```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                            +
+        BASICODE-3 - INFO +++ KURZINFORMATION               +
+                                                            +
+   Bearbeitet: H.-J. Bachmann,  Computer-Club robotron      +
+                                                            +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Literaturhinweis: FF-dabei – Serie Begleitmaterial Radio DDR "REM"

BASICODE ist ein Computer-Esperanto, welches von den unterschiedlichsten Computertypen verstanden und abgearbeitet werden kann. Voraussetzung dazu ist ein auf den jeweiligen Typ zugeschnittenes Konvertierungsprogramm oder ein BASICODE-Interpreter. BASICODE verlangt eine nach festen Regeln strukturierte Programmgestaltung.
Auf der Tagungskassette finden Sie für die Verwendung auf dem MRB Z1013 zwei Teilprogramme.

-BASCODER
-BASCON

Dieses sind noch Testversionen und werden vervollkommnet. Weiterhin ist vorgesehen eine einteilige 64k-Version zu schaffen.

*** BASCON - ist ein Konvertierungsprogramm
Damit wird der vom Rundfunk abgestrahlte BASICODE in ein KC- File umgesetzt.
Die Bedienung ist MENUE gesteuert.
Für die Weiterverarbeitung ist das umgesetzte KC- File abzusaven.
BASCON ist nur fuer den Z1013 relevant, andere Computertypen verarbeiten BASICODE direkt mit ihrem eigens dafuer vorhandenen
vorhandenen Interpreter!!!

*** BASCODER - Dieses ist der eigentliche Interpreter, der die mit BASCON konvertierten Files verarbeiten kann.

LADEN : LOAD#1">name<"
SAVEN : LIST#1">name<"

Mit BASCODER erstellte Programme sind vor Weitergabe grundsätzlich in das BASICODE-Format zu konvertieren, damit die Universalität des Austausches auch für andere Computertypen gewährleistet ist.

Programmierungshinweise in BASICODE entnehmen Sie bitte den Publikationen zu diesem Thema.

Weiterhin befindet sich auf der Tagungskassette ein BASICODE- File, welches zur Übung im Umgang mit BASCON noch von Ihnen konvertiert werden muß.

HINWEISE zu BASCON und BASCODER an:
URANIA Computerclub, Kreispionierhaus"G.Orbschat",
Martin Duchrow * PF 367 1093 Berlin

und zu den Rundfunkübertragungen an:
Radio DDR , Dr. J.Baumann , Redaktion Schulfunk
Nalepastr. 18-50 , 1160 Berlin

Software zur Weiterverbreitung an:
Computer-Club robotron, PF 180 , 7010 Leipzig
für alle nachfolgenden Typen.
Für den Vertrieb von BASICODE-3 ist der Einsatz der Schallplatte geplant.

BASICODE-3 wird verfügbar sein für:
MRB Z1013
alle KC-Typen
Sinclair Spectrum
Commodore 64 (C128) und 4000'er-Serie
Atari (noch in Arbeit)
MSX-Computer mit mind. 64k-RAM
IBM-Computer und kompatible
Schneider CPC
Weitere Entwicklungen nicht ausgeschlossen!

Wir wünschen Ihnen als USER mit BASICODE-3 viel Erfolg !
- H.-J.Bachmann - Computer-Club robotron
- Dr. J. Baumann - Radio DDR/Schulfunk
- M. Duchrow - Autor der Z1013-BASICODE-Version

(… unvollständig …)

*******MikroWORD-Textfile*******Stand:10/89(ba)**********

-tine den Wert von 10 SD nach unten auf eine ganze Zahl abrundet. Man bedenke des weiteren, daß diese Routine erst aufgegeben wird, wenn der Ton abgelaufen ist, somit also wenn die in SD angegebene Zeit verstrichen ist! Wenn das nicht exakt möglich ist, darf die Routine eine Fraktion eher stoppen (10 + SD nach unten abgerundet), um auf diese Weise die overhead-Zeit in BASIC ein wenig zurückzugewinnen.

Subroutine 450 kommt ebenfalls mit einem nicht-ganzen SD auf die Beine. Man sorge dafür, daß nach Beendigung nicht unglücklicherweise eine negative Zahl an SD geliefert wird und daß SD in den richtigen Einheiten ausgedrückt wird. Der Wert in IN stimmt wieder mit dem überein, was bei den Routinen 200/210 erwähnt wurde.

