanden, die länger ist als die zugelassenen 60 Zeichen. Das Umsetzen wird hierdurch NICHT aufhören. Wenn der Speicher zu klein ist für das unübersetzte und übersetzte Programm gleichzeitig, bekommen Sie nach dem Umsetzen die Meldung 'BASIC PROGRAM OVERWRITTEN'; sollte der Speicher auch zu klein sein für das übersetzte Programm allein, so folgt die Fehlermeldung 'OUT OF MEMORY'. In beiden Fällen sind Sie dann das ursprüngliche Programm los!

*W

Hiermit schreiben Sie das auf obige Weise umgesetzte Programm in BASICODE-Format in die Kassette. Sie können das so viele Male machen, wie Sie wollen. Das auf die Kassette weggeschriebene Signal wird auf dem Schirmrand gezeigt wie beim Einlesen. Sie können das unterbrechen mit der Leertaste.

*K

Diese ist die List-Option (Hinweis: Das LIST-Schlüsselwort sitzt auf der K-Taste). Ein Programm, das in BASICODE-Form (unmittelbar nach dem Einlesen oder vor dem Wegschreiben) im Speicher sitzt, können Sie normalerweise nicht LISTen, und daher diese Option. Sie können hiermit auch beurteilen, was nach einem Einlesefehler hineingekommen ist.

*S und *B

Mit der '*S'-Option können Sie das übersetzte BASICODE-Programm SAVEn in einem gewöhnlichen Spectrum-Format, so daß Sie es später direkt einladen können. Mit '*B' machen Sie ein 'backup' des Übersetzungsprogramms. Sie haben bei diesen Optionen die Wahl zwischen Kassette und Microdrive (sofern angeschlossen).
Anmerkung: Diese Optionen laufen über die BASIC-Zeilen 960 bzw. 980, so daß Sie eventuell an Ihr eigenes Speicherungssystem anpassen können.

Arbeiten mit BASICODE-3-Beständen

Die Bestandsroutinen arbeiten fuer NF=0 und NF=1 mit BASICODE-Beständen gemäß dem Protokoll. Soll ein Datenblock nach 'tape' abgeschrieben werden, so erscheint auf der untersten Schirmzeile die Meldung 'PRESS <PLAY> & <REC> THEN ANY KEY' und wartet Spectrum auf einen Tastendruck, bevor die Daten weggeschrieben werden. Beim Einlesen bekommen Sie die Meldung 'START TAPE …'. Wenn ein Block eingelesen wird, der neben dem erwarteten Block liegt, so erscheint die Meldung 'WRONG RECORD - REWIND TAPE' auf dem Schirm.

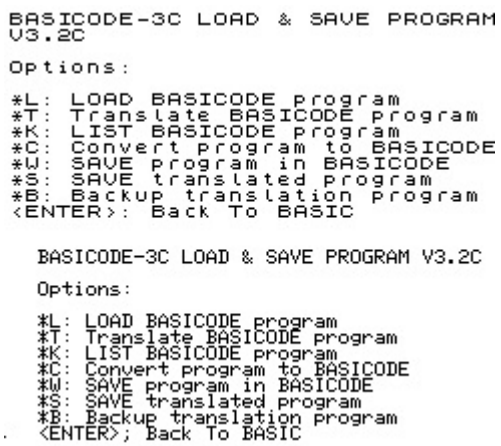
Wenn beim Einlesen eines Blocks ein Lesefehler entdeckt wird, kann einmal oder mehrmals die Meldung LOAD ERROR - REWIND TAPE auf dem Schirm erscheinen. Man spule in einem solchen Fall das Band ein Stückchen zurück, so daß der fehlerhaft gelesene Block erneut gelesen werden kann. Für die übrigen Werte von NF wird Microdrive als Speicherungsmedium verwendet.

