CLUB 80

Clubinfo

der

TANDY - GENIE - und KONTEK - ANWENDER

14. AUSGABE

Kontaktadresse : CLUB 80 / Peter STEVENS / Postfach 56 / 4600 Dortmund 1

Tel.: 0231 / 553953

Inhaltsverzeichnis

Seite:

Clubinternes

1. Vorstandssitzung . . . . . 01 - 02
Nochmals Fragebogen 03
Vorstellung eines Mitgliedes 04
Terminkalender 05

Software

EDIT-Befehl 06
Programmierwettbewerb 08
Mein 1. ASSEMBLER-Programm . . . . . 09 - 16
ASSEMBLER-Routinen . . . . . 17 - 20
Mehr Fernsehen für's Geld . . . . . 21 - 22
ASCII-Tab 23
HRG-Bildübertragung . . . . . 24 - 28
Grafik-Programme . . . . . 29 - 38
Wenn die Uhr mal stört . . . . . 39 - 42
Scripsit auf dem "alten" Genie . . . 43 - 44
Scripsit-Tabellen . . . . . 45 - 46
Freiräume in Scripsit . . . . . 47 - 48
Textumleitung . . . . . 49 - 51
Nachtrag zu "Sondertasten" . . . . . 53 - 54
Unterschiede . . . . . 55 - 56
Knöpfe . . . . . 57 - 58
Aufgestiegen . . . . . 59 - 60
Tips M4/4p . . . . . 61 - 62
Boot-ROM des 4p . . . . . 63 - 65
HRG für Model 3/4/4p . . . . . 67 - 68
DOS - (fast) ohne Floppy . . . . . 69 - 75
Rechenzeitverkürzung . . . . . 77 - 85

Hardware

Floppytrick's . . . . . 87 - 88
HRG-1b . . . . . 89 - 90
256k RAM für Z80 . . . . . 91 - 99
CLUB 80 ECB-Bus-System . . . . . 101 -103

Seite:

Börse

Wer hat was ???--Wer sucht was ??? 105 -106
Wer kann weiterhelfen . . . . . 107 -108

Sonstiges

Buchbesprechung . . . . . 109 -110

Programmbibliothek

Einleitung 111
Neue Programme . . . . . 112 -113

Die letzten Seiten

Impressum 114
Schluß 115
Clubmitgliederadressen . . . . am INFO-Ende
Anmeldung ECB-Bus . . . . . am INFO-Ende

1. VORSTANDSSITZUNG

am Sonntag, 29. 06. 1986 bei KaJot
Anwesende waren: Eckehard Kuhn
Klaus-J. Mühlenbein
Jens Neueder
Hartmut Obermann

Mit dieser Zusammenfassung über die 1. Vorstandssitzung möchte der Vorstand die Mitglieder über seine laufende Arbeit informieren.

Priorität in unserer Tagesordnung hatte natürlich unsere Fragebogenauswertung. Leider kamen von 53 Fragebogen -nach sechs Wochen !- nur insgesammt 19 zur Auswertung an uns zürück. Die Auswertung der Fragebogen wurde aus diesem Grund verschoben. Die aus der Fragebogenauswertung resultierenden Punkte wie:

-- Mitgliederaktivierung
-- neuer Mitgliederstand (Geräte, Interessen usw.)
-- Interessengruppen

mußten aufgrund der geringen Daten auch unter den Tisch fallen. Das Thema "Fragebogen" wird wegen seiner Wichtigkeit an anderer Stelle im INFO nochmals aufgegriffen.

Weiterhin sprachen wir über die Mitgliederwerbung und haben uns für folgendes entschlossen.

1. Nochmalige Puplikation unseres Clubs in den einschlägigen Zeitschriften, da uns dies durch die "Vorstandsänderung" nichts kostet.
2. Interessenten an unserem Club bekommen ein Vorstellungsblatt (Darstellung unseres Clubs und Clubsatzung), sowie ein Anmeldeformular (wie der Fragebogen vom letzten INFO) zugesandt. Weiterhin wird auf die Möglichkeit hingewiesen, ein Probeinfo für DM 5,-- zu beziehen. Wer Mitglieder werben möchte oder Interessierte an unserem Club kennt, melde sich bitte wegen der nötigen Formulare bei der Redaktion.

Desweiteren hatten wir die Möglichkeit ein Public-Domain-Software-Paket in die "Diskothek" zu übernehmen. Es sind über 200 neue Programme aus den USA. Ihr werdet im nächsten INFO Ausführliches darüber erfahren.

Alsdann beschlossen wir wieder ein Sonderheft herauszubringen. Wir haben dazu mehrere Themen in der engeren Wahl: Modem-Sonderheft, Grafik M4/4p und ROM-Sonderheft Auch möchten wir Kurzbefehlslisten über BASIC und DOS erstellen. Zu beiden "Aktionen" ist Eure Mitarbeit erwünscht. Bitte meldet Euch dazu bei der Redaktion zur Koordinierung.

In Zukunft möchten wir auch in der INFO einen Veranstaltungskalender einbauen. Als Grundstock soll uns "TEDAS" vom Franzisverlag dienen. In dieser Mailbox sind sicher die aktuellsten Meldungen über die Computerei vorhanden. Wer zudem noch Termine bekanntzugeben hat (Regionaltreffen usw.) melde sich bei der Redaktion.

Als nächstes diskutierten wir darüber, mit einem Tandy-Club in den USA in Briefwechsel zu treten. Die Entscheidung dazu fiel positiv aus, da wir uns davon einige Vorteile erhoffen.

Weiterhin wird Hartmut einen Hardware-Grundlagen-Kurs abhalten. Wann, wo und wie wird bekannt gegeben.

Als vorletzten Punkt kam nun noch unser neues Club-Konto an die Reihe. Die neuen Daten wie folgt:

Postgirokonto	Peter STEVENS
	Sonderkonto Computerclub
Konto-Nummer	285 491 - 465
Postgiroamt	Dortmund
BLZ	440 100 46

Da sich die "e.V. -Eintragung" noch etwas verzögert, wurde diese Möglichkeit gewählt. Nach der "e.V. -Eintragung" wird der Name des Inhabers in Club 88 e.V. geändert.

Abschließend entschlossen wir uns, innerhalb des Vorstandes, alle 14 Tage ein Rundschreiben kursieren zu lassen, damit neueste Informationen diskutiert werden können. Solltet Ihr also in Zukunft wichtige Neuigkeiten für die Mitglieder oder den Club betreffend haben, so ist jetzt die Möglichkeit gegeben, brandneue Informationen bei einem der Vorstandsmitglieder abzusetzen.

Die Redaktion.

FRAGEBOGEN

Wie schon vorher im INFO angeführt soll hier nochmals die Rede auf den Fragebogen Kommen.

Der Fragebogen ist dazu da, uns -Euren Vorstand- über Eure Wünsche, Interessen und Gerätekonfigurationen zu informieren. Nur durch die von uns gewünschten Informationen ist es uns erst möglich Euch ein aktuelles, interessantes und vor allem ansprechendes INFO zu bringen - deshalb der Fragebogen. Leider hat uns die Mehrzahl der Mitglieder im Stich gelassen. Nach sechs Wochen !!! kamen nur 19 Fragebögen von 53, die mit dem INFO verschickt wurden, zurück.

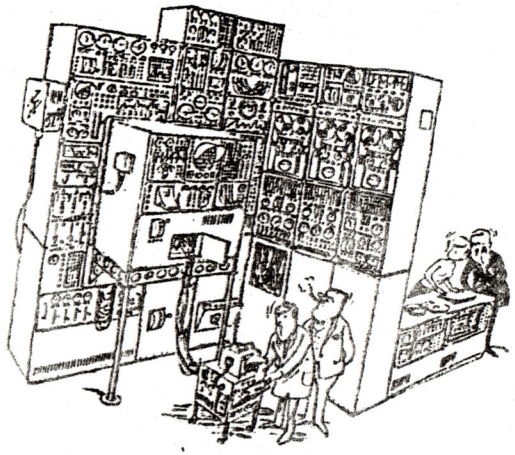
Ist es für Euch eigentlich so uninteressant, was in Eurer Club-INFO steht? Sicher nicht, oder ...?

Wir denken, daß diese Fragebogenaktion in Eurem Sinn ist und von allgemeinem Nutzen. Aus diesem Grund haben wir die Frist verlängert. Bitte seit so freundlich und sendet den Fragebogen umgehend zur Auswertung an die Redaktion.

In der Hoffnung, nun auch die fehlenden Fragebogen zurückzubekommen

Euer Vorstand

P S.: Wer seinen Fragebogen unauffindbar aufgeräumt haben sollte, kann von der Redaktion einen weiteren Fragebogen bekommen - Anruf genügt.



»Typisch unser Chef –
er läßt den Computer von einem Lügendetektor testen!«

Liebe Clubfreunde,

ich bin seit Februar diesen Jahres Mitglied im Club und moechte mich ebenso wie andere vor mir kurz vorstellen. Ich benutze ein Video-Genie System mit zwei Laufwerken, HRG-1B Grafik-Karte, 64K RAM und einer einfachen, von mir fuer den Drucker eingebauten RS-232c Schnittstelle. Als Drucker verwende ich einen H14 der Firma Heathkit. Er ist bereits reichlich aus der Mode, aber ein neuer kostet auch seine Mark und nur zum Papierschwaerzen erfuellt er seine Dienste bisher noch ausreichend. Mein bevorzugtes und einziges Betriebssystem ist Newdos80.

Was meine Aktivitaeten in Sachen computern betrifft, so erstelle ich meine kleinen Programm in Assembler, BASIC oder auch einmal in FORTRAN. Z. Zt. plage ich mich mit der Anpassung des RPNL-Compilers des Herrn Wostrack ab. Sollte sich Jemand im Club hier ebenfals schon einmal versucht haben, bin ich an einem Informationsaustausch interessiert. Der Compiler erscheint mir durchaus brauchbar, ermoeglicht er doch eine maschinennahe Programmierung in Assembler als auch die problembezogene Programmerstellung. Obwohl FORTH-Aehnlich, ist RPNL bei weitem nicht so kaotisch, da die Sprachstruktur an PASCAL angelehnt ist.

Tschuess

Kurt Mille

-- Termine -- Termine -- Termine --

An dieser Stelle unseres CLUB-INFO's möchten wir in Zukunft einen Terminkalender aufbauen.

Wir haben vor, hier über allgemeine Veranstaltungen der Computerei

-- Börsen, Treffen und Messen --

sowie über spezielle TRS-80-Termine zu berichten.
Auch die Termine unseres Clubs sollen hier bekanntgegeben werden.
Wir dachten dabei an:

Regionale Treffen, Stichtage von Aktionen usw.

Ihr als Mitglieder seit nun aufgefordert hierbei mitzuwirken.

Die ersten Termine:

Ende der ECB-Bus-Anmeldung	20. August 1986
Ende der Fragebogenaktion	29. August 1986
nächster Redaktionschluß	29. August 1986

Nachdem wir bei unserem regionalen Treffen bei Hans feststellen mußten, daß die EDIT-Möglichkeiten wohl wegen fehlender Handbücher o.ä. bei einigen Clubkameraden nicht bekannt sind, nachfolgende Kurzbeschreibung:

Editieren einer Programmzeile

In die gewünschte Programmzeile kommt man durch:

1. EDIT Programmzeilen-Nummer
2. EDIT . (die zuletzt bearbeitete Zeile wird aufgerufen)
3. E Programmzeilen-Nummer (nur in einigen BASIC-Versionen)
4. , (die zuletzt bearbeitete Zeile wird aufgerufen - nur in einigen BASIC-Versionen

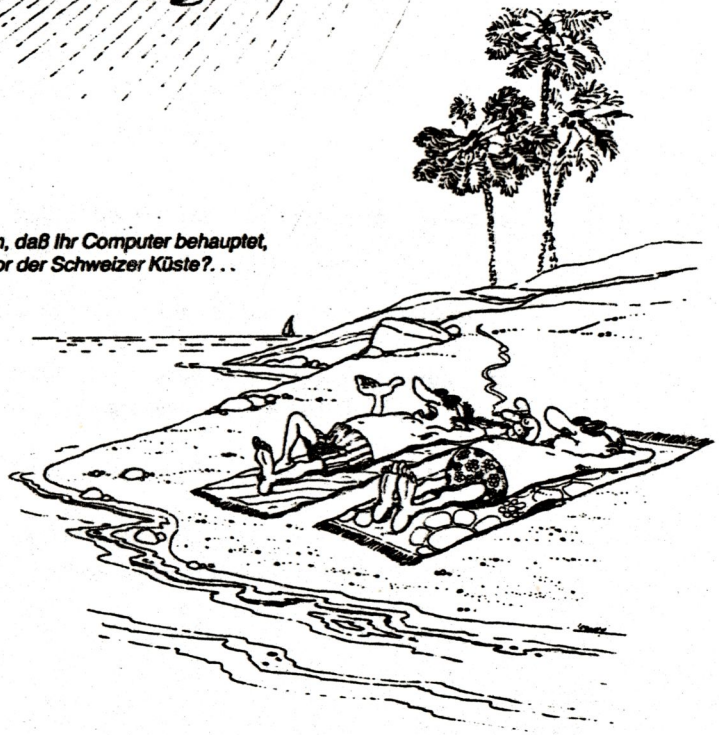
Unterbefehle und Funktionstasten im EDIT-Modus

(HP = Hochpfeil / LP = Linkspfeil)

ENTER	Editieren beenden und in den Befehlsmodus zurückkehren
SHIFT HP	Unterbefehl verlassen und im EDIT-Modus bleiben
n Leert.	Den Cursor um n Zwischenräume nach rechts verschieben
n LP	Den Cursor um n Zwischenräume nach links verschieben
L	den Rest der Zeile auflisten und zum Anfang der Zeile zurückkehren
X	den Rest der Zeile auflisten, den Cursor zum Ende der Zeile und den Insert(Einfüge-) Unterbefehl aufrufen
I	die folgenden Zeichen an der augenblicklichen Lage des Cursors einfügen; dieser Unterbefehl wird mit SHIFT HP verlassen
A	Änderungen rückgängig machen und den Cursor zum Anfang der Zeile zurücksetzen
E	Editieren beenden, alle Änderungen sichern und in den Befehlsmodus zurückkehren
Q	Editieren beenden, alle Änderungen rückgängig machen und in den Befehlsmodus zurückkehren
H	Rest der Zeile löschen und durch die folgenden Zeichen ersetzen; dieser Unterbefehl wird mit SHIFT HP verlassen
nD	von der augenblicklichen Lage des Cursors an die n folgenden Zeichen entfernen
nC	von der augenblicklichen Lage des Cursors an die n folgenden Zeichen durch die nächsten n Zeichen, die eingegeben werden, ersetzen
nSc	den Cursor von seiner augenblicklichen Lage aus an die Stelle fahren, wo das Zeichen c zum n-ten Male auftritt
nKc	alle Zeichen von der augenblicklichen Lage des Cursors an bis zu der Stelle entfernen, wo das Zeichen c zum n-ten Male auftritt

Sicherlich gibt es noch komfortablere bzw. bessere Editoren als diesen von Tandy (BASIC Level II). Aber immerhin ist es besser mit diesem zu arbeiten, als die Zeilen neu zu schreiben bzw. mit der Leertaste vorrücken zu müssen.

Klaus Hermann



Soll das heißen, daß Ihr Computer behauptet, wir seien vor der Schweizer Küste?...

Aber was werden wir tun, wenn Computer die ganze Arbeit machen?

WETTBEWERBSBEITRAG UM DAS 'PROGRAMM DES JAHRES' VON ARNULF SOPP

```
10 '     *****   BASIC-pur-Treiber für die HRG 1b von RB-Electronic  *****

         Die Graphik hat die Koordinaten X und Y (BASIC-Variable), die von
         0-383 (X) und von 0-191 (Y) gehen.
20 '     Dieses Programm stellt die Funktionen SET, RESET und POINT zur
         Verfügung, analog zur Klötzchengraphik.
30 '     Ein zu setzender, zu löschender oder zu prüfender HRG-Punkt wird
         mit den Koordinaten (Variablen) X und Y gekennzeichnet. Beispiel:
          X=50:Y=100. Dies ist der Punkt an der Stelle 50/100 auf dem Bildschirm.
40 '     SET wird mit GOSUB 10000, RESET mit GOSUB 20000, POINT mit GOSUB 30000
         realisiert. SET (10000) und RESET (20000) setzen bzw. löschen Punkte
          wie von Level-II-BASIC her gewohnt.
50 '     POINT (30000) lädt die Variable D mit -1 (Punkt gesetzt) oder 0
         (Punkt nicht gesetzt), entsprechend den wahr/unwahr-Usancen von
         Level II.

60 DEFINTA-Z'
         Dieser Befehl sollte allen SET/RESET/POINT-Operationen vorangehen,
         um die Ausführungsgeschwindigkeit zu erhöhen. Außerdem dient er als
          Lückenfüller, um dieses Programm wirklich zu einem Zehnzeiler zu
          machen. (Kicher!)

10000 GOSUB40000:OUT5%,AORC:RETURN'
      SET-Unterprogramm

20000 GOSUB40000:OUT5%,AAND(NOTC):RETURN'
      RESET-Unterprogramm

30000 GOSUB40000:D=((AANDC)<>0%):RETURN'
      POINT-Unterprogramm

40000 X1=INT(X/6%):B=X-X1*6%:Y1=INT(Y/12%):Y2=Y-Y1*12%:Z=Y2*1024%+Y1*64%+X1:
      ZM=Z/256%:ZL=ZAND255%:OUT2%,ZL:OUT3%,ZM:C=2%AB:A=INP(4%):RETURN'
      gemeinsames Unterprogramm, hauptsächlich Koordinaten bestimmen
```

```
* * *  Mein erstes 'ASSEMBLER-Programm'  * * *

====>  Schau Dir mal den Bildschirm an!  <====
```

```
8000             00010          ORG     8000H           ;EINSPRUNGSADR.
8000 21003C      00020 START    LD      HL,3C00H        ;1.BILDSCHIRM-POS.
8003 11013C      00030          LD      DE,3C01H        ;NAECHSTE POS.
8006 01FF03      00040          LD      BC,03FFH        ;ZAEHLE 1023 VORWAERTS
8009 36BF        00050          LD      (HL),0BFH       ;MACH 1.POS. WEISS
800B EDB0        00060          LDIR                    ;UND ALLE UEBRIGEN
800D CD5B80      00070          CALL    PSZYKL          ;ZUM PAUSE-ZYKLUS
-------------------------------------------------------------------------
8010 20          00130 REKLAM   DEFM    '   REKLAME IST GUT   ';BEHAUPTUNG
0015             00140 LAENG1   EQU     $-REKLAM        ;LAENGE 1.TEXT
8025 00          00150          DEFB    00H             ;WEITER GEHT'S
8026 20          00160 WAHRH    DEFM    ' WAHRHEIT IST BESSER ';GEGENBEHAUPTG.
0015             00170 LAENG2   EQU     $-WAHRH         ;LAENGE 2.TEXT
803B 00          00180          DEFB    00H             ;TEXTENDE
-------------------------------------------------------------------------
803C 011500      00190          LD      BC,LAENG1       ;CHR-ZAEHLER
803F 11133D      00200          LD      DE,3D13H        ;SCHAU NICHT HIN
8042 211080      00210          LD      HL,REKLAM       ;ANFANG DES UEBELS
8045 EDB0        00220          LDIR                    ;AUSGABE
8047 CD5B80      00230          CALL    PSZYKL          ;PAUSE-ZYKLUS
-------------------------------------------------------------------------
804A 011500      00250          LD      BC,LAENG2       ;CHR-ZAEHLER
804D 11D33D      00260          LD      DE,3DD3H        ;NOTABENE
8050 212680      00270          LD      HL,WAHRH        ;HEISST 'MERKE GUT'
8053 EDB0        00280          LDIR                    ;JETZT KOMMT'S
8055 C35580      00290 LOOP     JP      LOOP            ;SIEH DICH SATT
-------------------------------------------------------------------------
                 00305                                  ;SUBMARIN
8058 0603        00310 PSZYKL   LD      B,3H            ;PAUSE-WIEDERHOLFAKTOR
805A C5          00320 PAUSE    PUSH    BC              ;BEWAHRE B (C = EGAL)
805B 01FFFF      00330          LD      BC,0FFFFH       ;PAUSENLAENGE
805E CD6000      00340          CALL    60H             ;PAUSE (ROM-CALL)
8061 C1          00350          POP     BC              ;HOLE WIEDERHOLFAKTOR
8062 10F6        00360          DJNZ    PAUSE           ;WIEDERHOLE PAUSE
8064 C9          00370          RET                     ;WEITER
8000             00380          END     START           ;glückliches Ende!
00000 TOTAL ERRORS
33672  TEXT AREA BYTES LEFT
```

SYMBOL TABLE

```
LAENG1 0015 00140   00190
LAENG2 0015 00170   00250
LOOP   8055 00290   00290
PAUSE  805A 00320   00360
PSZYKL 8058 00310   00070 00230
REKLAM 8010 00130   00140 00210
START  8000 00020   00380
WAHRH  8026 00160   00170 00270
```


Schreiben, um verständlich zu machen

HEJ, Leute - verzagt nicht !
Auch wenn ZEUS, der Große Asseablierer, SEINEN Blitz auf Euch herabschleudert!
ER WIRD EUCH NICHT TREFFEN: Denn ich habe IHN zum Wanken gebracht!

Heute sage ich Euch (noch etwas hinter der hohlen Hand):
*** Macht IHN Euch untertan! ***
Von nun an wird "ER", der Göttervater, klein geschrieben!
Denn er läßt sich zu UNS herab...
- auf einer Leiter, mühsam gehalten von Zween von Euch, denen ich für ihre Hilfe beim mühevollen "Einstieg" (Aufstieg? Abstieg!) hier danken möchte, ohne ihre Namen zu nennen - denn sonst laßt Ihr Assembler-Willigen ihnen keine Ruhe mehr!
(Anm.: Bitte, schreibt doch dieses "ihre" und "ihnen" nicht immer groß, wenn, wie hier, in der 3.Person gemeint - tut uns den Gefallen, ja? Dankeschön! Dafür verrate ich Euch auch, wie man "ihn" - den Göttervater - in die Kniee zwingt...)

Hier nun das Ergebnis meines unermüdlichen Einsatzes im HEXen-Lotto (wie immer natürlich ohne Gewähr!)

Wer's besser weiß,
erhält 'nen Preis!

Das Thema des Programms ist völlig uninteressant (erst wird die Tafel gereinigt; dann schreibt ein Lümmel eine freche Behauptung drauf; schließlich wird er von weiser Hand eines Besseren belehrt - das ist alles). * Lediglich auf die Abfolge und Wirkung der Befehle ("Mnemonics" = Gedächtnisstützen) und ihre richtige Anwendung kam es mir an. Und Euch zunächst doch auch!? ("Euch" ist jetzt immer selektiv gemeint: mein "Seminar" ist Senf für die Experten; die sollen weiterblättern, es gibt Interessanteres in diesem Heft. Ich wende mich an die große Menge der Unschuldigen, der Unbedarften - als Einäugiger unter den Sehbehinderten...)

Seht Euch das Listing an: es ist fein säuberlich in sieben Spalten unterteilt. Die beiden ersten Spalten von links lassen wir da liegen, wo sie sind, nämlich links. Die hat der Assembler auf dem Gewissen. Die 3.Spalte, das sind die Zeilen-Nummern, die ich benutzt habe bei der Programm-Entwicklung.
Gehen wir nun nach diesen Zeilen-Nummern vor:
In der ersten Zeile habe ich "ihn" (den Assembler, künftig von mir nur noch "Ass" genannt) "angewiesen", daß er das von ihm Verdaute im Örtchen Nr. 8000H (Speicheradresse) abladen soll. Wohlgemerkt: ich habe ihn "angewiesen", ihm eine "Anweisung erteilt" - nicht einen Befehl, den er an die CPU (=Central Processing Unit = Leitzentrale unseres GENIE/TRS80) weitergeben solle. Dies ist ein feiner Schiedunter, wie Ihr noch merken werdet.
Zeilen 2 bis 4: Jetzt wird es ernst, sehr ernst sogar!
Ein ausnahmsweise erfolgreicher Sisyphos kreuzt auf: er schafft es in der Tat, ganze Blöcke zu verschieben, was dem König von Korinth sz. nie gelingen sollte...
Der "Block": das ist hier eine Folge von Bytes, z.B. eine Anzahl Zeichen gemäß ASCII-Code oder einfach eine Mitteilung (Deutsch: "message"), die man z.B. auf den Bildschirm bekommen will.
Wie geschieht das? Dies ist eins der wichtigsten Geheimnisse des Herrn Ass, das wir jetzt entschleiern wollen:
Wir alle wissen natürlich schon, daß die Seele unseres Spielzeugs im wesentlichen zweiteilig (nicht etwa zwiespältig) ist: sie hat ein Gedächtnis und ein Herz; auf "Deutsch": ein memory und eine CPU (s.o.). Letztere, die Leit- und Arbeitszentrale, besteht (unter anderem) aus 12 Registern - von den Ersatzregistern sehe ich einmal ab -, deren jedes 1Byte = 8 Bits faßt. Laßt Euch nicht durch deren Namen verwirren, bis auf vier sind es nichts anderes als

Haus-Nummern: B, C, D, E, H, L, IX und IY. Die vier übrigen sind die "Edleren", und zwar: der fleißige Akkumulator **A** (nomen est omen), die symbolisierende Fahne oder Flagge **F**, der Programm-Zähler oder Program Counter **PC** und der Stapel-Zeiger oder Stack Pointer **SP**. Die Funktionen all dieser Register lassen uns jetzt noch kalt.
Merken müssen wir uns schon, daß man einige davon zu Pärchen zusammenfügen kann. Das ist wichtig, wenn das Fassungsvermögen von einem einzigen Byte nicht ausreicht. Zum Beispiel bei Speicher-Adressen. Die gehen ja bis 65535 = FFFFH. Ein Byte kann jedoch höchstens FFH = 255 darstellen (die Kenntnis der HEXerei darf ich wohl voraussetzen).
Folgende Pärchen sind machbar: AF / BC / DE / HL.
AF - das fällt zunächst schwer; ist aber als halber Affe leicht zu merken. BC heißt zwar nicht Byte Counter, könnte aber. Dieses Pärchen countet häufig. DE = Du(mmer) Esel ist für fast alles da. Und HL = High/Low = HiLo merkt sich auch leicht, denn dessen Hauptfunktion ist in der Tat der "Hinweis" auf eine Speicheradresse: erst das "Hohe", spr. das vordere Byte (z.B. von "3FD7H" landet das Byte "3F" im Register H), dann folgt das "Lausige" - Verzeihung: das Lowere Byte (D7 kommt also ins Register L). Im Speicher steht's dann allerdings umgekehrt - aber diese Verwirrungs-Taktik zieht bei uns nicht mehr, wir Seeleute lassen uns nicht erschüttern (nach dem TSW-Motto "Trau, Schau, Wem").
So, nun kommt's:
Wir wollen zunächst einmal den Bildschirm ganz schnell weiß anstreichen. Hierfür bedienen wir uns des "White Byte". Mit dem "Weißen Byte" meine ich den Weißmacher, das Grafikzeichen mit dem ASCII-Code 191 = BFH. Wenn wir jede der 16 mal 64 = 1024 Positionen des Bildschirms mit dem Byte BFH besetzen, so wird er lückenlos aufgehellt.

ASS hat für diese und ähnliche Prozeduren zur Vereinfachung bereits bestimmte Register-Paare verpflichtet: das HL-Paar wird mit der ersten Adresse des Bildschirms (3C00H), das DE-Paar mit der nächsten (3C01H) "geladen" (LOAD, kurz LD):

```
LD HL, 3C00h
LD DE, 3C01h
```

D.h.: "LaDe HL mit 3C00h - LaDe DE mit 3C01h".
So einfach ist das. Leider wieder rückwärts. Also merkt Euch: HL ist immer der Geber (Hingebender Lieferant), DE = Der Empfänger.
Ass macht's stets so kurz wie möglich. Gerad so kurz, daß wir uns noch etwas dabei denken können. Und nichts anderes als solche Gedächtniskrücken ("MNEMONICS") versteht der (Götter-)Vater Ass; NUR DIE kann er der CPU schmackhaft machen, ja regelrecht "vorkauen". Denn dieses Baby - unser Spielzeug - will nicht einmal mit ALETE gefüttert werden, geschweige denn mit MNEMONICS; nur mit diesen langweiligen Flips und Flops: mit 1=höheres oder 0=niederes Potential! Ein elektronischer Homunculus...
Wer gibt hier Wen Was? Das Konzept lautet: Gib 3C00h (der ersten Bildschirm-Position) einen Weißmacher und veranlasse die CPU, daß dieser an die nächste Position weitergereicht wird, danach wieder an die nächste usf. Wie oft? Solange bis der Schirm voller BFh ist, also noch 1023 mal, nachdem die erste Position schon "eins drauf gekriegt hat".
Diese Anzahl kontrolliert der ByteCähler BC. Wir schreiben daher diesmal:

```
LD BC, 03FFh
```

weil 1023 (Dez) = 03FF (Hex).
Gleich kann's losgehen. Das einzige, was wir noch zu tun haben, ist, den ersten Weißmacher als Pionier in die Bildschirm-Speicherstelle 3C00H zu packen. Also:

```
LD 3C00H, BFh    ? ? ?  Mitnichten !!!!!
```

Nächster MERKSATZ:

Es können nie - niemals zwei Zahlenwerte im Befehl stehen !

Das wäre für Herrn ASS wie Höhere Mathematik: soviel verkraftet er nicht!
Wir müssen mit Zahlen stets den Umweg über ein Register einschlagen (meist A oder HL), wenn im OP(erations)-Code (=5.Spalte) oder im Operanden (=6.Spalte) schon eine Zahl steht! Und eine Speicher-Adresse ist ein reiner Zahlenwert

In diesem Fall wird das Register HL in Klammern gesetzt, dann weiß Ass bzw. die CPU: Nicht das Register HL, sondern die Speicherstelle, die in HL steht, soll mit dem Zahlenwert BFh gefüllt werden!
Steht nicht ein Register-Buchstabe, sondern eine Zahl in Klammern, so ist damit eine "Hausnummer", d.h. eine Speicher-Adresse gemeint.
Soll die Speicherstelle 3C00h mit BFh geladen werden (damit sie weiß wird), so müssen wir das also dem Register HL (in dem die Adresse 3C00h bereits steht) mitteilen, indem wir es einklammern.
Allgemein gilt immer, wenn eine Klammer auftritt:
"Lade die Speicherstelle, die du in Sinn hast, mit dem Wert sowieso!"
("Sinn" = Klammer.)

In unserem Falle ist der "Wert" ein Code, nämlich der ASCII-Code "BFh".
Und noch eine Falle hätte Herr Ass bei diesem Versuchs-Befehl vorgefunden: er hätte "BFh" für einen String, eben für die Buchstaben-Folge BFh gehalten. Wir meinten aber eine Hex-Zahl. Die akzeptiert er als solche nur, wenn sie mit einer Ziffer beginnt. Doch woher nehmen? Code 191 (Dez) fängt in HEX-Darstellung nunmal mit dem Buchstaben B an! "Was tun?" sprach ZEUS hier nicht; sondern: "Eine Null ist immer gut: sie schadet niemand - wenn was dahinter steckt!!" (Wie recht er hat...)
Jetzt ist also alles klar (??) :

```
LD (HL), 0BFh
```

ist richtig. Nicht HL wird mit BFh geladen, sondern (HL), das heißt:
die Speicherzelle, deren Adresse in HL steht,
und das ist ja (siehe 1.Befehl) die erste Bildschirm-Position!
Wir dürfen uns den Schweiß abwischen, den ZEUS vor unseren Erfolg gesetzt.
Denn nun arbeitet ASS für uns. Es bedarf nur noch eines Fußtrittes, unentschlossen wie ASS ist - aber dann rast er!
Der Tritt, der den Ball (Fußball 1986!) ins Sausen bringt:

LaDe den Inhalt der Quelle HL ins Ziel DE, kurz "LD";
Inkrementiere (erhöhe) beide um 1, kurz "I";
vermindere gleichzeitig den Zähler BC um 1;
Repeat (wiederhole) diesen Vorgang, kurz "ER"
und zwar solange, bis in BC eine Null steht.