Die beständigen Routinen 500, 540, 560 und 580 dürfen die unterste Schirmzeile für Anweisungen und Meldungen benutzen. Vor dem Verlassen der Routinen müssen diese Meldungen natürlich wieder entfernt werden. Es wird sogar noch eine schönere Lösung empfohlen: Vor dem Verlassen der Routine wird die unterste Schirmzeile wieder so hergestellt wie diese beim Anruf der Routine ausgesehen hat (man achte auf den Unterschied zwischen textlichen und graphischem Betrieb; man tue es nur, wenn es wirklich mit Leichtigkeit möglich ist: Im Buch steht immerhin, daß die unterste Zeile für Meldungen verwendet wird. Ein gutes BASICODE-Programm hält mithin die unterste Zeile prinzipiell frei). Während des Lesens und Schreibens von Bestandsblöcken geht auch der Inhalt dieses Blockes über diese unterste Schirmzeile.

Wohlgemerkt: Solange sich der Computer im graphischen Betrieb befindet, ist es für einige Computer sehr mühevoll, Meldungen auf die unterste Schirmzeile zu drucken/printen. Daher denken wir daran, in das Buch aufzunehmen, daß die Bestandsroutinen nur verwendet werden dürfen, wenn die Maschine im textlichen Betrieb steht. Wenn das tatsächlich aufgenommen wird (im nächsten Druck), wird es nötig sein, daß die Bestandsroutinen bei einer Fehlermeldung stoppen, wenn sie angesprochen werden, während sich die Maschine im graphischen Betrieb befindet. Sie werden noch von uns hören! Werden wir etwas hören, wenn Sie im graphischen Betrieb Probleme mit dem Printen/Drucken des Textes haben?

Bei NF=2 bis einschließlich NF=7 ist es möglich, daß Meldungen nicht auf die unterste Zeile gelangen. In diesen Fällen ist es immerhin das eigene 'operating system', das für die Datenübertragung/-beförderung sorgt, und das hat man nicht

immer ganz unter Kontrolle. Wenn es ausführbar ist, auch diese Meldung abzufangen und auf die unterste Schirmzeile zu plazieren, dann natürlich gern!

Diese Bestandsroutinen geben alle vier nach Ablauf eines jeden Anrufes in IN einen 'Statuscode':

IN=0 : alles ist gut gegangen
IN=1 : in INS wurde jetzt oder früher der letzte 'string' (engl. Strick, Seil) abgeliefert (nur während des Lesens von Beständen)
IN=-1: es ist ein nicht behebbarer Fehler aufgetreten.

Beim Lesen wird stets INS="" (und IN=1), wenn der letzte 'string' aus dem Bestand schon eher abgeliefert worden ist.

Wie Bestände genau auf dem Band stehen, steht im BASICODE-3-Buch. Es ist zu empfehlen, den letzten Bestandsblock mit hex 04 aufzufüllen.

Wenn sich ein Bestandsblock als nicht gut lesbar erweist, darf zweimal mittels einer Meldung auf der untersten Schirmzeile zum Zurückspulen des Bandes aufgefordert werden, wonach ein neuer Leseversuch stattfindet. Tritt dann noch immer ein Einleseproblem auf, so wird das mittels IN=-1 dem Programm gemeldet, und der Inhalt des (mit Fehlern) eingelesenen Blocks wird weiterhin auf dem Weg über Subroutine 540 ganz normal (also mitunter mit IN=-1) dem Programm angeboten.
Damit wollen wir erreichen, daß schlecht lesbare Bestände doch noch möglichst gut verarbeitet werden. Man vermeide auf jeden Fall, daß das Programm ein IN$ geliefert bekommt, in dem Zeichen auftreten, die in BASICODE-3 unmöglich sind, ganz davon zu schweigen, daß im 'string' fremde Kontroll-Codes stehen. Man ersetze eventuelle fremde Zeichen durch Gitter (#).

Als NF=2 oder NF=3 wird ein Bestand entweder auf eine Kassette oder auf eine Diskette geschrieben bzw. von dort abgelesen, nämlich nach/von dem, was in dem jeweiligen Computer Standard ist. Vielleicht kann das Übersetzungsprogramm einholen, was dort genau angeschlossen ist? In der Gebrauchsanweisung muß erwähnt werden, was das Übersetzungsprogramm bei diesen Werten von NF genau tut!

