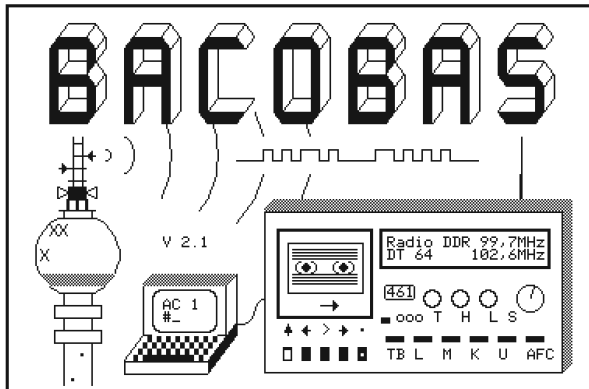


BACOBAS 3.0_{USB}

Was ist BACOBAS?

BACOBAS ist ein BASIC-Interpreter für den AC1, der "eingebaute" Mechanismen für die Arbeit mit BASICODE besitzt. Zusätzliche "BASCODER"-Programme wie bei anderen Rechnern sind damit nicht nötig. In erster Linie ist der Interpreter für BASICODE gedacht; es kann damit aber auch "normal" unter BASIC gearbeitet werden.



Ladebild BACOBAS 2.1
(erscheint nur beim Laden von Kassette)

Das Original:

„BACOBAS“ - AC1-BASCODER 2.1

F.Heyder & B.Nickel

Quelle: BASICODE-Schallplatte

Warum neue Version?

BASICODE ist ohne Zweifel "aus der Mode". Es besteht eigentlich keine Notwendigkeit mehr, es einzusetzen. Der ursprüngliche Zweck (standardisierter Programmaustausch) wird heute mit dem Medium Internet wesentlich besser erfüllt. Aber das Befassen mit BASICODE bewahrt ein Stück Geschichte, macht nicht dümmer und man kann sich mal ein damaliges originales BASICODE-Programm "reinziehen" und staunen ☺

BACOBAS2.1 weist einige Unzulänglichkeiten auf, die mit einer neuen Version beseitigt werden:

- x Dem damaligen Stand der Technik entsprechend wurden ein "Tondatenstream" und als Massenspeicher hauptsächlich die Kassettenaufzeichnung benutzt. Heute plagt sich kaum noch jemand damit herum. BASICODE-Programme sind im Internet jedoch als ASCII-Textdateien verfügbar.
- x Diese Version ist an den originalen AC1-Monitor "3.1" gebunden. Unter neueren Monitoren gibt es Probleme.
- x Es sind keine Pseudografikzeichen ($\geq 80h$) möglich.
- x Im Vergleich zum bekannteren und weitläufig benutzten "Grafik-Sound-BASIC" (E.Ludwig, SCCH) ist bei manchen Befehlen eine andere Syntax nötig und es werden andere Token benutzt.

Änderungen gegenüber Version 2.1:

- ✓ Anpassung an neuere Monitore und Zeichensätze
- ✓ USB- statt Kassettenschnittstelle
- ✓ Erweiterung/Modifizierung der Systemzeilen, z.B.:
 - Aufnahme GOSUB 500/540/560/580 (sequenzielle Dateiarbeit)
 - bei GOSUB 620, 630, 650 ist CN nun wirksam: CN=1 => Zeichnen, CN=0 => Löschen
 - schnelles Linienzeichnen mit Maschinencode
- ✓ Kommando-/Tokenangleichungen an GS-BASIC
- ✓ Implementierung der F-Tasten
- ✓ eingebauter Einfach-Druckertreiber (monitorbasiert)
- ✓ die Erweiterungen erfordern die Ausdehnung von 12kB auf 16kB (2000h...5FFFh)

Inhaltsverzeichnis

Programmstart.....	3
USB-Schnittstelle.....	3
Dateinamen und -besonderheiten.....	3
Laden und Sichern	4
Inhaltsverzeichnis anzeigen.....	4
Verzeichnis wechseln.....	4
USB-Fehlerbehandlung.....	4
Arbeit in BASICODE.....	5
Laden und Ausführen eines BASICODE-Programms.....	6
<i>ASCII-Format</i>	6
<i>BAC-Format</i>	6
Speichern eines BASICODE-Programms.....	7
<i>ASCII-Format</i>	7
<i>BAC-Format</i>	7
Listen/Drucken eines BASICODE-Programms.....	7
Sequenzielle Dateiarbeit.....	8
Arbeit im Normalmodus.....	13
Laden und Sichern von Programmen.....	13
Dateiarbeit im Normalmodus.....	14
Anhang 1.....	15
Kommandoangleichungen.....	15
Tastenbelegung.....	15
Tokentabelle.....	16
Zeichensätze/Kursorsymbole.....	17
Eingebauter Druckertreiber.....	18
Adressen.....	18
Hinweis zum GS-BASIC-Import.....	19
Anhang 2.....	20
Was ist BASICODE?.....	20
BASICODE-Standardroutinen.....	22
Anpassungshinweise für Programme.....	24
Anhang 3.....	25
Literatur und Internetlinks.....	25

Programmstart

BACOBAS 3 wird nach 2000h geladen (nicht EPROM-fähig) und ab dort gestartet:

J 2000 Kaltstart.

```
BACOBAS 3.0 USB
Max. RAM : ?
40646 Bytes frei
'HELP' : Neues
ok
>
```

Es wird die höchste von BASIC zu verwendenden RAM-Zelle abgefragt. Sinnvolle dezimale Endadressen liegen im Bereich von ca. 25000 (= 111 Bytes frei) bis ca. 65529 (= 40640 Bytes frei). Nur <ENTER> verwendet den aktuell verfügbaren Maximalwert. Man befindet sich nun im Normalmodus.

J 2003 Warmstart. Darf nur verwendet werden, wenn zuvor ein Kaltstart stattgefunden hat! Es wird der Zustand vor dem Verlassen wiederhergestellt.

Grundkommandos in BACOBAS 3:

- HELP** Das Kommando zeigt die wesentlichen Neuerungen gegenüber BACOBAS 2.1
- BACO** Der BASICODE-Modus wird aufgerufen
- NEW** Der BASICODE-Modus wird beendet, Rückkehr zum Normalmodus
- BYE** BACOBAS3 wird verlassen.

USB-Schnittstelle

Dateinamen und -besonderheiten

- Dateinamen zwingend in Gänsefüßchen
- Länge max. 8 Zeichen (längere Namen werden abgeschnitten)
- nur Großbuchstaben, Ziffern und einige Sonderzeichen
- keine Umlaute, keine Leerzeichen im Dateinamen
- Dateierweiterungen sind prinzipiell nicht mit anzugeben, sie werden automatisch aus der Art des Lade-/Speicher-Befehls bestimmt. Es wurde definiert:

BAS	BASIC-Programme im Normalmodus, tokenisiert ab 60F7h, <u>ohne Systemzellen!</u>	binär, letztes Zeichen in der Datei ist die 3. Null (Programmende)
BAC	BASICODE-Programme tokenisiert <u>ohne Systemzellen</u> und Bascoderzeilen	
ASC	BASICODE-Programme im ASCII-Format	blanker Text (Zeilenende=0Dh)
CSV	Daten im sequenziellen Dateimodus	strukturiert, siehe Elemente

- Der erste USB-Zugriff nach Start von BACOBAS3 initialisiert die USB-PIO und testet das VINCULUM incl. angeschlossenen USB-Medium. Daher dauert dieser etwas länger als alle folgenden Zugriffe.

Laden und Sichern

Je nach aktuellem Modus wird bei LOAD/SAVE die Endung „BAS“ (normale Basic-Datei) oder „BAC“ (BASICODE-Datei) automatisch verwendet. Beide Dateiformate sind nur unter BACOBAS3 verwendbar!

- **LOAD** „NAME“ Laden BASIC-Datei bzw. tokenisierte BASICODE-Datei
- **SAVE** „NAME“ Sichern einer BASIC-Datei bzw. tokenisierte BASICODE-Datei

ASCII-BASICODE-Programme sind direkt lad- und speicherbar. Es wird hier die Endung „ASC“ automatisch verwendet:

- **ALOAD** „NAME“ Laden ASCII-BASICODE-Programm
- **ASAVE** „NAME“ Sichern ASCII-BASICODE-Programm

Eine MERGE-Funktion ist möglich. Im Normalfall ist daher vor dem Laden eines neuen Programms das alte mit **NEW** zu löschen. Beim Hinzuladen ist auf höhere Zeilennummern zu achten, und es darf "0 REM" bzw. "1001 REM" nicht vergessen werden!