Fassen wir diese Abkürzungen zusammen, so erhalten wir das schlaue Kürzel

LDIR

das dies alles allein zuwege bringt.
Jetzt ist der Bildschirm zwar nicht sauber - aber rein. So schnell schafft das kein Persil.
Was steht denn da in der 4.Spalte? Sieht aus wie Aushänge-Schilder. Sind es auch, wie sie vor wichtigen Häusern hängen (z.B. bei Friseuren, Bäckern und Ähnlichem).
Gelegentlich muß man der CPU sagen, wo sie mit ihrer Arbeit anfangen und wo aufhören soll. Sie versteht "START" und "END". Nicht immer ist "START" identisch mit "ORG" (= origine = Ursprung, eigentlich also auch gleich Anfang; hier aber Anfang des Codes im Speicher, START hingegen ist das Startloch für die CPU, wie RUN für BASIC.
übrigens, der Ordnung halber sei's gesagt, obschon der Club80 wohl namensgerecht nur den Mikroprozessor Z80 oder dessen Abkömmlinge betreibt: Alles hier Gesagte betrifft eben diesen; ich weiß nicht, ob alles z.B. für den "6502" (hochwohllöblicher COMMODORE) oder andere auch zutrifft. Wie gesagt: "OHNE GEWÄHR!"

==== PAUSE! ====

Gut erholt??
Was, der Monitor will auch pausieren? Möchte eine Zeitlang weiß bleiben? (Welcher Bösewicht möchte das nicht?)
Kann er haben!
Meine Pause ist aber noch nicht zu Ende! Ich lasse jetzt mal andere für mich arbeiten! Wo sind doch die vielen Arbeitslosen? .... Aha - hier in Zauberkästchen vor mir mit dem Namen KB (Kleines Biest wie KeyBoard) schlummern sie vor sich hin. Weckt sie auf, die Bediensteten im ROM !! Was heißt überhaupt "ROM" ?
Ich glaube: Routines Organization for Management!(Wer's nicht
glaubt, verzichtet auf vieles!)
Unter den vielen Routinen, die es uns zu unserer Bequemlichkeit beim (Maschinen-)Programmieren bietet, befindet sich eine, die heißt "DELAY" (= Zaudere! Zögere!) Sie beginnt in der Speicherstelle 60H. Bevor wir diese in Anspruch nehmen, müssen wir aber unserem "Zählwerk" BC verraten, wie lange das "Bild" still stehen soll. Ich füllte BC daher mit der größten Zahl, die in dieses Doppel-Register (= 2 Byte) hineingeht (s. Zeile 330. Seht Ihr dort die Null zur Einleitung der HEX-Zahl? Erinnert Euch!)
Doch als ich das Programmchen dann laufen ließ, war mir das doch ein bißchen zu kurz: eine knappe Sekunde. Die Taktzeit des Delay-Befehls beträgt nur 14,7µs, wie man an passender Stelle nachlesen kann (z.B. bei Röckrath, "TRS80 ROM Listing", Seite 6). Dies mit FFFFh = 65535d multipliziert, ergibt 0,96 Sekunden.
Deshalb baute ich diese ROM-Routine in eine Schleife ein, die sie dreimal durchlaufen muß, ehe sie befreit wird, so daß die Pause wenigstens knapp drei Sekunden (2,88s) dauert.
Dies machte ich so: Ich packte die ROM-Routine in ein UP (Unterprogramm, "Deutsch":SUBMARINE (d.i. eine Sub-Routine, die unter Wasser schießt, so daß man sie innerhalb des Programms erst lange suchen muß und (siehe besagtes ROM-Listing) oft nur daran erkennt, daß sie mit einem "RET"(urn) aufhört).
Statt BASIC's "GOSUB" sagt man "CALL" (Zeile 70) und gibt die Zeilen-Nummer an, wo sich das UP befindet; gewöhnlich am Schluß eines Programms. Ich wählte die Nummer 310 (s.dort).
Schlau wie ich manchmal bin, sagte ich mir aber, daß das wohl nicht optimal ist, weil die Zeilen-Nummern dieses "Source-Codes" für Herrn Ass ziemlich uninteressant sind; er mißachtet sie und stopft die Bytes einfach hintereinanderweg in den Speicher, beginnend mit der Adresse, die ich ihm als "ORG" vorgegeben. Diese Adressen sind dann die eigentlichen Zeilen-Nummern.
Um mich von den allzu vergänglichen Zeilen-Nummern meines Codes (oder Kotes?) unabhängig zu machen, hängte ich wieder so ein Aushängeschild, auch "LABEL" genannt ("Etikett" wär' auch nett) vor den Beginn des UP und nannte es sinnig "PSZYKL" (PAUSE-ZYKLUS). Dieses steht nun stellvertretend für die Adresse des UP - egal, an welcher Stelle auch immer es nachher im Speicher abgelegt wird (anfängt).
Daraus sieht Zeile 70 so aus, wie sie aussieht: mit "PSZYKLUS" als Zielscheibe.
Wie wurde diese Schleife, dieser Zyklus, der ab 310 beginnt, nun programmiert?
Als Schleifenzähler dient immer Register B. Leider aber auch (zusammen mit C) als "Uhr" für die Dauer der Pause. Was nun?
Ich habe mir einen Trick ausgedacht. Den habe ich weder gelesen noch hat ihn mir jemand verraten, und deshalb ist er möglicherweise auch unbeholfen. Aber er läuft - Hauptsache! (Die Erfahrenen unter uns würden sicher einen besseren wissen; aber leider sind die ja längst zu interessanteren Themen übergegangen - nicht ohne vorher ein- bis dreimal über dieses "ihre Nase gerümpft" zu haben...)
Ich lade also zuerst B mit der Zyklenhäufigkeit 3 (Zeile 310)
Dann stoße (PUSHe) ich diesen Wert rücksichtslos ins Abseits, "STACK" genannt (Zeile 320). Aus diesem (un)heimlichen Schlupfloch kann ich ihn jederzeit wieder hervorholen; es ist wie ein Räubernest: immer der Letzte, der sich dorthin geflüchtet, muß als Erster wieder heraus (was, Ihr habt noch nicht von

"LIFO" gehört? "LAST IN, FIRST OUT"? Schämt Euch!)
Als Geisel nimmt B noch das völlig unschuldige Christkindlein C mit, denn Einzelpersonen öffnet der Zerberus vor dem Unterschlupf dessen Tore nicht; er will immer zwei Bytes auf einmal verschlingen. Macht nichts, denn C interessiert uns heute weder in Freiheit noch in Gefangenschaft. (Immer wird dies nicht so klappen, fürchte ich...)
Nun kann BC getrost mit der Schleifendauer geladen werden (Zeile 330); sein Inhalt ist sichergestellt.
Und jetzt: Nichts wie her mit der DELAY-Routine (Zeile 340), dieser Schlafmütze!
Es ist deren besondere Eigenart, weiter zu schlafen, wenn sie gerufen wird; aber genau das ist ihre Aufgabe: solange das Zählwerk BC noch nicht bei Null angelangt ist, geschieht - nichts, rein gar nichts, und wir können uns in aller Ruhe das Werk des Herrn Saubermann ansehen.
Hat sie ausgeschlafen, POP(elt) Zeile 350 das verstoßene Pärchen BC (d.h. seine alten Werte) aus dem Stack wieder in sein Heimat-Registerpaar BC zurück, so daß wir in B wieder den letzten Schleifenstand stehen haben.

Nun kommt schon wieder was Neues! (Lieber Leser: Hast Du noch ein paar Speicherplätze frei im Hirn? Sonst geh hier schlafen! ROMCALL 60 weckt Dich!!)

"Denk Jaa Nich, Z80,
der Anfänger lacht sich!"

Der lacht sich gar nicht. Bis er merkt, daß er die erste Zeile (nicht des CODE, sondern dieses "Gedichtes") leicht zusammenfassen kann:
DJNZ = Dekrementiere (and) Jump (if flag bit is) Not Zero!
Diesmal auf gut Deutsch: "Vermindere B um 1 und springe (zurück zum Label PAUSE), sofern das Null-Bit des FLAG-Registers nicht gesetzt ist, d.h. nicht Null anzeigt!" Alles restlos klar?? KLar für den, der Deutsch versteht. (Oder hättet Ihr es lieber in Spanisch? Dann fragt Arnulf. Er antwortet Euch Spanisch; teils wirklich, teils wird's Euch so vorkommen...)
Wie auch immer: die Pausenschleife wird solange durchlaufen, bis B=0. In unserem Fall also dreimal.
Puh - das wär's. Und das bei 34 Grad im Schatten des Odenwaldes...
Nachdem sich der Beschauer genügend bei der Betrachtung des gereinigten Bildschirms gelangweilt hat, soll endlich was passieren. Laßt uns auf die Tafel eine Botschaft schreiben.
"Botschaft" heißt bei ASS: message (nicht zu verwechseln mit Messe oder Kirmes - aber ich wittere eine gewisse Verwandtschaft!)
Was ist mit dem Zauberwort "LDIR"? Könnten wir das hier nicht auch gebrauchen?
Statt jeden Buchstaben unserer Message einzeln zu transportieren (das ginge auch), fassen wir sie wieder in einen "Block" zusammen und definieren einfach:
"Define Message" ===> DEFM '.....' (z.B. Zeile 130)
Das, was die Botschaft besagen soll, setzen wir dorthin, wo jetzt noch die Punkte stehen (Apostrophe nicht vergessen!) Ich glaube, ihre Länge ist beliebig (?)
Damit die CPU auch weiß, wann und wo die Message zu Ende ist, setzen wir noch ein Null-Byte hinterher, vermögens der Definition DEFB 00H (s.Zeile 150)
Trotzdem - die CPU ist ja sauduma - müssen wir ihr noch explizit sagen, wie lang das Ding ist. Das teilen wir wieder unserem "ByteZählwerk" BC mit:

LD BC, Länge der Botschaft

Zum Beispiel behauptet da einer " REKLAME IST GUT ", eine Sentenz mit je drei Blanks Anstands-Abstand vorn und hinten - weil Reklame sowas Vornehmes ist...
Also 21 Bytes (auch Blanks sind Bytes! Verachtet mir die Unscheinbaren nicht, sie leisten die wichtigsten Dienste!
Oderkannstedaslesen:Himmikruzifixibuasmirnicheeiheiligsblechle?
Somit: BC,15H (15H = 21dez)
Aber wir brauchen die Bytes auch nicht "von Hand" abzuzählen, wenn uns das schwerfällt. Wir können einen Trick anwenden.
Zunächst hängen wir wieder so ein "Schild" (Label) vor die Message-Definition.

Sinnvollerweise nannte ich es "REKLAM". Damit repräsentiert das Label "REKLAM" die uns verborgene Anfangs-Adresse der Message. Es ist dasselbe wie eine symbolische Konstante in der Mathematik, hinter der sich ein Zahlenwert versteckt, den wir noch gar nicht kennen.

ACHTUNG:
Länger als 6 alphanumerische Zeichen darf ein Label nicht sein und es muß mit einem Buchstaben beginnen!

Wenn die Message X Bytes lang ist, so belegt sie X Speicher-Adressen.
Wenn sie bei der Adresse REKLAM beginnt, befindet sich ihr letztes Byte demnach in der Adresse REKLAM+X-1 (rechnet nach!). Die nächste Zeile hat also die Adresse REKLAM+X. * Andersherum:

Die Länge X ist gleich der Adresse der nächsten Zeile minus "REKLAM"

Da die Adresse einer Zeile, in der man sich gerade befindet, sich durch das Symbol $ kennzeichnen läßt, ohne daß man ihren zahlenmäßigen Wert zu kennen braucht, lösen wir unser kleines Problem, indem wir
sofort in der nächsten Zeile folgende Gleichung (EQUATION) hinschreiben:

```
LAENG1  EQU  $-REKLAM      (Zeile 140)
```

Somit füllen wir den Zähler BC mit LAENG1:

```
LD BC,LAENG1   (Zeile 190)
```

Nun brauchen wir nur noch HL und DE, wie gehabt, zu beschicken.
Die Quelle, der Haus- & Hof-Lieferant HL, peilt - das haben wir mithilfe des Labels so geregelt - die Adresse "REKLAM" an. So ergibt sich Zeile 210.
Wohin mit der "Botschaft"? Auf den Bildschirm natürlich, damit er nicht ewig weiß bleibt. Wir wählen als Anfang den Punkt mit der Adresse 15632 = 3D10H (wie schön: fängt mit einer Ziffer an!) Somit erhält "Der Empfänger" DE die Zieladresse 3D10, s.Zeile 200.
(Die Reihenfolge, in der wir die Register-Pärchen beschicken, spielt offenbar keine Rolle. Deswegen ist dieses kleine Zeilen-Durcheinander hier nichts weiter als eine freche Abwechslung. Punkt.)

So, das wär's schon. Jetzt nur noch schnell **LDIR** befohlen (Zeile 220) - - -

und schon prangt da diese Behauptung der Sprüche-Macher auf dem schönen Laken...

Nur rein zur Übung wiederholen wir die Prozedur, diesmal mit einem VETO (Einspruch) - na, schaut es Euch selbst an.
Das Looping - der unbedingte Sprung (JUMP) "zurück" zum Label "LOOP" - kann unterbleiben, falls Ihr Euch das Wunderwerk nicht bis zu St.Nimmerlein's Tag ansehen wollt. Dann allerdings hilft nur noch das Fast-Allheilmittel "RESET" - so streng sind Asses Bräuche!! (Stimmt nicht: man kann auch den Dreitasten-Befehl 123 benutzen und dann im DEBUG "Q" sagen!).
Vergeßt nicht, am Schluß zu sagen "Nun Schluß!" oder vielmehr: "END(e)"
Wenn Ihr hinter "END" noch "START" setzt, tut Ihr, wie ich gehört habe, dem ASS oder auch der CPU einen Gefallen; dann weiß sie nämlich, wo sie starten soll, nachdem sie den ganzen Plan, nach dem sie marschieren soll, erst einmal durchgesehen und gründlich studiert hat. Ich glaube nämlich, genau das tut sie, bevor sie loslegt; wie könnte sie sonst wissen, was da alles definiert und equalisiert wurde, bevor sie die betreffenden Zeilen mit den Definitionen und Gleichungen überhaupt erreicht hat?? Das habe ich zwar nirgends gelesen und das hat mir auch niemand gesagt
(eben das ist ja der ganze Jammer bei diesem Lernvorhaben!)
aber mir sagt es schlichte Logik (die allerdings nicht immer ins Schwarze trifft...)

* * * * * *

So - und Ihr lieben geduldigen Leser tätet mir jetzt auch einen Gefallen, wenn Ihr mir bald mitteilen tuen tätet, ob Ihr daran interessiert seid, daß ich Euch künftig weiterhin von meinem ständigen (?) zähen Kampf mit ASS und Genossen laufend berichte - vielleicht hat doch der eine oder andere etwas davon, der sich bisher nicht "rantraute".
Die Experten unter uns machen sicher manche Wendung besser. Vielleicht steuern sie jetzt auch mal etwas unter ihrem Niveau hierzu bei...
(und weisen mich, bitte, auf Fehler in meiner "Anleitung" hin - das Prograämmchen läuft jedenfalls!)

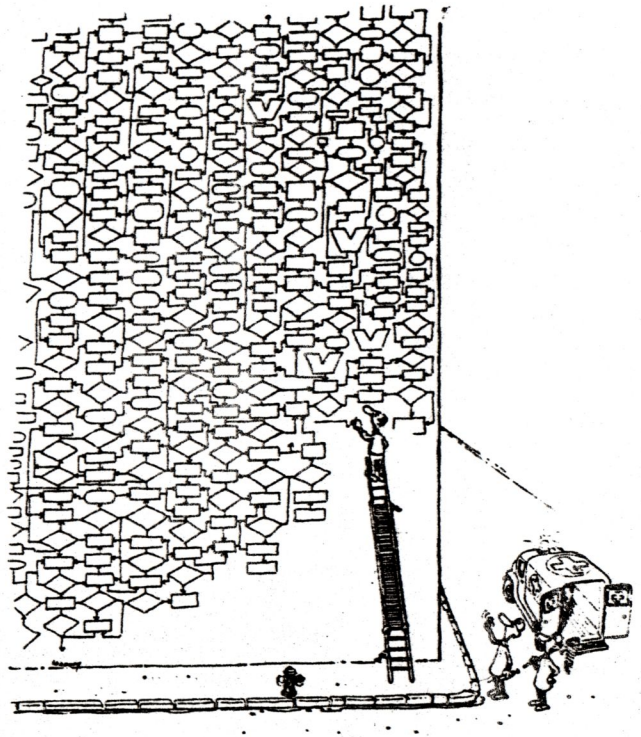
Glaubt nur nicht, ich wäre ein Menschenfreund.
Alles ist purer Egoismus!

Denn:

EAST WENN MAN ETWAS ZU BESCHREIBEN VERSUCHT,
MERKT MAN, OB MAN ES BEGRIFFEN HAT...

SO LONG ! (BUT NOT
"PING-PONG"!) *KA-JOT

Kajot



Einige Assembler-Routinen

Wenn man als Assembler-Laie einige professionelle Programme betrachtet, fallen kurze Unterprogramme ins Auge, die Funktionen ausführen, die man selber vergeblich zu programmieren versucht hat. Jedenfalls ging es mir so mit Scripsit. Ganz am Anfang meiner Bekanntschaft mit Assembler wollte ich einige kleine Spielprogramme schreiben, die aber schon im Ansatz scheiterten: Zufallszahlen und die Zahlenausgaben auf dem Bildschirm waren damals ein riesiges Problem. Da ich vermute, daß auch andere klein anfangen und ähnliche Schwierigkeiten haben, möchte ich einige kurze Routinen vorstellen, die aus Scripsit stammen, aber wohl ähnlich in jedem größeren Programm auftauchen. Beim Ansehen und Ausprobieren kann der Anfänger schon viel über Assembler lernen, aber auch für geübtere Anwender dürften diese Routinen zum Einbau in ihre Programme interessant sein.

Gerald Schröder

Zu AUSWERT:
Dieses Unterprogramm erwartet, daß HL auf einen Buffer zeigt, in dem eine Dezimalzahl steht (0-65535). Außer Ziffern sind nur Leerzeichen in dem Buffer zugelassen. Das erste Zeichen, das weder Space noch Ziffer ist, führt zum Rücksprung aus dem Unterprogramm. Dabei steht die umgesetzte Zahl in HL und kann somit weiterverarbeitet werden.

Zu UMWAND:
Hier zeigt HL auf das Low-Byte einer Zahl, die in der Reihenfolge Low-Byte / High-Byte im Speicher steht. Diese Zahl wird in eine Dezimalzahl (0-65535) umgewandelt, welche rechtsbündig in einen Buffer geschrieben wird. Von dort kann sie zum Beispiel auf den Bildschirm gebracht werden.

Zu RND:
Dieses Programm erzeugt eine Zufallszahl in A. Das Programm ist zwar sehr kurz, aber eine Folge von Zufallszahlen scheint sich nicht zu wiederholen. Scripsit benötigt die Zufallszahlen, um die zusätzlichen Spaces beim Justify (Randausgleich) über die Zeile zu verteilen. Da die RND-Tabelle vor jedem Ausdruck in den Urzustand zurückversetzt wird, sehen alle Ausdrucke desselben Textes absolut identisch aus.

Zu TEILEN:
Hier wird L durch B geteilt, was eigentlich nicht aufregend ist, doch das Ergebnis ist interessant: In BC steht sozusagen eine Kommazahl im Hexadezimalsystem. Das heißt nichts weiter, als daß B die Zahl vor dem Komma und C die nach dem Komma beinhalten. Einziger Unterschied zum Dezimalsystem: C muß durch 256 geteilt werden, um zum korrekten Ergebnis zu kommen.
Beispiele:

```
Eingabe  L=2   B=1      Ausgabe  BC=02 00  (=2.00 dezimal)
         L=3   B=2               BC=01 80  (=1.50 dez.)
         L=1   B=3               BC=00 55  (=0.33 dez.)
```

Somit kann man jetzt auf- oder abrunden oder auch zwei Kommazahlen addieren. Scripsit benutzt auch diese Routine im Justify. Es berechnet damit, wieviele Spaces in der Zeile an einem Wortzwischenraum noch eingefügt werden müssen (L=Anzahl der einzufügenden Spaces, B=Anzahl der noch verfügbaren Wortzwischenräume). Zu dem Ergebnis BC wird eine Zufallszahl aus der Routine RND addiert, so daß eventuell aufgerundet wird, aber nicht unbedingt. Es werden dann 8 Spaces an den aktuellen Wortzwischenraum eingefügt.


```
00002 ;    CALL AUSWERT                geschützte Register:
00003 ;                                alle außer HL
00004 ;    Zahl aus dem Buffer (HL)... auswerten.
00005 ;    Ausgabe: Zahl in HL
00006 ;
00007 AUSWERT PUSH    AF
00008         PUSH    BC
00009         PUSH    DE
00010         EX      DE,HL             ;Bufferzeiger NACH DE
00011         LD      HL,0000           ;HL LÖSCHEN
00012 AUSW1   LD      A,(DE)            ;ZEICHEN LADEN
00013         INC     DE                ;POINTER +1
00014         CP      20H               ;SPACE? => NÄCHSTES ZEICHEN
00015         JR      Z,AUSW1
00016         CP      30H               ;Ziffer?
00017         JR      C,AUSW2           ;nein, Absprung
00018         CP      3AH
00019         JR      NC,AUSW2          ;ebenso
00020 ;
00021 ;       ZIFFER GEFUNDEN
00022 ;
00023         ADD     HL,HL             ;ZAHL VERGRÖßERN (ZIFFERN LINKS)
00024         LD      B,H
00025         LD      C,L
00026         ADD     HL,HL
00027         ADD     HL,HL
00028         ADD     HL,BC
00029         SUB     '0'               ;AUS ZIFFER ZAHL ERZEUGEN
00030         LD      C,A               ;NACH BC UND AUF HL ADDIEREN
00031         LD      B,00H
00032         ADD     HL,BC
00033         JR      AUSW1             ;NÄCHSTE ZIFFER
00034 ;
00035 AUSW2   POP     DE                ;ABGANG
00036         POP     BC
00037         POP     AF
00038         RET

00040 ;   CALL UMWAND                  geschützte Register:
00041 ;                                alle
00042 ;   Wandelt Zahl in (HL)/(HL+1) in Ziffern (dezimal) um.
00043 ;   Die Zahl wird rechtsbündig in den Buffer geschrieben.
00044 ;
00045 UMWAND  PUSH    AF
00046         PUSH    DE
00047         PUSH    BC
00048         PUSH    HL
00049         PUSH    IY
00050         LD      IY,ZIFTAB
00051         LD      C,00H             ;WIEVIELE ZIFFERN
00052         LD      E,(HL)            ;DIVIDENDEN LADEN (LOW-BYTE)
00053         INC     HL
00054         LD      D,(HL)            ;(HIGH-BYTE)
00055         LD      HL,BUFFER         ;AUSGABE-BUFFER
00056         EX      DE,HL             ;NACH DE, ZAHL IN HL
00057 UMW1    PUSH    DE                ;MOM. BUFFERADRESSE RETTEN
00058         LD      A,'0'             ;ZIFFER "0"
00059         LD      D,(IY+01H)        ;DIVISOR LADEN (10.000, 1000, 100, 10, 1)
00060         LD      E,(IY+00H)
00061 UMW2    OR      A
00062         SBC     HL,DE             ;PABT WIE OFT IN IN DEN DIVIDENDEN?
```

```
00063          JR      C,UMW3          ;FERTIG MIT DIESEM DIVISOR
00064          INC     A               ;ZIFFER+1
00065          JR      UMW2            ;UND WEITER
00066 ;
00067 UMW3     ADD     HL,DE           ;DIVIDENDEN KORRIGIEREN
00068          INC     IY              ;NÄCHSTER DIVISOR
00069          INC     IY
00070          CP      '0'             ;DIVISOR>DIVIDEND?
00071          JR      NZ,UMW4         ;NEIN, DIVIDEND/DIVISOR>0
00072          INC     C               ;SCHON VORHER ZIFFER?
00073          DEC     C
00074          JR      NZ,UMW5         ;JA, ZIFFER "0" AUSGEBEN
00075          LD      A,' '           ;NEIN, FÜHRENDE LEERZEICHEN AUSGEBEN
00076          JR      UMW5            ;UND IN DEN BUFFER
00077 ;
00078 UMW4     LD      C,A             ;"SCHON VORHER ZIFFER"-FLAG SETZEN
00079 UMW5     DEC     E               ;DIVISOR=1? JA, FERTIG, AB
00080          POP     DE              ;BUFFERZEIGER ZURÜCK
00081          LD      (DE),A          ;ZIFFER ABSPEICHERN
00082          INC     DE              ;ZEIGER+1
00083          JR      NZ,UMW1         ;NOCH NICHT FERTIG? => NÄCHSTER DIVISOR
00084 ;
00085          INC     C               ;ZAHL=0? (KEINE ZIFFER?)
00086          DEC     C
00087          JR      NZ,UMW6         ;NEIN, AB
00088          DEC     DE              ;JA, "0" IN DEN BUFFER
00089          LD      A,'0'
00090          LD      (DE),A
00091 UMW6     POP     IY              ;AB
00092          POP     HL
00093          POP     BC
00094          POP     DE
00095          POP     AF
00096          RET
00097 ;
00098 ;        DIVISOREN FÜR VORSTEHENDES UNTERPROGRAMM
00099 ;
00100 ZIFTAB   DW      10000
00101          DW      1000
00102          DW      100
00103          DW      10
00104          DW      1
00105 ;
00106 ;        AUSGABE-BUFFER FÜR VORSTEHENDES UNTERPROGRAMM
00107 ;
00108 BUFFER   DS      10

00111 ;   CALL RND                     geschützte Register:
00112 ;                                 alle außer AF
00113 ;   Erzeugt eine Zufallszahl in A
00114 ;
00115 RND      PUSH    BC
00116          PUSH    HL
00117          LD      HL,RNDTAB       ;RANDOM-TABELLE
00118          LD      B,08H
00119          LD      A,(HL)
00120 RND1     RLCA
00121          RLCA
00122          RLCA
00123          XOR     (HL)
00124          RLA
00125          RLA
00126          DEC     HL
00127          DEC     HL
00128          DEC     HL
00129          LD      A,(HL)
00130          RLA
00131          LD      (HL),A
00132          INC     L
00133          LD      A,(HL)
00134          RLA
00135          LD      (HL),A
00136          INC     L
00137          LD      A,(HL)
00138          RLA
00139          LD      (HL),A
00140          INC     L
00141          LD      A,(HL)
00142          RLA
00143          LD      (HL),A
00144          DJNZ    RND1
00145          POP     HL
00146          POP     BC
00147          RET
00148 ;
00149          DB      34H,12H,78H
00150 RNDTAB   DB      56H

00152 ;   CALL TEILER                  geschützte Register:
00153 ;                                 alle außer BC
00154 ;   Teilt L durch B. Ausgabe: B=INT(L/B)
00155 ;                                 C=L/B - INT(L/B) in 256stel
00156 ;
00157 TEILER   PUSH    AF              ;RETTEN
00158          PUSH    HL
00159          PUSH    DE
00160          LD      H,00H           ;H UND C LÖSCHEN, D=2
00161          LD      D,02H
00162          LD      C,H
00163 TEIL1    LD      A,08H
00164 TEIL2    ADD     HL,HL
00165          OR      A
00166          SBC     HL,BC
00167          INC     HL
00168          JR      NC,TEIL3
00169          ADD     HL,BC
00170          RES     0,L
00171 TEIL3    DEC     A
00172          JR      NZ,TEIL2        ;8 MAL
00173 ;
00174          DEC     D
00175          JR      Z,TEIL4         ;2 MAL
00176          LD      E,L             ;L RETTEN
00177          LD      L,C             ;L = 0
00178          JR      TEIL1           ;ZUM 2. MAL
00179 ;
00180 TEIL4    LD      B,E             ;1. L NACH E
00181          LD      C,L             ;2. L NACH C
00182          POP     DE
00183          POP     HL
00184          POP     AF
00185          RET
```

Mehr Fernsehen für's Geld

Das Genie III s (und vermutlich einige andere Genies "darunter") arbeitet im Gegensatz zum Genie I/II mit einem Video-Controller (CRTC, cathode ray tube controller). Der hat eine Reihe von Registern, mit denen gewisse Bildschirmzustände programmiert werden. G-DOS 2.4 unterstützt lt. Handbuch diese Möglichkeiten. In einem Falle ist das aber pure Hochstapelei: Der Befehl ## soll angeblich in der Lage sein, u. a. die beiden Bildschirmseiten des Formats 16*64 Zeichen untereinander auszutauschen. Er hält sich aber vornehm zurück. Jedenfalls ist mir dergleichen noch nie gelungen. Bis heute.

Dazu ein wenig Theorie: In den Registern 12 und 13 des CRTC ist die relative Startadresse des angezeigten Bildschirms im Verhältnis zur Startadresse des Video-RAMs abgelegt. Dieses beginnt bei 3800h. Beim Kleinbildschirm 16*64 liegt die erste angezeigte Adresse bei 3C00h, also 0400h Bytes hoch im Video-RAM. Daher wird beim Umschalten auf diesen "Kleinbildschirm" 0400h in die Register 12 und 13 des Controllers eingeschrieben. Dazu wird zunächst 0Ch (12d) auf den Port F6h ausgegeben, um das Register Nr. 12 zu adressieren. Der Port F7h kriegt dann das MSB dieses Anzeige-Offsets mitgeteilt. Anschließend wird über F6h das Register 13 aufgeweckt, um über F7h das LSB ausgeben zu können.

Um nun eine Bildschirmseite in die andere zu kopieren, beide auszutauschen, die eine oder andere vollzumalen usw., muß demnach das MSB des Anzeige-Offsets entweder 00h oder 04h lauten. Auf diese Weise wird die obere oder die untere Bildschirmseite adressiert. Den CRTC interessiert nämlich nicht die Bohne, wo die Designer des G-DOS oder des Genie den Bildschirm gerne memory mapped haben möchten. Er schreibt stur dorthin, wo es ihm seine Register 12 und 13 vorschlagen. Schreibt man nun nach Ausgabe des Offsets 0000h einen Text ab 3800h (Großbildschirm) und gibt dann den Offset 0400h aus (Kleinbildschirm), dann ist der Text verschwunden, weil nicht dort gelesen wird, wo er steht.