Die graphischen Routinen 600, 620, 630 und 650 müssen die Werte, die in HO und VE angegeben werden, intern vervielfältigen mit der verfügbaren Zahl von Bildpunkten und dem Ergebnis den INT entnehmen. Das erbringt die Nummer des angegebenen Bildpunktes. Man beachte folgendes: (0,0) ist links oben, und (1/2,1/2) ist die Mitte des Schirmes.

Es darf nichts aus der Hand gehen, wenn HO und/oder VE kleiner als 0 oder größer als 1 sind. Was da wohl geschieht, ist nicht so von Bedeutung, wenn dort nur etwas

geschieht, woran der Programmierer eines BASICODE-3-Programmes sehen kann, daß etwas falsch gelaufen ist. Empfehlung: Genau so wie in Subroutine 110 wird bei negativen Werten so getan, als würde der Wert 0 aufgegeben, und bei Werten ab 1 wird so getan, als würde 1-1/HG bzw. 1-1/VG aufgegeben. Bei Werten ab 1 wird mithin so getan, das heißt, es wird auf dem äußersten Schirmrand plaziert. Darüber hinaus wird ein kurzes Tönchen aus dem Lautsprecher gegeben.

Man achte darauf, daß Subroutine 630 aufgerufen werden kann, wenn sich in HO, VE die Koordinaten des Punktes befinden, wo sich der graphische Cursor zufällig schon befindet. In diesem Fall wird natürlich nur dieser eine Punkt plaziert.

Subroutine 650 hat nur Einfluß auf die Bildpunkte, die mit dem gewünschten Buchstaben übereinstimmen.

Beim Drucken/Printen eines Zwischenraumes tut diese Routine überhaupt nichts, ungeachtet dessen, ob CN=0 oder CN=1. Es ist zugelassen, daß Routine 650 bei einem zu langen 'string' in dem eventuellen rechten 'border' weiter printet/druckt. Was dann wirklich nicht mehr darauf kann, muß auf jeden Fall 'irgendwo' auf dem Schirm unterkommen. Die in HO und VE angegebene Position stimmt auf jeden Fall überein mit dem linken Oberwinkel der Charakterzelle des ersten Zeichens in dem zu plazierenden 'string' (und somit nicht mit dem am meisten links oben plazierten Bildpunkt, der wirklich 'geplottet' wird).

In Zeile 950 (wohin mit GOTO gesprungen wird) wird die Maschine wieder in den normalen Stand versetzt. Diese Zeile enthält somit einen internen GOSUB 100, und um auf jeden Fall auf textlichen Betrieb umzuschalten, setzt der Cursor wieder ein, schaltet die Stopptaste auf 'arbeitend' um usw. Das Programm bleibt normal erhalten, ebenso wie alle Variablen.

Tipp: Man setze in Zeile 950 den Auftrag GOSUB 951:END und setze die benötigten Umschaltungen in eine Subroutine, die somit auf Zeile 951 beginnt.

Allgemein: Es ist möglich, daß das BASICODE-3-Programm einen Progammier-fehler enthält, bei dem der Computer mit einer Fehlermeldung stoppt. Es ist wohl ganz in Ordnung, wenn diese von den Subroutinen aufgefangen wird, die dann erst eine interne GOSUB 951 ausführen und erst danach die Fehlermeldung auf den Schirm kommen lassen. Auf dem Weg über/mittels "ON ERROR" oder durch das Umsetzen des 'Fehlermeldungsvektors' läßt sich das eine oder andere wohl regeln.

Die Subroutinen 200, 210 und 220 geben in der Variablen IN unter allen Umständen ausschließlich Zahlen von 32 bis einschl. 95 ab (oder die Codes 28 bis 31 und 127).

Subroutine 330 tut etwas Ähnliches: Nach dem Ablauf wird SR$ nur Charaktere mit ASCII-Codes von 32 bis einschl. 95 enthalten.

Einige Computer kennen zwei Logarithmen: den elog und den 10log. Wo im BASICODE-3-Buch von LOG die Rede ist, ist der elog gemeint. Wenn ein Computer intern für den elog ein anderes BASIC-Wort als LOG gebraucht, muß das Übersetzungsprogramm für eine korrekte Übertragung sorgen und muß in der Gebrauchsanweisung in dem Buch bei diesem Computer deutlich erwähnt werden, daß diese Übertragung notwendig ist und automatisch erfolgt.