Inhaltsverzeichnis anzeigen

DIR als Direktkommando zeigt sowohl im Normal- als auch BASICODE-Modus die auf dem USB-Medium vorhandenen Dateien an. Eine Einschränkung der Anzeige auf bestimmte Dateiformate oder Namensbestandteile (Dateimaske) ist aktuell nicht möglich.

Verzeichnis wechseln

Mit **CD** lässt sich das aktuell benutzte USB-Verzeichnis einstellen:

- CD „name“ Wechsel in ein Unterverzeichnis
- CD „.“ eine Ebene hoch
- CD ohne Parameter: INIT USB-PIO und Wechsel ins root-Verzeichnis

Bei erstmaliger Verwendung einer USB-Anweisung befindet man sich im root-Verzeichnis!

USB-Fehlerbehandlung

- Ist beim Aufruf von USB-Befehlen (noch) kein Medium vorhanden, so wird **"ND"** ausgegeben (1x oder endlose Folge). Das ist mit Strg+C abubrechen. Nach Anschluss des USB-Mediums kann der Befehl (hoffentlich) erfolgreich ausgeführt werden.
- Treten beim Schreiben oder Lesen ein Fehler auf (z.B. wegen eines nicht zulässigen Umlauts im Dateinamen), so erfolgt eine Meldung **„USB Fehler“** und der Vorgang wird abgebrochen.
- Wird versucht, eine Datei zu laden, die auf dem USB-Medium nicht existiert, so erfolgt die Meldung **„Datei nicht gefunden“**. Das passiert meist durch einen „Verschreiber“ beim LOAD-Befehl. Also bei Bedarf erst mal mit „DIR“ das Inhaltsverzeichnis anzeigen lassen und dann den LOAD-Befehl mit dem korrekten Namen wiederholen...
- Soll eine Datei auf das USB-Medium geschrieben werden und es existiert bereits eine gleichnamige, so erfolgt eine Rückfrage: **Datei vorhanden, Ueberschreiben(J) ?**. Nur durch Bestätigung mit **„J“** wird das Überschreiben wirksam, ansonsten wird die Operation abgebrochen.
- Die Steuerung wird nach Ausführung der USB-Befehle zum Laden und Sichern an die Kommandoebene zurückgegeben. DIR und CD können auch innerhalb von Programmen benutzt werden; das Programm wird danach regulär fortgesetzt, soweit kein Fehler auftrat.

Arbeit in BASICODE

Mit dem Befehl **BACO** schaltet man in den BASICODE-Modus um. Ein evtl. vorher im Speicher befindliches BASIC-Programm wird **gelöscht!** Der Cursor wandelt sich vom weißen Vollfeld in ein graues Vollfeld. Wurde die Tastatur mit Strg+S auf „Schreibmaschine“ (mit Shift => Großbuchstaben) umgeschaltet, so ist es nur ein dünner Unterstrich.

Nun bestehen prinzipiell drei Möglichkeiten:

1. Laden eines BASICODE-Programms im ASCII-Format,
2. Laden eines BASICODE-Programms (tokenisiert) oder
3. Erstellung eines eigenen BASICODE-Programms.

Verlassen wird der BASICODE-Modus durch das Kommando **NEW**. **BACOBAS3** löscht das aktuelle Programm und kehrt damit in den Normalmodus zurück. Ersichtlich ist das am nun weißen Cursor.

In BASICODE ist nur ein Teil der im Normalmodus vorhandenen Befehle und Funktionen verwendbar (siehe [Anhang](#), erlaubte Anweisungen).

BASICODE benutzt original nur ASCII-Dateien zur Weitergabe. Zur Ausführung oder Weiterbearbeitung werden diese mit dem BASICODE umgewandelt. Die umgewandelte Form lässt sich dann für eigene Zwecke im computereigenen Format „BAC“ sichern und spart so zukünftiges Umwandeln. Bei einer beabsichtigten Weitergabe muss natürlich im „ASC“-Format abgespeichert werden. Kennzeichen beider Formen ist, dass sie prinzipiell mit Zeile 1000 beginnen.

Nur bei mit **BACO** eingeschalteten BASICODE-Modus sind folgende Anweisungen wirksam:

LOAD	Laden eines BASICODE-Programms im tokenisierten Format (BAC)
SAVE	Sichern eines BASICODE-Programms im tokenisierten Format (BAC)
ALOAD	Laden eines BASICODE-Programms im ASCII-Format (ASC)
ASAVE	Sichern eines BASICODE-Programms im ASCII-Format (ASC)
ALIST	Listen eines BASICODE-Programms im ASCII-Format Dabei handelt es sich die Ausgabe eines Listings auf dem Bildschirm analog zu LIST. Eine direkte Ausgabe auf den Drucker existiert jedoch nicht.
AEDIT	Editieren eines BASICODE-Programms im ASCII-Format Auch diese Anweisung ist vergleichbar mit der im Normalmodus. Es kann damit eine Programmzeile editiert werden. Zu beachten ist jedoch, dass der Aufruf mit AEDIT „zeilennummer“ erfolgt (Gänsefüßchen!)
TRANS	Konvertieren des ASCII-Format in tokenisiertes BASICODE-Programm Hiermit wird ein mit ALOAD eingelesenes BASICODE-Programm vom ASCII-Format ins tokenisierte BASIC-Format umgewandelt.
CONV	Konvertieren eines tokenisierten BASICODE-Programms in ASCII-FORMAT Das ist die Umkehrung zu TRANS.

Das Erstellen eines BASICODE-Programms ist etwas gewöhnungsbedürftig und bedarf weitergehender Kenntnisse. Ein [Crash-Kurs](#) im Anhang...

Nachfolgend wird der prinzipielle Umgang mit BASICODE anhand von Standardabläufen erläutert.

Laden und Ausführen eines BASICODE-Programms

ASCII-Format

- **ALOAD „TEST“** => lädt ein ASCII-BASICODE-Programm (TEST.ASC)
- Quittung des erfolgreichen Ladens mit "ASCII-File im Puffer" und Anzeige der möglichen Befehle, Kursorsymbol: "Herz"

Anmerkungen:

Der Befehl ist auch im Normalmodus möglich. Ein evtl. vorher im Speicher befindliches Programm wird gelöscht.

ASCII-Dateien haben nur ein "0Dh" als Zeilenende. Beim Laden werden aber auch Textdateien mit dem herkömmlichen "0D/0A"-Trenner akzeptiert! Am Ende einer Textdatei sollte immer ein 0D oder 0D/0A stehen.

Nicht wundern, wenn ein im „Umlauf befindliches“ ASCII-BASICODE-Programm nicht funktioniert. Ggf. ist trotz Standard noch Nacharbeit nötig (z.B. Tastaturcodes).

- **ALIST** => listet den ASCII-Puffer
- **AEDIT „1040“** => Editieren Zeile 1040 des ASCII-Puffers (beachte Gänsefüßchen!)
- **TRANS** => Übersetzen und Tokenisieren (Pflicht für Ausführen!)

Anmerkung:

In der letzten BS-Zeile wird der Übersetzungsvorgang angezeigt. Wurde der Vorgang erfolgreich beendet, so wandelt sich der "Herz"-Kursor zum grauen Vollfeld (bzw. dünnen Unterstrich) und es kann im BASICODE-Modus weitergearbeitet werden.

Das anschließende Sichern des BASICODE-Programms als BAC-Datei erspart ein erneutes Übersetzen der ASCII-Datei.

SAVE "name" => Sichern als BAC-Datei

- **LIST** => Kontrollanzeige
- **RUN** => Ausführen

BAC-Format

- **BACO** => Umschalten in den BASICODE-Modus (Kursor ändert sich)
- **LOAD "name"** => lädt die (tokenisierte) BASICODE-Datei name.BAC

Anmerkungen:

Existieren die Programme „TEST.BAS“ und „TEST.BAC“ gleichzeitig auf dem USB-Medium, so lädt LOAD automatisch das richtige, nämlich im BASICODE-Modus „TEST.BAC“.

Ein Hinzuladen (MERGE) ist ebenfalls möglich. Da wegen der Einschränkungen (Zeilen <1000 sind geschützt) ein „Neupointern“ durch Eingabe von **0 REM** nicht zulässig ist (? BC Error) muss statt dessen z.B. **1001 REM** eingegeben werden. Voraussetzung ist, dass das hinzuzuladende Programm höhere Zeilennummern als das schon vorhandene besitzt!