Arbeiten mit BASICODE-1

Es ist mit dem Übersetzungsprogramm möglich, gemäß dem alten BASICODE-1-Standard alle Programmzeilen ab Zeilennummer 2 auf Kassette zu schreiben. Dazu dienen zwei POKE-Befehle, die eingetippt werden:

POKE 54580,2 und
POKE 5481,0.

Nach diesen Befehlen ist das Arbeiten gemäß der BASICODE-3-Norm nicht mehr möglich. Man muss das Übersetzungsprogramm erneut einlesen.



BASICODE-3C LOAD & SAVE PROGRAM V3.2C
Options:
*L: LOAD BASICODE program
*T: Translate BASICODE program
*K: LIST BASICODE program
*C: Convert program to BASICODE
*W: SAVE program in BASICODE
*S: SAVE translated program
*B: Backup translation program
<ENTER>; Back To BASIC

Inbetriebnahme BASICODE-3

Der Textschirm umfaßt 24 Zeilen mit 42 Zeichen. Ein eventuell angeschlossener ZX-Drucker oder da-mit vergleichbarer Drucker gibt ebenfalls 42 Zei-chen je Zeile.

Der graphische Bildschirm umfaßt 192 Linien mit jeweils 256 Punkten je Linie (die zwei untersten Schirmzeilen machen also auch mit). Der gra-phische Schirm kann 24 Zeilen Text mit 42 Zeichen wiedergeben.

Die Musik umfaßt alle zugelassenen Werte von SP (0 bis 127); es ist jedoch nur eine Lautstärke möglich (bei SV=0 gibt Subroutine 400 kein Geräusch, bei allen anderen

Werten allerdings und in der gleichen Stärke). Der Ton kann maximal 10 Sekunden dauern (SD=104).

Sie dürfen NICHT das Copyright-Symbol (Code 127) verwenden, weil dieser Code in BASICODE-3 fuer die DELETE-Funktion gebraucht wird.

Die Kontroll-Codes der 42-zeiligen Schirmroutine sind folgendermaßen:

16 bis einschl. 23 als normal,
24, gefolgt von N, stellt die Zahl der Schirmzeilen ein (...24),
25 = PRINT Komma.
26 = ständig 'scrollen',
27 = man warte fuer 'scroll' auf Tastendruck,
28 bis einschl. 31 und 127 gemäß BASICODE-3-Standard.

Besonderheiten

Dieses Übersetzungsprogramm enthält einen speziellen BASIC-Dolmetscher, der Spectrum-BASIC mit BASICODE-3 in Übereinstimmung bringt. Sie dürfen jetzt also Zeilenzahlen über 9999 sowie String-, Array- und FOR-NEXT-Variable namentlich mit maximal zwei Buchstaben verwenden. Gleichzeitig brauchen Sie bei 'string-arrays' nicht mehr eine Extra-Dimension anzugeben, und es ist auch ein Index 0 zulässig.

Die Funktionen LEFT$, MID$ und RIGHT$ sind vorhanden, jedoch in der Form von 'user-defined' Funktionen. Sie müssen sie somit als FN LEFT$ usw. eintippen.

Nachbemerkung: Die bei Sinclair gebräuchliche Notierung A$ (X TO Y) ist NICHT zulässig.

ON-GOTO und ON-GOSUB sind vorhanden in der Form GOTO/GOSUB * V; 2000, 3000, 4000 ... anstelle der gebräuchlichen ON V GOTO/GOSUB 2000, 3000, 4000 ... (man lese das Sternchen als ON).

Es ist nun aber wohl ein 'CLEAR,A' erforderlich (vgl. Zeile 20), dessen Wert abhängig ist von dem gesamten Speicherraum an 'strings', die Sie mit INPUT und 'string'-Kopplung zusammenstellen. Für die meisten Programme genügt ein Wert von 100 bis 250. Wenn Sie dieses zu klein wählen, bekommen Sie eine Fehlermeldung 'Out of string space'. Berücksichtigen Sie vor allem die obenstehenden Dinge beim Schreiben eines BASICODE-3-Programms. Das Umsetzprogramm kann nicht alle Fehler aufstöbern!