Paragraph 3: BASICODE-1, -2 oder -3

Obwohl die BASICODE-1-Methode kaum mehr angewandt wird, haben wir auf unsere Fahne geschrieben, daß das immer noch möglich sein muß. Somit muß normalerweise ein Leseprogramm für die Subroutine sorgen und ab Zeile 1000 laden; ein Schreibprogramm muß ab Zeile 1000 schreiben.

Aber ... es muß daneben ein Kommando verfügbar sein, vorzugsweise nur erwähnt in der schriftlichen Gebrauchsanweisung, wodurch das Leseprogramm die Subroutine NICHT klarstellt und nur immer so drauflos einliest. So muß es auch möglich sein, mittels eines geeigneten Kommandos das SAVEN von der ersten Programmzeile an starten zu lassen.

Wenn möglich, muß ein Leseprogramm auch in der Lage sein einzulesen, während bereits ein BASIC-Programm im Gedächtnis steht, und dann alles, was hereinkommt, einzufügen (zu MERGEN) in das, was bereits vorhanden war. In der Praxis wird diese Möglichkeit kaum genutzt. Daher kann man sich eventuell mit einem Leseprogramm begnügen, das die neu eingelesenen Zeilen nicht dazwischenfügt, sondern hinter der letzten Zeile plaziert (APPEND).

BASICODE-2 ist veraltet, seit es BASICODE-3 gibt. Lese- und Schreibprogramme brauchen mithin nicht nach dem BASICODE-2-Verfahren arbeiten zu können.

Paragraph 4: BASICODE einlesen

Das BASICODE-Einleseprogramm muß ohne weiteres zuverlässig einlesen, auch und gerade dann, wenn die Flanken im Signal nicht exakt auf der richtigen Stelle stehen (jitter).

Lesefehler können als solche kaum festgestellt werden, nur wenn die Charakter/Zeichen außerhalb des Bereiches von ASCII 32 bis ASCII 126 hineinkommen und/oder wenn an Ende die 'checksum' nicht stimmt, stellt sich heraus, daß da etwas schief gegangen ist.

Das Lesen von BASICODE kann auf 4 Arten enden:

a) dadurch, daß das ganze Programm geladen ist, bis einschließlich ETX, 'checksum' und Auslaufton;
b) wenn entweder kein Signal oder aber ein erkennbarer Auslaufton gelesen wird;
c) wenn der Nutzer das Lesen mit einem Druck auf die STOP- oder BREAK-Taste der Tastatur und/oder dadurch, daß er den Recorder stoppt, abbricht;
d) dadurch, daß der verfügbare Speicher voll läuft; das Leseprogramm muß daraufhin testen und rechtzeitig abbrechen, so daß keine lebenswichtigen Bestandteile des Systems durch Überschreibung in Betriebsunfähigkeit geraten können.

Auf jeden Fall muß der eingelesene Teil als BASIC verfügbar sein (oder mindestens mit einem Druck auf eine Taste in BASIC umgesetzt werden). Im Falle einer nicht stimmenden 'checksum' oder anderer erzwungener Abbrechungen des Leseprozesses, muß auf dem Schirm eine Meldung erscheinen, daß das Eingelesene kein vollständiges und richtiges Programm ist (z.B. LOAD ERROR oder LESEFEHLER).

Ein gutes Leseprogramm reagiert folgendermaßen auf das ETX-Zeichen:

a) es schaut, ob das folgende Zeichen eine über 8 'bits' stimmende 'checksum' sein kann, und behält, daß eventuell eine Fehlermeldung nötig ist;
b) es schaut, ob danach tatsächlich ein Auslaufton folgt.

Wenn nicht, dann war ETX offensichtlich ein verstümmeltes Zeichen, und es kommt noch viel mehr. Das Einlesen geht als normal weiter. Wenn dann doch ein Auslaufton folgt, wird in Abhängigkeit von der 'checksum' wohl oder nicht eine Fehlermeldung gegeben. Der Recorder wird am Ende des Auslauftones stillgesetzt.