- **LIST** => Kontrollanzeige
- **RUN** => Ausführen des Programms

Speichern eines BASICODE-Programms

ASCII-Format

- Voraussetzung ist, dass sich eine ASCII-Datei im Puffer befindet. ("Herz"-Kursor). Sollte das noch nicht der Fall sein, so ist das zuvor geladene oder erstellte Programm zunächst in ASCII umzuwandeln:
- **CONV** => wandelt das aktuelle Programm in ein ASCII-BASICODE-Programm ("De-Tokenisieren")

Anmerkung:

Die Umwandlung erfolgt in zwei Läufen. Der Prozess kann am unteren Bildrand verfolgt werden. Treten dabei Fehler auf (z.B. zu lange Zeilen, Verwendung nicht zugelassener Befehle), so werden diese angezeigt. Ist nach einer entsprechenden Korrektur das Ergebnis fehlerfrei, so kann mit dem nächsten Schritt fortgesetzt werden.

- **ASAVE "TEST"** => sichert das BASICODE-Programm als ASCII-Datei unter „TEST.ASC“

BAC-Format

- Voraussetzung ist, dass man sich im BASICODE-Modus befindet und ein Programm im Speicher ist. Ggf. Kontrolle mit LIST.
- **SAVE "TEST"** => sichert BASICODE-Programm ab Zeile 1000 unter „TEST.BAC“

Anmerkung:

Das Sichern im BASICODE-Modus erfolgt ab Zeile 1000 als tokenisiertes Basic-Programm, welches nur für den AC1 unter BACOBAS3 geeignet ist. Damit wird ein erneuter Übersetzungsvorgang eingespart (z.B. für Weiterbearbeitung). Soll das Programm weitergegeben werden, ist ein Sichern als ASCII nötig!

Listen/Drucken eines BASICODE-Programms

LIST Anzeige Programmlisting auf dem Schirm,
Variante: LIST 10 mit Zeile 10 beginnend

ALIST dto. für ASCII-Text
Eine evtl. Zeilennummer muss hier in Gänsefüßchen stehen.

Bei Ausgabe längerer Listings wird nach einer bestimmten Zeilenzahl angehalten.

- Strg+C bricht die weitere Ausgabe ab
- alle anderen Tasten setzen fort

Die Zeilenanzahl, nach der angehalten wird, kann mit der Anweisung **LINES n** festgelegt werden.

LLIST Drucken des Programmlistings per V.24-Schnittstelle

Die mit LINES eingestellte Zeilenanzahl für einen Halt ist hier nicht wirksam, es kann jedoch mit Strg+C abgebrochen werden.

Ein Drucken des ASCII-Textes im Puffer (Kursor: "Herz") ist nicht möglich.

Sequenzielle Dateiarbeit

"BACOBAS 2.1" unterstützte keine Dateiarbeit. Mit dieser neuen Version "BACOBAS 3" wird diese nun möglich. Bei der Anpassung fertiger Originalprogramme an den AC1 sind jedoch einige rechnerspezifische Eigenarten zu beachten!

BASICODE-3-Standard

Die sequenzielle Dateiarbeit hat prinzipiell folgende Besonderheiten/ Einschränkungen:

- Eine Datei wird durch ihren Dateinamen gekennzeichnet.
- Sie besteht aus aneinandergereihten gleichartigen Elementen.
- Es kann nur der Reihe nach gelesen werden. Soll das n-te Element gelesen werden, so sind alle vorherigen n-1 Elemente auch zu lesen (als "Dummy").
- Geschrieben wird immer im Anschluss an das zuletzt geschriebene Element (bzw. ab Dateianfang). Es ist nicht möglich, nur das n-te Element einer sequenziellen Datei neu zu schreiben! Ist das erforderlich, so müssen alle Elemente in ein Array eingelesen, das gewünschte Element geändert und das Array komplett wieder neu in die Datei geschrieben werden.
- Wie mit den Elementen (z.B. Unterteilung eines Elements in einzelne Felder, erste Zeile = Überschrift etc.) umgegangen wird, das ist vom BASICODE-Programm zu organisieren.

Folgende Subroutinen werden beim Umgang mit sequenziellen Dateien verwendet:

- GOSUB 500 Öffnen einer Datei
- GOSUB 540 Lesen eines Elements aus der geöffneten Datei nach IN\$
- GOSUB 560 Schreiben eines Elements SR\$ in die geöffnete Datei
- GOSUB 580 Schließen einer Datei.

Generell gilt:

DIESE GOSUB NIE OHNE VORHERIGE FESTLEGUNG VON NF\$ UND NF AUFRUFEN!!!

Für das Öffnen werden vorher in der Variablen NF\$ der Name der Datei sowie in der Variablen NF der Öffnungs-Modus festgelegt. Die Öffnungsmodi NF=0 und NF=1 werden in (fast) allen bekannten Anwenderprogrammen mit sequenzieller Dateiarbeit benutzt. Sie bewirken original das Lesen bzw. (Neu-)Schreiben einer BASICODE-Datei als Ton-Datenstream (Kassette). Andere Öffnungsmodi und ihre Wirkungsweise hängen vom benutzten Rechnertyp ab und stehen daher nicht für jeden Rechner zur Verfügung.

Realisierung am AC1

Wie auch für das Laden und Speichern von BASICODE-Programmen wird die USB-Schnittstelle eingesetzt (VINCULUM). Alle mit o.a. GOSUBs zu schreibenden oder zu lesenden Dateien beziehen sich auf das USB-Medium als Ablageort.

Hinsichtlich der Öffnungsmodi wurde aus den o.a. Gründen für den AC1 definiert:

- Mit NF=0 bzw. NF=1 wird gelesen bzw. geschrieben. Auf eine Beibehaltung der historischen Wirkungsweise dieser Modi ("Kassettenbetrieb") wurde verzichtet.
- Die Modi NF=2 bzw. NF=3 wirken analog, sie wurden nur aus Kompatibilitätsgründen aufgenommen.
- Die Modi 4/5 sowie 6/7 werden wegen der in Anwenderprogrammen meist auf spezielle Rechner ausgelegte Besonderheiten nicht zugelassen (Fehlermeldung).
- Für experimentelle Zwecke wurde ein neuer Schreibmodus definiert. **NF=9** bewirkt, dass jetzt beim Schreiben immer angehängt wird.

Vergleich zur OPEN-Anweisung anderer BASIC-Dialekte:

NF=0 NF=2	INPUT	Datei wird zum Lesen geöffnet existiert die Datei nicht: Fehlermeldung (IN=-1)
NF=1 NF=3	OUTPUT	eine (neue) Datei wird zum Schreiben geöffnet. Existiert die Datei noch nicht, wird sie erstellt. Existiert sie schon, dann wird der Inhalt gelöscht!
NF=9	APPEND	Datei wird für Anfügen von Daten an existente Datei geöffnet. Existiert die Datei noch nicht, wird sie erstellt. Siehe Anmerkungen unter "Spezialmodus"!

Für die USB-Dateien sind folgende Regeln zu beachten:

- Dateierweiterung *.CSV für ASCII-Daten unter BACOBAS3
Diese Erweiterung ist in NF\$ nicht mit anzugeben; sie wird automatisch verwendet!
- Die Länge des Namens (ohne Erweiterung) darf max. 8 Zeichen betragen. Leerzeichen oder Umlaute sind nicht erlaubt. Der USB-Standard ("VINCULUM") lässt lediglich einige Sonderzeichen zu. Enthält ein Dateiname ein unzulässiges Zeichen, so wird das Öffnen einer Datei verweigert. Eine Ausnahme bilden Leerzeichen; diese werden automatisch entfernt!
- Es kann zu einem Zeitpunkt immer nur eine Datei geöffnet sein.
- Während einer geöffneten Datei ist kein Zugriff auf andere Dateien des USB-Sticks möglich, auch keine DIR-Anzeige!
- Die erzeugten CSV-Dateien können von EXCEL/OpenOffice gelesen werden!

Hinsichtlich der Elemente einer Datei gilt:

- Durch das Schreiben und Einlesen mit Stringvariablen (SR\$ bzw. IN\$) ist der Datentyp festgelegt. Der Inhalt der Variablen ist gemäß Standard grundsätzlich auf druckbare Zeichen beschränkt.
- Experimentell ist auch die Verwendung aller Codes $\geq 0Dh$ (also einige Steuerzeichen sowie Pseudografik) möglich!
- Die Länge der einzelnen Elemente kann unterschiedlich sein und bis zu 255 Bytes betragen (= max. Länge der Stringvariablen).
- Das Trennzeichen "0Ah" grenzt die einzelnen Elemente voneinander ab (es steht auch nach dem letztem Element).