Das ist das Funktionsprinzip der beiden Bildschirmseiten, nach dem der ##-Befehl - angeblich, leider nicht wirklich - arbeitet. Um in einem Assembler- oder BASIC-Programm dennoch dieses interessante Feature des CRTC nutzen zu können, möchte ich hier ein Strickmuster vorstellen, nach dem der Bildschirm dem User aus der Hand frißt. Weil sie einfacher zu verstehen ist, zunächst die BASIC-Variante:

In der Zeile 10 wird mit dem Library-Befehl ##,V der Bildschirm auf 32*64 Zeichen eingestellt. Er liegt nun im Bereich 3800-3FFFh. CLS löscht ihn. Links oben wird der Text mit dem Hinweis auf die Seite 0 angezeigt. Mit dem Library-Befehl 64 wird auf den Bildschirm mit 16*64 Zeichen umgeschaltet (3C00-3FFFh). Wieder links oben, aber ein ganzes Kilobyte höher wird der Hinweis auf die Seite 1 geprintet. Danach wird das Offset-Register 12 adressiert. Es genügt nämlich, das MSB des Offsets zu verändern. Das LSB ist immer 00h. In der Zeile 20 wird eben dieser Offset abwechselnd zu 0000h oder zu 0400h definiert. Dazwischen liegt jeweils eine Verzögerungsschleife, um den Effekt beobachten zu können: Beide zuvor in den Bildschirm geladenen Texte kommen abwechselnd zur Anzeige.

Das darüber gelistete Maschinenprogramm tut genau dasselbe. Die ausführlichen Kommentare erklären gewiß, was sich da tut. Ich möchte darüber hinaus lediglich noch auf die Unterprogramme eingehen, die gecallt werden:

Das UP an 3497h schaltet auf die maximale Auflösung mit 32*64 Zeichen. Dazu holt es die entsprechenden CRTC-Parameter aus 32F0h ff. und gibt sie an den Controller aus. 34A1h schaltet auf die gleiche Weise auf 16*64 Zeichen, die kleinste Auflösung. Der Vollständigkeit halber seien noch die beiden Unterprogramme für die zwei verbleibenden Bildschirmformate genannt: 34A6h schaltet auf 24*64, 3490h auf 25*80 Zeichen. Das UP an 01C9h löscht den Bildschirm bei jedem Format. Mit der Routine an 44e7h wird ein Text, auf den HL zeigt, auf den Bildschirm übertragen. Er muß mit 0Ch oder 00h abgeschlossen sein.

Beide Programmvarianten lassen sich mit den von Fall zu Fall erforderlichen Modifikationen in eigene Programme einbauen. Eine denkbare Anwendung: Auf Knopfdruck soll der User ein Menü angezeigt bekommen. Dazu wird nach Umschalten auf den Großbildschirm das Menü einfach mit PRINT in den Bildschirm geladen. Nach Rückkehr zum Kleinbildschirm bleibt dieser Text erhalten. Er wartet sozusagen im Keller. Bei Bedarf braucht nur das Register 12 des CRTC über Port F6h adressiert zu werden, damit man sodann 00 auf Port F7h ausgeben kann. Damit ist der Bildschirm ab 3800h definiert. Der versteckte Text erscheint, der zuletzt angezeigte verschwindet. Nach Ausgabe von 04h auf Port F7h werden beide Bildschirmseiten wieder zurückgetauscht. Es ist, als wäre nichts geschehen.

Arnulf Sopp

```
5200                    00001           ORG     5200h
5200  CD9734            00002 start     CALL    3497h           ;auf 32*64 Z. schalten
5203  CDC901            00003           CALL    01c9h           ;Bildsch. ab 3800 löschen
5206  215B52            00004           LD      HL,text0        ;Text "Seite 0"
5209  CD6744            00005           CALL    4467h           ;anzeigen
520C  CDA134            00006           CALL    34a1h           ;auf 16*64 Z. schalten
520F  214A52            00007           LD      HL,text1        ;Text "Seite 1"
5212  CD6744            00008           CALL    4467h           ;anzeigen
5215  3E0C              00009           LD      A,0ch           ;Anz.-Offset-Reg. d. CRTC
5217  D3F6              00010           OUT     (0f6h),A        ;adressieren
5219  AF                00011           XOR     A               ;A <- 00 (Offset-Byte)
521A  F5                00012 loop      PUSH    AF              ;Akku retten
521B  01FFFF            00013           LD      BC,0ffffh       ;Zähler für Warteschleife
521E  CD6000            00014           CALL    0060h           ;etwas trödeln
5221  F1                00015           POP     AF              ;Akku zurück
5222  EE04              00016           XOR     04h             ;Offset-Bit umschalten
5224  D3F7              00017           OUT     (0f7h),A        ;neu ausgeben
5226  18F2              00018           JR      loop            ;und so weiter
5228  44                00019 text0     DM      'Dies ist Seite 0 des Bildschirms.',0Dh
524A  55                00020 text1     DM      'Und dieses hier ist Seite 1.',0Dh
                        00021 endprg
5200                    00022           END     start

00000 Fehler
```

Und hier dasselbe in BASIC:

```
10 CMD"##,V":CLS:PRINT"Dies ist Seite 0 des Bildschirms.":CMD"64":
PRINT"Und dieses hier ist Seite 1.":OUT&HF6,12
'Texte schreiben, Anzeige-Offset-Register adressieren

20 FOR I=0TO500:NEXT:OUT&HF7,4:FOR I=0TO500:NEXT:OUT&HF7,0:GOTO20
'Zwischen den Offsets 0000 und 0400 hin- und herschalten
```

```
1 'ASCIITAB VON GERALD SCHRÖDER
2 'DIESES PROGRAMM GIBT EINE VIERSPALTIGE TABELLE (64 ZEILEN) AUF DEM
3 'DRUCKER AUS. JEDE SPALTE HAT DAS FORMAT:
4 ' DEZIMALZAHL   HEXADEZIMALZAHL   BINÄRZAHL   ASCII-CODE
5 'DIE TABELLE     KANN VIELSEITIG BEIM PROGRAMMIEREN GENUTZT WERDEN.
6 'VIELEN DANK AN KAJOT FÜR DIE DEZ/HEX-ROUTINE
7 'ACHTUNG: ZEILE 20 MUß JE NACH DRUCKER GEÄNDERT WERDEN!
8 '
10 CLS:CLEAR70:DEFINTA,I-N:DEFSTRH,X:RESTORE
20 LPRINTCHR$(15) '132 ZEICHEN/ZEILE BEI GEMINI 10X
30 FORDA=0TO63:FORDB=0TO3:DE=DA+DB*64:LPRINTUSING"###":DE;:LPRINT" ";
40 ' *** HEXADEZIMALWERT ***
50 I=0:D1=DE
60 D2=INT(D1/16):B(I)=D1-D2*16:IFD2>0THENI=I+1:D1=D2:GOTO60
70 FORJ=0TOI:IFB(J)<10THENX(J)=RIGHT$(STR$(B(J)),LEN(STR$(B(J)))-1):ELSEX
(J)=CHR$(B(J)+55)
80 NEXTJ:HE="":FORJ=ITO0STEP-1:HE=HE+X(J):NEXTJ:N=2-LEN(HE):NU$=STRING$(N
,"0")
90 LPRINTNU$+HE;" ";
100 ' *** BINÄRWERT ***
110 I=0:D1=DE
120 D2=INT(D1/2):B(I)=D1-D2*2:IFD2>0THENI=I+1:D1=D2:GOTO120
130 FORJ=0TOI:X(J)=RIGHT$(STR$(B(J)),1)
140 NEXTJ:HE="":FORJ=ITO0STEP-1:HE=HE+X(J):NEXTJ:N=8-LEN(HE):NU$=STRING$(
N,"0")
150 LPRINTNU$+HE;" ";
160 ' *** ASCII-CODE ***
170 IFDE>32ANDDE<127THENX=CHR$(DE)ELSEIFDE<128THENREADXELSEX=" "
180 LPRINTUSING"%   %";X;:IFDB<3THENLPRINT"      ";
190 NEXTDB:LPRINT:NEXTDA:END
200 ' *** DATA'S FÜR ASCII-CODE ***
210 DATANUL,SOH,STX,ETX,EOT,ENQ,ACK,BEL,BS,HT,LF,VT,FF,CR,SO,SI,DLE,DC1,D
C2,DC3,DC4,NAK,SYN,ETB,CAN,EM,SUB,ESC,FS,GS,RS,US,SPACE,DEL
```

87	57	01010111	W		151	97	10010111			215	D7	11010111		
88	58	01011000	X		152	98	10011000			216	D8	11011000		
89	59	01011001	Y		153	99	10011001			217	D9	11011001		
90	5A	01011010	Z		154	9A	10011010			218	DA	11011010		
91	5B	01011011	Ä		155	9B	10011011			219	DB	11011011		
92	5C	01011100	Ö		156	9C	10011100			220	DC	11011100		
93	5D	01011101	Ü		157	9D	10011101			221	DD	11011101		
94	5E	01011110	^		158	9E	10011110			222	DE	11011110		
95	5F	01011111	_		159	9F	10011111			223	DF	11011111		
96	60	01100000	'		160	A0	10100000			224	E0	11100000		
97	61	01100001	a		161	A1	10100001			225	E1	11100001		
98	62	01100010	b		162	A2	10100010			226	E2	11100010		

HRG — Bildübertragung

Schon vor einiger Zeit hat mich Ulrich Böckling auf einen Service des WDR-Computerclubs aufmerksam gemacht. In der Mailbox dieses Clubs kann man neben sehr vielen anderen Informationen und Programmen seit einiger Zeit hochauflösende Bilder abrufen. Dies geschieht nach einem weiter unten genau beschriebenen Verfahren, auf das ich hier nicht weiter eingehen möchte.
Interessant finde ich die Möglichkeit, nun endlich Bilder vom Model 1 auf's Model 3/4 übertragen zu können und natürlich sind auch Genie I/II/III und IIs/IIIs -Bilder übertragbar. Vielleicht ein Grund mehr, sich eine HRG anzuschaffen!
Die für TRS 80 M.1 und M.3/4 sind übrigens ab sofort beim Kajott in der Programmbibliothek abrufbar. Wer HRG-Bilder mit mir tauschen möchte, kann sich direkt an mich wenden. Vielleicht entsteht ja mal eine ganze Sammlung von für alle Tandygeräte (und viele andere Computermodelle) lesbaren HRG-Bildern wie z.B. Schaltplänen, Funktionsgrafiken usw.. Ich würde mich sehr darüber freuen! Euer

Hartmut Ostermann

Bildübertragung per DFÜ

Als wir im Computerclub vor etwa einem Jahr mit der Installation unseres KOMCOM begannen, sah die Welt der Datenfernübertragung noch anders aus als heute. Damals war die Verbreitung von Akustikkopplern als recht dürftig zu bezeichnen. Datenfernübertragung war einigen wenigen Spezialisten vorbehalten. Die Hardware, wie Akustikkoppler, war noch immens teuer und die Software war auf dem Hobbymarkt kaum zu bekommen.
Dieses hat sich geändert - und zwar mit großer Geschwindigkeit. Heute sind viele unserer Zuschauer in der Lage, mit ihrem Heimcomputer DFÜ zu machen. Wir zählen in unserem KOMCOM nach etwa einem Jahr immerhin 65000 Anrufe. Wenn wir davon ausgehen, daß die Technik mithin bekannt ist, so können wir nun in der zweiten Phase damit beginnen, die Angebote in INFO-BOXEN etwas bunter zu gestalten. Es soll ein zusätzlicher Service sein, der dem einen Nutzer gefällt, dem zweiten Nutzer vielleicht nicht behagt.
In der Computerclubsendung vom 2.6. propagierten wir eine neue Möglichkeit, die Datenfernübertragung (auch im 300-Baud Modus) zur Übermittlung von Bildern zu nutzen. Vorweg gesagt: Zum Rennwagen können wir die Box nicht umfunktionieren. Die Übertragung eines hochaufgelösten Bildes vom Apple oder vom C64 dauert 7 Minuten, das Bild vom Viktor oder vom QX10 mit mehr Bildpunkten entsprechend länger.
Doch wir sehen schon einen Sinn in der Möglichkeit der Übertragung solcher Bilder. Zum Beispiel kann man sich Schaltbilder direkt nach Hause holen und sie dann ausdrucken. Die Realisierung kann dann sofort beginnen.
Stichwort Schaltbild. Gerade dies ist ein Beispiel, bei dem wir in Zukunft unserer Mailbox noch Beine machen können. Denn hier finden wir viele Flächen "schwarz" und einige Linien "weiß" vor. Durch Datenkompression (indem man am Anfang einer Zeile zum Beispiel sagt :200 Punkte schwarz) läßt sich hier noch ein erheblicher Verkürzungsfaktor in der Übertragung von Bildern einbauen. Diese Kompressionsmethode werden wir in einer der nächsten Sendungen vorstellen. Vorausgesetzt, das Übertragen von Bildern wird als Service überhaupt von Ihnen genutzt. In der letzten Sendung begannen wir damit, den von uns erdachten Standard vorzustellen. Wohlgemerkt, es kann ein vorläufiger Standard sein, wenn Ihnen dazu eine bessere Methode einfällt.

Nun zu unserem Prinzip:
Ein beschriebener Computerbildschirm liefert seine Information, indem alle dargestellten Zeichen als kleinste Einheit eine Anzahl von Bildpunkten benutzen. Bei dem einen Rechner ist der Bildschirm in xxx Punkte horizontal und yyy Punkte vertikal aufgeteilt. Ein zweiter Rechner benutzt ein ganz anderes Format. Die Auflösung des Bildschirms ist umso besser, je größer die Anzahl der Bildpunkte horizontal und vertikal ist. Dieses stellt natürlich ein Problem bei der übertragung dar. Wir könnten natürlich Basicprogramme mit Grafikbefehlen ablegen, zum Beispiel zeichnet der Befehl: LINE (1,1)-(300,1) eine waagrechte Linie ===> bei einigen Computern. Andere Computer wiederum benötigen zur gleichen Darstellung den PLOT - Befehl. Also hierüber ist keine Normierung zu finden!
Unser eingeschlagener Weg führt über eine "Krücke" zum Ziel. Wir scannen den Bildschirm Punkt für Punkt ab und legen den Inhalt als eine ASCII-Datei ab. Diese Datei können wir einfach übertragen und zu Hause kann sie mit einem kleinen BASIC- Programm wieder in den Grafikmodus umgewandelt werden. Wie gesagt: kein Rennwagen, dafür aber ein Auto, das jeder fahren kann.

Wie geht das?
Wir stellen uns einmal eine Linie auf dem Bildschirm vergrössert vor. Die könnte so aussehen:
.. .. u.s.w..
Nunmehr packen wir 6 Bildpunkte als eine Gruppe zusammen und geben dieser Gruppe einen bestimmten Zahlenwert. Aus dem Zahlenwert machen wir dann ein ASCII-Zeichen, also CHR$(Zahl). Dieses Zeichen legen wir in der Bilddatei ab.
Es kann dann übertragen werden und zu Hause wieder in eine Punktgruppe zurückverwandelt werden.
Nun einmal im Einzelnen:
Wir untersuchen die ersten 6 Bildpunkte einer Zeile. Sie mögen so aussehen :
1 2 3 4 5 6
. . . .
Nun geben wir jeder Stelle eine Wertigkeit nach dem Muster 2↑n: 1 2 3 4 5 6
. . . .
1 2 4 8 16 32 =Wertigkeit des Punktes
Wir addieren die Wertigkeit der gesetzten Punkte. Das macht:
1 + 4 + 8 + 32 = 45
Nur ein einziges Muster, nämlich das oben dargestellte, kann die Zahl 45 ergeben. Deshalb kann ich dies zu Hause auch wieder in ein eindeutiges Muster zerlegen. Wir müssen aber noch etwas berücksichtigen. Bei 6 Punkten "schwarz" würde die Addition 0+0+0+0+0+0 = 0 ergeben. Wenn wir dieses kleinstmögliche Ergebnis in der Datei ablegen und es später wieder auslesen, so sind wir damit im Bereich der Steuerzeichen, die wir möglichst vermeiden müssen. Trick: wir zählen überall 32, den Wert für daß Leerzeichen (SPACE), hinzu. Damit ergibt sich für unser Ergebnis 45 : 45+32 = 77. Umgesetzt in der ASCII-Tabelle steht hierfür das Zeichen "M". Dieses legen wir in der Bilddatei ab. Für alle Punkte "schwarz", also für die Addition 0 ergibt sich entsprechend 0+32 = 32, oder als Zeichen eben " " = SPACE.
Dieses waren die ersten 6 Bildpunkte. Nun kommen die nächsten 6 Bildpunkte. Dann die nächsten, bis die erste Zeile abgelegt ist. Dann kommt die nächste Zeile dran, dann die nächste usw. usw..

Unser Programm zur Rückverwandlung braucht also zwei Schleifen:
Für VERTIKAL = 0 TO ANZAHL BILDPUNKTE VERT.
Für HORIZONTAL = 0 TO ANZAHL BILDPUNKTE HORIZ.
Eine so abgelegte Zeile kann deshalb als Datei so aussehen:
M AN/ /))D %VF ZF%§§§
Jedem Buchstaben bzw. Zeichen entsprechen also 6 Bildpunkte.
Nun müssen wir noch ein gemeinsames Kopfformat festlegen, damit wir zum einen erkennen, daß es sich um eine Bildübertragung und nicht um eine Nachricht handelt. Zum anderen wollen wir automatisch das Bildschirmformat des abgelegten Bildes erkennen. Der Anfang der Datei ist deshalb folgendermaßen festgelegt:
BILD (in Grossbuchstaben Super-Calculator GTX (Zugspitze im Nebel) (Rechner + Beschreibung)
H300 (horizontale Bildpkte. mit führender Null)
V099 (vertikale Bildpkte. mit führender Null)
(leer, für spätere Erweiterung)
(leer, für spätere Erweiterung)
M §§§ % B T D %%%% (Daten zur Bildbeschreibung)
§ " FGT M BB)
dto.
dto.
§ A G T FE (JJJ E
(Carriage Return)
(Carriage Return)
Die beiden CR am Ende signalisieren das Ende der Bilddatei.

Soweit also zum Prinzip.
Wie komme ich aber später, wenn ich die Daten als Textfile zu Hause habe, wieder an die Wertigkeit der Punkte. Nun, hier kann man sich sicherlich seine eigenen Gedanken machen, doch man wird nach der schnellsten Methode suchen.
Hier mein Vorschlag zur Lösung:
Wir finden wieder das Zeichen "M" vor.
Der Befehl ASC("M") ergibt 77. Davon ziehen wir 32, unseren "Offset" ab. Damit sind wir wieder bei 45 gelandet. Diese Zahl wollen wir in ein eindeutiges Muster zurückverwandeln. Wir nehmen dazu die Vergleichsoperation AND. Schreiben Sie sich doch einmal das kleine Programm:

```
10 A=45
20 PRINT A AND 1
30 PRINT A AND 2
40 PRINT A AND 4
50 PRINT A AND 8
60 PRINT A AND 16
70 PRINT A AND 32
```

Nun, das Ergebnis wird folgendermaßen aussehen:
0 4 8 0 32
Genau das, was wir benötigen. Wenn wir das Ganze in IF-Abfragen kleiden und nach "wahr" oder "nicht wahr" abfragen, so können wir nun unsere Punkte auf dem Bildschirm setzen. Ich kann es hier nur für die Rechner machen, mit denen ich arbeite, bei denen ich die Grafikbefehle kenne (z.B. MBASIC oder GWBASIC). Hier gibt es den Befehl PSET (x,y) zum Setzen eines Punktes. Also sieht die Abfrage bei Einlesen des Zeichens "M", dann bei Umwandlung in den ASCII-Wert =77 und bei Subtraktion des Offsets von 32, also A=45 folgendermaßen aus:

```
IF A AND 1 THEN PSET (X+0,Y) (wahr, Punkt setzen)
IF A AND 2 THEN PSET (X+1,Y) (nicht wahr, Punkt nicht setzen)
IF A AND 4 THEN PSET (X+2,Y) (wahr, .....)
IF A AND 8 THEN PSET (X+3,Y) (wahr, .....)
IF A AND 16 THEN PSET (X+4,Y) (nicht wahr, .....)
IF A AND 32 THEN PSET (X+5,Y) (wahr, .....)
```

Bei anderen Rechnern muß eben der Befehl zum Setzen eines Punktes entsprechend geändert werden.

Ebenso verfährt man dann, wenn man ein Bild in eine Textdatei umwandeln möchte. Hier noch einmal das Format:

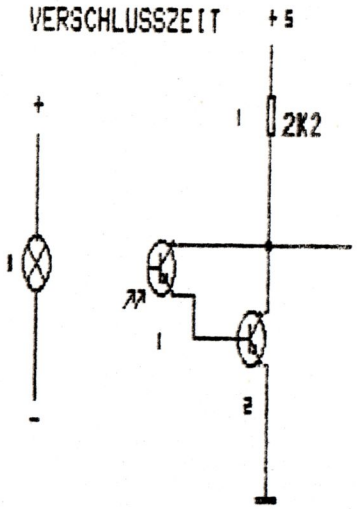
```
BILD
Beschreibung Rechner, Beschreibung Bild
Hnnn
Vnnn

Daten,Daten,CR+LF
dto.
dto.
dto.
Daten,Daten,CR+LF
CR
CR
```

Ein Beispiel wieder für MBASIC bzw. GWBASIC. Hier gibt es den Befehl zum Abfragen eines gesetzten Punktes, der POINT heißt. Also z.B. IF POINT (X,Y)=1 (bei GWBASIC) oder IF POINT (x,y)=7 (bei MFBASIC) gibt bei gesetztem Punkt das Ergebnis "wahr". Somit sieht das Programm dann grob so aus:

```
PRINT #1,"BILD"
PRINT #1,"RECHNER +BESCHREIBUNG"
PRINT #1,"H";H
PRINT #1,"V";V
PRINT #1,""
PRINT #1,""
FOR VERTIKAL = 0 TO V
FOR HORIZONTAL = 0 TO H STEP 6
IF POINT (VERTIKAL ,HORIZONTAL) =1 (7) THEN Z=Z+1
IF POINT (VERTIKAL+1,HORIZONTAL) =1 (7) THEN Z=Z+2
IF POINT (VERTIKAL+2,HORIZONTAL) =1 (7) THEN Z=Z+4
IF POINT (VERTIKAL+3,HORIZONTAL) =1 (7) THEN Z=Z+8
IF POINT (VERTIKAL+4,HORIZONTAL) =1 (7) THEN Z=Z+16
IF POINT (VERTIKAL+5,HORIZONTAL) =1 (7) THEN Z=Z+32
PRINT #1,CHRZ+32);
Z=0
NEXT HORIZONTAL
PRINT #1,""
NEXT VERTIKAL
PRINT #1,CHR13)
PRINT #1,CHR13)
```

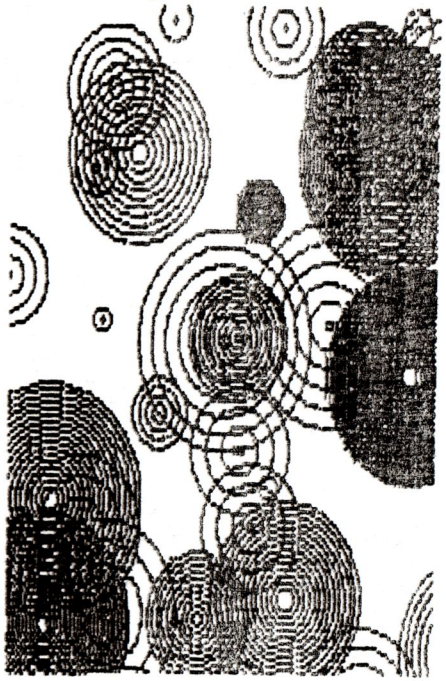
Man sollte nur mit INTEGER-Variablen arbeiten, um das Programm schneller zu machen. Also DEFINT oder Variable% benutzen. Beispiele für die Programmgestaltung für verschiedene Rechner findet man im Menuepunkt "BILDUEBERTRAGUNG" im KOMCOM. Wir hoffen natürlich, daß dieser Service Ihren Zuspruch findet und wir schon bald mit der Datenkompression beginnen können. Für Verbesserungsvorschläge sind wir auf jeden Fall dankbar.

Ihr Wolfgang Back.
(Dieser Beitrag wurde der WDR-Mailbox entnommen!)



LAMPE BPX 43 BC 187 AUSG.

Verschlusszeitenschaltung (C 64) und "Kreise" (Victor Sirius) aus der WDR-Mailbox (0221 / 371876)



Ein einfaches Grafik-Programm: PLOT/BAS (und seine Nachkommen)

Ich möchte Euch ein kleines Basic-Programm vorstellen, das einer meiner Freunde auf einem C64 auf meine Bitte hin entwickelt hat. Es zeichnet einen Ausschnitt einer dreidimensionalen Funktion mit (vom Beobachtungspunkt abhängigen) verdeckten Linien.

Funktionsweise: Die Achsen des Koordinatensystems sind: x = rechts/links, y = vorne/hinten, z = oben/unten. Vom jeweiligen Punkt (px/py/pz) wird eine Gerade zum Beobachtungspunkt (ox/oy/oz) gezogen. Die Gerade trifft bei y=-20 auf eine Ebene. Der Schnittpunkt wird mit (x/z) bezeichnet, aus dem sich je nach Lage des Ursprungs des Koordinatensystems (xx/zz) der auf dem Bildschirm zu setzende Punkt (sx/sz) berechnet.
Um festzustellen, ob ein Punkt sichtbar ist, schreitet der Computer die Gerade vom Punkt (px/py/pz) bis zum Schnittpunkt mit der Ebene ab. Findet er einen Punkt der Funktion auf der Gerade, so ist der Punkt (px/py/pz) nicht sichtbar und wird nicht gesetzt.

Programmierung:
Es werden folgende grafikspezifische Angaben benutzt, die je nach Grafikkarte und -treiber zu ändern sind:

HON	Grafikbildschirm einblenden
HCLS	" löschen
HMOVE sx,191-sz,2	Punkt invertieren (Grafik-Nullpunkt unten links)
HMOVE sx,191-sz,1	Punkt setzen
191	höchster Y-Wert (vertikal)
479	höchster X-Wert (horizontal)
100/240	Hälfte der obigen Werte

Programmlauf:
Zuerst wird abgefragt, wie groß der anzuzeigende Ausschnitt sein soll. Je tiefer die Funktion in den Raum angezeigt werden soll (y-Richtung) desto länger dauern die Berechnungen!
Es folgt die Abfrage des Beobachtungspunktes. Da die Projektionsebene fest ist, verzerrt sich das Bild mit wachsenden X-Werten erheblich. Empfohlene Größenordnung: 90,800,200 (Blick etwas von rechts, sehr weit entfernt, etwas von oben).
Das Berechnen dauert je nach Größe des Ausschnitts und Blickwinkel mehrere Stunden. Empfohlene Zeit für den Programmlauf: nachts! Zur Beschleunigung dient die Space-Taste: Wenn eine Linie offensichtlich sichtbar oder unsichtbar ist (auf blinkenden Punkt achten!), kann die Berechnung "verdeckt oder nicht" entfallen! Sie wird mit der Space-Taste übergangen. Der Punkt wird gesetzt, wenn der vorherige gesetzt wurde. Ist jemand bei einer verdeckten Linie zu weit vorgestürmt, kann er mit "0" (Taste lange festhalten) einige Punkte zurückspringen. Aber Achtung: schon gesetzte Linien werden nicht gelöscht!

Fragen an den Programmierer über Gerald Schröder

Idee und Programmierung: Stephen Arps
Text: Gerald Schröder

Das Programm: PLOT/BAS

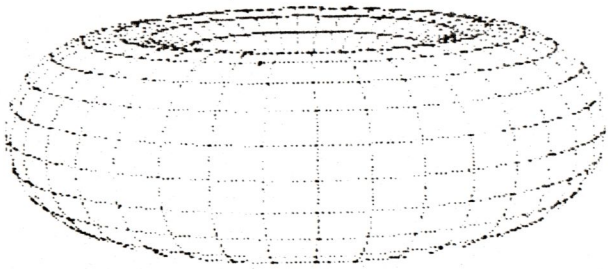
```
5 'PLOTTEN EINER FUNKTION
6 '1986 BY STEPHEN ARPS
7 '
10 CLS:HCLS
50 INPUT"X-RICHTUNG: VON ";VX
60 INPUT"            BIS ";BX
70 IFVX>BXTHEN50
80 INPUT"Y-RICHTUNG: VON ";VY
90 INPUT"            BIS ";BY
100 IFVY>BYTHEN80
110 PRINT"BEOBACHTUNGSPUNKT":INPUT"   X (RECHTS/LINKS)";OX:
    INPUT"   Y (VORNE/HINTEN)";OY:INPUT"   Z (OBEN / UNTEN)";OZ
120 PX=0:PY=0:PZ=0
130 GOSUB600
140 XX=240-X:ZZ=100+Z
150 '
200 HON:CLS
210 FORPY=BYTOVYSTEP-20
220 FORPX=VXTOBXSTEP3
225 IFINKEY$="0"THENPX=PX-6:IFPX<VXTHENPX=VX
230 GOSUB900:GOSUB500:NEXTPX,PY
250 '
300 FORPX=VXTOBXSTEP20
310 FORPY=BYTOVYSTEP-3
315 IFINKEY$="0"THENPY=PY+6:IFPY>BYTHENPY=BY
320 GOSUB900:GOSUB500
330 NEXTPY,PX
400 END
410 '
500 GOSUB600
505 SX=INT(XX+X+.5):SZ=INT(ZZ-Z+.5):IFSX<0ORSZ<0ORSX>479ORSZ>191THEN550
507 HMOVE SX,191-SZ,2
510 IFPEEK(14400)<>128THENGOSUB700
515 HMOVE SX,191-SZ,2
520 IFF=1THEN550
540 HMOVE SX,191-SZ,1
550 RETURN
560 '
600 YP=PY-BY-20:D!=YP/(OY-YP)
610 X=PX-(OX-PX)*D:Z=PZ-(OZ-PZ)*D
620 RETURN
630 '
700 DX=OX-PX:DY=OY-YP:DZ=OZ-PZ:LX=DX/DY:LZ=DZ/DY
710 S1=PX:S2=PY:S3=PZ
720 PY=YP+BY+21:PX=S1+LX
730 GOSUB900:C0=SGN(PZ-S3-LZ):IFC0=0THENF=1:GOTO795
740 LY=YP+1
750 HMOVE SX,191-SZ,2:ZW=LY-YP:PY=LY+BY+20:PX=S1+LX*ZW:HMOVE SX,191-SZ,2:
    IFPX<VXORPX>BXTHEN790
760 GOSUB900:C1=SGN(PZ-S3-LZ*ZW)
770 IFC1<>C0THENF=1:GOTO795
780 LY=LY+3:IFLY<-19THEN750
790 F=0
795 PX=S1:PY=S2:PZ=S3:RETURN
810 '
900 FU=SQR(PX*PX+PY*PY)
910 PZ=3A(4-FU/100)*COS(FU/25-.4)
920 RETURN
```

```
998 'ANDERE FUNKTIONEN
999 '
1000 PZ=PX*PX/200-150
1099 '
1100 PZ=PX*PX/200-PY*PY/200
1199 '
1200 PZ=(-(PX-100)*(PX-100)-PY*PY)/300+100:IFPZ<50THENPZ=50
1299 '
1300 PZ=SIN(PX/10)*SIN(PY/20)*10
1399 '
1400 PZ=3A(5-SQR(PX*PX+PY*PY)/50):IFPZ>120THENPZ=120
1499 '
1500 PZ=1.5*SQR(PX*PX+PY*PY)-100
1597 '
1598 '
1599 ' PYRAMIDE (VX=BY, VY=BY)
1600 AX=ABS(PX):AY=ABS(PY):PZ=AY-70:IFAX>AYTHENPZ=AX-70
```

Nun kann man nicht nur rechteckige Auschnitte von Funktionen plotten, sondern auch runde. Dazu müßt Ihr folgende Änderungen an dem vorstehenden Programm vornehmen:

RUNDPLOT/BAS '

```
5 'PLOTTEN EINES RUNDEN AUSSCHNITTS EINER FUNKTION
20 PI=3.14159
50 INPUT"RADIUS ";A:A2=A*A
60 '
70 '
80 '
90 '
100 '
210 FORRA=ATO1STEP-20
220 FORWI=0TO2*PISTEPPI/180
225 IFINKEY$="0"THENWI=WI-PI/90:IFWI<0THENWI=0
227 T1=SIN(WI):T2=COS(WI)
230 GOSUB895:GOSUB500:NEXTWI,RA
300 FORWI=PI/60TO2*PI+PI/60STEPPI/15
302 T1=SIN(WI):T2=COS(WI)
310 FORRA=ATO0STEP-3
315 IFINKEY$="0"THENRA=RA+6:IFRA>ATHENRA=A
320 GOSUB895:GOSUB500
330 NEXTRA,WI
600 YP=PY-A-20:D=YP/(DY-YP)
720 PY=YP+A+21:PX=S1+LX
750 HMOVE SX,191-SZ,2:ZW=LY-YP:PY=LY+A+20:GOSUB900:HMOVE SX,191-SZ,2:
    IFPX*PX+PY*PY>A2THEN790
760 C1=SGN(PZ-S3-LZ*ZW)
795 RETURN
895 PX=RA*T1:PY=RA*T2
```



Aber sicherlich wollt Ihr auch noch mehr als solche simplen mathematischen Funktionen plotten. Also noch was: der Torus. Am vorstehenden Programm RUNDPLOT müßt Ihr dazu folgendes ändern:

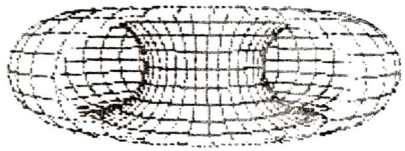
TORUS/BAS

```
5 'TORUS
10 CLS:HCLS:DEFINTA,F,K,O,R:PI!=3.14159
20 '
50 INPUT"RADIUS DES TORUS";A
60 INPUT"RADIUS DES TORUS-KÖRPERS";R
70 IFR>ATHEN50
80 R2=R*R:DI%=A+R+20
210 FORW2=0TO2*PISTEPPI/10
211 T1=R*SIN(W2):T2=R*COS(W2)+A
220 FORW1=0TO2*PISTEPPI/180
225 IFINKEY$="0"THENW1=W1-PI/90:IFW1<0THENW1=0
227 U1=SIN(W1):U2=COS(W1)
230 GOSUB800:GOSUB500:NEXTW1,W2
300 FORW1=PI/80TO2*PI+PI/80STEPPI/20
302 U1=SIN(W1):U2=COS(W1)
310 FORW2=0TO2*PISTEPPI/90
315 IFINKEY$="0"THENW2=W2-PI/45:IFW2<0THENW2=0
317 T1=R*SIN(W2):T2=R*COS(W2)+A
320 GOSUB800:GOSUB500
330 NEXTW2,W1
600 YP=PY-DI%:D=YP/(DY-YP)
720 '
730 '
750 HMOVESX,191-SZ,2:ZW=LY-YP:PY=LY+DI%:PX=S1+LX*ZW:GOSUB900:HMOVESX,191-SZ,2:
    IFK=0THENIFPZ>=S3+LZ*ZWTHENIF-PZ<=S3+LZ*ZWTHENF=1:GOTO795
760 '
770 '
800 PX=T2*U1:PY=T2*U2:PZ=T1:RETURN
900 FU!=SQR(PX*PX+PY*PY)-A:FU!=R2-FU!*FU!:IFFU!<0THENK=1:RETURN
910 K=0:PZ=SQR(FU!):RETURN
```

- alle folgenden Zeilen bis zum Ende löschen!