Das Leseprogramm sorgt in Wirklichkeit für zwei Dinge: erstens für das Lesen von Kassettensignalen und deren Umsetzung in 'bits' und 'bytes', und zweitens für das Umsetzen der eingelesenen Reihe ASCII-Zeichen in ein BASIC-Programm bzw. in Zeichencodes, so wie diese in dem betreffenden Computer erforderlich sind (beispielsweise: Computer, die nicht mit Kleinbuchstaben zurecht kommen, müssen hereinkommende Kleinbuchstaben in Großbuchstaben umsetzen. Das gleiche gilt für andere abweichende Codes. man denke nur an die Möglichkeit, daß durch Lesefehler Kontroll-Codes hereinkommen: Diese müssen ebenfalls abgefangen und unschädlich gemacht werden. Empfehlung: Man setze schädliche Kontrollzeichen in 'Gitterchen'(#) um).

Beide Aufgaben können mitunter teilweise erledigt werden, indem man bereits im System vorhandene Subroutinen (passen Sie dabei auf, falls von Ihrem Computer verschiedene Versionen bestehen!) aufruft.

Einige Computer können die von der Kassette hereinkommenden Signale über einen 'hardware-interrupt' verarbeiten, so daß das Umsetzen in BASIC anscheinend gleichzeitig geschieht. Andere Computer müssen diese zwei Aufgaben notgedrungen nacheinander erledigen, wobei die eingelesene Zeichenreihe erst im RAM gespeichert wird (hinter dem schon vorhandenen Programm!) und wonach erst in einer zweiten Phase die ersetzt werden.

Ungeachtet dessen, welches System verwendet wird, muß der Nutzer ständig auf dem Bildschirm verfolgen können, daß da was geschieht. Am schönsten ist es wohl, wenn die hereinkommenden Zeilen nacheinander über den Schirm (oder zumindest über die unterste Schirmzeile) 'scrollen', doch andere Lösungen sind auch möglich.

Dieses 'Folgenkönnen' gilt sowohl für die Lesephase als auch für die Umsetzphase (vgl. den nächsten Paragraphen).

Paragraph 5: BASICODE wegschreiben

Das BASICODE-Schreibprogramm schreibt das BASIC-Programm (ab Zeile 1000, wenn BASICODE-3, ab Zeile 0, wenn BASICODE-1) in BASICODE-Format nach Band.

Ebenso wie das Leseprogramm erfüllt auch das Schreibprogramm zwei Funktionen: einerseits das Umsetzen des Speicherinhalts in ASCII-Zeichen (durchweg ist das ganz einfach das LISTen des Programms, auf jeden Fall das Umsetzen von TOKENS in Zeichen), und als zweites das richtige Beschreiben des Bandes. Auch hier hängt es von der Hardware ab, ob das eine oder das andere in zwei Runden nacheinander kommen muß oder aber scheinbar zugleich geschieht durch die Unterbrechungsmöglichkeiten ('interruptfaciliteiten').

Wiederum gilt die Forderung, daß der Nutzer auf dem Schirm das Nötige geschehen sieht, wobei das 'summum' (der Gipfel) der Schönheit das mit-scrollen desjenigen ist, was gleichzeitig auf Band geschrieben wird.

Bei einem Schreibsystem in zwei Runden, also erst LISTen und dann Schreiben, kann ein Raumproblem entstehen, weil das gleiche Programm sowohl 'tokenized' als auch in ASCII aufbewahrt/gelagert wird. Das ist schlauer möglich, aber vielleicht arbeitet das in einigen Computern unerträglich langsam:

Es ist möglich, immer sobald das 'listing' einer Zeile gemacht ist, die ursprüngliche BASIC-Version von dieser Zeile zu entfernen. Sobald das 'listing' fertig ist, ist das BASIC-Programm verschwunden. Das 'listing' wird dann auf 'tape' geschrieben, und sobald das geschieht bzw. geschehen ist, wird das 'listing' wieder in BASIC umgesetzt mittels des betreffenden Teils aus dem Leseprogramm. Dieser 'truc' bringt einige Risiken mit sich und ist daher nur ausführbar, wenn sich die Kontrolle über den Computer ganz in den Händen des Programms befindet. Sollte der Nutzer das

Programm abbrechen wollen, so sorgt das Programm auf jeden Fall erst noch für die Wiederherstellung des ursprünglichen Programms.