GOSUB 500: ÖFFNEN

- Vor dem Aufruf von GOSUB 500 ist zu bestücken:
 - ➔ **NF\$**: Dateiname (ohne Dateierweiterung!). Anwenderprogramme sind dahingehend zu prüfen/anzupassen, dass der Dateiname max. 8 Zeichen lang ist und keine Leerzeichen oder Umlaute enthält! Wird ein ungültiger Dateiname angegeben, so schlägt das Öffnen fehl, wie auch bei Nichtexistenz der Datei.
 - ➔ **NF**: Modus 0/2=> Öffnen zum Lesen 1/3=> Öffnen zum Schreiben

Wird eine vorhandene Datei statt für Lesen versehentlich für das Schreiben geöffnet, so wird sie gelöscht und der ehemalige Inhalt ist verloren! Ausnahme: Spezialmodus NF=9.

- Rückmeldung: IN=0 fehlerfrei IN=-1 Fehler
- Position nach dem Öffnen: Dateianfang (Ausnahme: siehe Spezialmodus NF=9)
- Beispiel:

```
2000 NF$="TEST":NF=0:GOSUB 500: REM ÖFFNEN ZUM LESEN
2010 IF IN<>0 THEN 950: REM FEHLERTEST UND REAKTION
```

GOSUB 540: LESEN

- Jeder Aufruf von GOSUB540 liest der Reihe nach immer ein Element¹. Direkt nach dem Öffnen ist die Startposition des Lesens immer am Dateianfang. Ein Lesen vor der aktuellen Position ("Rückwärtslesen") ist nicht möglich.
- Das gelesene Element befindet sich anschließend in der Stringvariablen IN\$.
- Rückmeldung: IN=0: fehlerfrei, IN=-1: Fehler
- Wird versucht nach dem letzten Element noch etwas zu lesen, ist IN=1 und IN\$ ein leerer String. Soll der gesamte Inhalt einer Datei gelesen werden, so ist also ein GOSUB 540 mehrfach solange auszuführen, bis IN=1.
- Um ein bestimmtes Element einzulesen, muss man entweder dessen Position kennen oder die Elemente müssen eine eindeutige Kennung besitzen, auf welche der eingelesene String zu testen ist. In beiden Fällen muss jedoch mehrfach gelesen werden, bis das gewünschte Element erreicht ist.
- Beispiel:
2010 GOSUB 540: PRINT IN\$:REM LESEN+ANZEIGEN EINES ELEMENTS

GOSUB 560: SCHREIBEN

- Der zu schreibende Wert muss in der Variablen SR\$ abgelegt werden.
- GOSUB 560 schreibt ihn in die geöffnete Datei.
- Die erste Schreibposition ist der Dateianfang (Ausnahme: Spezialmodus NF=9)
- An das geschriebene Element wird automatisch immer der Trenner "0Ah" angehängt.
- Rückmeldung analog zum Lesen (IN=0: fehlerfrei, IN=-1: Fehler)
- Beispiel:
2010 SR\$="HALLO": GOSUB 560:REM SCHREIBEN EINES ELEMENTS

GOSUB 580: SCHLIEßEN

- **NF\$**: Dateiname, **IN**: Modus (beide Variablen enthalten den Wert beim Öffnen, sie dürfen zwischendrin nicht geändert werden!)
- Rückmeldung analog zum Lesen (IN=0: fehlerfrei, IN=-1: Fehler)
- Beispiel:
2020 GOSUB 580:REM SCHLIESSEN DER DATEI

Der in der **gelb markierten** Zeile beispielhaft angeführte Fehlertest empfiehlt sich nach jeder Dateioperation!

¹ Gelesen wird intern von USB zeichenweise. Wird das Trennzeichen "0Ah" erkannt, so ist der Lesebefehl (aktuelles Element) beendet. Beim Schreiben erfolgt intern nur ein Schreibvorgang mit LEN(SR\$) Bytes.

Spezialmodus NF=9

Dieser nur hier in AC1-BASICODE verfügbare Modus ist für experimentelle Zwecke vorgesehen. Da es sich um keinen "offiziellen" Modus handelt, sollten damit erstellte Anwenderprogramme nicht oder nur mit ausdrücklichem Hinweis weitergegeben werden!

Mit NF=9 wird die in NF\$ angegebene Datei für ein Anhängen neuer Elemente geöffnet. Unmittelbar nach dem Öffnen befindet sich der Dateizeiger am Dateiende. Damit können nun neue Elemente angehängt werden, ohne den bisherigen Inhalt erst in ein Array einzulesen.

Existiert die Datei noch nicht, so wird sie angelegt.

Es ist damit möglich, Dateien zu verwenden, die die Kapazitätsgrenze des im Normalfall für die Aufnahme der Elemente verwendeten Arrays überschreiten.

Das empfiehlt sich jedoch nur dann, wenn sie nicht mehr geändert werden müssen. Ein komplettes Einlesen zwecks Änderung ist ja dann wegen des begrenzten Array-Raumes meist nicht möglich..
Beispiel:

```
2000 NF$="TEST":NF=9:GOSUB 500: REM ÖFFNEN FÜR ANHÄNGENDES SCHREIBEN
2010 SR$="HALLO": GOSUB 560:REM SCHREIBEN EINES ELEMENTS
2020 GOSUB 580:REM DATEI SCHLIESZEN
```

Tipps & Tricks

- ➔ Originale BASICODE-Programme verwenden ggf. rechnerspezifische Besonderheiten. Daher empfiehlt es sich, von einem RUN zumindest die Programmteile für die Dateiarbeit zu untersuchen. Solche Besonderheiten werden regulär ab Programmzeile 20000 angeführt und sollten normalerweise dort auch dokumentiert sein.
- ➔ Achtung! Einige BASICODE-Programme erfragen beim Einlesen/Abspeichern im Modus NF=0/NF=1 (BASICODE-Kassette) keinen Dateinamen. In diesen Fällen muss das Programm entsprechend angepasst bzw. NF=2/3 verwendet werden!
- ➔ Das Programm ist für die Stringarbeit mit der Reservierung eines ausreichend großen Stringraums zu beginnen (z.B. 1000 A=1000:GOTO 20). Ansonsten kann es passieren, dass bei vielen/langen ein Strings ein "OS-Error" auftritt.
- ➔ Die Inhalte der Variablen NF\$ und NF dürfen zwischen Öffnen und Schließen nicht verändert werden. Der Inhalt der Rückmeldevariablen IN ist im BASICODE-Programm auszuwerten. Je nach Anforderungen ist entsprechend darauf zu reagieren.
- ➔ Es ist gerade in der Erstellungsphase von Programmen nicht auszuschließen, dass undefinierte Zustände beim USB-Lesen auftreten (Programmabsturz, Schließen einer Datei vergessen...). In diesen Fällen zeigt auch DIR nichts bzw. nur "Unsinn" an. Abhilfe schafft das Kommando "CD". Damit werden die USB-PIO und das VINCULUM neu initialisiert.
- ➔ Auch für den AC1 gibt es zur Dateiarbeit nichtstandardisierte rechnerspezifische Besonderheiten, die benutzt werden können. Damit erstellte Programme sollten aber nicht weitergegeben, zumindest aber gut ersichtlich dokumentiert werden.
 - Modus NF=9 (APPEND), siehe oben
 - Verwendung von nicht druckbaren Zeichen als Elementinhalt, siehe oben
 - Unterverzeichnis-Wechsel: Im Zusammenhang mit der ChangeDirectory-Anweisung der USB-Schnittstelle können die BASICODE-Daten auch in einem separaten Unterverzeichnis geführt werden. Beispiel:

```
1010 GOSUB20000:REM VERZEICHNIS FUER CSV-DATEIEN DEFINIEREN
...
9999 GOSUB20020:GOTO950:REM VOR ENDE ALTES VERZEICHNIS EINSTELLEN!
...
20000 CD"CSVDATEN":RETURN:REM AC1-BACOBAS3-SPEZIFISCH!
20020 CD"..":RETURN:      REM AC1-BACOBAS3-SPEZIFISCH!
```

Da der verwendete BASIC-Interpreter keine OPEN-, WRITE#- und INPUT#-Anweisungen besitzt, erfolgte die Realisierung der Unterprogramme komplett in Maschinencode. Die Systemzeilen 500-580 enthalten daher nur die entsprechenden CALLs:

```
500 IN=0:CALL*2010:RETURN
540 IN$="":CALL*2013:RETURN
560 CALL*2016:RETURN
580 CALL*2019:RETURN
```

Die CALLs lesen und schreiben die BASICODE-Variablen NF, NF\$, IN\$ und IN direkt aus dem Maschinencode heraus. Die Angabe von IN bzw. IN\$ in den Zeilen 500 und 540 ist nur nötig, damit die beiden Variablen ordnungsgemäß initialisiert und so auch gefunden werden können (NF, NF\$ und SR\$ werden vor dem GOSUB benutzt und sind daher schon "vorhanden").