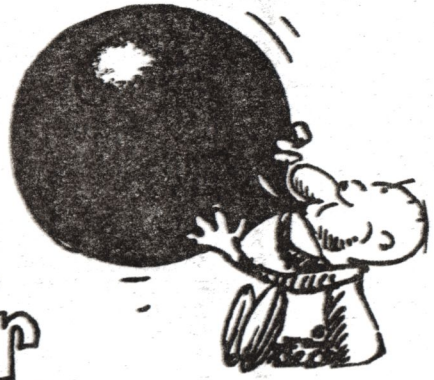
Nun kann man den Torus auch aufschneiden und in ihn hineinsehen. Das bewerkstelligt das folgende Programm. Ihr müßt diesmal die folgenden Zeilen in TORUS/BAS ändern!

TORUSINN/BAS



```
5 'TORUS AUFGESCHNITTEN
10 CLS:HCLS:PI=3.141
80 R2=R*R:DI=A+R/2
211 T1=R*SIN(W2):T2=A+R*COS(W2)
317 T1=R*SIN(W2):T2=A+R*COS(W2)
501 IFYP>0THENRETURN
600 YP=PY-DI:D=YP/(DY-YP)
750 PY=LY+DI+1:PX=S1+LX:GOSUB900:C0=0:IFK=0THENIFPZ>S3+LZTHENIF-PZ<S3+LZ
    THENC0=1
780 LY=YP+1
781 HMOVESX,191-SZ,2:ZW=LY-YP:PY=LY+DI:PX=S1+LX*ZW:GOSUB900:C1=0:
    HMOVESX,191-SZ,2:IFK=0THENIFPZ>S3+LZ*ZWTHENIF-PZ<S3+LZ*ZWTHENC1=1
782 IFC1<>C0THENF=1:GOTO795
783 LY=LY+3:IFLY<3THEN781
```

Ein Generalist fängt damit an, daß er **WENIG** über **VIELES** weiß. Im Laufe der **Zeit** weiß er immer weniger über immer mehr, bis er zuletzt **NICHTS** über **ALLES** weiß.



Ein Spezialist weiß am Anfang ziemlich **VIEL** über **ziemlich WENIG**. Nach und nach weiß er immer **MEHR** über immer **WENIGER**, bis er schließlich **ALLES** über **NICHTS** weiß.

(LORD ELTON)

Linienmodelle in 3 Dimensionen

Jeder von Euch kennt Baupläne. Dort sieht man einen Gegenstand meist mit einfachen Linien von mehreren Seiten. Das folgende Basic-Programm kann nun solche Pläne erstellen, indem es einen Gegenstand, den man aus Linien zusammensetzt, aus verschiedenen Blickwinkeln, von verschiedenen Standpunkten, in unterschiedlicher Größe etc. abbildet. Allerdings verschwinden die Linien, die "hinten" liegen, also verdeckt wären, nicht.
Das Programm entstand erst kurz vor Readaktionsschluß und ist noch dementsprechend kurz. Es enthält aber alle wichtigen Routinen. Der weitere Ausbau ist Geschmackssache.

Verwendet wurden folgende Grafikangaben, die abzuändern sind:

HON	Grafikbildschirm einblenden
HCLS	" löschen
HCOLOR 1	alle Linien "SET", also setzen
HPLOT qx,191-qz,0	Punkt als Startpunkt für eine Linie
HPLOT sx,191-sz,1	Linie vom vorherigen Startpunkt bis zu diesem Punkt ziehen
479	höchstzulässiger X-Wert
191	" Y-Wert
240/100	Hälfte der höchstzulässigen X- und Y-Werte

Disk-Ein/Ausgabe wurde für GDOS erstellt.

Idee und Programm: Stephen Arps (Original auf C64)
Text: Gerald Schröder

Fragen zu dem Programm leitet Gerald Schröder weiter.

LINPROG/BAS

```
1 'LINIEN-MODELLE ZEICHNEN
2 '1986 BY STEPHEN ARPS
3 '
10 DIMP1%(400,2),P2%(400,2)
20 CLS:PRINT"DATEN LADEN (J/N)? ";
30 GOSUB1500
40 IFJATHEN550ELSE60
50 '
60 CLS
70 PRINT"SIE MÜSSEN LINIEN DURCH JEWEILS EINEN ANFANGS- UND ENDPUNKT";
   PRINT"DEFINIEREN. AUS DIESEN LINIEN ENTSTEHT IHR LINIENMODELL."
75 PRINT"SIE MÜSSEN DIE KOORDINATEN IN DER FORM 'X,Y,Z' EINGEBEN.":
   PRINT"  X = HORIZONTAL":PRINT"  Y = IN DEN RAUM":PRINT"  Z = VERTIKAL":
   PRINT
80 PRINT"WENN SIE BEI 'VON' NUR <ENTER> EINGEBEN, WERDEN DIE WERTE DES":
   PRINT"LETZTEN 'BIS' GENOMMEN."
90 PRINT"DREI KOMMANDOS SIND BEI DER 'BIS'-EINGABE MÖGLICH:":
   PRINT" 'Z,0,0' = VORHERIGE LINIE NOCHMAL EINGEBEN.":
   PRINT" 'E,0,0' = EINGABE BEENDEN.":
   PRINT" 'S,0,0' = SPEICHERN DER BIS JETZT EINGEGEBENEN DATEN."
95 PRINT:INPUT"<ENTER>";FR$
100 NR%=0
110 NR%=NR%+1
120 PRINTNR%;TAB(5);". LINIE";
130 PRINTTAB(14);:INPUT"VON ";P1%(NR%,0),P1%(NR%,1),P1%(NR%,2)
140 P2$="":PRINTTAB(14);:INPUT"BIS ";P2$,P2%(NR%,1),P2%(NR%,2)
150 IFP2$="Z"THENNR%=NR%-2:GOTO190
160 IFP2$="E"THENNR%=NR%-1:GOTO210
170 IFP2$="S"THENNR%=NR%-1:GOSUB500:GOTO190
180 P2%(NR%,0)=VAL(P2$)
190 IFNR%<400THENP1%(NR%+1,0)=P2%(NR%,0):P1%(NR%+1,1)=P2%(NR%,1):
    P1%(NR%+1,2)=P2%(NR%,2):GOTO110
200 '
210 PRINTSTRING$(35,"-")
220 INPUT"BEOBACHTUNGSPUNKT (X,Y,Z) ";OX,OY,OZ
230 INPUT"X-/Z-VERSCHIEBUNG (X,Z) ";X,Z
240 XX=240-X:ZZ=100+Z
250 INPUT"GRÖßENFAKTOR ";GF:INPUT"DREHPUNKT (X,Y,Z) ";GX,GY,GZ
260 INPUT"WINKEL HORIZONTAL ";WH:INPUT"        VERTIKAL    ";WV
270 WH=WH*3.14159/180:WV=WV*3.14159/180:HS=SIN(WH):HC=COS(WH):VS=SIN(WV):
    VC=COS(WV)
280 '
290 HON:HCLS:CLS:HCOLOR 1
300 FORZA=1TONR%
310 PX=P1%(ZA,0):PY=P1%(ZA,1):PZ=P1%(ZA,2)
320 GOSUB1000
330 GOSUB790
340 S1=GF*X:S2=GF*Z
350 PX=P2%(ZA,0):PY=P2%(ZA,1):PZ=P2%(ZA,2):GOSUB1000
360 GOSUB790:X=GF*X:Z=GF*Z
370 GOSUB700
380 NEXTZA
390 '
400 IFINKEY$=""THEN400
410 HOFF 'Grafik ausblenden
420 PRINT"NOCHMAL (J/N)? ";
430 GOSUB1500
440 IFJATHEN210
450 PRINT"ABSPEICHERN (J/N)? ";
460 GOSUB1500
470 IFJATHENGOSUB500
480 END
490 '
491 '
492 ' FILE SPEICHERN
493 '
500 GOSUB640
510 OPEN"D",1,N$,"MU"
520 PUT1,,,NR%:FORKL=1TONR%:PUT1,,620
530 NEXTKL:CLOSE:RETURN
540 '
541 ' DATEN LADEN
542 '
550 GOSUB640
560 OPEN"I",1,N$,"MU"
570 GET1,,,NR%:FORKL=1TONR%:GET1,,620
580 NEXTKL:CLOSE:PRINT"ERWEITERUNG (J/N) ?";
590 GOSUB1500
600 IFJATHEN110ELSE210
609 '
610 'IGEL
611 '
620 P1%(KL,0),P1%(KL,1),P1%(KL,2),
630 P2%(KL,0),P2%(KL,1),P2%(KL,2);
635 '
636 'FILENAMEN EINGEBEN UND BEARBEITEN
637 '
```

```
640 INPUT"FILENAME OHNE EXTENSION ";N$:IFLEN(N$)>8THEN
    N$=LEFT$(N$,8)
650 IFINSTR(N$,"/")>0THENN$=LEFT$(N$,INSTR(N$,"/")-1)
660 N$=N$+"/LIN"
670 RETURN
690 '
691 ' LINIE ZIEHEN, SOWEIT SICHTBAR
692 '
700 QX=INT(XX+S1+.5):QZ=INT(ZZ-S2+.5)
710 SX=INT(XX+X+.5):SZ=INT(ZZ-Z+.5)
720 IFQX<0ORQZ<0ORQX>479ORQZ>191THEN840
730 IFSX<0ORSZ<0ORSX>479ORSZ>191THEN920
740 IFRO=1THENIFSX=QXTHENIFSX=QZTHENRETURN
750 QX=INT(QX):QZ=INT(QZ):SX=INT(SX):SZ=INT(SZ)
760 HPLOT QX,191-QZ,0:HPLOT SX,191-SZ,1
770 RETURN
780 '
790 D=PY/(QY-PY)
800 X=PX-(QX-PX)*D
810 Z=PZ-(QZ-PZ)*D
820 RETURN
830 '
840 RO=0:IFSX<0ORSZ<0ORSX>479ORSZ>191THENRO=1
850 GOSUB1090
860 IFQX<0THENGOSUB1120
870 IFQZ<0THENGOSUB1140
880 IFQX>479THENGOSUB1160
890 IFQZ>191THENGOSUB1180
900 GOTO730
910 '
920 IFQX<0ORQZ<0ORQX>479ORQZ>191THENRETURN
930 GOSUB1100
940 IFSX<0THENGOSUB1200
950 IFSZ<0THENGOSUB1220
960 IFSX>479THENGOSUB1240
970 IFSZ>191THENGOSUB1260
980 GOTO740
990 '
1000 I1=PX-GX:I2=PY-GY:I3=PZ-GZ
1010 ZX=I1*HC+I2*HS:ZY=I2*HC-I1*HS
1020 YZ=ZY*VC-I3*VS:CZ=ZY*VS+I3*VC
1030 PX=GX+ZX:PY=GY+YZ:PZ=CZ+GZ
1040 RETURN
1080 '
1090 K1=SZ-QZ:K2=SX-QX:K3=QX-479:K4=QZ-191:RETURN
1100 K1=QZ-SZ:K2=QX-SX:K3=SX-479:K4=SZ-191:RETURN
1110 '
1120 IFK2<>0THENQZ=QZ-QX*K1/K2:QX=0:GOTO1090
1130 RETURN
1140 IFK1<>0THENQX=QX-QZ*K2/K1:QZ=0:GOTO1090
1150 RETURN
1160 IFK2<>0THENQZ=QZ-K3*K1/K2:QX=479:GOTO1090
1170 RETURN
1180 IFK1<>0THENQX=QX-K4*K2/K1:QZ=191
1190 RETURN
1200 IFK2<>0THENSZ=SZ-SX*K1/K2:SX=0:GOTO1100
1210 RETURN
1220 IFK1<>0THENSX=SX-SZ*K2/K1:SZ=0:GOTO1100
1230 RETURN
1240 IFK2<>0THENSZ=SZ-K3*K1/K2:SX=479:GOTO1100
1250 RETURN
1260 IFK1<>0THENSX=SX-K4*K2/K1:SZ=191
1270 RETURN
1490 '
1491 'EINGABE JA/NEIN; BEI JA IST VARIABLE JA=TRUE
1492 '
1500 FR$=LEFT$(INKEY$+"$",1):IFINSTR("J,Nn",FR$)=0THEN1500
1510 PRINTFR$;IFINSTR("J,",FR$)=0THENJA=0ELSEJA=1
1520 RETURN
```

Und noch ein kleines Beispiel (jeweils "VON" / "BIS"):

20	-20	-110	50	-50	-110
50	-50	-110	50	-250	-110
50	-250	-110	30	-250	-110
30	-250	-110	30	-310	-110
30	-310	-110	-30	-310	-110
-30	-310	-110	-30	-250	-110
-30	-250	-110	-50	-250	-110
-50	-250	-110	-50	-50	-110
-50	-50	-110	-20	-20	-110
-20	-20	-110	20	-20	-110
20	-20	-110	20	-20	-30
50	-50	-110	50	-50	-30
50	-250	-110	50	-250	-30
30	-250	-110	30	-250	30
30	-310	-110	30	-310	30
-30	-310	-110	-30	-310	30
-30	-250	-110	-30	-250	30
-50	-250	-110	-50	-250	-30
-50	-50	-110	-50	-50	-30
-20	-20	-110	-20	-20	-30
20	-20	-30	50	-50	-30
50	-50	-30	50	-250	-30
30	-250	30	30	-310	30
30	-310	30	-30	-310	30
-30	-310	30	-30	-250	30
-30	-250	30	30	-250	30
-50	-250	-30	-50	-50	-30
-50	-50	-30	-20	-20	-30
-20	-20	-30	20	-20	-30
30	-250	30	0	-280	130
30	-310	30	0	-280	130
-30	-310	30	0	-280	130
-30	-250	30	0	-280	130
50	-250	-30	0	-250	30
-50	-250	-30	0	-250	30
0	-250	30	0	-70	30
0	-70	30	-50	-50	-30
0	-70	30	-20	-20	-30
0	-70	30	20	-20	-30
0	-70	30	50	-50	-30

Empfohlene Werte:
Beobachtungspunkt 0,200,0
Drehpunkt 0,-165,0
Verschiebung 0,-10
Vergrößerung 1.5
Winkel beliebig

Wenn die Uhr mal stört

Der UHR-Befehl unseres DOS ist gewiß eine feine Sache, aber die angezeigte Zeit übertüncht natürlich alles, was in der rechten oberen Bildschirmecke steht. Meist stört das nicht besonders. Im Augenblick aber läuft auf meiner Maschine TSCRIPS, und da hätte ich rechts oben ganz gerne den Text anstatt der Zeit. Rechts unten in der Kommandozeile ist dafür Platz genug. Es sind eine Menge weiterer Fälle denkbar, wo das wünschenswert ist.

Hier möchte ich zwei Versionen eines Programms vorstellen, das die Uhrzeit am Ende des Bildschirms anzeigen läßt. Um es ein bißchen komfortabler zu machen, hat es noch drei weitere Fähigkeiten. Gleich zu Beginn prüft es, ob beim Aufruf mit dem Namen UHRUNTEN noch ein Parameter angehängt wurde, z. B. UHRUNTEN,N für "nein". Ist das der Fall, so wird der Patch rückgängig gemacht und die Anzeigestelle wieder an die obere Bildschirmecke gerückt. Dabei ist es übrigens gleichgültig, welches Zeichen man dem Aufruf nachstellt. Zweitens wird automatisch der DOS-Befehl UHR durchgeführt; die Anzeige, wo auch immer, erfolgt also in jedem Falle.

Und drittens wird ein Mangel behoben, wo die Designer von NEWDOS-80 offenbar gepennt haben: Die Uhrzeit wird nicht ganz rechts angezeigt, sondern drei Stellen in die Zeile eingerückt. Das ist zwar nicht viel, aber es ist nicht einzusehen, was das soll. Deshalb wird - oben oder unten - der äußerste rechte Rand zur Anzeige benutzt. Das bleibt auch so, wenn mit UHRUNTEN,N wieder auf Anzeige oben zurückgeschaltet wird. Mit einem einfachen Patch kann das übrigens zurechtgebogen werden, ohne UHRUNTEN zu fahren:

Für das Genie I und seine Geschwister:
in SYS0/SYS, rel. Sekt. 01, Byte 92 von 35 auf 38 zappen

Für das Genie III s:
in DVL4/SYS, rel. Sekt. 09, Byte 53 von 08 auf 08 zappen

Zunächst möchte ich die für beide Versionen gleichen Teile erläutern. Der CALL nach 4CD5h überprüft, ob nach dem Programmnamen noch ein Parameter folgt. Wenn das der Fall ist, kehrt 4CD5h mit der NZ-Bedingung zurück, andernfalls mit Z. Das Entsprechende kann nun ablaufen. Am Ende des Programms wird HL als Zeiger auf das 0Dh-Byte (das entspricht ENTER) geladen. Was das bedeutet, kommt gleich. Danach werden C mit 02h und A mit E5h geladen. E5h ist der Requestcode, der über RST 28h SYS3/SYS aufruft. Der Inhalt von C entscheidet mit darüber, welche der Routinen aus SYS3/SYS angesprungen wird. Für UHR muß es 02h sein. Beim Einsprung in die UHR-Routine wird zuerst ebenfalls 4CD5h geCALLt, um die Parameter zu checken. Wenn in (HL) (Zeiger auf das Ende des Aufrufbefehls) nur ENTER (0Dh) steht, bedeutet das dasselbe wie UHR,J. Deshalb wird mit RST 28h die Uhr nun auf jeden Fall angezeigt, wenn HL auf 0Dh steht.

Nun zur Version für das Genie III s. Ab 3594h steht die Routine zur Anzeige der Uhrzeit. Dort wird u. a. die je nach Bildschirmformat zutreffende Stelle errechnet. Dabei werden vom Beginn der zweiten Zeile 11 Bytes subtrahiert, um nicht ganz ans Ende der ersten zu gelangen. Dieser Subtrahend steht in 35A3h. Dort schreibt das Programm nun stattdessen 8 ein, dazit das äußerste Ende benutzt wird. An der Stelle 359Dh steht der Befehl LD DE,(3408h). In 3408h ist die Zeilenlänge je nach Bildschirmformat abgelegt. Die Summe von Bildschirmstart und Zeilenlänge ergibt den Beginn der zweiten Zeile (s. o.).

Zur Anzeige rechts unten muß aber die Summe von Bildschirmanfang und seiner Länge errechnet werden, um das Bildschirmende zu erhalten. Die Länge steht in 3408h. Beide Adressen unterscheiden sich nur im LSB, das entweder 00h (oben) oder 0Eh (unten) lautet. Daher genügt es, wenn der Akku in Zeile 7 mit 00h oder in Zeile 10 mit 0Eh geladen wird, je nach dem. Dieses LSB wird nun nach 359Fh gepatcht.

Beim Genie I läuft es etwas anders. An der Stelle 44A4h wird HL als Zeiger auf die (fast) rechte obere Bildschirmecke, nämlich 3C3Eh geladen. Wir nehmen lieber 3C3Fh, um ganz rechts zu landen. Rechts unten lautet diese Adresse 3FFFh. Je nach Wunsch wird nun eine dieser beiden Adressen

nach DE geladen. In der Folge (Zeilen 11-13) patcht das Programm die gewünschte Adresse nach 44A5h, wo der Operand des Ladebefehls steht.

Etliche Clubmitglieder haben die 80-Zeichen-Karte von RB-Elektronik. Es wäre interessant, im nächsten Info eine Version zu finden, die mit dieser Karte läuft. Da es beim Kontek, dem Genie III (ohne "s") und ähnlichen Computern auch Unterschiede geben dürfte, sind wir auf weitere Versionen gespannt.

```
                00001 ;                 UHRUNTEN/CMD
                00002 ; Programm zur Anzeige der Uhrzeit in der unteren Ecke
                00003 ; Version für Genie I, TRS-80 usw.:
                00004
5200            00005           ORG     5200h
5200  CDD54C    00006 start     CALL    4cd5h           ;folgt ein Parameter?
5203  11F83F    00007           LD      DE,3ff8h        ;Anz. am Bildschirmende
5206  2803      00008           JR      Z,unten         ;falls kein Parameter
5208  113E3C    00009           LD      DE,3c3Eh        ;rechte obere Ecke
520B  21A544    00010 unten     LD      HL,44a5h        ;Adresse für Ladebefehl
520E  73        00011           LD      (HL),E          ;Adreß-LSB des Ladebef.
520F  23        00012           INC     HL              ;auf MSB erhöhen
5210  72        00013           LD      (HL),D          ;MSB patchen
5211  211A52    00014           LD      HL,cr           ;ENTER für UHR-Befehl
5214  0E02      00015           LD      C,02h           ;Zeiger auf UHR in SYS3
5216  3EE5      00016           LD      A,0e5h          ;Requestcode für SYS3
5218  EF        00017           RST     28H             ;aufrufen (= UHR<ENTER>)
5219  C9        00018           RET                     ;zur. ins Betriebssystem
521A  0D        00019 cr        DB      0dh             ;ENTER für UHR-Befehl
5200            00020           END     start

00000 Fehler
```

```
                00001 ;                 UHRUNTEN/CMD
                00002 ; Programm zur Anzeige der Uhrzeit in der unteren Ecke
                00003 ; Version für das Genie III s:
                00004
5200            00005           ORG     5200h
5200  CDD54C    00006 start     CALL    4cd5h           ;folgt ein Parameter?
5203  3E08      00007           LD      A,08h           ;für 3408h und 8 Stellen
5205  32A335    00008           LD      (35a3h),A       ;8 Stellen vor Zeilenende
5208  2802      00009           JR      Z,unten         ;falls kein Parameter
520A  3E08      00010           LD      A,08h           ;DE aus 4308h laden
520C  329F35    00011 unten     LD      (359fh),A       ;Adresse für Ladebefehl
520F  211852    00012           LD      HL,cr           ;ENTER für UHR-Befehl
5212  0E02      00013           LD      C,02h           ;Zeiger auf UHR in SYS3
5214  3EE5      00014           LD      A,0e5h          ;Requestcode für SYS3
5216  EF        00015           RST     28h             ;aufrufen (= UHR<ENTER>)
5217  C9        00016           RET                     ;zur. ins Betriebssystem
5218  0D        00017 cr        DB      0dh             ;ENTER für UHR-Befehl
5200            00018           END     start

00000 Fehler
```

Arhulf Sopp

Nachtrag zu UHRUNTEN/CMD

Bei meinem Programm zur Anzeige der Uhrzeit in der rechten unteren Ecke des Bildschirms habe ich gleich zweimal gepennt. Das soll jetzt richtiggestellt werden:

Der RET-Befehl nach dem RST 28h erubrigt sich, weil RST 28h, obgleich ein Unterprogrammaufruf (wie CALL bzw. GOSUB), wie ein JP (entspr. GOTO) behandelt wird. Das RET stort aber auch nicht weiter.

Viel gravierender ist ein anderer Fehler, der aber nur in der Version für das Genie III s vorkommt: Bei dem Bildschirmformat 24*64 Zeichen erfolgt die Anzeige nicht rechts unten, sondern in einer mittleren Zeile, je nach dem. Das ist mit dem anschließend vorgestellten Listing behoben. Dieses arbeitet bei allen möglichen Bildschirmformaten fehlerfrei. Auf die Programmlogik will ich jetzt nicht eingehen. Die drei Patches, die das Programm setzt, erklären sich aus den nachfolgenden Erläuterungen, hoffe ich.

Das DOS des Genie III s holt seine Informationen über das Bildschirmformat nicht direkt vom Videocontroller, sondern aus dem RAM, wo die wichtigen Daten ähnlich wie in einem DCB niedergelegt sind. Ab 3400h finden wir folgende Daten:

3400/01	physikalische Anfangsadresse des Bildschirms
3402/03	mit PRINT, CLS usw. adressierbare Anfangsadresse
3404/05	physikalische Endadresse des Bildschirms +1
3406/07	wie oben adressierbare Endadresse +1
3408/09	Länge des wie oben adressierbaren Bildschirms
340A	unbenutzt?
340B	Länge einer Bildschirmzeile

Bei der Anzeige der Uhrzeit wird eine Routine angesprungen, die ich zum Verständnis (auch meines Programms) ab 359Ah vorstellen und kommentieren möchte:

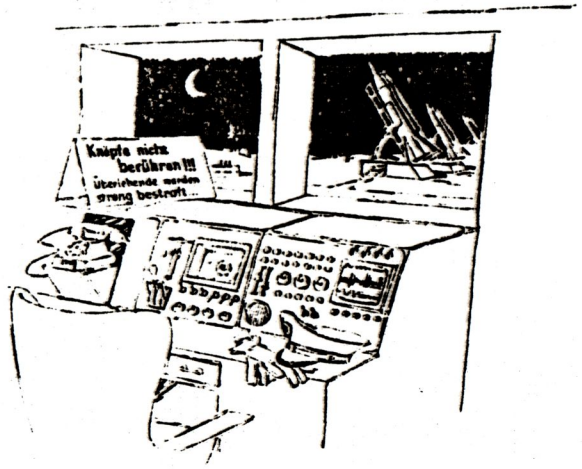
```
LD      HL,(3400h)      ;Anfangsadresse des Bildschirms
LD      DE,(340Bh)      ;Anzahl der Zeichen pro Zeile
ADD     HL,DE           ;ergibt Anfang der 2. Zeile
LD      DE,0008h        ;11 Bytes zurück
SBC     HL,DE           ;ergibt ca. Ende der 1. Zeile
```

Nun zeigt HL auf die Bildschirmstelle, ab wo die Uhrzeit erscheinen soll. Mein Programm subtrahiert nicht 11, sondern 8 Stellen vom Beginn der 2. Zeile, damit der äußerste rechte Rand erreicht wird. Wenn die Uhr unten erscheinen soll, wird nicht der Anfang, sondern eben das Ende des Bildschirms geladen. Dann werden ein paar Bytes übersprungen, um sofort die 8 Stellen abzuziehen.

Der oben beschriebene Fehler trat deswegen auf, weil die erste Version zur Anfangsadresse nicht die tatsächliche Länge des Bildschirms, sondern die mit den Standardbefehlen erreichbare Länge addierte, die in 3408/09h abgelegt ist. Da zeigt sich mal wieder, daß auch der Assembler-Fachidiot gelegentlich sein BASIC anschmeißen sollte, um Fehler zu bemerken, die eben nur unter BASIC auftreten. Und auf der folgenden Seite steht das korrigierte Programm.

Arnulf Sopp

```
                00001 ;       U H R U N T E N / C M D
                00002 ; Version für das Genie III s, korrigierte Fassung
                00003
5200            00004           ORG     5200h
5200 CDD540     00005 start     CALL    4cd5h           ;folgt ein Parameter?
5203 3E08       00006           LD      A,08h           ;8 Stellen vor Zeilenende
5205 32A335     00007           LD      (35a3h),A       ;für Anzeige patchen
5208 0F         00008           RRCA                    ;3404h als Adreßquelle
5209 211803     00009           LD      HL,0318h        ;= JR 07h
520C 2804       00010           JR      Z,unten         ;falls kein Parameter
520E AF         00011           XOR     A               ;3400h als Adreßquelle
520F 21ED5B     00012           LD      HL,5bedh        ;überschriebener Code
5212 329B35     00013 unten     LD      (359bh),A       ;Adresse für Ladebefehl
5215 229D35     00014           LD      (359dh),HL      ;Programmcode patchen
5218 212052     00015           LD      HL,cr           ;ENTER für UHR-Befehl
521B 0E02       00016           LD      C,02h           ;Zeiger auf UHR in SYS1
521D 3EE5       00017           LD      A,0e5h          ;Requestcode für SYS1
521F EF         00018           RST     28h             ;aufrufen (= UHRENTER)
5220 0D         00019 cr        DB      0dh             ;ENTER für UHR-Befehl
5200            00020           END     start

00000 Fehler

cr    5220   start 5200   unten 5212
```



Scripsit auf dem "alten" Genie

Schon auf dem Clubtreffen im April wurde ich durch unser Clubmitglied Hans-Martin Stephan auf ein Problem hingewiesen, daß einige Besitzer von VideoGenie 1 - Geräten älterer Bauart haben.