Einige Forderungen werden ohne weiteres gestellt:

1 Das Signal, so wie es auf Band geschrieben wird, muß EXAKT der BASICODE-Norm genügen. Hierbei sei insbesondere an die 'timing' der Signale gedacht, sowohl zwischen den einzelnen 'bits' als auch zwischen den nächsten 'bytes' und auch im An- und Auslaufton. Desweiteren sei verwiesen auf die Tatsache, daß das MSB eines jeden 'byte' in der Praxis immer 1 ist, außer eventuell in der 'checksum'. Des weiteren die Reihenfolge der Zeichen in einem Programmblock: erst 5 Sekunden Anlaufton, dann STX ($82), dann CR ($8D), die 'listing' mit immer einem Zwischenraum zwischen Zeilennummer und Zeilentext, und am Ende einer jeden Zeile ein CR ($8D), nach dem letzten CR ein ETX ($83) und unmittelbar danach die 8-bit-checksum, der 1 Sekunde Auslaufton folgt.

2 Das Wegschreiben muß vom Nutzer abgebrochen werden können (STOP-Taste auf der Tastatur);

3 eine BASICODE-Zeile so wie diese auf dem Band angebracht wird, darf grundsätzlich nicht länger als 60 Zeichen sein (plus einem CR); somit dürfen zwischen zwei aufeinanderfolgenden CR prinzipiell höchstens 60 andere Zeichen stehen. Der einzige Grund, weshalb auf dem Band mehr als 60 Zeichen gesetzt werden können, kann derjenige sein, daß das Übersetuungsprogramm einen Zwischenraum oder mehrere Zwischenräume hinzugefügt hat, entsprechend dem, was nach Punkt 7 erwähnt wird.

4 Es dürfen unter keiner Bedingung andere ASCII-Codes darin vorkommen als Hex 20 (Zwischenraum) bis einschließlich Hex 7E. Auf dem Band werden es dann Hex C0 bis einschließlich Hex FE. Daneben kommt CR (Hex 8D) vor, Hex 82 zu Beginn und Hex 83 am Schluß. Nur die 'checksum',die über 8 bits den Exor all des Vorangegangenen bildet, kann alle 8-bit Code sein. Es sei erwähnt, daß die Buchstaben von BASIC-Variablen und die Buchstaben der BASIC-keywords immer einen ASCII-Code von Hex 41 vor dem A bis einschließlich Hex 5A (vor dem Z) haben müssen. Buchstaben von Hex 61 bis Hex 7A sind ausschließlich zwischen 'quotes' und in REM-Zeilen erlaubt;

5 Nach dem $82 zum Beginn folgt immer erst ein $8D (CR). Jede Zeilennummer wird von dem eigentlichen Zeilentext getrennt durch einen Zwischenraum. Dieser Zwischenraum zählt durchaus mit bei der Kontrolle der Zeilenlänge der 60 Zeichen. Die CR am Ende und eventuell hinzugefügte Zwischenräume zählen nicht mit.

Um die Forderungen 2, 3 und 4 zu erfüllen, wäre zu empfehlen, das Schreibprogramm beim Wegschreiben in BASICODE-3 in zwei Phasen arbeiten zu

lassen: In beiden Phasen wird das BASIC-Programm ausgelesen und umgesetzt in ASCII. Dabei wird je Zeichen und je Zeile die Zahl der erlaubten Zeichen und die Zeilenlänge kontrolliert. Sollte ein Verstoß gegen eine dieser Regeln festgestellt werden, so wird der Wegschreibprozeß abgebrochen mit einer Fehlermeldung, die angibt, in welcher Zeile welcher Fehler entdeckt worden ist.

Phase 1 und Phase 2 sind in dieser Hinsicht identisch.

Der Unterschied besteht darin, daß in Phase 1 nichts nach Band geschrieben wird und in Phase gerade doch. Phase 1 verläuft somit ziemlich schnell und stoppt bei einem Fehler. Der Nutzer behebt diesen und startet aufs neue. Erst wenn in Phase 1 keine Fehler entdeckt werden, wird sofort danach von selbst Phase 2 ausgeführt. Darin kann dann natürlich kein Stop als Folge einer Fehlermeldung mehr vorkomen, so daß dann das Programm insgesamt gut auf das Band kommt (wenn nicht der Nutzer mit einer Stopp-Taste das Schreiben unterbricht).

Beim Wegschreiben gemäß dem BASICODE-1-Format wird Phase 1 nicht ausgeführt und wird ohne jegliche Kontrolle nur Phase 2 erledigt. Auf diese Weise kann man BASICODE-1 für etwas mißbrauchen, wofür es ursprünglich nicht bestimmt war, nämlich für die Übertragung eines Programmtextes in einen anderen Computer, ungeachtet dessen, was da genau in diesem Text steht.