Zusätzlich zur Rückmeldevariable IN=-1 werden Fehlermeldungen wie

- "Dateiname ungültig"
- "Dateiöffnen fehlgeschlagen (existiert nicht)"

und aktuell auch die Datei-Aktionen in der letzten Bildschirmzeile als Inverstext ausgegeben.

Die VINCULUM-Befehle setzen standardmäßig beim Dateiöffnen zum Schreiben den Dateizeiger immer an das Dateende. Um aber gemäß den BASICODE-Regeln ab Dateianfang zu schreiben wird eine ggf. bereits existierende gleichnamige Datei automatisch gelöscht. Danach wird eine neue Datei erstellt und somit ab Anfang geschrieben.

Normalerweise erfolgt beim Schreiben zunächst das Befüllen eines Ausgabe-Puffers. Dessen Inhalt wird erst beim Schließen in die Datei geschrieben. Darauf wird in BACOBAS3 verzichtet. Alle mit GOSUB 560 geschriebenen Elemente landen sofort in der Datei. Damit steht mehr Speicherplatz für ein Array zur Verfügung.

Arbeit im Normalmodus

Neben dem ursprünglichen Verwendungszweck der BASICODE-Arbeit kann man in BACOBAS3 auch im Normalmodus arbeiten. Er entspricht im wesentlichen dem von GS-BASIC:

- der Schlüsselwortsatz von GS-BASIC ist eine Untermenge von BACOBAS3
- gleiche verwendete Token
- gleicher Programmanfang ab #60F7
- gleiche Systemzellen für Programmende (#60D2)

Unterschiede zu GS-BASIC:

- Der Bereich 2000...3FFFh darf nicht für andere Zwecke (z.B. Maschinenprogramme) benutzt werden!
- Während GS-BASIC je nach Monitor und ggf. Zusatztools für die Programmlage Kassette, Diskette oder USB bzw. GIDE-Festplatte benutzt, besitzt BACOBAS3 nur ein USB-Interface.
- Es ist kein „fullscreen“-Editor vorhanden. Um einzelne Zeilen zu bearbeiten, müssen sie mit EDIT aufgerufen werden. Der Cursor ist nur in der aktuellen Kommandozeile beweglich. Ein „Anführungszeichen-Modus“ wie in GS-BASIC existiert nicht.
- Steuerzeichen in PRINT-Anweisungen können nicht direkt eingegeben werden. Ist deren Verwendung nötig, so ist die PRINT CHR\$-Funktion zu benutzen. Sind in importierten GS-BASIC-Programmen Steuerzeichen in PRINT-Anweisungen enthalten, so werden sie aber korrekt ausgeführt und bei LIST angezeigt (als Grafikzeichen mit Code+E0h).
- Eine vorhandene Grafiktaste wird immer ausgewertet. Ist sie gedrückt, so wird jeder Tastaturcode im Bereich 20h...7Fh (Leertaste bis Unterstrich) als Grafikzeichen interpretiert und auf den Schirm gebracht (nicht nur in PRINT-Anweisungen).
- Abbruchtaste ist (neben NMI) immer **STRG+C**. (GS-BASIC hat Strg+R, Strg+S).

Erweiterungen:

- Drucken: LPRINT und LLIST
- LOAD/SAVE für Laden und Sichern von Programmen
- DIR für Verzeichnisanzeige
- CD für Verzeichniswechsel
- schnelle Linienroutine DRAW (TO)
- sequenzielle Dateiarbeit mit Systemcalls

Programme aus GS-BASIC sind nach Import in den meisten Fällen sofort lauffähig. Ggf. müssen sie manuell geringfügig angepasst werden. Etwas mehr Arbeit ist insbesondere dann nötig, wenn sie Maschinencode im Bereich 2000h...3FFFh verwenden. Im umgekehrten Fall sind Programme natürlich nicht unter GS-BASIC lauffähig, wenn die neuen Befehle und Funktionen von BACOBAS3 genutzt werden.

Laden und Sichern von Programmen

Nach

- Kaltstart von BACOBAS
- NEW

befindet man sich im Normalmodus (Kennzeichen: weißer Vollkursor bzw. weißer Halbkursor im Schreibmaschinenmodus). Dann kann ein "normales" BASIC-Programm geladen oder ein vorhandenes gesichert werden:

LOAD "NAME"	Lädt das (tokenisierte) BASIC-Programm NAME.BAS von USB. Ein Hinzuladen ("MERGE") ist möglich.
SAVE "NAME"	Sichert das aktuelle BASIC-Programm unter NAME.BAS auf USB.

Dateiarbeit im Normalmodus

Dem zugrundeliegenden Standard-BASIC fehlen -wie bereits erklärt- Anweisungen wie OPEN, INPUT# und PRINT#. Unter Beachtung einiger Besonderheiten und Verwendung von System-CALLs ist dennoch im Normalmodus von BACOBAS3 die Arbeit mit sequenziellen Dateien möglich:

CALL*2010	Datei Öffnen, zuvor: - Dateiname in NF\$ ablegen - Modus definieren: NF=0 Lesen NF=1 Scheiben NF=9 Schreiben/Anhängen
CALL*2013	Element einlesen, steht dann in Variable IN\$
CALL*2016	Element aus Variable SR\$ in Datei schreiben
CALL*2019	Datei schließen

Prinzipiell sind wie in BASICODE immer 3 Schritte auszuführen:

1. Datei öffnen
2. Lesen oder Schreiben
3. Datei schließen

Die beim Öffnen verwendeten Werte von NF und NF\$ dürfen bis zum Schließen nicht verändert werden.

Alle CALLs geben in der Variablen IN eine Information über die ordnungsgemäße Ausführung zurück:

IN=0 alles o.k.

IN=-1 ein Fehler ist aufgetreten

IN=1 es wurde versucht, nach Dateiende etwas zu lesen ("EOF")

Beispiel:

```
10 REM DEMO SEQUENZIELLE DATEI
15 CLS
20 NF$="TEST":REM DER INTERNE DATEINAME LAUTET TEST.CSV
25 REM *** LESEN ***
30 NF=0:CALL*2010:REM DATEI ÖFFNEN ZUM LESEN
31 IF IN<>0 THEN GOTO 90: REM TEST AUF FEHLER UND ENTSPRECHENDE REAKTION
35 CALL*2013:PRINT IN$:REM ELEMENT LESEN UND ANZEIGEN
40 CALL*2019:REM DATEI SCHLIESSEN
45 REM *** NEU-SCHREIBEN ***
50 NF=1:CALL*2010:REM DATEI ÖFFNEN ZUM SCHREIBEN
55 SR$="HALLO":CALL*2016:REM WERT IN SR$ IN DATEI SCHREIBEN
60 CALL*2019:REM DATEI SCHLIESSEN
65 REM *** ANHÄNGEND SCHREIBEN ***
70 NF=9:CALL*2010
75 SR$="HALLO":CALL*2013
80 CALL*2019
85 END
90 PRINT "DATEI NICHT VORHANDEN!"
```

Der in der gelb markierten Zeile beispielhaft angeführte Fehlertest empfiehlt sich nach jeder Dateioperation!

Anhang 1

Kommandoangleichungen

Syntax/Anweisungen

Gegenüber BACOBAS 2.1 beginnen in BACOBAS3 alle **A**SCII-bezogenen Anweisungen nunmehr mit **A** (BLOAD => ALOAD,...). Das C für Cassette bei Lade-/Sicherungsanweisungen ist entfallen.

BACOBAS 3	BACOBAS 2.1	GS-BASIC
LOAD, SAVE	CLOAD, CSAVE	CLOAD, CSAVE
ALIST, AEDIT, ASAVE, ALOAD	BLIST, BEDIT, BSAVE, BLOAD	-
HELP, DIR, CD	-	-
DRAW, DRAWTO	-	-
BRON, BROFF, LPRINT, LLIST, BACO, CONV, TRANS, BEEP	-	-
-	-	CLOAD*, CSAVE*
EDIT	-	Onscreen-Editor

Tastenbelegung

Im Zusammenhang mit der Implementierung der F-Tasten wurden gegenüber BACOBAS 2.1 einige Steuertasten umdefiniert. Es gilt für BACOBAS3 folgende Tabelle:

Taste	Code		Wirkung
Strg+C	03h		allgemeines Abbruchkommando (Programmlauf, LIST, Editieren)
Strg+D	04h	DEL	Zeichen löschen, rechter Rest der Zeile rückt nach
Strg+E	05h	INS	Leerzeichen einfügen, rechter Rest der Zeile rückt nach
Strg+F	06h		Kursor an Zeilenanfang (im Editor hinter die Zeilennummer)
Strg+H	08h	←	Kursor nach links
Strg+I	09h	→	Kursor nach rechts
Strg+M	0Dh	ENTER	Abschluss des Direktkommandos (Im Editor der Bearbeitung)
Strg+O	0Fh	TAB	Kursor ans Zeilenende (im Editor zum Anfang nächster Befehl)
Strg+R	12h	BSP	rückwärts Zeichen löschen
Strg+S	13h		Umschaltung Computermode/Schreibmaschinenmode
Strg+T	14h	KEY1	LIST
Strg+U	15h	KEY2	RUN
Strg+V	16h	KEY3	LOAD“
Strg+W	17h	KEY4	SAVE“
Strg+X	18h	KEY5	DIR
Strg+Y	19h	KEY6	(unbelegt)
Strg+Z	1Ah	KEY7	(unbelegt)
ESC	1Bh	KEY8	(unbelegt)

BACOBAS3 lädt beim Kaltstart automatisch diese F-Tastenbelegung.
Mit der Anweisung **KEY** lässt sich die aktuelle Belegung anzeigen.