Die ersten Nachbauten des guten alten TRS80 Model 1 haben von Ihren fernöstlichen Geburtshelfern statt eines "memory-mapped" Druckeranschlusses (wie ihn der TRS80 besaß) einen "portgesteuerten" erhalten. Dies hat den Nachteil, daß Programme, die direkt auf die TRS80-Hardware (und in diesem speziellen Fall auf die Druckeradresse 37E8h) zugreifen, auf diesen Geräten nicht laufen! Diesen Mangel stellte man schon bald ab, indem man späteren Auflagen des VideoGenie eine "doppelte" Zugriffsmöglichkeit einbaute. Bei "neueren" VideoGenie 1 - Modellen kann man den Drucker sowohl unter der Memoryadresse 14312 (37E8h) als auch unter der Portadresse 253 (FDh) erreichen.

Den Besitzern der ersten Genies bleibt aber das Problem erhalten, daß z.B. Scripsit auf ihren Maschinen nicht läuft. Abhilfe kann man da auf zwei Arten schaffen:
1. nachträglicher Einbau der Decodierung der Adresse 37E8h
2. Anpassung der Programme an das "abnorme" Gerät!
Obwohl die erste Möglichkeit durchaus realisierbar ist, wird wohl meist die Softwarelösung bevorzugt, die ich aus diesem Grunde hier ausführlich erläutern möchte.

LD contra IN/OUT

LD A,(37E8h) und LD (37E8h),A sind normalerweise die Befehle, mit denen der memory-mapped Druckeranschluß des TRS80 abgefragt bzw. beschickt wird. Mit dem ersten Befehl kann der Druckerstatus abgefragt werden. Der zweite gibt ein im Accu zwischengespeichertes Zeichen an den Drucker aus. Aus den Memoniks (so nennt man die Befehlsworte im Assemblerformat) werden nach der Assemblierung folgende Maschinencodes: LD A,(37E8h) --> 3AE837
LD (37E8H),A --> 32E837
Diese Codes sind mit entsprechenden Hilfsprogrammen (FED, SuperZap usw.) in den umzuschreibenden Programmen sehr leicht zu finden und zu ändern!
IN A,(FDh) und OUT (FDh),A sind die entsprechenden Befehle, die eingesetzt werden müssen. Die Maschinencodes der beiden Befehle sind nur zweistellig, so daß eine Änderung ohne Probleme durchführbar ist! Der Maschinencode 3AE837 ist in DBFD00 (IN A,(FDh) zu ändern, aus dem Code 32E837 wird D3FD00. Da die IN/OUT-Befehle, wie schon erwähnt, nur Zweibytebefehle sind, muß jeweils "00" (NOP = keine Operation) angehängt werden.

Finden und Ändern

Zum Finden der betreffenden Stellen in /CMD-Files und zum Ändern der Codes benutzt man am besten Superzap oder FED (Fileeditor). Der Diskdateneditor DDE aus GDOS ist für diese Aufgabe kaum brauchbar, da mit ihm keine Suchoperationen durchgeführt werden können.
Im Folgenden wird ein Such/Änderungsprozess mit Superzap im Detail durchgesprochen, damit auch diejenigen, die mit dem Zappen nicht so vertraut sind, ohne größere Probleme Programme anpassen können. Als Beispielprogramm nehmen wir SCRIPSIT.

1. Superzap laden.
2. DFS (Display File Sector) aufrufen.
3. Als Filenamen SCRIPSIT/CMD (oder wie das Programm sonst heißen mag) eingeben.
4. Auf die Frage nach dem "File relativ Sector" 0 eingeben.
5. Mit der Eingabereihenfolge "F,3A,E8,37" bzw. "F,32,E8,37" nach den zu ändernden Codes suchen und deren Position (File relativ Sector und relativ Byte im Sector) notieren.
6. Mit der Eingabe "F" nach weiteren Vorkommen des gesuchten Codes fahnden.
7. Wenn alle zu ändernden Stellen gefunden sind, kann mit der Änderung begonnen werden. Dazu den entsprechenden Sector mittels der Tasten "+ und -" aufsuchen und mit dem Kommando "Modxx" in den Änderungsmodus gehen. "xx" steht dabei für das relative Byte im Sector, welches zuvor beim Suchen notiert wurde.
8. Der Cursor blinkt jetzt an der zu ändernden Stelle und man kann die neuen Codes (siehe oben) eingeben. Mit <ENTER> wird die Änderung auf die Diskette geschrieben.
9. Den Änderungsvorgang so lange wiederholen, bis alle gefundenen Codes umgeschrieben sind!


```
DRV 00F E1C1 4704 D1F1 3235 7C21 D27D 7EC9 FE9E ..G...25è!.ûß..
1   10, 2014 1B1A 3C28 AF1B 1A13 3C20 0332 617C  ...<(....< .2a
1H  203 042B 362E 18A1 FE8E 2006 E1C1 D1C3 765E .+6..... .....v^
    30A CD2B 5FCB 0102 005F 7928 10FE 3A30 8CFE .+_....._y(..:0.
DRS 40, 2E28 08FE 2C28 04FE 3038 8012 0520 0404 .(..,(..08 .. ..
363 50E C393 5ECD 445F CB71 CA92 5E78 3DC2 925E .^.D_.q..^x=.^
16BH608 1318 95FE 2038 03FE 80D8 E521 7479 3281 .. . 8.. ..!ty2
    70, 7923 BE23 20FB 7EE1 C93A 307C 083A 637C y## .ß..:0ü.:cü
    803 CB7F CA8E 5FE6 60C0 FDCB 0E4E 280A CD84 . _'....N(..
    907 5F08 E67F D3EB 1814 CD04 603A E837 CB7F _.....`:.7
    A0  20F6 08FD CB34 6600 00C3 8A7A FE0D 2804  ....4f...z..(.
    B0  FE0A 2004 FD35 4BC9 FD35 4CC9 CD04 60DB .. ..5K..5L...`.
FRS C0  EACB 7728 F7C9 E60F 201C 08CB 7F28 16FE ..w(.... ..(..
13  D0  A030 10E5 2166 7932 7379 23BE 2320 FB7E  0..!fy2sy## .ß
DH  E0  E118 02CB BF77 23FD 3406 3A35 7CFE 40C9 ....w#.4.:5ü.@.
    F0  E506 41CD 725E E1CD 355A FE59 C8FE 4E20 ..A.r^..5Z.Y..N
```

Scripsit auch für Model 3

Die obenstehende Anleitung ermöglicht es übrigens auch Model 3 - Besitzern Scripsit und natürlich auch die Scripsitabkömmlinge wie z.B. TSCRIPS4 umzuschreiben. Beim Model 3 ist nur statt des Prots FDh der Port F8h einzusetzen.
Zusätzlich muß Scripsit mitgeteilt werden, daß die Speicherstelle in der der HIMEM abgelegt ist, sich von 4049h beim Model 1 auf 4411h beim Model 3 geändert hat. Ansonsten behauptet das Programm unrichtigerweise, daß nicht genügend Speicherplatz vorhanden sei.
Ein zusätzliches Problem stellt sich den Model 3 - Usern mit deutscher Tastatur. Scripsit benutzt nämlich seine eigene Treiberroutine und erzeugt damit eine schier unendliche Verwirrung auf der Tastatur. Dies zu Ändern bedeutet jedoch mehr als nur ein paar Zaps und sprengt den Rahmen dieses Beitrags!

So, ich glaube damit wäre einige Probleme aus der Welt geschafft!? Sollte es Fragen geben werde ich selbstverständlich versuchen sie zu klären, Euer

Reinhard Überreiter

Scripsit-Tabellen

Scripsit (oder Tscrips etc.) ist ziemlich offen aufgebaut, denn es benutzt möglichst oft Tabellen, die man meist ohne Probleme ändern kann. So wäre eine Umstellung von Englisch auf Deutsch nicht schwer, aber wie bei Newdos wohl eher unsinnig. Interessanter sind andere Bereiche, die durch einfache Zaps von jedem geändert werden können.

Als erstes wäre eine Tabelle zu nennen, die bei "BREAK P,I", also dem Ausdrucken mit Steuerzeichen, diese Steuerzeichen umwandelt. So wird bei mir u.a. aus der Paragraph-Markierung (auf dem Bildschirm "-") ein "$" auf dem Papier. Eine Verwendung, über die man streiten kann. Wer andere Sonderzeichen auf seinem Drucker ansteuern kann, sollte diese Tabelle bei 7446h vielleicht ändern. In der Tabelle steht immer das Scripsit-interne Steuerzeichen vor dem Zeichen, in das es zum Ausdrucken umgewandelt wird. Das stellt sich so dar:

```
M7446   DB      8Dh,'-'         ;Enter wird zu "-"
        DB      8Eh,'$'         ;Paragraph zu "$"
        DB      97h,'Ä'         ;Blockanfang zu "Ä"
        DB      9Bh,'Ü'         ;Blockende zu "Ü"
        DB      8Ch,'ö'         ;Seite zu "ö"
        DB      00h,' '         ;unbekannte Zeichen werden zu Space
```

Bei Bildschirmausgaben gibt es eine ähnliche Tabelle. Wer auf dem Bildschirm andere Zeichen als normal darstellen kann (z.B. mit der 80-Zeichen-Karte) sollte sie unbedingt ändern, denn "ö" und "Ä" sind sehr mißverständlich. Die Tabelle steht bei 7967h und sieht im Urzustand folgendermaßen aus:

```
M7967   DB      8Dh,8Ch         ;Enter in Grafikblock
        DB      97h,0B7h        ;Blockanfang in Grafik-"{"
        DB      9Bh,0BBh        ;Blockende in Grafik-"}"
        DB      8Eh,'-'         ;Paragraph in "-"
        DB      8Ch,'ö'         ;Seite in "ö"
        DB      8Bh,'Ä'         ;Copy (Break C) in "Ä"
        DB      00h,'Ü'         ;unbekanntes Zeichen in "Ü"
```

Nachdem man "BREAK" eingegeben hat, können zum Beispiel Zeichenfolgen gesucht werden (mit "F"). Um auch nach bestimmten Steuerzeichen suchen zu können, wurden einige Funktionstasten auch im Break-Modus weiter zugelassen bzw. umcodiert. Die Tabelle dazu sieht folgendermaßen aus:

```
M7975   DB      18h,8Dh         ;§X wird zur "Enter"-Markierung
        DB      03h,8Eh         ;§C wird zur "Paragraph"-M.
        DB      11h,97h         ;§Q wird zu Blockanfang
        DB      16h,8Ch         ;§V wird zur "Seite"-Markierung
        DB      1Ch,9Bh         ;§ DOWN ARROW wird zu Blockende
        DB      9Bh,8Bh         ;Hochpfeil wird zur "Copy"-Markierung
        DB      00h,20h         ;andere werden zu Space
```

Beim Ausdruck werden immer bestimmte Default-Druckparameter angenommen, die im Handbuch auch extra aufgeführt werden. Wem diese nicht passen (z.B. Arnulf und sein LM=0), der kann sie problemlos im Programm selbst zappen. Die Tabelle bei 7A15h ist dafür zuständig. Der Normalzustand:

```
M7A15   DB      3Ch             ;Zeichen/Zeile: 60 (s. auch LM/RM)
        DB      0Ch             ;LM=12  (Linker Rand)
        DB      48h             ;RM=72  (Rechter Rand)
        DB      01h             ;PF=1   (Paragraph-Einschub)
        DB      'Y'             ;J=Y    (Randausgleich an)
        DB      'N'             ;C=N    (keine Zentrierung)
        DB      'N'             ;FR=N   (nicht rechtsbündig schreiben)
        DB      'Y'             ;unbenutzter Parameter
        DB      'Y'             ;WS=Y   (Sinn mir unbekannt)
        DB      01h             ;LS=1   (eine Zwischenzeile)
        DB      06h             ;TM=6   (Druck beginnt in Zeile 6)
        DB      3Ch             ;BM=60  (Druck endet in Zeile 60)
        DB      'N'             ;VC=N   (keine vertikale Zentrierung)
        DB      42h             ;PL=66  (eine Seite hat 66 Zeilen)
        DB      80h             ;H=Y    (sollte nicht geändert werden)
        DB      80h             ;F=Y    (ebenso)
```

Sollte noch jemand spezielle Fragen haben, kann er sich gerne an mich wenden. Ich bin noch dabei, das Listing des Scripsit (bzw. Tscrips) zu kommentieren.

Gerald Schröder

Sicher ist sicher ...

Freiräume im Tscrips

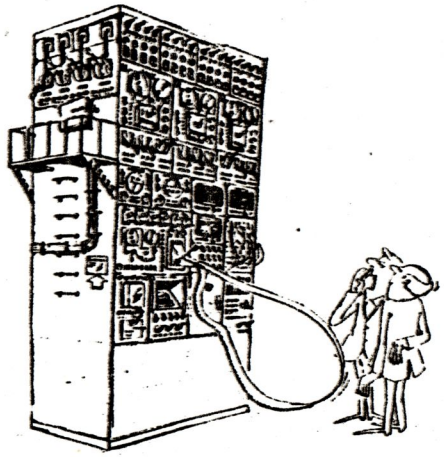
Im Tscrips 4.0 gibt es einige Freiräume, die für jeden interessant sind, der Tscrips erweitern will. Allerdings würde ich für ein ernsthaftes Arbeiten eine vollständige Neu-Assemblierung empfehlen. Aber für kleinere Erweiterungen genügen diese Bereiche, in die man zappen kann. Die Angaben erfolgen jeweils einschließlich des ersten und letzten Bytes, das verändert werden darf.

Bereich	dort stand vorher...
5848h-5869h	nur Quark
6188h-61A1h	ein Unterprogramm ohne erkennbaren Ansprung
6561h-6567h	Quark
6C17h-6C29h	ein Unterprogramm ohne Ansprung
6FB8h-6FC8h	Text: "TAB COMMAND MODE" (gibt es nicht mehr)
7AA4h-7C28h	Sicherungsroutine für Tscrips 4.0 (vorher 52F2h-5241h mit NOP's füllen; Achtung bei Tscrips 5.4: 7BAEh-7BF5h werden noch anderweitig benutzt, also nicht überschreiben!)
8DF1h-8E4Ah	Quark (nur bei Tscrips 4.0)

Außerdem sind meines Erachtens die Tape-I/O-Routinen überflüssig, weil niemand ohne Disk Tscrips benutzen kann. Anschließend also noch die Zaps, die die Tape-I/O abschaffen und einen Absturz durch versehentliche Eingabe der Kommandos "S,T", "L,T" oder "V" verhindern.

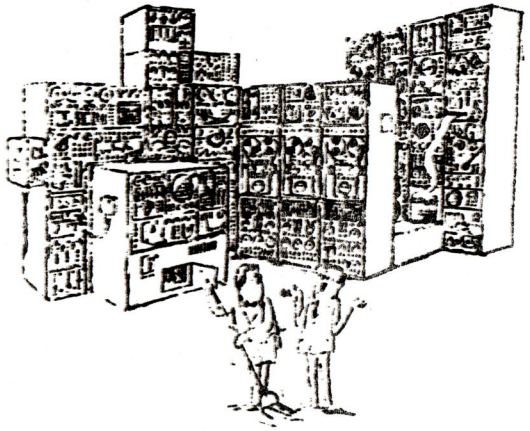
Gerald Schröder

```
00001 ; PATCH #1
00002 ; ABSCHAFFEN DER TAPE-I/O
00003 ; DADURCH FREIRAUM VON 626Ah BIS 6445h
00004 ; LETZTE ANDERUNG: 7.4.86 BY GS
00005 ;
00006          ORG     6238H
00007 TEIL1    LD      DE,6F8BH        ;CLEAR-ROUTINE
00008          LD      (7C29H),DE
00009          LD      D,H
00010          LD      E,L
00011 LOOP2    LD      A,(HL)
00012          OR      A               ;ENDE DES TEXTES? (00)
00013          RET     Z               ;JA => ZURÜCK
00014          CP      20H             ;<20H? (FUNKTIONSCODE)
00015          JR      NC,LOOP1        ;NEIN
00016          SET     07H,(HL)        ;JA => BIT 7 SETZEN
00017 LOOP1    INC     HL
00018          JR      LOOP2
00019 ;
00020 ;
00021 TEIL2    LD      DE,6F8BH        ;CLEAR-ROUTINE
00022          LD      (7C29H),DE
00023          XOR     A
00024          LD      DE,(7C43H)
00025          LD      (DE),A          ;TEXTENDE MARKIEREN
00026          LD      (7C2BH),DE      ;TEXTENDE SPEICHERN
00027          SET     01H,(IY+34H)
00028          CALL    65B0H
00029          LD      A,01H           ;"NO MORE ROOM"-ERROR
00030          JP      6F77H
00031 ;
```



»Um Himmels willen! Jetzt fährt er schon Selbstgespräche!«


```
00032 ;
00033 ;        ANDERUNGEN IM DISK-I/O-TEIL
00034 ;
00035          ORG     5DB4H
00036          CALL    TEIL1
00037 ;
00038          ORG     5DBDH
00039          JP      C,TEIL2
00040 ;
00041 ;
00042 ;        ANDERUNGEN IM I/O-AUFRUF
00043 ;
00044          ORG     66EEH
00045          JP      6F76H           ;BEI TAPE-I/O "INVALID COMMAND"
00046 ;
00047          ORG     66F9H
00048          JP      6F76H           ;          "
00049 ;
00050 ;
00051 ;        TAPE VERIFY (BREAK V) WEG
00052 ;
00053          ORG     6AF1H
00054          DB      00
00055 ;
00056          ORG     6AF5H
00057          DW      0
00058 ;
00059 ;
00060 ;
00061          END
```



»Aber Chef, Sie haben doch selbst gesagt, der Computer solle alle Mitarbeiter ihren Fähigkeiten entsprechend einsetzen!«

Textumleitung für Tscrips und andere

In der letzten Ausgabe hat Arnulf ein Programm vorgestellt, mit dem Sektordumps u.ä. in einen Tscrips-Text eingebaut werden können. Leider gab es dabei ein Problem: Tscrips muß schon im Speicher sein und laufen. Wenn ein anderes Anwenderprogramm Tscrips (zer)stört, funktioniert die Umleitung nicht.
Nun wollte ich beweisen, daß Arnulfs und Hartmuts Artikel auch einem Assembler-Neuling etwas bringen und habe mir deshalb meine eigenen Gedanken dazu gemacht. Und siehe da, auf einmal erinnerte ich mich an den von mir noch nie benutzten "ROUTE"-Befehl des Newdos. Da ich vermute, daß Ihr diesen Befehl auch selten benutzt, will ich ihn kurz erklären: ROUTE leitet Ein- oder Ausgaben um. Wenn man zum Beispiel keinen Drucker hat oder kein Papier verschwenden will, kann "ROUTE PR DO" eingegeben werden und alle Druckerausgaben, die auf den Druckertreiber zurückgreifen, werden auf den Bildschirm umgeleitet.
Dieses machte ich mir nun zunutze: mit "ROUTE PR MM=xxxxH" werden Druckerausgaben an eine Routine weitergereicht, die bei xxxxH steht. Ein Beispiel für eine solche Routine seht Ihr unten. Die Routine packt in diesem Fall alles, was eigentlich auf dem Drucker erscheinen würde, in den File "ROUTE/DVC" (u.a. LPRINT und LLIST von BASIC, JKL-Hardcopies, DIR P, H#:* bei Edtasm).
Diese Umleitung wird beendet, wenn Ihr ein 01 an den Drucker schickt (z.B. "LPRINT CHR$(1)"). Dann wird der File geschlossen und die Umleitung durch "ROUTE CLEAR" rückgängig gemacht.
Der entstandene File kann von anderen Programmen benutzt werden. Er läßt sich in dieser Form von Tscrips laden, womit wir wieder den Anschluß an Arnulfs Artikel gefunden hätten.
Ein Nachteil: Es müssen dann noch $ etc. in &24& etc. umgewandelt werden. Wem die Replace-Funktion dazu nicht ausreicht, der kann sich ja noch ein Konvertierungsprogramm schreiben.
Ein Vorteil: Das Programm arbeitet unabhängig von Tscrips. Also lassen sich auch Superzap-Dumps und Edtasm-Programme in einen Text einbauen, allerdings nicht direkt, sondern nur über den Ladebefehl "L.C filename" in Tscrips.
Das nachfolgende Listing wurde folgendermaßen in den Text eingefügt:

1. Listing mit ZEUS erstellen.
2. Abspeichern mit "S*ROUTE" (für Edtasm, weil ZEUS eine eigene Druckroutine besitzt und Übertragung nur durch JKL möglich ist).
3. ZEUS verlassen, "ROUTE/CMD" eingeben, "EDTASM/CMD" eingeben.
4. In Edtasm: "L D=ROUTE/SRC".
5. "H#:*" (Listing an Drucker).
6. Edtasm verlassen.
7. Im GDOS "DR #A" (gibt 01 an den Drucker aus).
8. Tscrips starten und BREAK "L ROUTE/DVC".
9. Text um das Listing herum konstruieren.

Das ganze sieht sehr umständlich aus, ist es aber nicht! Die Möglichkeiten sind fast unbegrenzt. Einige Grenzen: FE00 ff. darf nicht überschrieben werden; Treiberadresse muß unverändert benutzt werden; DIR 1 P ist nicht möglich, wenn dort "ROUTE/DVC" steht (warum? Null Ahnung!); File darf nicht zu lang für Tscrips sein.

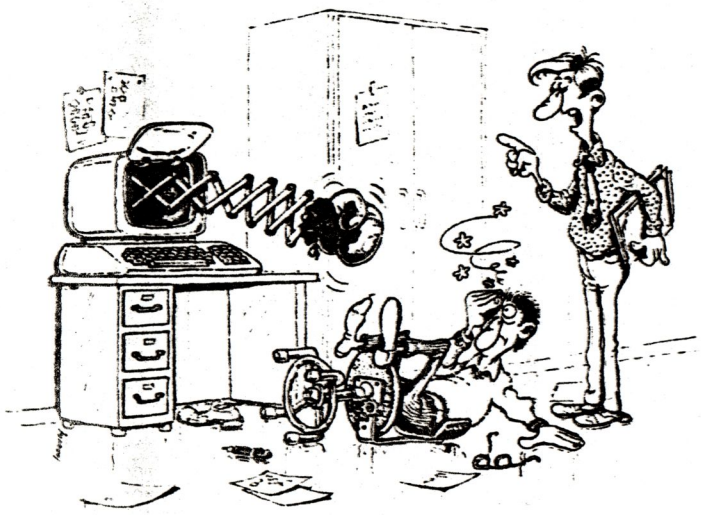
Anwendungen und Verbesserungen denkt Ihr Euch am besten selbst aus. Aber berichtet darüber im Clubinfo!

Gerald Schröder

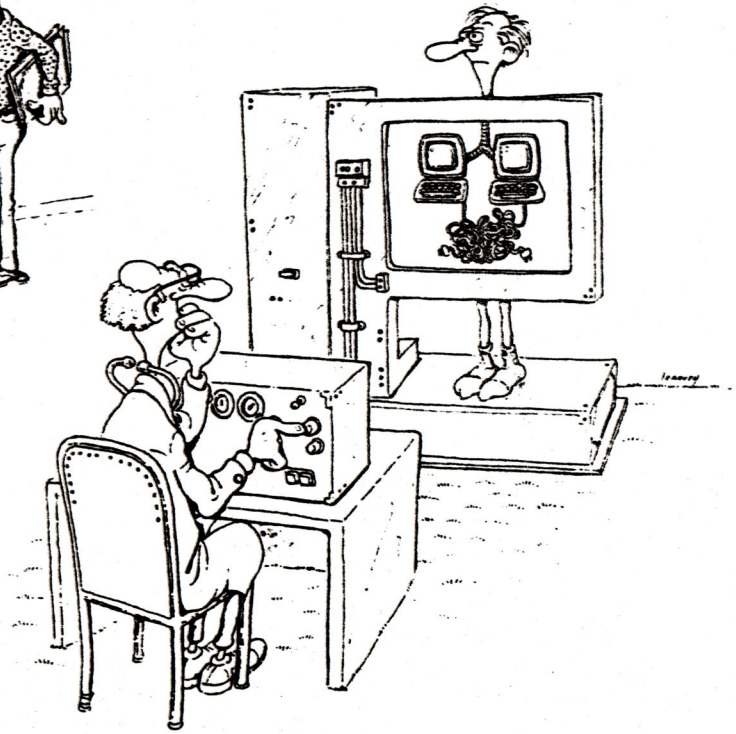
```
00001 ;       ROUTE von Gerald Schröder
00002 ;       8.6.86
00003 ;       LEITET DRUCKERAUSGABE IN DEN FILE "ROUTE/DVC" UM
00004 ;       FÜR SPÄTERE NUTZUNG DURCH TSCRIPS
00005 ;
00006 ;       KONSTANTEN
00007 ;
00008 WRIBYT  EQU     001BH           ;ein Byte ausgeben
00009 DOS     EQU     402DH           ;DOS-Abgang
00010 HIMEM   EQU     4049H           ;HIMEM-Zeiger
00011 DERROR  EQU     4409H           ;DOS-Fehlerausgabe
00012 DOSCAL  EQU     4419H           ;ruft DOS-Befehl auf
00013 EXFIL   EQU     441CH           ;Filennamen bearbeiten
00014 DINIT   EQU     4420H           ;File eröffnen
00015 CLOSE   EQU     4428H           ;File schließen
00016 ;
00017         ORG     0E000H
00018 INIT    LD      HL,START-1
00019         LD      (HIMEM),HL
00020         LD      HL,FILENAME     ;Filename-Zeiger
00021         LD      DE,FCB          ;File Control Block-Zeiger
00022         CALL    EXFIL           ;Filenamen übertragen
00023 ;
00024         LD      DE,FCB          ;Zeiger auf FCB
00025         LD      HL,BUFFER       ;Zeiger auf Sektor-Buffer (256 Bytes)
00026         LD      B,00H           ;RECORD-Länge (256)
00027         CALL    DINIT           ;File eröffnen
00028         JP      NZ,DERROR       ;Error ausgeben, wenn aufgetreten
00029 ;
00030         LD      HL,ROUTEON      ;Ausgabe-Umleitung
00031         CALL    DOSCAL
00032         JP      NZ,DERROR
00033         RET
00034 ;
00035         ORG     0FE00H
00036 START   DW      00,00,00,00,00,00       ;12 Byte für das DOS
00037         PUSH    AF
00038         PUSH    HL
00039         PUSH    DE
00040         PUSH    BC
00041 ;
00042         LD      A,C             ;auszugebendes Zeichen nach A
00043         LD      DE,FCB          ;FCB-Zeiger
00044         CP      01H             ;Ende der Ausgabe?
00045         JR      Z,ENDE          ;Ja
00046 ;
00047         CALL    WRIBYT          ;Zeichen ausgeben
00048         CALL    NZ,DERROR       ;falls Fehler aufgetreten
00049 ;
00050         POP     BC
00051         POP     DE
00052         POP     HL
00053         POP     AF
00054         RET
00055 ;
00056 ;
```

```
00057 ENDE    CALL    CLOSE           ;File schließen
00058         CALL    NZ,DERROR
00059 ;
00060         LD      HL,0FFFFH       ;HIMEM wieder zurück
00061         LD      (HIMEM),HL
00062 ;
00063         LD      HL,ROUTEOFF     ;Ausgabe-Umleitung zurück
00064         CALL    DOSCAL
00065         CALL    NZ,DERROR
00066         JP      DOS             ;Absprung ins DOS
00067 ;
00068 ;
00069 FILENAME      DM      'ROUTE/DVC',0DH
00070 ROUTEON       DM      '§ DR ST=FE00H',0DH
00071 ROUTEOFF      DM      '§ KEINE',0DH
00072 FCB           DS      32
00073 BUFFER        DS      256
00074 ;
00075         END     INIT
```

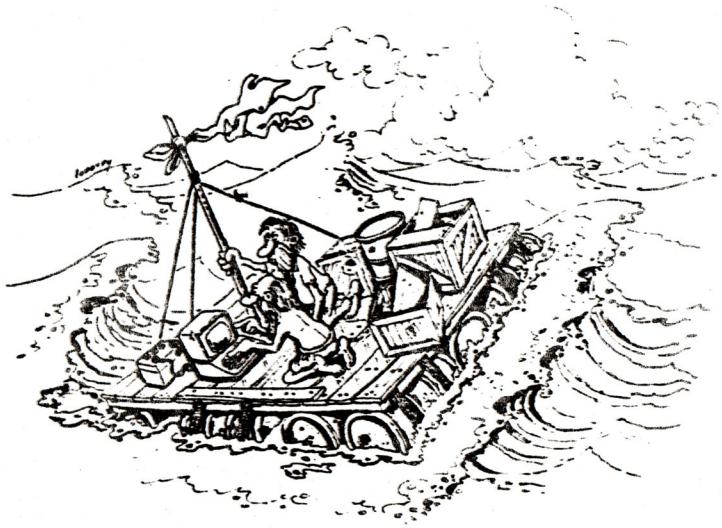
Anmerkung: Die Ausgabeumleitung gilt für GDOS. Bei Newdos heißen die Befehle:
ROUTEON: "ROUTE PR MM=FE00H"
ROUTEOFF: "ROUTE CLEAR"



Das versucht er mit jedem. Zeig ihm, wer der Boß ist!



Er sagt: „Verlange Barzahlung, der überlebt die Nacht nicht mehr."



Die Chance gerettet zu werden steht drei Milliarden zu eins.
Gut, daß wir den Computer haben, sonst hätten wir es nie erfahren!

Nachtrag zu "Sondertasten ..."

In meinem Beitrag "Sondertasten diverser Genies" frägte ich, ob jemand schon herausgefunden habe, wie die Doppelnull erzeugt wird. Wieso habe ich mir diese Frage nicht selber gestellt? Inzwischen habe ich es nämlich raus:

Auf der nächsten Seite ist oben ein Listing abgedruckt. Der obere Teil ist die Routine, die die Doppelnull auf den Bildschirm bringt. Da wird zunächst die Anzahl der zusätzlich zu printenden Zeichen, hier 1, nach 4019h geladen. Dort ist normalerweise für die F-Tasten die Länge ihrer Strings abgelegt. Die Adresse des einen zusätzlichen Zeichens, 3E54h, wird nach 401Ah übertragen. Dort ist sonst der Beginn eines F-Strings gespeichert. Bei 3E47h geht es dann weiter, wo alles angezeigt wird.

Jede dieser Informationen kann geändert werden: Die um 1 verminderte Anzahl der Zeichen darf beliebig sein. Sie wird entsprechend nach 3E4Fh gepatcht. Das erste anzuzeigende Zeichen in 3E54h darf auch beliebig lauten. Schließlich kann man in 3E56/57h die Adresse des zweiten und der folgenden Zeichen ebenfalls nach Gusto ändern. Auf diese Weise ist es möglich, die 00-Taste wie eine zusätzliche F-Taste zu gebrauchen. Ihr String kann beliebig lang sein (solange er kürzer als 256 Bytes ist). Sie ist auch abschaltbar, wenn in 3E54h z. B. eine logische 00 oder ein anderer unwirksamer Code steht. In diesem Falle kann die 00-Taste z. B. während einer INT-Bearbeitung als Steuertaste für jedwede Funktion dienen, ohne daß sie außerhalb der INT-Routine stören würde.