In Anbetracht dessen, daß Kassettenlesezeit ein rarer Artikel ist, ist es erwünscht, daß das Schreibprogramm alle Zwischenräume - außer zwischen "" und nach REM - entfernt. Es muß ein Zwischenraum eingefügt werden, und zwar vor den BASIC-Kommandos GOTO, THEN, GOSUB, TO und STEP, sofern solch ein Kommando nicht das erste einer Zeile ist oder unmittelbar einem Doppelpunkt folgt (:).

Nachbemerkung: Es ist somit kein Zwischenraum erforderlich für AND und OR.

Diesen muß in BASICODE 3 immerhin ein Häkchen vorangehen und folgen. Aus diesem Grunde hinzugefügte Zwischenräume zählen nicht mit für die maximale Zeilenlänge von 60 Zeichen.

Eine Kontrolle über erlaubte BASIC-Kommandos ist fein. Insbesondere Instruktionen, wie POKE, USR oder CALL (Instruktionen, die somit bei falschem Gebrauch den Computer auf den Grund laufen lassen) sollten eigentlich durch das Schreibprogramm messerscharf abgelehnt werden. In einem REM-Auftrag oder als Text zwischen "" nach beispielsweise PRINT kann natürlich wohl so etwas wie POKE stehen (z.B. in einem Poker-Programm). Es wäre nicht schön, wenn das Schreibprogramm auch daraufhin anhalten würde.

Paragraph 6: Polarität, Tondetektion und Anlaufdetektion

6.1 Schreiben auf und Lesen von der Kassette

Es erweist sich, daß Schreiben und Lesen ausgezeichnet geht mit einem sehr einfachen 'interface'. Alles, was benötigt wird, um zu schreiben, ist ein Drähtchen/Fädchen, auf dem die Spannung bei der Steuerung des Schreibprogramms hoch und niedrig gehalten werden kann.

Mit einem geeigneten Serienwiderstand läßt sich das Signal abschwächen bis zum Stand des Mikrophoneingangs.

Oft ist der Standard-Kassetteneingang auch schon so gemacht. Bei Computern ohne geeigneten Kassetteneingang läßt sich meistens mit wenig Mühe etwas finden, was sich geeignet machen läßt: ein 'game-output', und ein einziger Draht von einem parallelen 'output', beispielsweise dem 'printer' oder einem der begleitenden Drähte des V24-Eingangs.

Auch der 'input' kann sehr einfach sein. Das Program muß feststellen können, ob die Spannung auf einem Draht niedrig oder hoch ist. Der Lautsprecherausgang eines Kassettenrecorders gibt auf einem durchschnittlichen Stand/Niveau 100 mV ab, wenn er hart steht, mehr als 1 Volt. Diese (Wechsel-) Spannung reicht aus, um einen TTL-Eingang oder einen V24-Eingang anzusteuern. Zwar ist es dabei nötig, den Gleichspannungsstand mit einigen Widerständen so einzustellen, daß der Eingang genau "auf der Grenze zwischen 0 und 1" steht. Wer einen Eindruck davon bekommen möchte, wie so etwas möglich ist, sehe nur mal im BASICODE-3-Buch unter P2000 nach. Dort wird unter anderem so beim 'printerconnector' verfahren.

Was nicht ohne weiteres gebraucht werden kann, ist der Daten-Ein- und Ausgang eines serienmäßigen Eingangs.

6.2. Programmtips für die Schreibroutine

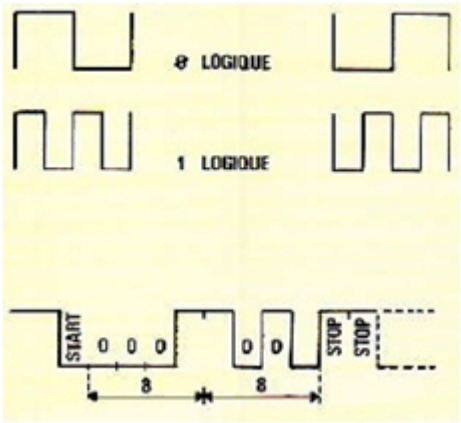
- Man versuche, das 'timing' der Wellenform so genau wie nur möglich zu machen.
- Man kalkuliere auch die Zeit ein, die nötig ist, um das nächste Zeichen aus dem Speicher heranzuholen.
- Man sei gefaßt auf eine eventuelle Systemunterbrechnung (interrupt); man schalte diesen/diese aus, wenn er/es das 'timing' stören kann.