Tokenabelle

Alle Schlüsselworte von END (80h) bis TROFF (D8h) wurden GS-Basic-kompatibel gemacht und fehlende Anweisungen aufgenommen (z.B. MODE).

	8x	9x	Ax	Bx	Cx	Dx	Ex
0	END	OUT	LIST	^	EXP	SET	CONV
1	FOR	ON	CLEAR	AND	COS	RESET	TRANS
2	NEXT	NULL	LOAD*	OR	SIN	RENUMBER	ALIST
3	DATA	WAIT	SAVE*	>	TAN	LOCATE	AEDIT
4	INPUT	DEF	NEW	=	ATN	SOUND	ASAVE
5	DIM	POKE	TAB(<	PEEK	INKEY	ALOAD
6	READ	DOKE	TO	SGN	DEEK	MODE	DIR
7	LET	AUTO	FN	INT	POINT	TRON	CD
8	GOTO	LINES	SPC(ABS	LEN	TROFF	BEEP**
9	RUN	CLS	THEN	USR	STR\$	HELP	DRAW***
A	IF	WIDTH	NOT	FRE	VAL	EDIT	
B	RESTORE	BYE	STEP	INP	ASC	BRON	
C	GOSUB	KEY	+	POS	CHR\$	BROFF	
D	RETURN	CALL	-	SQR	LEFT\$	LPRINT	
E	REM	PRINT	*	RND	RIGHT\$	LLIST	
F	STOP	CONT	/	LN	MID\$	BACO	

identisch mit GS-BASIC

- *) benutzt USB statt Kassette
- **) wegen Kompatibilität mit GS-Basic (SOUND-Anweisung!) geändert:
Tonausgabe an PIO1/B0
- ***) für Beschleunigung der BASICODE-Linienfunktion (GOSUB 630)
auch im Normalmodus nutzbar:






DRAW x1,y1,x2,y2[,f]	x1,y1 = Koordinaten Anfangspunkt
	x2,y2 = Koordinaten Endpunkt

bzw. Linie ausgehend vom letzten Punkt zum nächsten Punkt weiter:

DRAWTO x2,y2[,f]	x2,y2 = Koordinaten nächster Punkt	
	ohne Parameter f bzw.	f=0 zeichnen
		f=1 Linie löschen

Zeichensätze/Kursorsymbole

Mit folgender geänderten Symbolik wird sowohl mit originalem Zeichensatz (ACC) als auch mit alternativem Zeichensatz (SCCH) eine eindeutige Zuordnung des Cursors zur Betriebsart erreicht:

	Normal-modus	Basicode-modus	ASCII geladen
Computer			
Schreibmaschine			

BACOBAS3 arbeitet nach Kaltstart wie GS-BASIC standardmäßig mit oberen Zeichensatz, sofern vorhanden (SCCH). Der Zeichensatz lässt sich einstellen:

MODE0 schaltet auf den oberen Zeichensatz (SCCH)
MODE wechselt zwischen beiden

Alle Grafikzeichen (>=80h) lassen sich mit `CHR$(x)` oder `POKE bildschirmadresse,x` ausgeben. Das Erscheinungsbild auf dem Schirm ist abhängig vom installierten Zeichensatz.

Eingebauter Druckertreiber

PUSH IX PUSH HL PUSH DE PUSH BC PUSH AF LD A,21h ;Ausgabe auch an Drucker LD (1821h),A ;E/A-Byte setzen POP AF RST 10h ;Zeichen an Schirm LD A,11h ;normal LD (1821h),A ;E/A-Byte setzen POP BC POP DE POP HL POP IX RET	<p>Soll ein eigener Druckertreiber verwendet werden, so ist dessen Anfangsadresse in 2007h einzutragen (siehe folgende Tabelle).</p> <p>Ist eine Drucker-Initialisierung nötig, so ist deren Anfangsadresse in 200Ah einzutragen.</p> <p>Bei Programmstart wird die Routine auf 200Ah immer aufgerufen. Standardmäßig ist sie mit MS30 belegt (der Minimaltreiber benötigt kein Druckerinit).</p>
---	---

Adressen

2000h	Sprung Kaltstart
2003h	Sprung Warmstart
2006h	Port-Nummer für Lautstärkesteuerung (Standard: 80h)
2007h	Druckerausgabe (Zeichen in A)
200Ah	Drucker-Initialisierung (Standard: MS30)
2010h	Datei öffnen
2013h	Datei lesen
2016h	Datei schreiben
2019h	Datei schließen
201Ah...204Fh	Arbeitszellen
5FEFh...5FFFh	Arbeitszellen
6000h...60F6h	Systemzellen BASIC, u.a.:
6004h	USR-Adresse
60D2h	Programmende
60D4h	Ende Variablenliste
60D6h	Ende Feldliste
60F7h	Beginn BASIC-Programm (bzw. BASICODE-Teil)

BACOBAS3 ist wie das Original nicht EPROM-fähig und daher für die Abarbeitung in den RAM zu laden (2000h...5FFFh).

Im Übrigen gelten die Angaben der originalen Anleitung (Buch mit BASICODE-Schallplatte).

Hinweis zum GS-BASIC-Import

BACOBAS3 speichert wie BACOBAS2.1 ein BASIC-Programm ohne Systemzellen. Ein GS-BASIC-Programm muss deshalb je nach vorliegendem Format bearbeitet werden:

Mit einem HEX-Editor am PC:

1. Eventuellen z80-Header entfernen
2. Systemzellen bis einschließlich 60F6h entfernen
3. Ende prüfen, letztes Byte muss die 3. Null sein, ggf. korrigieren!
4. Speichern mit der Endung BAS auf USB-Stick
5. Laden in BACOBAS3 mit LOAD

Alternativ kann man versuchen (keine Garantie!):

1. Kaltstart BASIC, mit BYE wieder verlassen
2. Laden des GS-BASIC-Programms nach 6000h (z.B. per DVU)
3. Warmstart BASIC
4. Abspeichern mit SAVE als *.BAS
5. Verlassen und Kaltstart von BASIC (andere Systemzellenbelegung!)

Liegt das GS-BASIC-Programm bereits ohne Systemzellen vor, so ist ein Umbenennen in *.BAS und Laden mit "LOAD" oft auch erfolgreich.

Die meisten GS-BASIC-Programme können so im Normalmodus ausgeführt werden. **Nicht bzw. nur mit Änderungen lauffähig sind Programme, die den Adressbereich 2000...3FFFh benutzen (z.B. für Maschinenprogramme, Zwischenablagen etc.)**

Was ist BASICODE?

Wer sich mit alten Computern befasst, kommt an BASICODE nicht vorbei. Anfang der 1980er Jahre gab es einen Boom in der Entwicklung des Programmaustauschs: die Übertragung per Radio! Die „öffentliche“ Übertragung war das eine - die Programme sollten aber auch universell einsetzbar sein, also auf möglichst vielen Computern. Dazu mussten Standards geschaffen werden. BASICODE als Kassettenspeicherformat wurde geboren und weiterentwickelt. Für die verschiedenen Kleincomputer wurden „Übersetzungsprogramme“ geschrieben, die das universelle Format in ein rechnerspezifisches umwandelten. Nun konnte man die gesendeten und auf Kassette aufgenommenen Programme auch mit dem eigenen Kleincomputer unter der Programmiersprache BASIC benutzen. Anfang der 1990er kam BASICODE aus der Mode; mit BTX und Internet gab es neue/bequemere Möglichkeiten des Programmaustauschs.