Das Erkennen und Anzeigen der PRINT-Taste ist im unteren Teil des Listings wiedergegeben. Wie man sie abschalten oder anders belegen kann, habe ich bereits im o. g. Artikel beschrieben. Zusätzlich hier ein Tip, wie ihr ebenfalls ein beliebig langer Code zugeordnet werden kann: Die Anzahl der Zeichen ist in 3E6En abgelegt. Sie darf geändert werden. Die Adresse des Strings auch, die in 3E70h steht. Man definiert einen eigenen String und patcht seine Adresse dorthin.

Die gelistete Routine steht "neben" dem Bildwiederholspeicher in der Nachbarbank. Den Änderungen muß deshalb ein entsprechendes Banking vorausgehen: Beim Genie III s muß beim Systembyte 1 (Port FAh) dazu das Bit 0 gesetzt werden. Hierzu genügt ein CALL 06A0h. Jetzt sind die Tastatur und der Bildschirm ausgeblendet, stattdessen ist das RAM zugänglich. Nach der Manipulation wird das Bit 0 wieder zurückgesetzt, wofür ein CALL 06ABh langt. Es empfiehlt sich, zuvor die Interrupts mit DI zu sperren, weil evtl. Tastatur- oder Bildschirmzugriffe (Uhr usw.) sonst das Nachbar-RAM adressieren.

In den beiden Sektorsums unter dem Listing ist alles unterstrichen, was hier zur Änderung vorgeschlagen wird. Das sind die Sektoren 04 und 05 von OVL4/SYS für das Genie III s. gepatchte G-DOS-Version. Bei den anderen Overlays für die anderen Genies wird man die entsprechenden Codes leicht finden können, weil in ihrer Umgebung so ziemlich dasselbe stehen dürfte.

Arnulf Scpp

```
3E4E  3E01      00224          LD      A,01h          ;1 zusätzliches Zeichen
3E50  321940    00225          LD      (fcount),A     ;als Zeichenzähler ablg.
3E53  3E30      00226          LD      A,'0'          ;zweite Null bei 00-Taste
3E54            00227 key00    EQU     $-1            ;zu verdoppelndes Zeichen
3E55  21543E    00228          LD      HL,key00       ;Adr. des Doppelzeichens
3E58  221A40    00229          LD      (faddr),HL     ;als Stringadresse ablg.
3E5B  18EA      00230          JR      m3e47          ;dort weiter
                00231
3E5d  FE3F      00232          CP      '?'            ;PRINT-Taste gedrückt?
3E6B  C0        00233          RET     NZ             ;falls nein
3E69  CD7E3D    00240          CALL    m3d7e          ;Cursor aus?
3E6C  C8        00241          RET     Z              ;falls ja
3E6D  3E06      00242          LD      A,06h          ;6 Zeichen von PRINT
3E6F  21753E    00243          LD      HL,kyPRINT     ;Belegg. der PRINT-Taste
3E72  C3763D    00244          JP      m3d76          ;dort weiter
3E75  50        00245 kyPRINT  DM      'PRINT '       ;Belegung PRINT-Taste

00000 Fehler
```

```
DRV 00  00DC 7E23 052B 0485 6F18 F732 1940 221A .<ß#.(..o..2.§".
1   10  40AF C94F 3A22 40B7 79C8 3A6E 43CB 4779 §..O:"§.y.:nC.Gy
1H  20  C9CD 99ED ED4B 4845 CD21 91C3 4A45 2185 ...=.KHE.!..JE!.
    30  3DCD 2101 5FAE 73A3 200C 7AC6 0857 2379 =.!._.s...z..W#y
DRS 40  C620 4F20 ECC9 E1C3 5E45 0000 0000 280E ..O.....^E....(.
864 50  2100 002B 22BC 3D7C B53E 00C2 8945 2100 !..+".=d.>...E!.
360H60  0022 9C3D CD55 4521 0000 22BC 3D32 4F45 ."..=.UE!..".=2OE
    70  AFC9 219F 377A 2334 3528 05ED A120 F77E ..!.7z#45(.....ß
    80  FE0B 2806 FE41 D8FE 5FD0 4FCD 003D E642 ..(..A.._.O..=.B
    90  79C8 E61F FE0B C001 4038 CD21 01E6 083E y........§8.!...>
    A0  0BC5 3E5B C9D6 3E38 79C8 3DC8 2118 0136 ..>A..88y.=.!..=
    B0  C9E5 CD24 3EE1 36CD C9FE 0730 1207 4FCD ...$>.6....0..O.
FRS C0  0102 3C4F 003D 0F79 3001 3C21 7B3E CD46 ..<O.=.y0.<!{>.F
4   D0  047A C9D6 79FE 88D8 E67F FE12 3007 C625 .z..y....  ..0..
4H  E0  CD4B 047A C920 0F3E 0132 1940 3E30 2154 .K.z...>.2.§>0!T
    F0  3E22 1A40 18EA D613 C921 883E CD33 3EFE >".§.....!.>.5>.
```

```
DRV 00  3FC0 CD7E 3DC8 3E06 2175 3EC3 763D 5052 ?..8=.>.!u>.v=PR
1   10  494E 5420 0000 181B 0A1A 081B 282B 0000 INT.........+*..
1H  20  0000 2020 203F AF32 2104 CD00 3DC3 0E04 ...=.?.2!...=...
    30  CDB5 0621 4140 0502 3E23 32BC 3E16 0A06 ...!A§..>#2.>...
DRS 40  037E D96F 2600 3E0A CDB4 4CD9 CD19 3FD9 .ß.o&.>...L...?.
865 50  7DD9 CD19 3F23 10E9 167A 3E2B 32BC 3E2E ß...?#...z>+2.>.
361H60  460D 20DB AF16 0ACD 193F CDBE 06DD E5DD F........?......
    70  212C 3F2E 440E 023E 2332 F53E 16CC 060D !.?.D..>#2.>....
    80  CD0A 3F97 5F87 9783 5FCD 0A3F 9377 2310 ..?._...._..?.w#.
    90  EF16 5C3E 2B32 F53E 2543 0D20 E1DD E1AF ..8>+2.>.C......
    A0  D35B C97A D35B 1610 9257 DD23 DB5A DDA6 .A.z.A...W.#.Z..
    B0  00C9 5F7A D35B 1610 9257 E6F0 FE60 7B20 ..z.A...W...`ß.
FRS C0  02C5 3FD3 0110 3C50 5AC9 0F0F 010F 030F .....<PZ.......
5   D0  030F 070F 070F 010F 0645 00F5 3A05 45EE .........E..:.E.
5H  E0  20D2 7304 F100 0001 0C5E 04ED 5B41 36F5 .Os......^..AA6.
    F0  19F1 C342 3401 2780 04E5 ED5B 4136 B7ED ...B4.'....AA6..
```

Unterschiede

Wer, wie z.B. ich, von einem Model 1 auf ein Model 3 (bzw. Model 4/4P im Model 3-Mode) umsteigt, wird mit dem Problem konfrontiert, daß manche Programme, die man auf dem Model 1 schätzen gelernt hat, nicht mehr laufen. Um diese Programme umzuschreiben, benötigt man, außer etwas Erfahrung im Umgang mit der Maschinensprache, eine Auflistung der Unterschiede zwischen den beiden Maschinen.

In einer kleinen Serie von Artikeln möchte ich diese Unterschiede aufzeigen. Den Anfang machen die ROM-Differenzen, danach (ev. noch in dieser Ausgabe) folgen die Tastaturabweichungen. Ein Beitrag über die NewDOS-Unterschiede wird im nächsten Info erscheinen. Weiterhin werde ich für verschiedene, von mir angepasste Programme, Patchs veröffentlichen. Hier also zunächst die ROM-Unterschiede!

Wenn man sich die beiden Röckrath'schen ROM-Listings genauer betrachtet, sind natürlich erhebliche, meist hardwarebedingte Unterschiede festzustellen. Betrachtet man dagegen nur die Adressenauflistung der Unterroutinen, so wird man feststellen, daß sich hier nur wenig geändert hat. Die ROM-Macher haben also, um die Model 1-Kompatibilität zu zu wahren, zwar den Programmcode geändert, aber wo möglich die Einsprung -Adressen und -Bedingungen beibehalten. Die untenstehende Liste führt alle Änderungen und Erweiterungen auf.

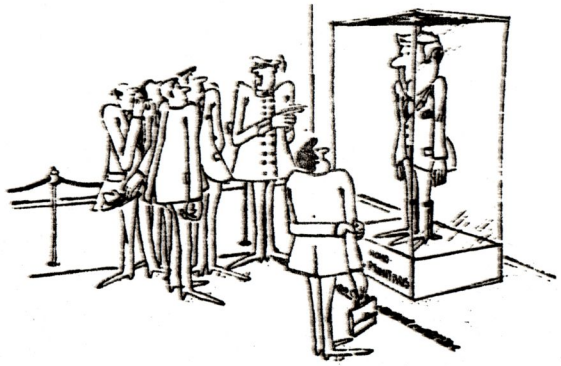
ADDR M. 1-ROM	M. 3-ROM	Bedeutung
0050 -	RSRCV	Holen eines Zeichens von der RS-232 (BREAK bewirkt Sprung über 4203 nach 1A19)
0055 -	RSTX	Ausgabe eines Zeichens an die RS-232 (BREAK bewirkt Sprung über 4203 nach 1A19)
005A -	RSINIT	Initialisierung der RS-232
0069 -	INITIO	Initialisiert alle I/O-DCB's einschl. I/O-Routes
006C -	ROUTE	Führt Routing der I/O-Treiber durch (gibt es nur in der amerikanischen Version!)
01D9 PULSE	*31A5*	Gibt Impuls auf Kassette aus
01D9 -	PRSCN	Druckt Bildschirm aus (Codes 80-FF als "."!)
01FE CASND	-	Decodiert Kassettenrecorder-Nr. und schaltet Kassettenrecorder ein
0214 -	CRPR	CR (Carrige Returne) auf Drucker ausgeben
0215 CASON	*31E8*	Schaltet Kassettenrecorder ein
021B -	TEXT	Text auf Video ausgeben
021E RESINP	-	Setzt den Eingabeport Bit 7 von Port FF zurück
0221 PORTCT	-	Steuert Port FF
022C BLINK	-	Schaltet Stern in Bildschirmecke ua
0241 READBI	*3220*	Liest Bit von Kassette
0287 WRITES	-	Schreibt Syncronbytes
028D -	BREAK	BREAK-Taste abfragen
0296 READS	-	Liest Syncronisation
029B -	CLKON	Uhrzeit jede Sekunde anzeigen
02A1 -	CLKOFF	Uhrzeit-Anzeige abschalten
03C2 CALL	*0674*	Ruft I/O-Routinen über DCB auf
044B *05D1*	PRRDY	Testet ob Drucker bereit
05D1 PRRDY	*044B*	Testet ob Drucker bereit
0674 *03C2*	CALL	Ruft I/O-Routinen über DCB auf

ADDR M. 1-ROM	M. 3-ROM	Bedeutung
3033 -	DATE	Datum ab (HL) abspeichern
3036 -	TIME	Zeit ab (HL) abspeichern
3042 -	SETCAS	Fragt den Benutzer nach der Kassetten-BAUD-Rate (gibt es nur in der amerik. Version!)
31A5 *01D9*	PULSE	Gibt Impuls auf Kassette aus (500 BAUD)
31D1 -	MOTON	DE und BC retten, Kassettenrec. einschalten
31E8 *0215*	CASON	Kassettenrecorder einschalten
3220 *0241*	READBI	Liest Bit von Kassette (500 BAUD, BREAK bewirkt einen Sprung über 4203 nach 1A19)
3335 -	WRITBI	Bit auf Kassette schreiben (1500 BAUD)
3350 -	WAITBI	Auf nächstes Bit von Kassette warten (1500 BAUD, BREAK bewirkt einen Sprung über 4203 nach 1A19)
337C -	AUSWBI	Wartezeit für gelesenes Bit von Kassette auswerten (1500 BAUD)
3505 -	PRNCND	Druckt Bildschirm aus (Codes 80 - 8F als "."). Gibt es nur in der deutschen Vers.!)

Erklärung zu Spalte 2 und 3:
"-" = Routine in der ROM-Version nicht vorhanden!
xxxx = Routine in der ROM-Version an anderer Stelle!

Wie schon erwähnt sind viele Probleme im Model 3 erheblich anders gelöst als im Model 1. Ein gravierendes Beispiel hierfür ist die Abfrage der Tastatur. Diese Lösungswege interessieren aber bei der Anpassung von Programmen nur selten und sollen deshalb hier auch weiter nicht erwähnt werden. Ich glaube, daß mit der obenstehenden Liste schon die meisten Anpassprobleme zu lösen sind. Sollte jemand weitere Unterschiede in Erfahrung bringen oder schon Progamme angepasst haben, möge er doch bitte sein Wissen im Info weitergeben! Viel Spaß beim Patchen euer

Reinhard Ohlemann



»Beachten Sie bitte die Schädelbildung aus der Zeit vor der Erfindung des Computers«

Knöpfe

Die Tastatur aller TRS80-Modelle und auch der Kompatiblen ist, wie schon mehrfach im Clubinfo festgestellt, nichts weiter als eine Matrix aus lauter Schließern. Leider sind die einzelnen Tasten bei den unterschiedlichen Geräten in der Matrix unterschiedlich angeordnet. So besitzt das M. 1 z.B. zwar zwei SHIFT-Tasten, diese sind jedoch elektrisch parallelgeschaltet. Beim Model 3/4/4P hingegen wird zwischen zwei SHIFT-Tasten unterschieden. Und sogar innerhalb eines Models gibt es erhebliche Unterschiede, wenn man sich eine "amerikanische" und eine "deutsche" Tastatur ansieht! Diese unterschiedliche Belegung der Matrix kann bei bestimmten Programmen (SCRIPSIT, FED usw.) zu Problemen führen. Aus diesem Grund sind hier einmal die Belegungslisten aller TRS-80- und VideoGenie-Geräte veröffentlicht.

Tastaturmatrix TRS 80 Model 1, Genie I und II:

Adresse:	I	Datenleitung Bit 0	1	2	3	4	5	6	7
3801h	I	ALPHA	A	B	C	D	E	F	G
3802h	I	H	I	J	K	L	M	N	O
3804h	I	P	Q	R	S	T	U	V	W
3808h	I	X	Y	Z					
3810h	I	0	1	2	3	4	5	6	7
3820h	I	8	9	:	;	,	-	*	/
3840h	I	ENTER	CLEAR	BREAK	HOCH	RUNTER	LINKS	RECHTS	SPACE
3880h	I	SHIFT							

Tastaturmatrix Genie III:

Adresse:	I	Datenleitung Bit 0	1	2	3	4	5	6	7
3801h	I		A	B	C	D	E	F	G
3802h	I	H	I	J	K	L	M	N	O
3804h	I	P	Q	R	S	T	U	V	W
3808h	I	X	Y	Z	;	Ä	:	ALPHA	
3810h	I	0	1	2	3	4	5	6	7
3820h	I	8	9	ß	ö	,	-	.	/
3840h	I	ENTER	CLEAR	BREAK	HOCH	RUNTER	LINKS	RECHTS	SPACE
3880h	I	SHIFT	CTRL						LP
38A0h	I	F1	F2	F3	F4	F5	F6	F7	F8
38C0h	I	0	1	2	3	4	5	6	7
38E0h	I	8	9	00	LOCK	,	-	.	

Tastaturmatrix Genie IIIs:

Adresse:	I	Datenleitung Bit 0	1	2	3	4	5	6	7
3801h	I	ALPHA	A	B	C	D	E	F	G
3802h	I	H	I	J	K	L	M	N	O
3804h	I	P	Q	R	S	T	U	V	W
3808h	I	X	Y	Z	Ä	ö	ü	ß	_
3810h	I	0	1	2	3	4	5	6	7
3820h	I	8	9	:	;	,	-	.	/
3840h	I	ENTER	CLEAR	BREAK	HOCH	RUNTER	LINKS	RECHTS	SPACE
3880h	I	SHIFT	CTRL	ESC	P5	P4	P3	P2	P1
38A0h	I	F1	F2	F3	F4	F5	F6	F7	F8
38C0h	I	0	1	2	3	4	5	6	7
38E0h	I	8	9	00	LOCK	,	-	.	PRINT

Tastaturmatrix TRS 80 Model 3 (amerik. Version)

Adresse:	I	Datenleitung Bit 0	1	2	3	4	5	6	7
3801h	I	ALPHA	A	B	C	D	E	F	G
3802h	I	H	I	J	K	L	M	N	O
3804h	I	P	Q	R	S	T	U	V	W
3808h	I	X	Y	Z					
3810h	I	0	1	2	3	4	5	6	7
3820h	I	8	9	:	;	,	-	*	/
3840h	I	ENTER	CLEAR	BREAK	HOCH	RUNTER	LINKS	RECHTS	SPACE
3880h	I	LSHFT	RSHFT						

Tastaturmatrix TRS 80 Model 3 (deutsche Version)

Adresse:	I	Datenleitung Bit 0	1	2	3	4	5	6	7
3801h	I	>	A	B	C	D	E	F	G
3802h	I	H	I	J	K	L	M	N	O
3804h	I	P	Q	R	S	T	U	V	W
3808h	I	X	Z	Y	Ü	+	Ö	Ä	#
3810h	I	0	1	2	3	4	5	6	7
3820h	I	8	9	ß	´	ALPHA	,	.	-
3840h	I	ENTER	CLEAR	BREAK	HOCH	RUNTER	LINKS	RECHTS	SPACE
3880h	I	LSHFT	RSHFT						

Tastaturmatrix TRS 80 Model 4/4P

Adresse:	I	Datenleitung Bit 0	1	2	3	4	5	6	7
3880h	I	LSHFT	RSHFT	CTRL	CAPS	F1	F2	F3	

Hier wird nur die letzte Adressenreihe angegeben, da sich ansonsten gegenüber der Model 3 - Tastaturmatrix (amerik. und deutsch) keine Änderungen ergeben haben!

Hier noch ein kleines Beispiel:
Beim DEBUG des NewDOS läßt sich durch Eingabe von <+> und <ENTER> eine Seite vor-, und durch Eingabe von <-> und <ENTER> eine Seite zurückblättern. Dies funktioniert beim Model 3 nicht bzw. nicht richtig! Beim Model 1 liegt das <+> über dem <;>, während es beim Model 3/4 zusammen mit dem <*> auf einer Taste liegt und ohne Shift erreichbar ist. DEBUG fragt aber in beiden Versionen (Mod.1 und 3 NewDOS) ab, ob das <;> gedrückt wurde. Da beim Model 3/4 das <;> über dem <,> liegt, muß man <SHIFT><,> und <ENTER> drücken, um den gewünschten Erfolg zu haben! Damit man auch beim M.3/4 mit <+> arbeiten kann, ist folgendes Mini-Patch nötig:
SYS5/SYS Rel.Sector 1 Rel. Byte FD
von FE 3B in FE 2B ändern

So einfach ist manchmal das Ausmerzen unschöner Fehler und Schwächen, probierts doch mal! Viel Glück euer

Hartmut Obermann

m4/4p ecke

Aufgestiegen?

(Günther Wagner)

Auch ich gehöre jetzt seit 2 Wochen zum erlauchtem Kreis der Model 4 - Besitzer. Ich selbst habe einen 4P wobei das P nicht für Profi sondern für transportabel steht. Lange Zeit habe ich mir auch einen IBM-Kompatiblen überlegt - vor allem die vorhandene und dann nicht mehr lauffähige Software hat mich davon abgehalten.

Mein altes Model III konnte ich (mit 2 Laufwerken 40/ss/dd) für 1300 DM recht gut verkaufen.
Das neue 4P bekam ich sehr günstig und ich habe bei dieser Gelegenheit auch eine neue interessante Adresse aufgetan.

Ein Spezialist für TRS-80 ist :

Anthony Barber
Computer Stube
Fuchskamp 4
3032 Fallingsbostel
Tel. 05162 - 3818

Warum ich für diese Adresse Reklame mache ? Ganz einfach - alle Mitglieder des CLUB 80 erhalten 3 % Skonto. Ein Beispiel:

Das Model 4P kostet bei A. Barber mit 128 K RAM inklusive aller Nebenkosten (wie Versand) 1798 DM (schon sehr günstig) - nach Abzug der 3 % Skonto bezahlt man dann 1744 DM (Preise vom Mai).

Man bekommt bei A. Barber außer Hardware auch die nötige Software für TRS-DOS 1.3, 6.2 und für CP/M. Wer was braucht, kann sich ja mal das Angebot zusenden lassen.

Nun aber zurück zum eigentlichen Artikel:

Bereits beim Studium des neuen Handbuches (sehr umfangreich, meist sehr gut gemacht, manchmal etwas merkwürdig ins Deutsche übersetzt) konnte ich mit großer Freude feststellen, daß das neue TRS-DOS 6.2 ein hervorragendes Betriebssystem ist, welches speziell auf die Möglichkeiten des 4P zugeschnitten ist.

Man kann es auch so ausdrücken: Vergleicht man das TRS-DOS 1.3 mit einem VW-Käfer so hat man mit NEWDOS-80 einen VW-Golf und mit TRS-DOS 6.2 einen BMW oder Mercedes.

Mit sehr einfachen Befehlen kann man eine RAM-Disk initialisieren. Ebenso einfach ist das Einrichten dieser RAM-Disk als System-Laufwerk. Das Arbeiten mit der RAM-Disk ist natürlich viel schneller als mit normaler Diskette.

Ebenso einfach ist das Einrichten des Spoolers. Einfachste Befehle erlauben das Spoolen in den Speicher und/oder in einen File. Nur das Einschalten des Spoolers muß man besorgen - um alles andere kümmert sich das DOS.
Das DOS unterstützt auch sehr gut (wenn auch etwas umständlich) die Kommunikation über die RS232.
Nicht vergessen möchte ich die 'Job Control Language' (JCL), eine eigene Programmiersprache im DOS mit sehr guten Möglichkeiten.
Sehr gut gefällt mir die Verwendung von Drucker, Tastatur, Bildschirm etc. als logische (und gleichwertige) Geräte, die ohne weiteres miteinander verbunden werden können. Dabei können hier Filterprogramme zwischengeschaltet werden, die es z.B. ermöglichen, bestimmte Zeichen zu unterdrücken, abzuändern oder zusätzliche Zeichen auszugeben.
Das weitere Aufzählen wirklich leistungsfähiger Merkmale würde hier zu weit führen - erlaubt sei mir noch ein Hinweis auf das vorzügliche BASIC, welches mehrere neue leistungsfähige Befehle aufweist. Es handelt sich hierbei um Microsoft-Basic Version 5.

Will man Model III - Software laufen lassen, so muß man sich zunächst ein Model III - ROM-Image von Diskette einladen. Anschließend kann man ganz normal die Model III - System-Diskette einlegen. Bisher lief bei meinem Model 4 im Model III - Modus sämtliche Software, welche auch beim Model III lief. Und diese Software läuft sogar doppelt so schnell. Man muß einfach über die Tastenfolge 1-2-3 mit M 4210 auf die betreffende Adresse gehen und dort 20 in 60 abändern und siehe da - die Model III - Software läuft mit 4 MHz.

Auch CP/M läuft auf dieser wirklich feinen Maschine. Hierzu kann ich mich allerdings noch nicht weiter auslassen, da ich mit CP/M noch nicht viel gemacht habe.

Fazit : Ich betrachte mich als echten Aufsteiger - ich habe jetzt einen transportablen, formschönen Computer mit 128 K in dem eigentlich 3 Computer stecken:

- ein Model III
- ein Model 4P mit TRS-DOS 6.2
- ein CP/M - Computer

An dieser Stelle darf ich noch über bereits vorgenommene Veränderungen am Model 4P berichten. Ich darf dabei nochmals dem Hartmut danken, der diese für mich vorgenommen hat.

- zunächst kamen die internen 40 Spur-Laufwerke ins externe Gehäuse und die externen 80 Spur-Laufwerke in den Computer rein
- damit man die externen Laufwerke auch am 4P anschließen kann, baute mir der Hartmut eine entsprechende Buchse in den 4P ein
- die in der 13. Ausgabe auf Seite 40-41 beschriebenen Floppy-Tricks wurden ebenfalls mit Erfolg ausgeführt. Ich habe jetzt die Möglichkeit, mein Laufwerk 0 mit 40 Spuren oder 80 Spuren laufen zu lassen. Ferner habe ich für jedes Laufwerk einen Schreibschutz-Schalter.

m4/4p ecke

Tips für Model 4/4P

Aufruf des RAM-Tests:

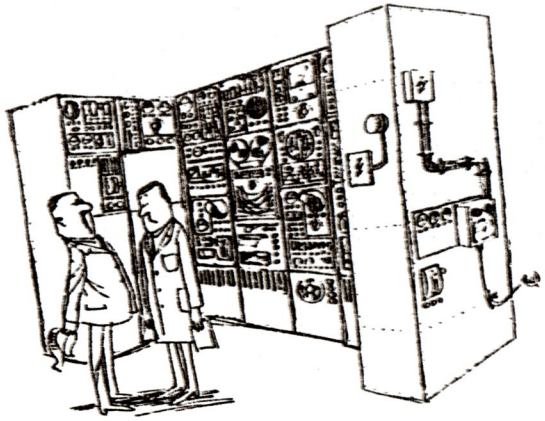
folgende Tasten zusammen drücken < 6 8 0 >
oder < . >
und anschließend den RESET-Knopf drücken.

Patch für den KILL-Befehl:

Im TRSDOS wurde der KILL-Befehl entfernt und statt dessen ein REMOVE installiert. Um den KILL-Befehl wieder einzusetzen, kann man folgenden Patch verwenden:

```
PATCH SYS1/SYS.LSIDOS (D02,81=4B:F02,81=00)
```

Der KILL-Befehl hat dieselbe Syntax wie der REMOVE-Befehl.



„Müller, der Computer hat sich über Sie beschwert"

Aufruf der SuperVisor Calls im Basic

(aus 80Micro, April 1986, Seite 102ff, Hardin Brothers)

TRSDOS 6.xx benutzt für seine Ein- und Ausgabeoperationen die sog. SuperVisor Calls (SVC). Damit lassen sich diese Operationen ziemlich einfach durchführen, wie z.B. Directory lesen, Bildschirminhalt abspeichern etc. Dies ist jedoch nur in Assembler möglich. Am nächsten kommt diesem der SYSTEM-Befehl in Basic. Damit lassen sich jedoch längst nicht alle gewünschten Systeminformationen in ein Programm (-variable) einlesen. Es ist zwar möglich z.B. aus dem Directory Daten über einen File einzulesen und dies auch in ein Programm einzubinden. Will man aber mehrere versch. Systeminfos, so wäre für jede eine spezielle Routine erforderlich. Das in dem o.g. Artikel vorgestellte Programm läßt sich dagegen als GOSUB-Routine in jedes Basicprogramm einbinden. (das Programm befindet sich demnächst auch in der Clubbibliothek - "SVC/BAS" u. "DEMO/SVC")

Nun zur Funktionsweise:
Um von einem Assemblerprogramm einen SVC aufzurufen, wird die SVC-Nummer in das A-Register geladen, weitere erforderl. Register werden ggf. gesetzt. Danach wird ein RST 28H-Befehl ausgeführt. TRSDOS sucht sich nun die aktuelle Adresse des SVC in einer Tabelle und führt anschließend den SVC aus.
Im Prinzip funktioniert so auch das SVC-Programm. Es definiert für jedes Z80 Register eine Variable. über eine Funktion (Zeile 50108) werden zwei dieser 8Bit Register zu einem Registerpaar kombiniert. Um nun einen SVC aufzurufen, müssen nur die richtigen Werte in die Registervariablen geladen werden. Danach wird mit GOSUB 50000 zum SVC-Unterprogramm gesprungen. Die Registerübergabe und der Aufruf des SVC werden durch ein kleines Assemblerprogramm (Z 50100-50138) erledigt. Nach Ausführung des SVC werden die Ergebniswerte wieder in die Registervariablen zurückgeladen, sodaß man darauf vom Basicprogramm aus zugreifen kann. über die Z.FLAG-Variable kann man kontrollieren, ob der SVC-Aufruf erfolgreich war. Sollte für entsprechende Systeminformationen ein Datenfeld notwendig sein, kann man dieses zunächst mit dem DIM-Befehl erstellen und anschließend mit dem ERASE Befehl wieder löschen.

Die einzelnen SVC's und ihre Registerwerte sind im techn. Handbuch für Model 4/4P beschrieben.

Einige der Möglichkeiten, einen SVC von einem Basicprogramm aus aufzurufen, werden in dem DEMO-Programm dargestellt (Scrollschutz für 1-7 Zeilen, Umschaltung Groß-/Kleinschreibung, Abspeichern eines Teils des Bildschirms, womit sich eine Fenstertechnik realisieren läßt, und Einlesen von Teilinformationen aus dem Directory).

Bei der Anwendung sollte man jedoch aufpassen, damit keine falschen Werte in die Registervariablen gelangen. Ansonsten dürfte der RESET-Knopf wohl die letzte Rettung sein.

Klaus Hermann

m4/4p ecke

Das Boot-ROM des 4P

Wie allgemein (?) bekannt, besitzt die tragbare Version des Model 4 kein BASIC- sondern nur ein sog. Boot-ROM. Dieses 4kByte große ROM wird "nur" dazu benutzt, ein Disk Operating System (TRSDOS 6.x, DOSPlus, MultiDOS, CP/M usw.) ins RAM zu laden und zu starten. Will man im Model 3 - Mode arbeiten, muß man ein BASIC-ROM ins RAM laden; auch das erledigt das Boot-ROM!

Diese Operationen sind den meisten Usern bekannt. Weit weniger bekannt sind die weiteren Möglichkeiten, die in diesem 4k-EPROM versteckt sind!

RAM - Test
Hält man beim Einschalten des 4P oder beim RESET drücken die Taste <.> (Dezimalpunkt), startet man ein ins Boot-ROM eingebautes Diagnoseprogramm. Leider sind im "Technical Reference Manual" (aus diesem stammen alle hier abgedruckten Informationen) keine weitere Erläuterung zu diesem Programm zu finden. Da (hoffentlich bleibt das noch lange so) meine 128k-RAM in Ordnung sind, weiß ich nicht, wie eine ev. Fehlermeldung aussehen würde. Sollte jemand mehr zu diesem Teil des Boot-ROM herausbekommen, wäre ich für jede Information sehr dankbar!

Boot - Funktionen
Nachdem der RESET-Schalter betätigt wurde, führt das Boot-ROM einige wichtige Funktionen durch (Bildschirmspeicher löschen, CRT- und Floppy-Controler initialisieren usw.), die uns hier nicht weiter interessieren sollen. Nachdem diese Operationen beendet sind (dauert nur Millisekunden) wird ein sog. Keyboard-Scanner angesprungen. Dieses Programm ermittelt, ob und welche Tasten auf der Tastatur gedrückt wurden. Um dem Benutzer Zeit zur Eingabe seiner Wünsche zu lassen, wird die Durchsicht in einer Schleife für ca. 2-3 Sekunden ständig wiederholt. Erst danach wird das eigentliche Boot-Programm aufgerufen.