- Man sei gefaßt auf 'wait-cycles' des Prozessors; einige Systeme fügen diese selbständig hinzu.
- BASICODE ist unabhängig von der Polarität. Es ist daher erwünscht, die Routine so zu machen, daß er wegzuschreibende Programme entweder 'random' oder abwechselnd in die eine oder in die andere Polarität schreibt (man vergleiche Seite 107 im Buch).

- Schreibroutine muß grundsätzlich zu jedem Zeitpunkt vom Nutzer abgebrochen werden können.

6.3 Programmiertips für die Leseroutine

- Man ermittle sowohl die aufwärts gehenden Flanken (0 auf 1) als auch die abwärts gehenden Flanken (1 auf 0). Bei 1200 Hz hat man dann 2400 mal je Sekunde Information über die Tonhöhe, und die braucht man ja auch.
- Man berechne immer die Zeit zwischen 2 übereinstimmenden Flanken (durch die zwei letzten Zeiten miteinander zu addieren). Es kann immerhin ganz leicht geschehen, daß die Wellenform symmetrisch wird. Die positiven Impulse werden dann kürzer als die negativen, oder umgekehrt, jedoch zusammen sind sie immer gleich lang. Man monitoriere somit immer die gemeinsame Zeit der beiden letzten halben Perioden.
- Man halte beim Lesen gesondert die Zeit bei, um die 'bits' einzunehmen. Man vertraue also nicht auf die Beziehung 'bits'-Wellenform. Bei Störungen und 'dropouts' im Band hält man so besser Schritt mit den Zeichen.
- Man warte nach dem Ende eines Zeichens wiederum auf einen 'startbit', unabhängig von der Polarität. Man vertraue nicht auf die 8 Perioden.
- Man mache ein 'timeout' beim Warten auf den 'startbit'.
- Man wähle die 'Bit-Detektion' so, daß eine halbe Periode von 2400 Hz plus einer halben Periode von 1200 Hz als 0 wiedererkannt wird.
- Man mache eine Anlauf-Tondetektion, in der 2 Sekunden hintereinander ein Ton von 2400 Hz +/- 10% detektiert wird. Das verhindert, daß man mit Sprecheinlesen und Musik beginnt.
- Die Leseroutine muß zu jeder Zeit vom Nutzer abgebrochen werden können.

6.4 Graphische Übersicht über den Lesealgorithmus



1 Man warte auf den 'startbit'
2 Man warte 1 1/2 bit
3 Man warte 1 bit
4 feste Zeit!
5 feste Zeit!
6 ausgelesene Zeit
7 detektierter bit

t ist die Zeit zwischen zwei Nulldurchgängen
T ist die Zeit zwischen den letzten beiden übereinstimmenden Nulldurchgängen
Das Messen und Beibehalten von T ist ein frei laufender Prozeß.

Das Warten auf einen 'startbit' bedeutet, daß gewartet wird, bis T einen bestimmten Wert Überschreitet, im obigen Beispiel z.B. 10. Danach in festen Zeitabständen nachsehen, was T ist, und auf diese Weise die 'bits' zusammenbringen. Man achte

darauf, daß die 'sample'- Momente am Ende der 'bits' liegen. Dann erst ist die Information ausreichend stabilisiert.

Wenn Sie das Leseprogramm so machen, daß ein 'startbit' dadurch entdeckt wird, daß hintereinander vier Zyklen von 2400 Hz, gefolgt von einem Zyklus von 1200 Hz, vorbeikommen, so wird erreicht, daß man nach einer Signalstörung am schnellsten wieder in die Synchronisierung gelangt.

Es ist zu empfehlen, in das Übersetzungsprogramm eine klare Copyright-Angabe aufzunehmen. Außerdem wäre es gut, an einer oder an mehreren Stellen im Übersetzungsprogramm eine verborgene Identifizierungsmöglichkeit aufzunehmen, an der bei einer gesetzwidrigen Kopierung festgestellt werden kann, daß es tatsächlich Ihr Programm ist.
Sie werden gebeten, eventuelle Fehler im Übersetzungsprogramm nicht nur zu verbessern, sondern auch die Leitung der Stiftung über diese Fehler zu informieren.

- Ende des Textes BASICODE -

F.d.R.z.(Seite 17 BIS 27):
Friedrich Henlich
Hübnerstr.26
Dresden 8027