BASICODE-Programme zeichnen sich u.a. dadurch aus, dass nur ein standardisierter eingeschränkter Befehlssvorrat verfügbar ist. Anweisungen, die in verschiedenen BASIC-Dialekten unterschiedliche Bezeichnungen haben, wurden durch (maschinenspezifische) Subroutinen ersetzt.

Beispiel Löschen Bildschirm: Anstelle von „CLS“ ist „GOSUB 100“ zu verwenden.

Übersicht der erlaubten Anweisungen:

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

Funktionen und Operatoren:

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

AND	OR	NOT
------------	-----------	------------

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

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

Die Folge der Zeichen der Vergleichsoperatoren "<=", ">=" und "<>" ist vorgeschrieben.

Auch bei den Variablennamen bestehen Einschränkungen. Wie bei anderen Interpretern dieser Klasse (8k-Kern) sind lediglich zwei Zeichen des Namens signifikant. Generell verbotene Variablennamen sind:

AS, AT, DI, DI\$, DO, DO\$, DS, DS\$, EI, EI\$, EL, ER, FN, GO, GO\$, GR, HC, IF, LN, MA, MP, PI, SQ, SQ\$, ST, ST\$, TI, TI\$, TO, TO\$

Die Subroutinen verwendeten folgende Variablen, teilweise zu unterschiedlichen Zwecken.

A	Wert für Speicherreservierung mit CLEAR
CC	aktuelle Farbe im grafischen Betrieb
CN	Farbwert (AC1: 0/1) für grafische Ausgaben
CT	Formatierung numerischer Ausgaben, Unterscheidung Groß-/Kleinbuchstabe
FR	Größe Stringraum+freie Bytes
HG,VG	Grenzwerte Horizontal/Vertikal für grafische Ausgaben
HO,VE	HO(rizontal), VE(rtikal) Koordinaten für spalten-/zeilenweise Cursorpositionierung Textmodus: 0...63, 0...31, Grafik-Modus: 0...(1), 0...(1)
IN	Tastaturwert, Statuswert bei Dateioperationen
IN\$	Tastaturwert, Leseergebnis aus Datei
NF	Code für Dateioperation (Lesen/Schreiben)
NF\$	Dateiname
RV	Zufallswert
SD	Tondauer, Restzeit bei Tastaturabfrage
SP	Tonhöhe
SR	Bilden einer Stringvariablen
SR\$	Ergebnis Stringvariable
SV	Lautstärke

Neben diesen Festlegungen und Einschränkungen werden an den Programmaufbau bestimmte Forderungen gestellt:

- Ein Anwenderprogramm beginnt immer ab Zeile 1000. Diese hat einen festgelegten Aufbau und dient der Initialisierung des Systems:
`1000 A=50:GOTO 20:REM PROGRAMMNAME` (Wert für A kann variieren)
- Auch die Zeile 1010 muss existieren. Sie ist die erste „eigene“ Programmzeile.
- Ab Zeile 20000 stehen ggf. Subroutinen, welche rechner-spezifische (im Standard unerlaubte) Befehle verwenden.
- Ab Zeile 30000 stehen Erklärungen zum Programm,
- ab Zeile 32000 formale Fakten (Autor, Datum, Versionsnummer, verwendeter Rechner...).
- Die Zeilenlänge darf maximal 60 Zeichen betragen.
- u.a.m.

Die Arbeit unter BASICODE ist abhängig vom jeweiligen BASICODE-Programm und je nach Kleincomputer unterschiedlich.

Gemeinsam ist jedoch eine Eigenart: Am Anfang jedes auf dem Anwenderrechner lauffähig gemachten Basicode-Programms befindet sich ein (versteckter und nicht editierbarer) Bereich mit den Zeilennummern 10...950. Darin sind die rechnerbezogenen Subroutinen enthalten. Im BASICODE-3C-Standard sind sie wie folgt definiert:

BASICODE-Standardroutinen

GOSUB	vergleichbar	Zweck	Variablen
20	CLEAR	Programmstart, System-Reset, Variablen löschen	A
100	CLS	Beenden Grafikmode, Text-Modus einschalten und Bildschirm löschen.	-
110	LOCATE	Kursor-Position Spalte, Zeile setzen	HO,VE
120	POS	Kursor-Position holen	HO,VE
150	PRINT	String auffällig anzeigen	SR\$
200	INKEY\$	Daten einer eventuell gedrückten Taste holen	IN\$, IN
210	INKEY\$	wie 200, jedoch mit Warten auf Tastendruck.	IN
220	PEEK	Holen des Zeichens an Schirmposition	HO,VE=>IN
250	SOUND	Aufmerksamkeitston	-
260	RND	Zufallsvariable erzeugen ($0 \leq RV < 1$)	RV
270	FRE(x)	Garbage collection, Speicherplatzberechnung	FR
280	MODE	Aus- bzw. Einschalten der STOP/BRK - Taste	FR
300	STR\$	numerischen Wert in String wandeln	SR=>SR\$
310	PRINT USING	wie 300, jedoch mit CT und CN formatiert	SR,CT,CN
330	UPPER\$	Kleinbuchstaben in SR\$ in Großbuchstaben wandeln.	SR\$
350	LPRINT	Übergabe von SR\$ an den Drucker.	SR\$
360		Neue Zeile an Drucker (CRLF)	-
400	SOUND	Erzeugung eines Tons (Dauer, Höhe, Lautstärke)	SD,SP,SV
450	PAUSE n	Warten von maximal $SD * 0,1$ s auf einen Tastendruck.	SD
500	OPEN	Datei mit dem Namen NF\$ öffnen	NF\$, NF
540	INPUT#	Einlesen String in IN\$ aus der Datei	IN\$
560	PRINT#	String SR\$ wird in die Datei geschrieben	SR\$
580	CLOSE	Datei schließen	NF\$, NF
600	CLS	Graphischer Betrieb und Bildschirm löschen	-
620	PSET	Setzen Punkt an die Position HO, VE mit Farbe CN	HO,VE,CN
630	DRAWTO	Zeichnen Linie von akt. Kursorposition zum Punkt HO, VE in Farbe CN.	HO,VE,CN
650	TEXT	SR\$ an der Position HO, VE anzeigen (Grafik-Mode)	HO, VE,SR\$
950	END	Beenden des Basicode-Mode, Beenden Grafik-Mode	-

Besonderheiten AC1 BASICODE3:

- Grafischer Betrieb: Blockgrafik 128*64 Punkte
- GOSUB 150 gibt den "auffälligen" Text **invers** aus
- GOSUB 200/210 liefern auch bei den Kursortasten den ASCII-Code (nicht 28...31)
- Parameter SV (Lautstärke) wird berücksichtigt, benötigt jedoch Hardwareerweiterung!
- GOSUB 310: Zahl formatieren nicht voll realisierbar (wie bei Bacobas 2.1)
- GOSUB 620, 630, 650: CN (Farbe) entscheidet über Zeichnen/Löschen:
CN=1 bewirkt normale Ausgabe, CN=0 Löschen Text/Pixel/Linie