Je nachdem, welche Tasten oder Tastenkombinationen während des Keyboard-Scannings gedrückt wurden, werden verschiedene Funktionen ausgeführt. Hier eine ausführliche Zusammenstellung der möglichen Optionen:

Taste(n)	Funktion
F1 oder 1	Hard-Disk - Boot
F2 oder 2	Floppy-Disk - Boot (norm. Funktion)
F3 oder 3	Model 3 - Mode
rechte Shift	RS 232 - Boot

Wie man sieht, kann man dem 4P sogar dann Leben einhauchen, wenn die Drives mal streiken und eine Harddisk nicht vorhanden ist. Man genötigt allerdings ein zweites Model 3/4/4P, von dem man die benötigte Software über die RS 232-Schnitt- stelle bereitgestellt bekommt.

Wird während der Abfrage des Keyboards keine Taste gedrückt, wird zunächst versucht, ein DOS von der Diskette einzulesen. Gelingt dies nicht, wird nach einem Model 3 ROM-Image-File gesucht und dieses (falls vorhanden) geladen. Das Programm schaltet nach einer nicht erfolgreichen DOS-Suche automatisch auf Model 3 - Mode. In diesen Mode stehen dem Benutzer noch drei weitere Optionen offen.

Taste	Funktion
P	wenn das ROM-Image geladen wurde, wird es normalerweise sofort gestartet. Wurde während der Tastaturabfrage <P> eingegeben, wird nach Beendigung des Ladevorgangs eine Meldung angezeigt und auf die Eingabe von ENTER gewartet, bevor es weitergeht!
N	Unterdrückt das Laden des ROM-Image, es sei denn, es ist Bestandteil des DOS (NewDOS für Model 4-Mode) und wird automatisch mitgeladen!
L	Erzwingt das Laden des ROM-Images, auch wenn sich dieses (von einem früheren Bootvorgang) noch im RAM befindet! Bemerkung: Das Boot-ROM erkennt, ob das ROM-Image schon geladen wurde und unterläßt in diesem Fall ein nochmaliges Laden. Die beiden letztgenannten Optionen (N und L) sind vor allem dann brauchbar und notwendig, wenn man am "ROM" (im RAM) Änderungen vorgenommen hat und entweder ein Überschreiben verhindern oder erzwingen will!

Sondertasten
Wie schon erwähnt, tritt durch die Tastaturabfrage eine Wartezeit von ca. 2-3 Sekunden auf. Ist einem diese Zeitspanne zu lange, kann man sie durch Drücken von ENTER abbrechen. Will man z.B. TRSDOS 2.3 laden und drückt direkt nach RESET die Taste ENTER, wird das Booten um ca. 2-3 Sekunden beschläunigt!

Eine weitere Sonderstellung nimmt die BREAK-Taste ein. Mit ihr kann man dem DOS (nur im Model 4 - Mode) einen "Wink" geben. Wird während der Keyboadabfrage BREAK gedrückt, wird die RAM-Speicherstelle 405Bh (16475d) auf ungleich 00 gesetzt. Wurde BREAK nicht gedrückt, findet das DOS an dieser Stelle später den Wert 00 wieder! Damit kann man also schon während des Bootvorganges dem DOS eine Mitteilung zukommen lassen und ev. eine besondere Aktion starten.

Beispiel
Nehmen wir an, man hat im Model 3 - Mode ein bißchen am "ROM" herumgezapt und will nun neu booten. Dazu wären folgende Tasteingaben nötig:

F3 oder 3	(muß nicht unbedingt sein, da, wenn kein DOS auf der Disk gefunden wird, sowieso nach dem ROM-Image gesucht wird!)
L	(das ROM-Image wird "zwangsweise" geladen!)
P	(es soll die Meldung ausgegeben und auf ENTER gewartet werden!)
ENTER	(muß nicht sein, kürzt aber die Wartezeit ab!)

Übrigens!
Wenn man Tastenkombinationen wählt, die sich widersprechen (z.B. N und L), dann wird jeweils die zuletzt gegebene Instruktion (hier z.B. L) ausgeführt!

Sonst noch was?
Wenn man sich das Boot-ROM einmal ansieht (siehe Listing), wird man feststellen, daß über 50% des EPROM mit Texten (sprich Fehlermeldungen) in drei Sprachen (engl., deutsch und franz.) gefüllt ist. Die eigentliche Programm-Area kommt einem da ziemlich kurz vor und bringt einen (zumindest mich) auf dumme Gedanken. Diese bestehen darin, die "überflüssigen" Texte (Fehlermeldungen in einer Sprache sind meiner Meinung nach genug) aus dem Boot-ROM zu entfernen und den entstehenden Platz mit nützlichem zu füllen.

Denkbar wären z.B.: ein Monitor-Programm, eine Bildschirm-Hardcopy-Routine (für Programme, die auf JKL wegen abgeschalteter Interrupts nicht reagieren), eine Password-Abfrage nach der Inbetriebnahme usw. usw.. Es sind sicher noch viele Anwendungen und Spielereien, die sich hier unterbringen ließen. Für Vorschläge bin ich immer offen und dankbar!

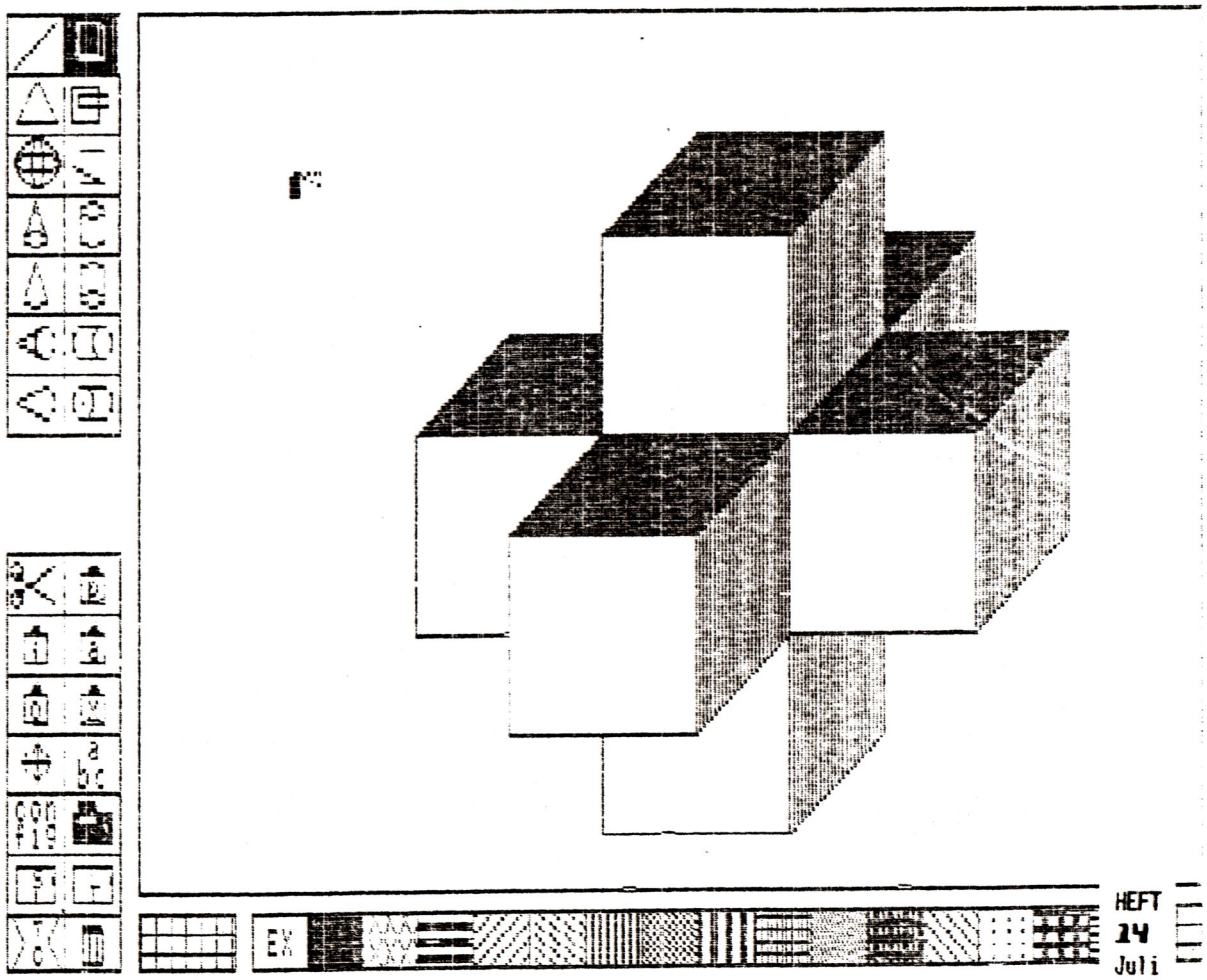
Read Boot-ROM
Das folgende kleine Assemblerprogramm ermöglicht es, das Boot-ROM, welches normalerweise in einer nicht ohne weiteres erreichbaren Speicherbank liegt, ins RAM zu verschieben (Topmem). Von dort kann man es mit dem DOS-Befehl DUMP BOOTROM/CIM F000h FFFFh 0000h 0000h auf Diskette abspeichern.

```
        ORG       5200H       'los gehts!
START   DI                    'keine Unterbrechungen bitte
        LD        A,01h       '01 in den Accu
        OUT       (9CH),A     'Boot-ROM einschalten
        LD        HL,0000h    'Quelladresse in HL
        LD        DE,0F000h   'Zeiladresse in DE
        LD        BC,0FFFh    'Anzahl der zu
                              verschiebenden Bytes
        LDIR                  'Verschieben
        XOR       A           '00 in den Accu
        OUT       (9CH),A     'Boot-ROM ausblenden
        EI                    'jetzt darf wieder
                              interruptet werden!
        JP        402Dh       'zurück zu DOS Ready
        END       START       'das bittere Ende!!!
```

Das wäre erstmal alles zum Boot-ROM. Sollte jemand weitere Informationen haben, möge er sie bitte im Interesse aller Clubmitglieder weitergeben!
Wer sich für das von Tandy vertriebene "Technical Reference Manual" für das Model 4/4P interessiert, kann es sich bei mir (kurzzeitig) ausleihen. Das Buch enthält viele interessante Informationen und Hinweise, ist jedoch für den "Normalanwender" zu umfangreich (nur absolute Freaks werden sich wohl für das Datenblatt des Bildschirmcontrolers interessieren) und zu teuer (ca. 70,- Deutsche Mark!).

Damit erstmal genug für heute. Happy computing wünscht

Hartmut Obermann



M4 Grafik

HRG für Model 3/4/4P!

Seit kurzem bin ich stolzer Besitzer einer Grafikkarte (µ-Labs Graphyx Solution) mit einer Auflösung von 640 * 240 Punkten (512 * 192 im Model 3-Modus). Hier ein kurzer Erfahrungsbericht über den Einbau und die ersten Programmierversuche.

Die µ-Labs-Karte ist sehr sauber aufgebaut (sauberes Layout) und macht einen stabilen Eindruck. Letzteres ist besonders wichtig, da die Karte bei mir ja in ein tragbares Gerät (4P) eingebaut wird! Grafyx Solution gibt es übrigens auch für's Model 3 und 4(Tischmodel). Der Einbau der Schaltung ist genauestens beschrieben, sodaß auch ein Hardwarelaie keinerlei Probleme haben dürfte. Lötungen sind keine vorzunehmen; die Karte wird auf einen Pfostenfeldstecker aufgesetzt und eine Leitung per Klemme mit einem IC-Beinchen verbunden. Weiterhin ist noch ein Jumper umzustecken, um dem Gerät das Vorhandensein der Karte zu signalisieren, dann kann der Grafikspaß auch schon beginnen!

Zusätzlich zur Karte bekommt man ein relativ dünnes Handbuch und eine Diskette mit Software für Model 3 und 4. Trotz des geringen Umfanges des Handbuches kommt man eigentlich von Anfang an sehr gut mit der HRG zurecht. Dazu tragen vor allem die über 20 mitgelieferten, teils sehr kurzen Beispielprogramme und Utilitys bei.

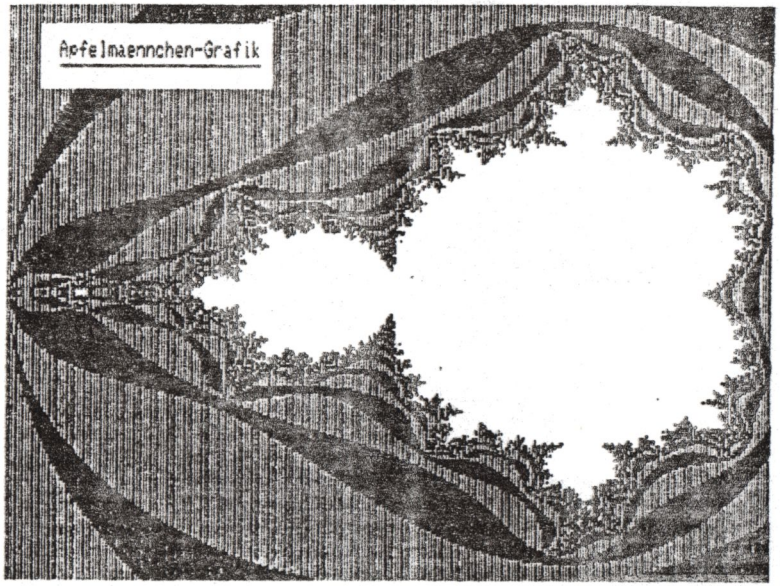
Die mitgelieferte BASIC-Erweiterung ermöglicht eine sehr komfortable Programmierung von Grafik. Ihr Befehlsumfang ist reichlich und umfaßt neben "normalen" Grafikbefehlen wie CLS, PLOT, POINT usw. auch Erweiterungen zum schnellen Zeichnen von Kreisen (CIRCLE) und Rechtecken (BOX), eine Füllroutine mit 256 möglichen Mustern und eine integrierte LOAD/SAVE-Möglichkeit (ca. 3-5 Sekunden Ladezeit) für die Bilder.

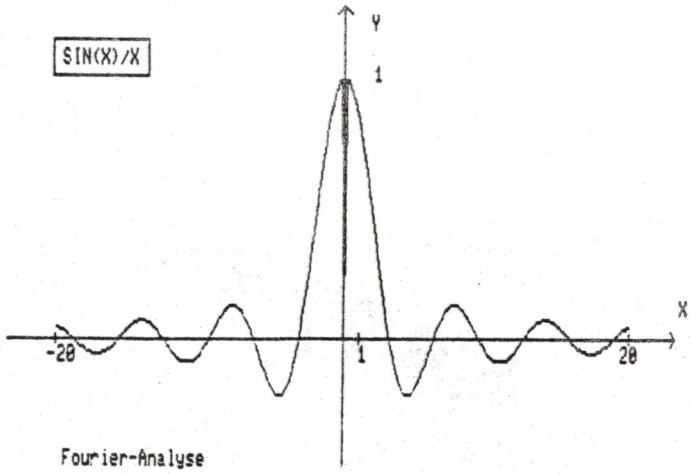
Natürlich will man die Grafikkarte nicht nur unter BASIC sondern auch unter Assembler ansprechen. Dazu liefert µ-Labs neben der genauen Beschreibung der Ansteuerung der Karte über die Ports 128 - 131 eine CLS- und eine POINT-Routine, sowie ein Beispielprogramm in dem diese Routinen benutzt werden. Damit gibt man dem Benutzer weiterhin die Möglichkeit, nach einigen kleinen Änderungen in der CLS/POINT-Routine, die HRG auch von Microsoft-Sprachen wie Fortran 80, Cobol 80 und BASCOM aus zu gebrauchen. Dadurch wird die Grafik auch in CP/M nutzbar.

Zusammen mit einem Bekannten der normalerweise auf einem MS-DOS-Rechner arbeitet, habe ich die Grafik auch für Turbo-PASCAL angepaßt. Dabei wurden wieder die CLS/POINT-Routinen verwendet (als Inline-Code direkt in den Turbo-Sourcecode eingebaut) und weitere Proceduren dazugeschrieben (PLOT). Wer die angepassten Routinen für BASCOM und Turbo-PASCAL haben möchte, kann sich direkt an mich wenden. Sicherlich könnte man auch die Model 1 HRG 1B für Hochsprachen in CP/M nutzbar machen!

Die beiden nebenstehenden Grafiken sind übrigens mit Turbo-Pascal-Programmen entstanden, die von einem IBM-Kompatiblen auf Montezuma CP/M transferiert und an die hochauflösende Grafik angepaßt wurden!

Hartmut Ebermann





Fourier-Analyse

DOS — (fast) ohne Floppy

128 kB oder mehr sind für die meisten heute verkauften Computer Standard geworden. Helmuts Banker für 256 kB bis 1 MB RAM macht aus den in unserem Club vertretenen Computern Maschinen, die sich in mancher Hinsicht ohne weiteres mit sehr viel teureren Rechnern messen können. Die Datenmengen, die wir zu verarbeiten haben, passen zwar normalerweise in die serienmäßigen 48 kB. Aber unser vergleichsweise leistungsfähiges DOS ist dafür bereits zu umfangreich. Deshalb wird es je nach Bedarf in kleinen Portionen in den Speicher geladen. Die Folge: Die Floppy steht kaum still und wird entsprechend belastet.

Hier soll das Programm RAMSYS/CMD vorgestellt werden, das zunächst alle Systemdateien des NEWDOS-80/G-DOS/H-DOS in den Speicher lädt, die nach dem Booten noch gebraucht werden. Danach residiert das DOS in zweien der verfügbaren Banks und braucht nicht von Fall zu Fall vom Systemlaufwerk nachgeladen zu werden. Eine Ausnahme bilden lediglich INHALT/SYS (DIR/SYS), SYS4/SYS (DOS-Fehlermeldungen) und SYS8/SYS (DIR-Befehl). INHALT/SYS muß auf der Diskette verwaltet werden, um beim erneuten Booten keine bösen Überraschungen zu erleben. Wie sich bei Tests herausstellte, treten beim Aufruf eines RAM-residenten SYS4/SYS und SYS8/SYS Fehler auf, weil diese offenbar die RST-28h-Routine modifizieren. Sie werden deshalb wie gehabt von der Diskette abgerufen.

Das Programm beginnt mit dem FCB, der zum Laden der SYS-Files gebraucht wird. Es schließt sich eine Tabelle an, in der die diskrelativen (physikalischen) Adressen der Systemdateien verzeichnet sind und deren Länge in Sektoren. Mit diesen Informationen kann das Programm die Systemmoduln laden. Das Byte, das normalerweise die Länge angibt, enthält an zwei Stellen eine andere Information: Die Speicherblocks, die der Banker zugänglich macht, sind je 32 kB lang. Das ist zu wenig für das gesamte DOS. Nach dem Laden von SYS12/SYS wird deshalb von Bank 1 auf Bank 2 umgeschaltet, um dort den Rest unterzubringen. Das Signal zum Umschalten gibt das Byte 255 in Zeile 48. Am Ende der Tabelle findet sich eine 0, die dieses Ende kenntlich macht.

Das Programm beginnt beim Label start (Z. 85). Dort wird der Banker durch einen beliebigen Output auf den Port 224 so konfiguriert, daß er die Umschaltungen im Adreßbereich ab 8000h vornimmt. Nicht alle Bernhardt-Banker haben dieses Feature. Die Umschaltung muß, wie auch immer, A15 gegen *A15 austauschen. Das geht auch zu Fuß mit einem gewöhnlichen Schalter. Ebenso ist es möglich, die normale Speicherkonfiguration beizubehalten. Dann liegt der Common-Bereich in der oberen Speicherhälfte, was für den Betrieb mit CP/M besser ist. Nun kann z. B. ein Programm im Himem die Arbeit übernehmen, das sich vom hier vorgestellten nur um wenige Bytes unterscheiden müßte.

Anschließend an diese Operation wird mit dem Output 1 auf den Port ECh in diesen Adreßraum die Bank 1 selektiert, wo die erste Hälfte des DOS geladen werden soll. Das passiert in der Schleife tabloop. Der FCB wird von Sektor zu Sektor jeweils aktualisiert. Wird das Byte 255 angetroffen, schaltet der Banker um auf Bank 2. Dort wird die zweite Hälfte wieder ab 8000h geladen. Wenn das 0-Byte erkannt wird, ist tabloop fertig abgearbeitet, und es geht bei tabdone (Z. 127) weiter.

Hier wird das Programm zum Interpretieren eines DOS-Requestcodes, Label loadsys (Z. 150), an seine endgültige Arbeitsadresse geladen. Sie lautet hier 7900h. Dort ist nämlich das "sichere Plätzchen" von Helmut Bernhardt (beschrieben in 6/85). Es handelt sich um freies RAM in dem Adreßbereich, wo ab Werk die Tastatur überflüssigerweise Speicherplatz verschwendet. Wer das "sichere Plätzchen" nicht eingebaut hat, kann einen beliebigen anderen Speicherbereich zwischen 5000h und 7FFFh wählen (oder 8000–FFFFh, wenn der Common-Bereich oben liegt, s. o.). Helmut hat bisher zwei Versionen seiner RAM-Platine in 7900–79FFh veröffentlicht (eine davon im Info 10/85 des Bremerhavener Clubs).

Ein anderer Ausweg ist der EG 64 MBA. Er erlaubt es z. B., die RST-28h-Umleitung "neben" den Bildschirm ab 3C00h oder in den Adreßbereich des Sonder-ROMs ab 3000h zu legen. Dazu werden auf den Port DFh die Codes 0Ah und 0Bh ausgegeben, um den Speicherbereich 3000–37FFh auf RAM zu schalten. Dann lautet der offset (Z. 150) $-3000h.

Leider führten Versuche zu Systemabstürzen, bei denen das Arbeitsprogramm im Bereich der ursprünglichen RST-28h-Routine lief. Dies geschah jedoch nur bei einigen wenigen DOS-Anforderungen. Bei entsprechenden Checks und Ausschlüssen bestimmter Requestcodes (s. u.) wäre das aber machbar. Die meisten und wichtigsten Systemfiles konnten sogar ohne solche Vorsichtsmaßnahmen geladen und benutzt werden, z. B. auch Debug.

Um Speicherplatz zu sparen, wird der größte Teil der Arbeit zum Finden der zuständigen SYS-Datei in den beiden reservierten Banks erledigt, und zwar von dem Programmsegment findsys (Z. 178). Dieser Abschnitt kommt in den oberen, frei gebliebenen Teil der beiden Nachbarbanks 1 und 2.

Zuletzt wird die ursprüngliche Speicherkonfiguration wiederhergestellt. Bank 0 wird zugeschaltet (0 auf Port ECh) und die RST-28h-Routine gepatcht: Bis zur Adresse 4BDFh hat sie die nötigen Vorarbeiten geleistet, ohne die auch RAMSYS nicht auskommt. Von hier ab geht es aber darum, das Directory nach dem zuständigen Systemfile abzusuchen, was ohne rotierende Floppy nicht zu machen ist. Und genau das soll mit RAMSYS aber vermieden werden. Deshalb wird an der Stelle 4BE0h ein Sprung nach loadsys gepatcht (ab Z. 142). Damit sind alle Initialisierungen erledigt, und mit RET kehrt das Programm ins DOS zurück.

In der RST-28h-Umleitung loadsys wird geprüft, ob der Requestcode eines der kritischen Files (s. o.) betrifft. In diesem Falle geht es in der alten RST-28h-Routine weiter, die dann in Gottes Namen die Drives ankurbelt. Erlaubte DOS-Moduln werden aber aus dem RAM geholt. Da sie auf zwei Banks verteilt sind, muß die richtige zunächst herausgefunden werden. Sie geht aus dem Requestcode hervor. Wenn er 0Fh oder mehr beträgt, liegt die Datei in Bank 2, sonst in Bank 1. Dabei bezieht sich die Angabe des Requestcodes übrigens auf seinen um die Auswahl-Bits 5–7 verkürzten Rumpf. Die entsprechende Bank wird angewählt. Da unter BASIC und bei vielen Anwenderprogrammen der Stack im oberen Adreßbereich ab 8000h liegt, lädt die Routine den Stackpointer mit dem Himem der Nachbarbank. Jetzt wird findsys aufgerufen, um das zuständige Systemfile zu finden:

Da nicht alle SYS-Files gleich lang sind (SYS6/SYS umfaßt 35 Sektoren) und da in beiden Banks im Akku für die erste Datei der Requestcode 0 erwartet wird, muß anfangs ein wenig gerechnet werden (Zz. 190 ff.). Nun hat der Akku seinen korrekten Wert, und die SYS-Dateien können abgezählt werden, bis die richtige gefunden ist (ab sectok). In der Folge werden die einzelnen Bytes des Record-Headers nach ihrer Bedeutung interpretiert und ausgewertet. Die eigentlichen Programm- bzw. Datenbytes werden in den Speicher geladen, wo sie auch ab Floppy hingekommen wären. Dabei sorgt eine Sicherung dafür, daß Kommentar-Records schnuppe im Niemandsland, nämlich in einem freien Bereich der Nachbarbank verschwinden.

Wenn die Systemdatei vollständig übertragen ist, kehrt findsys zum CALLer loadsys zurück. Der restauriert als erstes wieder den Stack und schaltet die Bank 0 ein. Die vorsichtshalber abgeschalteten Interrupts werden wieder zugelassen. An der Stelle 4C19h, wohin loadsys zuletzt springt, wird hauptsächlich das Systemprogramm angesprungen. Seine Startadresse wurde in findsys in den Adreß-Operanden des CALL-Befehls bei 4C1Dh geladen. Im weiteren Verlauf macht das DOS genau dasselbe, als hätte es die Systemdatei von der Diskette geholt.

Diese DOS-Vergewaltigung ist relativ umfangreich, aber dennoch um ein Vielfaches schneller als die ursprüngliche RST-28h-Prozedur. Das Auffinden der Systemdatei im Inhaltsverzeichnis und das Laden von der Floppy erübrigen sich nämlich. Zum Test kann der Leser beispielsweise JKL eingeben. Der Drucker beginnt sofort mit der Arbeit, die Laufwerke bleiben stehen.

Zuletzt noch ein Hinweis: Die DOS-Moduln liegen in den verschiedenen DOS-Versionen an unterschiedlichen physikalischen Disk-Adressen. Mit SUPERZAP oder einem ähnlichen Programm muß daher zunächst überprüft werden, ob vielleicht Änderungen in der Tabelle tab (Zz. 24 ff.) erforderlich sind. Die Programme werden (bei SUPERZAP) mit DFE aufgerufen. In der Anzeige ist danach ablesbar, um welchen diskrelativen Sektor es sich handelt (drs).

Es gibt auch DOSses, bei denen mehr Systemdateien als nur SYS6/SYS über 5 Sektoren lang sind. Das gilt z. B. für SYS25/SYS von G-DOS 2.1b. SUPERZAP und DDE helfen dabei, es herauszufinden. In diesem Falle muß ähnlich wie wegen SYS6/SYS in Z. 183 der Requestcode bereinigt werden. Da es sich dabei aber um simple binäre Arithmetik handelt, sollte dies kaum abschrecken.

U. U. kann es auch erforderlich werden, den Bankwechsel anderswo als bei SYS13/SYS vorzunehmen. Man sollte vorsichtshalber alle relevanten Systemsektoren addieren, die Anzahl halbieren und die Trennung entsprechend vornehmen. Da genügend Platz vorhanden ist, kommt es auf ein Ungleichgewicht von ein paar Sektoren aber nicht an. Wichtig ist dabei nur, daß für das Unterprogramm findsys und eventuelle Kommentarrecords noch 315 Bytes übrig sind. Sie an das oberste Ende der Banks 1 und 2 zu legen, ist bei äußerster Platznot auch keine Hürde: Das Label offset2 kann mit maximal FEC4h statt F000h definiert werden. Wichtig ist lediglich, daß die erste Systemdatei der neuen Bank fünf Sektoren lang ist. RAMSYS unterstellt nämlich, daß bei dem angeblichen Sektorzähler 255 in Wirklichkeit 5 gemeint ist.

Hier ein Hinweis für die dann notwendigen Änderungen: In der Zeile 161 wird der Akku mit 0Fh verglichen. Das ist der Rumpf-Requestcode für SYS12/SYS. Er erhöht bzw. erniedrigt sich mit der laufenden Nummer der Systemdatei. Er ist immer um zwei höher als deren Nummer. In den Zz. 184 und 186 hat er eine Subtraktion um 3 hinter sich. Hier beträgt die Differenz nunmehr minus eins. Ist es also SYS13/SYS, dann muß hier mit 0Eh+1Eh verglichen werden. 0Eh ist dezimal 11 (12-1).

Das liest sich furchtbar trocken und ist es wohl auch. Solche Modifikationen sind aber nur bei DOS-Exoten wie DOSPLUS, Genie-Text u. dergl. zu erwarten. Es sei den geneigten Leser empfohlen, ein rechtschaffenes System wie NEW-, G- oder H-DOS zu fahren. Alles andere verdirbt eh' den Charakter.