Bei Verwendung von 'diskinterfaces' anders als die Sinclair Interface1 können Sie Schwierigkeiten bekommen bei Aufträgen, die den Gebrauch der 'interface' betreffen. Vermeiden Sie hierbei die Verwendung von Variablen und tippen Sie Zahlen wie VAL "getal" ein.

Nachbemerkung: Zeile 1 darf niemals aus dem Übersetzungsprogramm entfernt werden! Ohne diese Zeile ist das Arbeiten mit BASICODE nicht zuverlässig.

Jan Bredenbeek

Anhang 2

Der Unterschied zwischen BASICODE-2 und BASICODE-3

Dieser Anhang behandelt die Unterschiede zwischen BASICODE-2 und BASICODE-3. Das eine und das andere wird speziell für diejenigen erwähnt, die mit BASICODE-2 vertraut sind und in aller Kürze überschauen möchten, was sich an BASICODE-3 verändert hat und was ihm hinzugefügt wurde.

A2.1 Allgemeine Vereinbarungen

In BASICODE-2 wurde der Bildschirm definiert als 24 Zeilen von 40 Zeichen. Die Computer, die mehr oder weniger zu bieten hatten, erwiesen sich in einigen Programmen als abweichend reagierend. In BASICODE-3 ist daher das Maß von 24 zu 40 nur als vernünftiger Durchschnitt erwähnt. Jedes Programm kann sich jetzt einfach anpassen an das wirkliche Bildformat und zwar dadurch, daß in Zeile 1010 in HO und VE die maximal zulässigen Werte für die jeweiligen Computer verfügbar sind.

Neu ist der graphische Bildschirm, der ein eigenes Koordinatensystem bekommen hat: (0,0) in der linken oberen Ecke, (1,0) genau neben der rechten oberen Ecke und (0,1) genau unter der linken unteren Ecke. Desweiteren wurde festgelegt, daß die Höhe des graphischen Bildschirms genau 3/4 der Breite beträgt.

```
10 Seiten Abschrift beendet !!!
     Dieter Schleinitz
     Fominstr. 61/204
     Ruf: 2985
     R A D E B E R G
     ---------------
     8142
```


BASICODE-3
Boek + cassette ISBN 90 201 1949 4
© 1986 Kluwer Technische Boeken BV
De cassette bevat 10 vertaalprogramma's, geschikt voor meer dan 30 micro-computers, en drie demonstratie-programma's in BASICODE-3.
Kant A
– Acorn BBC Modellen B, B+64K en B+128K
– Acorn Electron
– Apple II, II+, IIe en de meeste Apple-compatibles
– Commodore 64
– Exidy Sorcerer
– Alle MSX-1 en MSX-2 computers
– Philips P2000M
Kant B
– Philips P2000T
– Sinclair Spectrum en Spectrum+
– Spectravideo SV.318 en SV.328
gevolgd door drie programma's in BASICODE-3:
– Sterrenhemel
– Muziek
– adresbestand
gevolgd door een bestandje in BASICODE-3 met allerlei nuttige adressen.
08-42875
stichting
BASICODE
1
BASICODE - 3
Verzamelcassette
© 1987
BASICODE - 3
Verzamelcassette 1
KANT A
BASICODE-3
Kant A
08-42875
BOEK + CASSETTE ISBN 90 201 1949 4
© 1986 KLUWER TECHNISCHE BOEKEN BV
BASICODE - 3
Verzamelcassette 6
BASICODE - 3
Verzamelcassette 7
BASICODE - 3
Verzamelcassette 8
BASICODE - 3
Verzamelcassette 9
stichting
BASICODE
BASICODE - 3
Verzamelcassette 2
stichting
BASICODE
BASICODE - 3
Verzamelcassette 3
stichting
BASICODE
BASICODE - 3
Verzamelcassette 4
stichting
BASICODE
BASICODE - 3
Verzamelcassette 5