Die BASICODE-Systemzeilen

Beim Umschalten mit dem BACO-Kommando erzeugt BACOBAS3 folgende (geschützte) BASICODE-Systemzeilen:

```
10 GOTO1000:REM AC1-SUBROUTINEN V3.3
20 CLS:CLEARA:HO=63:VE=31:HG=96:VG=64:GOTO1010
100 CLS:RETURN
110 LOCATE(HO,VE):RETURN
120 HO=ABS(DEEK(6144)-6143):VE=0
121 HO=HO-64:VE=VE+1:IFHO>=0THEN121
122 HO=HO+64:VE=VE-1:RETURN
150 PRINTCHR$(17);"> ";SR$;" <";CHR$(16);:RETURN
200 IN=INP(4):IFIN>128THENIN=IN-128:IN$=CHR$(IN):RETURN
205 IN=0:IN$="":RETURN
210 IN=INP(4):IFIN>128THENIN=IN-128:IN$=CHR$(IN):GOTO212
211 GOTO210
212 POKE24561,IN
213 IN=INP(4):IFIN>0THEN213
214 IN=PEEK(24561):RETURN
220 POKE24567,HO:POKE24568,VE:CALL12305:IN=PEEK(24568):RETURN
250 BEEP:RETURN
260 RV=RND(1):RETURN
270 FR=FRE(A):IFFR<0THENFR=65535+FR:RETURN
280 IFFR=1THENBROFF:RETURN
281 IFFR=0THENBRON
282 RETURN
300 SR$=STR$(SR):IFSR>=0THENSR$=MID$(SR$,2)
301 RETURN
305 REM*GOSUB310 nicht voll realisierbar!*
310 SR$=STR$(SR):IFLEN(SR$)<=CTTHENRETURN
311 POKE24568,CN:SR$=""
312 FORCN=1TOCT:SR$=SR$+"*":NEXT
313 CN=PEEK(24568):RETURN
330 IFLEN(SR$)=0THENRETURN
331 POKE24568,CT:CT=LEN(SR$):POKE24567,SD:FORSD=1TOCT
332 POKE6400+SD,ASC(RIGHT$(SR$,CT-SD+1)):NEXT:SR$=""
333 FORSD=1TOCT:SR$=SR$+CHR$(PEEK(6400+SD)AND95):NEXT
334 CT=PEEK(24568):SD=PEEK(24567):RETURN
350 LPRINTSR$;:RETURN
360 LPRINT:RETURN
400 SD=INT(SD):IFSD<1THENS=1
401 BEEPSP,SD,SV:RETURN
450 POKE24568,INT(SD):CALL12308:IN=PEEK(24567):IN$=CHR$(IN)
451 SD=PEEK(24568):RETURN
500 IN=0:CALL*2010:RETURN
540 IN$="":CALL*2013:RETURN
560 CALL*2016:RETURN
580 CALL*2019:RETURN
600 CLS:RETURN
620 IF CN=0 THEN SET(INT(HO*95),INT(VE*63)):RETURN
621 RESET(INT(HO*95),INT(VE*63)):RETURN
630 DRAWTO INT(HO*95),INT(VE*63),CN:RETURN
650 LOCATE(INT(HO*63),INT(VE*31))
651 IF CN=0 THEN PRINT SR$:RETURN
652 FOR CN=LEN(SR$)TO1 STEP-1:PRINT " ";:NEXT:RETURN
950 END
```

Gelb markiert: Änderungen gegenüber Version 2.1

Diese Zeilen sind normalerweise mit LIST nicht zu sehen. Erst ein "LIST 10" macht sie sichtbar. Eine Veränderung der Zeilen mit EDIT ist nicht möglich.

Anpassungshinweise für Programme

BASICODE ist kein 100%iger Standard; es gibt zu viele Unterschiede zwischen den einzelnen Rechnertypen. Nicht immer sind Programme auf anderen Rechnern sofort lauffähig. Im folgenden werden einige "Problemstellen" aufgezeigt, die am AC1 auftreten können.

Tastencodes

Beim Aufruf der Subroutinen GOSUB 200 und GOSUB 210 ("INKEY\$") liefert BACOBAS 3 wie BACOBAS 2.1 den ASCII-Code der gedrückten Taste. Das ist aber nicht bei allen Rechnern der Fall. Teilweise geben Tasten einen abweichenden Code zurück. Das betrifft insbesondere die Kursortasten. In einigen Beispielprogrammen findet man Abfragen mit den Codes 28...31 wie folgt:

...	für den AC1 ist zu ändern:
1280 GOSUB 200:IF IN=0 THEN 1280	28 = Cursor nach links => 8
1290 IF IN<28 THEN 1280	29 = Cursor nach rechts=> 9
1300 IF IN>31 THEN 1280	30 = Cursor nach unten => 10
1310 Z=I:S=J	31 = Cursor nach oben => 11
1320 IF IN=28 THEN S=S-1	
1330 IF IN=29 THEN S=S+1	Nach "GOSUB 200" oder "GOSUB 210" immer
1340 IF IN=30 THEN Z=Z+1	prüfen, welche Codes erwartet werden und
1350 IF IN=31 THEN Z=Z-1	ggf. ändern. Das gilt auch für
...	Groß-/Kleinschreibung eingegebener Zeichen!

Grafischer Modus

Im grafischen Modus arbeitet BASICODE nicht mit absoluten Koordinaten sondern mit Werten 0...<1. Damit werden unterschiedliche Bildschirmauflösungen der verschiedenen Rechner berücksichtigt. Der Bascoder enthält dazu in den Variablen HG und VG die Maximalwerte des konkreten Rechners als Vorgaben. Am AC1 wird eine "Klötzchengrafik" von 128x64 "Pixeln" realisiert. Benutzt werden jedoch horizontal nur 96 von den 128 Pixeln. Damit erfolgt eine gewisse Angleichung im Aussehen von Grafiken auf verschiedenen Rechnern, aber der AC1-Bildschirm wird nicht komplett genutzt. Durch Modifizieren der Standardwerte für HG und VG oder von Parametern in der Berechnungsmethode von Pixelpositionen lässt sich das Aussehen am AC1 verändern/verbessern (damit ein Kreis auch "rund" wird...).

Infolge der Relativwerte der Positionsangaben sind auch Verschiebungen in der Bildschirmdarstellung möglich, wenn Grafikelemente und grafischer Text (GOSUB 650) zusammen platziert werden. Da nur eine Positionierung im 128x64 Raster möglich ist, werden die Relativwerte nach der Multiplikation mit dem Endwert gerundet. Das kann dazu führen, dass der Text nicht an der gewünschten Stelle in Bezug auf die Position der Grafikelemente erscheint. Als Abhilfe müssen die Positionsangaben (des Textes oder des Zeichnungselements) meist nur geringfügig geändert werden.

Ein weiteres Problem kann auftreten, wenn Programme die Position von Pixeln berechnen und sich dabei (aufgrund von Rundungs- oder anderen Fehlern) negative Werte ergeben. Diese führen bei Funktionen wie SET(x,y) zum Abbruch des Programms mit der Fehlermeldung "FC-Error". Abhilfe ist zu schaffen, indem vor dem Setzen des Pixels dessen Koordinaten geringfügig verändert werden. So ergibt z.B. bei x=-0,0001 die Verwendung von INT(x) einen kleinen positiven Wert, der dann im Wertebereich für SET als "0" interpretiert wird. Problem gelöst...

Farbe

Der Standard-AC1 kann nur 2 "Farben" darstellen (die Farbkarte des AC1-2010 wird nicht unterstützt). Der bei grafischen Ausgaben (GOSUB 620, 630 und 650) farbbestimmende BASICODE-Parameter CN hat also am AC1 nur zwei mögliche Werte:

CN=0 => Zeichnen in Hintergrundfarbe (= Löschen Pixel) und

CN=1 => Zeichnen in Vordergrundfarbe (= Setzen Pixel).

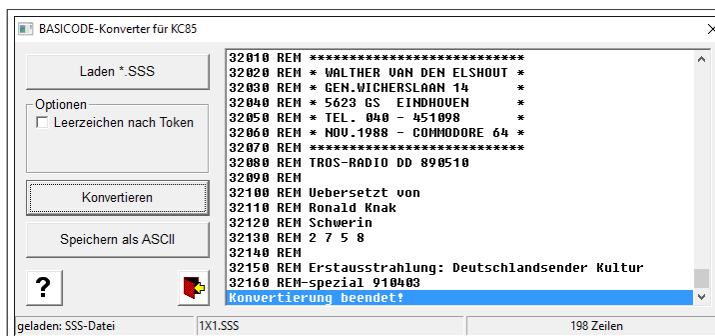
Im Einzelfall sind die CN-Werte daher ggf. anzupassen.

Literatur und Internetlinks

- [1] Völz, Horst: Basicode. Mit Programmen auf Schallplatte für Heimcomputer.
Verlag Technik, Berlin, 1990, 200 Seiten
AC1: S. 112-121
- [2] Völz, Horst: REM-Computermagazin "BASICODE"
Begleitmaterial zur Rundfunksendung, 22 Seiten
- [3] <https://de.wikipedia.org/wiki/BASICODE>
Grundlageninformationen
- [4] <http://www.basicode.de/>
Thomas Rademacher
(viele BASICODE-Programme im ASCII-Format)
- [5] <https://www.iee.et.tu-dresden.de/~kc-club/09/RUBRIK05.HTM>
(BASICODE-Programme überwiegend im *.SSS-Format des KC85-BASIC)

Achtung:

Die *.SSS-Dateien müssen erst nach ASCII umgewandelt werden. Wer keinen KC85 besitzt, der kann das z.B. mit JKCEMU oder meinem "KC85-BASICODE-Konverter" (unter Windows) tun:



Mit 3 Klicks wird ein SSS-Typ geladen, konvertiert, angezeigt und als ASC abgespeichert.

Lauffähig ab WinXP, auch Windows 10/ 64bit

Ist im Paket von BACOBAS 3 enthalten.

Erstellt: Rolf Weidlich, Stand: Dezember 2015

Anmerkung:

Der Entwickler von *BACOBAS 2.1*, Frank Heyder, wurde hinsichtlich meiner Weiterentwicklungen informiert. Bislang liegen keine Rückinformationen bzw. Einwände seinerseits vor.