Arnulf Sopp

Bitte beachtet dazu auch die Seiten 91 – 99 dieses INFO's


```
                00001 ;*****************************************************
                00002 ;*                                                   *
                00003 ;*                    R A M S Y S                    *
                00004 ;*                                                   *
                00005 ;*      Utility zum RAM-residenten Betrieb von       *
                00006 ;*      NEWDOS-80, G- und H-DOS mit dem 256-kB-      *
                00007 ;*            Banker von Helmut Bernhardt            *
                00008 ;*                                                   *
                00009 ;*              (C) 1986 by Arnulf Sopp              *
                00010 ;*                                                   *
                00011 ;*****************************************************
                00012
5300            00013          ORG     5300h
                00014
                00015 ;FCB für diskrelative Sektor-I/O
5300 82         00016 fcb      DB      82h,28h,00h             ;diskrelativ usw.
5303 0052       00017          DW      5200h                   ;Sektorpuffer
5305 00         00018          DB      00h,00h,00h,00h,00h     ;hier belanglos
530A 0000       00019 sector   DW      0000h                   ;Sektornummer
530C FFFF       00020          DW      0ffffh,0ffffh,0ffffh,0ffffh,0ffffh ;s. o.
5316 FFFF       00021          DW      0ffffh,0ffffh,0ffffh,0ffffh,0ffffh ;s. o.
                00022
                00023 ;Tabelle der File-Längen und diskrelativen Sektoradressen
5320 05         00024 tab      DB      5               ;Länge in Sektoren von
5321 9605       00025          DW      0596h           ;SYS1/SYS, Startsektor
5323 05         00026          DB      5               ;dto. SYS2
5324 9B05       00027          DW      059Bh
5326 05         00028          DB      5               ;SYS3
5327 BE05       00029          DW      05beh
5329 05         00030          DB      5               ;SYS4
532A C305       00031          DW      05c3h
532C 05         00032          DB      5               ;SYS5
532D D205       00033          DW      05d2h
532F 23         00034          DB      35              ;SYS6
5330 D705       00035          DW      05d7h
5332 05         00036          DB      5               ;SYS7
5333 7805       00037          DW      0578h
5335 05         00038          DB      5               ;SYS8
5336 C805       00039          DW      05c8h
5338 05         00040          DB      5               ;SYS9
5339 CD05       00041          DW      05cdh
533B 05         00042          DB      5               ;SYS10
533C 8C05       00043          DW      058ch
533E 05         00044          DB      5               ;SYS11
533F 6905       00045          DW      0569h
5341 05         00046          DB      5               ;SYS12
5342 FA05       00047          DW      05fah
5344 FF         00048          DB      255             ;SYS13. Bank umschalten
5345 9705       00049          DW      0587h
5347 05         00050          DB      5               ;SYS14
5348 7705       00051          DW      0577h
534A 05         00052          DB      5               ;SYS15
534B 6E05       00053          DW      05eeh
534D 05         00054          DB      5               ;SYS16
534E 6405       00055          DW      05e4h
5350 05         00056          DB      5               ;SYS17
5351 5F05       00057          DW      05f4h
5353 05         00058          DB      5               ;SYS18
5354 8205       00059          DW      0582h
5356 05         00060          DB      5               ;SYS19
5357 7D05       00061          DW      057dh
5359 05         00062          DB      5               ;SYS20
535A 9105       00063          DW      0591h
535C 05         00064          DB      5               ;SYS21
```

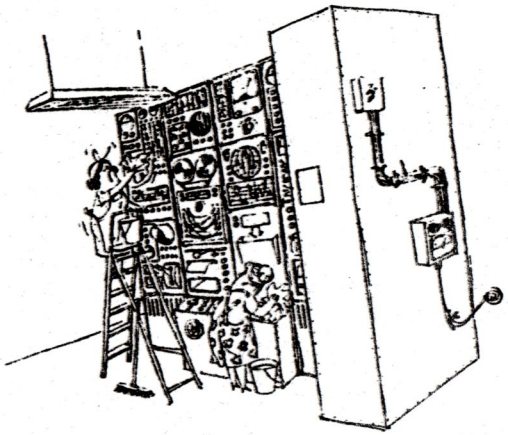
```
535D FF05      00065          DW      05ffh
535F 05        00066          DB      5               ;SYS22
5360 0E06      00067          DW      060eh
5362 05        00068          DB      5               ;SYS23
5363 5A05      00069          DW      055ah
5365 05        00070          DB      5               ;SYS24
5366 1306      00071          DW      0613h
5368 05        00072          DB      5               ;SYS25
5369 0406      00073          DW      0604h
536B 05        00074          DB      5               ;SYS26
536C 4605      00075          DW      0546h
536E 05        00076          DB      5               ;SYS27
536F 4B05      00077          DW      054bh
5371 05        00078          DB      5               ;SYS28
5372 5005      00079          DW      0550h
5374 05        00080          DB      5               ;SYS29
5375 5505      00081          DW      0555h
5377 00        00082          DB      0               ;Flag für Tabellenende
               00083
               00084 ;SYS-Dateien in die Puffer laden
5378 D3ED      00085 start    OUT     (0edh),A        ;Common-Bereich unten
537A 3E01      00086          LD      A,1             ;Bank 1 zuschalten
537C D3EC      00087          OUT     (0ech),A
537E 212053    00088          LD      HL,tab          ;Anfang der Sektortabelle
5381 F3        00089          DI                      ;vorsichtshalber
5382 46        00090 tabloop  LD      B,(HL)          ;Länge des SYS-Files
5383 04        00091          INC     B               ;Ende 1. Hälfte?
5384 200B      00092          JR      NZ,noend        ;falls noch nicht
5386 3E02      00093          LD      A,2             ;sonst Bank 2 zuschalten
5388 D3EC      00094          OUT     (0ech),A
538A 3E80      00095          LD      A,80h           ;MSB Sektorpuffer
538C 32AD53    00096          LD      (filbuf),A      ;neu festlegen
538F 0606      00097          LD      B,6             ;Sektorzähler + 1
5391 05        00098 noend    DEC     B               ;Tabelle ganz zuende?
5392 2829      00099          JR      Z,tabdone       ;falls ja
5394 23        00100          INC     HL              ;LSB der Sektornummer
5395 5E        00101          LD      E,(HL)          ;DE <- Sektornummer, LSB
5396 23        00102          INC     HL              ;auf MSB stellen
5397 56        00103          LD      D,(HL)          ;D <- MSB
5398 ED530A53  00104          LD      (sector),DE     ;Sektornummer in den FCB
539C 23        00105          INC     HL              ;auf nächstes Längenbyte
539D E5        00106          PUSH    HL              ;retten
539E 11005C    00107          LD      DE,fcb          ;zum Lesen der Sektoren
53A1 210052    00108          LD      HL,5200h        ;Sektorpuffer
53A4 CD3644    00109 readloop CALL    4436h           ;Sektor einlesen
53A7 D9        00110          EXX                     ;Register retten
53A8 210052    00111          LD      HL,5200h        ;Sektorpuffer
53AB 118080    00112          LD      DE,8080h        ;Filepuffer
53AD           00113 filbuf   EQU     $-1             ;MSB veränderlich
53AE 010001    00114          LD      BC,0100h        ;Sektorlänge
53B1 EDB0      00115          LDIR                    ;Sektor übertragen
53B3 21AD53    00116          LD      HL,filbuf       ;MSB der File-Pufferadr.
53B6 34        00117          INC     (HL)            ;256 Bytes weiterstellen
53B7 D9        00118          EXX                     ;Register restaurieren
53B8 10EA      00119          DJNZ    readloop        ;bis 8 Sekt. eingelesen
53BA E1        00120          POP     HL              ;Tabellenzeiger
53BB 18C5      00121          JR      tabloop         ;nächstes SYS-File
               00122
53BD 21EF53    00123 tabdone  LD      HL,loadsys      ;Anf. der RST-28h-Routine
53C0 110039    00124          LD      DE,loadsys-offset1      ;dorthin versch.
53C3 013100    00125          LD      BC,findsys-loadsys      ;Länge der Rout.
53C6 EDB0      00126          LDIR                    ;übertragen
53C8 2120E4    00127          LD      HL,findsys      ;Anf. der RST-28h-Umleit.
53CB 1100F0    00128          LD      DE,findsys+offset2      ;dorthin versch.
53CE 013B00    00129          LD      BC,endprog-findsys      ;Länge der Uml.
53D1 E5        00130          PUSH    HL              ;alles retten
53D2 D5        00131          PUSH    DE
53D3 C5        00132          PUSH    BC
53D4 EDB0      00133          LDIR                    ;Umleitung übertragen
53D6 3E01      00134          LD      A,1             ;Bank 1 selektieren
53D8 D3EC      00135          OUT     (0ech),A        ;(seit tabloop Bank 2)
53DA C1        00136          POP     BC              ;Register restaurieren
53DB D1        00137          POP     DE
53DC E1        00138          POP     HL
53DD EDB0      00139          LDIR                    ;Uml. auch in Bank 1
53DF AF        00140          XOR     A               ;Arbeitsbank 0
53E0 D3EC      00141          OUT     (0ech),A        ;zuschalten
53E2 3EC3      00142          LD      A,0c3h          ;JP-Opcode
53E4 32E04B    00143          LD      (4be0h),A       ;dort laden
53E7 210039    00144          LD      HL,loadsys-offset1      ;Anf. RST 28h
53EA 22E14B    00145          LD      (4be1h),HL      ;umpatchen
53ED FB        00146          EI                      ;INTs wieder zulassen
53EE C9        00147          RET                     ;fertig, zurück ins DOS
               00148
               00149 ;Programm zum Laden und Anspringen der RAM-SYS-Files
1AEF           00150 offset1  EQU     $-3900h         ;zum Verladen
53EF 77        00151 loadsys  LD      (HL),A          ;SYS-File vermerken
53F0 FE0B      00152          CP      0bh             ;SYS9/SYS oder höher?
53F2 300D      00153          JR      NC,ramsys       ;falls ja
53F4 E607      00154          AND     07h             ;signif. Teil d. Rec.-Cd.
53F6 FE03      00155          CP      03h             ;ab SYS1, ohne SYS8
53F8 DAE34B    00156          JP      C,4be3h         ;andere normal bearbeiten
53FB FE06      00157          CP      06h             ;SYS4/SYS?
53FD CAE34B    00158          JP      Z,4be3h         ;normal bearbeiten
5400 7E        00159          LD      A,(HL)          ;Requestcode restaurieren
5401 57        00160 ramsys   LD      D,A             ;und retten
5402 FE0F      00161          CP      0fh             ;SYS13 oder höher?
5404 3E01      00162          LD      A,1             ;Voreinstellung Bank 1
5406 3801      00163          JR      C,bankok        ;falls SYS1 - SYS12
5408 3C        00164          INC     A               ;Bank 2 selektieren
5409 D3EC      00165 bankok   OUT     (0ech),A        ;zuständ. B. zuschalten
540B F3        00166          DI                      ;vorsichtshalber
540C ED733839  00167          LD      (spbuf),SP      ;Stackpointer retten
5410 310000    00168          LD      SP,0000h        ;Stack im Himem
5413 CD00F0    00169          CALL    findsys+offset2 ;SYS-File aus RAM laden
5416 310000    00170          LD      SP,0000h        ;Dummy-Operand
3938           00171 spbuf    EQU     $-2-offset1     ;hier Pointer gepuffert
5419 AF        00172          XOR     A               ;A <- 0 für Bank 0
541A D3EC      00173          OUT     (0ech),A        ;Bank 0 zuschalten
541C FB        00174          EI                      ;INTs wieder zulassen
541D C3194C    00175          JP      4c19h           ;SYS-File starten u. zur.
               00176
               00177 ;Programm zum Herunterladen der SYS-Files
98E0           00178 offset2  EQU     0f000h-$        ;zum Verladen
5420 7A        00179 findsys  LD      A,D             ;Requestcode
5421 D603      00180          SUB     03h             ;an SYS-File# angleichen
5423 FE05      00181          CP      05h             ;SYS6/SYS oder höher?
5425 3808      00182          JR      C,sectok        ;falls darunter
5427 C61E      00183          ADD     A,1eh           ;sonst 30 Sektoren mehr
5429 FE2A      00184          CP      0ch+1eh         ;SYS13/SYS oder höher?
542B 3802      00185          JR      C,sectok        ;falls darunter
542D D62A      00186          SUB     0ch+1eh         ;an SYS13 ff. angleichen
542F 210005    00187 sectok   LD      HL,0500h        ;HL = 1280 Bytes = 5 Skt.
5432 CD944C    00188          CALL    4c94h           ;AHL <- HL * A
5435 118080    00189          LD      DE,8080h        ;Pufferanfang
5438 19        00190          ADD     HL,DE           ;HL <- Adr. d. SYS-Files
5439 5E        00191 movelcc  LD      E,(HL)          ;1. Byte des Rec.-headers
543A 23        00192          INC     HL              ;auf nächstes Byte
```

```
543B  0600      00193          LD      B,00h           ;Byterähler 00xx
543D  7E        00194          LD      A,(HL)          ;Längenbyte
543E  FE03      00195          CP      03h             ;256, 257 oder 258?
5440  CB10      00196          RL      B               ;Zähler-MSB anpassen
5442  4F        00197          LD      C,A             ;BC <- Zählbyte
5443  7B        00198          LD      A,E             ;Typbyte
5444  23        00199          INC     HL              ;auf Adreß-LSB
5445  5E        00200          LD      E,(HL)          ;DE <- Zieladresse
5446  23        00201          INC     HL
5447  56        00202          LD      D,(HL)
5448  ED531E4C  00203          LD      (4c1eh),DE      ;Startadresse
544C  23        00204          INC     HL              ;1. Programmbyte
544D  3D        00205          DEC     A               ;Typbyte 01?
544E  2805      00206          JR      Z,cmdrec        ;falls ja
5450  3D        00207          DEC     A               ;Typbyte 02?
5451  C8        00208          RET     Z               ;Ende, falls ja
5452  113BF0    00209          LD      DE,endprog+offset2      ;Komment. dorthin
5455  0B        00210 cmdrec   DEC     BC              ;um 2 vermindern, weil
5456  0B        00211          DEC     BC              ;Adreßbytes mitgezählt
5457  EDB0      00212          LDIR                    ;Record übertragen
5459  18DE      00213          JR      movelop         ;bis SYS-File übertragen
545B            00214 endprog  EQU     $
                00215
5378            00216          END     start

00000 Fehler

bankok  5409    cmdrec  5455    endprog 545B    fcb     5300    filbuf  53AD
findsys 5420    loadsys 53ER    movelop 5439    noend   5391    offset1 1AEF
offset2 9BE0    ramsys  5401    readlop 53A4    sectok  542F    sector  530A
spbuf   3928    start   5378    tab     5320    tabdone 53BD    tabloop 5382
```

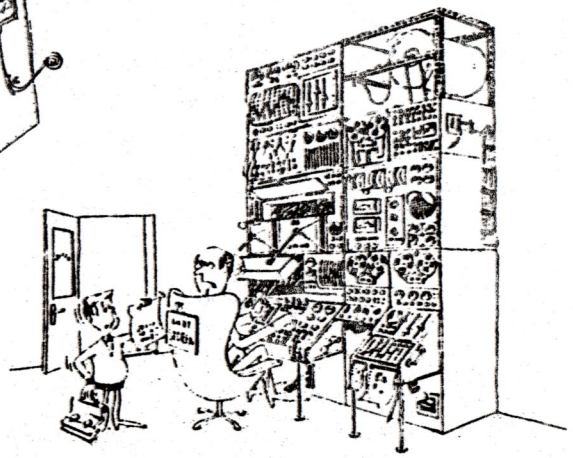
Eine Elektrogeräte-Fabrik in Los Angeles produziert batteriegetriebene Spaghetti-Gabeln, die Nudeln automatisch aufrollen. Eine Ersatzbatterie wird serienmäßig mitgeliefert, damit stets ein ungetrübter und ungestörter Eßgenuß gesichert ist.

Über einen Schwachstromtechniker, der sein Hinterteil mit Stanniol eingeschalt, durch die Verpackung dann Strom geleitet und dabei schwere Verbrennungen erlitten hatte, urteilt die Wiener »Klinik Hoff« in einem psychiatrischen Gutachten: »Dieser Mann ist völlig normal. Seine Handlung läßt lediglich die besonders innige Beziehung zu seiner beruflichen Tätigkeit erkennen.«

Bedingt durch vollautomatischen Rechnungsverkehr erhielten in einer bundesdeutschen Großstadt wohnende Bürger, die wochenlang verreist waren und deshalb keinen Strom verbrauchten, von den Stadtwerken Rechnungen in Höhe von 0,00 Mark. Wird der Betrag nicht fristgerecht eingezahlt, bitten die Stadtwerke »dringend, den Rechnungsbetrag von 0,00 Mark und 0,70 Mark Mahnkosten« zu überweisen. Im Weigerungsfalle – so wird den Kunden angedroht – würden sie »ohne weitere Mahnung« von der Strombelieferung ausgeschlossen.



»Wenn der schon alles kann, warum kann er sich dann nicht selbst abstauben?«



»Ist mir egal, ob dein Opa hier der Chef ist – mach' deine Schulaufgaben allein!«

Wie der Natuerliche Logarithmus die Rechenzeit verkuerzen kann

MITHILFE DES NATUERLICHEN LOGARITHMUS KOENNEN KOMPLIZIERTE MATHEMATISCHE AUSDRUECKE, DIE AUS MULTIPLIKATION, DIVISION, EXPONENTIATION UND WURZELAUSDRUECKEN BESTEHEN, IN EINE AUCH FUER 8-BIT CPU's HANDHABBARE FORM GEBRACHT WERDEN.

Peter Guagliano, COMTECH GOVERNMENT SYSTEMS

Um auch komplizierte arithmetische Probleme mit 8-bit CPU's in Assembler loesen zu koennen, sei sich der einfachen Handhabung der Logarithmen erinnert.
Solch ein Gedanke kann - richtig angewandet - den Einsatz eines Arithmetik-Prozessors ueberfluessig machen. Sinn und Zweck dieser Ueberlegungen ist es, die Benutzung von langsamen High-Level-Language Bibliotheksroutinen ueberfluessig zu machen, die nicht selten ein festes Parameteruebergabeformat bedueurfen.
Mithilfe des Natuerlichen Logarithmus ist es nun moeglich, zu multiplizieren, dividieren (alles trotz fehlender Hardware-Befehle) oder sogar Wurzelrechnung und Exponentiation. Eine kurze Uebersicht ueber die moeglichen logarithmenverknuerfungen zeigt die nachfolgende Tabelle :

- log(ab) = log(a) + log(b)
- log(a/b) = log(a) - log(b)
- log(sqrt(a)) = 1/2 * log(a)
- log(a*a) = 2 * log(a)
- log(a) fuer a <= 0 ist nicht definiert !

Um obige Gleichungen in Mikroprozessor-Applikationen nutzen zu koennen, wird ein Algorithmus zur Umwandlung von Binaerzahlen in den binaeren Logarithmus benoetigt.
Da negative Zahlen und Null nicht definiert sind, bleiben diese unberuecksichtigt. Damit der verwendete Algorithmus einfach gehalten werden kann, erfolgt eine Einengung des Wertebereiches auf Zahlen >= 1. Sollen dennoch einmal positive ganze Zahlen mit gebrochenen Zahlen multipliziert oder dividiert werden, ist die inverse Operation zu benutzen. Ein Beispiel :

```
        y = 2 / 0.1
statt dessen
        y = 2 * 10
```

NATUERLICHER LOGARITHMUS EINFACH INTERPOLIERT

Bild 1 zeigt den Funktionsgrafen des nat. Log., in Bild 2 ist ein vergroesserter Ausschnitt wiedergegeben. In diesem Bereich ist die Funktion nahezu linear, ein Umstand der fuer einige Vereinfachungen im Umwandlungs-Algorithmus genutzt werden kann.
Nun jedoch zu den Einzelheiten des benutzten Algorithmus.

Es sei X eine binaere Zahl. Der Exponent zur Basis 2 sei N, gegeben durch die Position der hoechstwertigsten 1 in der Binaerzahl. Zur Erinnerung : Die Nullen und Einsen repraesentieren die Koeffizienten a der Stellenwertigkeiten.

$$a_0 2^0 + a_1 2^1 + \ldots + a_k 2^k$$

Demzufolge ist a der Koeffizient von 2^N . Im einzelnen :

```
00010110, N = 4
01011001, N = 6
11010011, N = 7
```

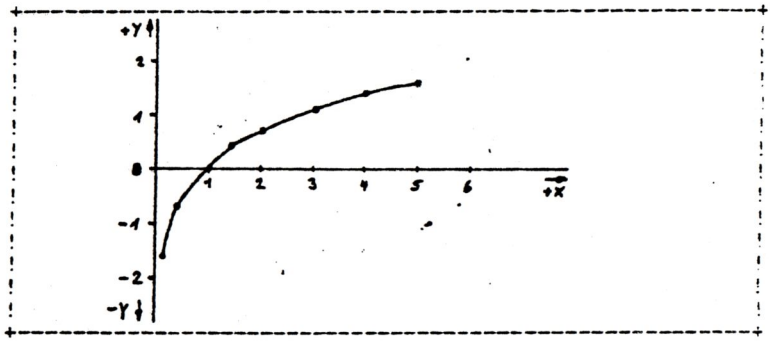


Bild 1: Lineare Approximation der Funktion $y = \log_2(x)$ im Intervall 1<X<2. Die durchgezogene Line stellt dabei die beste Annaeherung an den tatsaechlichen Funktionsverlauf dar.

Durch anwenden des Algorithmus kann die Charakteristik N des binaeren Logarithmus einer Zahl (ganzzahlig) sowie der gebrochene Anteil (=Mantisse) bestimmt werden.
Das Folgende zeigt, wie N und Mantisse bestimmt werden koennen. Hierbei sei X eine positive ganze Zahl, die folgendermassen dargestellt werden kann :

$$X = 2^N + (X - 2^N)$$

Umformen ergibt

$$x = 2^N + 2^N \left[\frac{X - 2^N}{2^N} \right]$$

Als Produkt

$$X = 2^N \left[1 + \frac{X - 2^N}{2^N} \right]$$

Logarithmieren obiger Gleichung und umschreiben in die Form Log(ab) = Log(a) + Log(b) ergibt

$$\log(X) = \log(2^N) + \log \left[1 + \frac{X - 2^N}{2^N} \right]$$

und weiter

$$\log(X) = N + \log\left[1 + \frac{X - 2^N}{2^N}\right] \qquad (2)$$

Hierbei nimmt der Term (X - 2)/2 - er ergibt den Divisionsrest, nachdem das MSB-Bit per Shift entfernt wurde - die Werte zwischen Null und Eins an.

Die gestrichelte Linie in Bild 2 ergibt fuer Gleichung 1 folgende lineare Approximation fuer Werte zwischen 1 und

$$\log(X) = N + \frac{X - 2^N}{2^N}$$

Diese Interpolations-Gl. ergibt jedoch einen max. Fehler von 9%.Hinzu kommt ausserdem,dass der Fehler einseitig und die ermittelten Werte stets zu klein sind.
Der mittlere Fehler beluaeuft sich somit auf 4%, summiert sich aber mit jeder durchgefuehrten Rechnung weiter auf.

Ein erster Ansatz, den mittleren Fehler zu reduzieren, besteht darin, eine guenstiger liegende Interpolations-Gerade zu waehlen (siehe Bild 2).

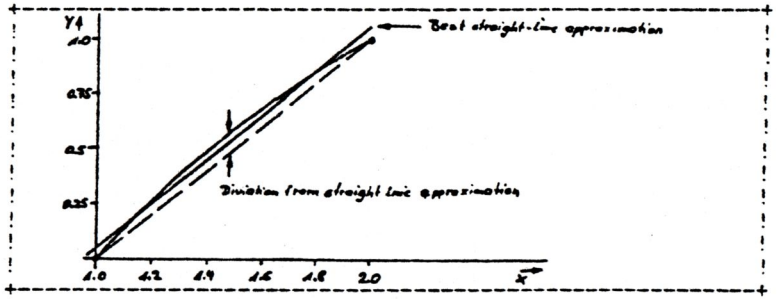


Bild 2: Lineare Interpolation fuer y = log (X) im Intervall 1 < X < 2. Die durchgezogene Gerade stellt die guenstigste Loesung dar.

Um die gewuenschte Interpolations-Gerade zu erhalten, genuegt die Addition einer Konstanten in Hoehe von 5% oder 13 Teile von 256. Die Addition dieser Konstanten reduziert den max. Fehler auf 4% und den mittleren Fehler auf Null. Damit wird der Rechenfehler zu Null, sofern die Verteilung der auftretenden Fehler gleichmaessig ist. Andernfalls koennen bei einseitiger Fehlerverteilung dennoch groessere Rechenfehler entstehen, z.B. wenn der Logarithmus von Zahlen in Intervallmitte oder an den Intervallgrenzen gebildet wird.

Eine bessere Loesung ist es, den Term $(X - 2^N)/2^N$ dazu zu benutzen, eine 256 Byte umfassende Tabelle mit den tatsaechlichen Logarithmen-Werten zu adressieren. Der Worst-Case-Fehler bei dieser Methode ist dann kleiner 1%, wobei der Rechenfehler auch hier gegen Null geht. Diese Operation benoetigt zusaetzliche 3 usec (Z80, 4Mhz) und zwei Tabellen (fuer den Logarithmus/Antilogarithmus).


```
! COMPUTELOG: LD    B,0FH          ;MAX. LOOP-COUNT      !
! LOGLOOP:    ADD   HL,HL          ;SHIFT HL LEFT        !
!             JR    C,GOTN         ;FOUND N ON CARRY     !
!             DJNZ  LOGLOOP        ;LOOP                 !
!             LD    H,B            ;RETURN ZERO IN HL    !
!             RET                  ;END OF CONVERSION    !
! GOTN:       LD    L,H            ;L GETS 8 BIT REMAIN- !
!                                  ;DERFRACTION          !
!             LD    H,LOGTABLE     ;H GETS PAGE OF LOG-  !
!                                  ;LOOK-UP TABLE        !
!             LD    A,(HL)         ;PERF. TABLE LOOK-UP  !
!             LD    L,A            ;RETURN MANTISSA      !
!             LD    H,B            ;RETURN CHARACTERISTIC!
!             RET                  ;END OF CONVERSION    !
! LOGTABLE:   EQU   080H           ;PAGE OF LOG LOOK-UP  !
!                                  ;TABLE                !
```

Bild 3: Die Z80 Assembler-Routine, welche den Binaer-Logarithmus fuer 16-Bit Integerzahlen ermittelt

```
! ANTILOG:    LD    A,H            ;GET CHARACTERISTIC   !
!             CP    10H            ;OVERFLOW ?           !
!             JP    OVERFLOW                             !
!             LD    B,H            ;MOVE CHARACT. INTO   !
!                                  ;LOOP COUNT           !
!             INC   B              ;INC LOOP-CNT BY 1    !
!             LD    H,INVLOG       ;H GETS PAGE OF INV.  !
!                                  ;LOG TABLE            !
!             LD    A,(HL)         ;PERFORM INV. LOOK-UP !
!             LD    HL,0000                              !
!             STC                  ;SET CARRY            !
! INVLOOP:    RL    L              ;CY <- L <- CY        !
!             RL    H              ;CY <- H <- CY        !
!             SLA   A              ;CY <- A <- CY        !
!             DJNZ  INVLOOP        ;LOOP                 !
!             RET                                        !
! OVERFLOW:   LD    HL,0FFFFH      ;MAX-INTEGER          !
!             RET                                        !
! INVLOG:     EQU   089H           ;PAGE OF INV. LOG TAB.!
```

Bild 4: Sobald das Endergebnis in logarithmischer Form vorliegt,kann diese Z80 Routine den inversen Logarithmus bilden.

EIN BEISPIEL BEWEIST DIE BRAUCHBARKEIT

Die Routine im nachfolsenden Anwendunssbeispiel uebernimmt den zu verarbeitenden Wert in einem 16-Bit Resister. Der zurueckselieferte Losarithmus ist ebenfalls im 16-Bit Format. Die Charakteristik wird in einem 8-Bit Resister uebersehen, ebenso wie die 8-Bit Mantisse (der verbleibende Rest). Die Ersebenisse aller Berechnunsen sind grundsaetzlich in 16-Bit Format. Der sich daraus ersebende Wertebereich ist somit :

$$1 <= X <= 2^{16} - 1$$

Soll die Losarithmen-Rechnung zum Loesen von Gleichunsen benutzt werden, aussedrueckt in Form eines 16-Bit Losarithmus, so besitzen die Zwischenwerte einen Wertebereich von 1 ... 2^{255} . Dies ersibt sich aus der Tatsache, dass eine 8-Bit Charakteristik verwendet wird, die einen Maximalwert von 255 ermoeslicht, bevor ein Ueberlauf erfolst. Die derzeitise Genauiskeit ist auf 8-Bit besrenzt, kann jedoch leicht erweitert werden.

Bild 3 zeist eine Z80 Assembler-Routine fuer die Berechnung des 16-Bit Inteser-Losarithmus (unter der Vorraussetzung das die Zahl positiv und sroesser Eins ist). Der 8-Bit Exponent wird im Resister H, die Mantisse im Resister L uebersehen. Beim verwendeten Alsorithmus wird das HL-Resister so lanse nach links seschoben, bis ein Uebertras erfolst. Die Subtraktion der dazu noetisen Anzahl von Schritten von 16 ersibt H, Resister H enthaelt den Rest. Mithilfe dieses Restes kann nun eine 256 Byte umfassende Los.-Tabelle adressiert werden, welche den senauen Losarithmus enthaelt.
Die Laufzeit dieser Routine haenst von den zu verarbeitenden Daten ab, duerfte im Mittel jedoch (bei einem 4Mhz Takt bei ca. 100 usec) liesen.
Bild 4 zeist eine Routine fuer die inverse Operation, also Los. -> Inteser. Auch sie basiert auf dem oben beschriebenen Verfahren, wendet es jedoch invers an.
Wiederrum ist hier die Laufzeit von den Daten abhaensis, wie auch hierfuer eine 256 Byte lanse Tabelle vorhanden sein muss. Die mittlere Verarbeitunsszeit (4Mhz Takt) betraest ca. 120 usec.
Bild 5 zeist ein PASCAL-Prosramm zur Berechnung der beiden Tabellen.

IM MITTEL WERTEUNABHAENGIGE LAUFZEIT

Es ist zu beachten, dass die Zeit zur Berechnung des Antilosarithmus fuer srosse Werte hoeher liest als die zur Berechnung des Losarithmus (bei sleichem Wert). Aehnlich verhaelt es sich mit kleinen Werten. Hier ist jedoch die Rechenzeit fuer den Losarithmus laenser als fuer den Antilosarithmus. Deshalb wird die Gesamtzeit zur Berechnung des Losarithmus einer Zahl mit anschliessender Rueckwandlung im Mittel ueber den in Frase kommenden Zahlenbereich konstant sein.
Als eine Art Anwendunsbeispiel zeist Bild 6, wie die Routinen aus Bild 3 & 4 fuer die Berechnung der Gleichung $Y = X^{2.5}$ benutzt werden koennen.

```
! PROGRAM LOG.TABLES (INPUT,OUTPUT)                          !
! VAR                                                        !
!     I:INTEGER                                              !
! BEGIN                                                      !
!     WRITELN (' LOG TRANSFORMATION TABLE ...');             !
!     WRITELN (' ============================');             !
!     FOR I:=0 TO 255 DO                                     !
!       WRITELN(I:5,ROUND(256*LN(1+I/256)/LN(2)):9);         !
!     FOR I:=0 TO 10 DO  WRITELN;                            !
!     WRITELN (' ANTILOG TRANSFORMATION TABLE ...');         !
!     WRITELN (' ==================================');       !
!     FOR I:=0 TO 255 DO                                     !
!       WRITELN (I:5,ROUND(256*(EXP(I/256*LN(2))-1)):9);     !
! END.                                                       !
```

Bild 5: Das PASCAL-Prosramm fuer die Erstellung der Los und Antilos-Tabellen zu den Routinen aus Bild 3 und Bild 4.

Das Resister HL uebersibt das Arsument X bei Aufruf und und enthaelt das Resultat nach Rueckkehr von den jeweilisen Wandlunss-Routinen. Bemerkenswert ist die Methode, wie das Prosramm in Bild 6 die Quadratwurzel (mittels Rechtsschieben) und das Quadrat (mittels Linksschieben) bildet.

SO WERDEN DIE KOSTEN FUER EINEN CO-PROZESSOR GESPART

Die mittlere Ausfuehrunsszeit der Routine aus Bild 6 liest bei ca. 250 usec (4Mhz CPU Takt). Im Gesensatz hierzu benoetist ein INTEL 8231A, ein adequater Co-Prozessor fuer einen Z80, 2.5 msec um einen Zahl zu quadrieren (Der Ehrlichkeit halber muss jedoch sesast werden, dass der 8231A fuer das senannte Beispiel etwas zu starker Toback ist. Beherrscht er doch 16 und 32-Bit Gleitkomma-Arithmetik in den ueblichen mathematischen Funktionen).

Ein weiteres Beispiel (Bild 7) soll die Moeslichkeiten der Routinen aus Bild 3 & 4 zeisen. Die Routine in Bild 7 berechnet die nachfolsende Gleichung :

$$X = \sqrt{\frac{A^2}{313} + \frac{B^2}{462}}$$

Das Arsument A (16 Bit) und das Arsument B (ebenfalls 16 Bit) wird in den Resistern HL und DE uebersehen. Das Ersebnis wird im Resister HL zurueckseliefert. Beide Arsumente muessen >= 1 sein.
Hervorzuheben ist, dass die Zwischenersebnisse bei der Quadrierung von A und B im losarithmischen Masstab vorliesen und somit nicht durch ein Zahlenueberlauf verfaelscht werden koennen, auch nicht bei sehr srossen

Zahlen. Eine Codeoptimierung ist nicht erfolgt, trotzdem betraegt die Laufzeit nur 700 usec bei 4Mhz Takt. Der 8231A benoetigt fuer die gleiche Aufgabe ca. 500usec.

Man kann die in Bild 3 & 4 gezeigten Routinen auf manche Weise erweitern. Eine waere z.B. die Erhoehung der Genauigkeit, eine andere die Erweiterung auf die Moeglichkeit zur Verarbeitung von gebrochenen Ergebnissen.

```
EXP2FIVE: CALL COMPUTELOG
          PUSH HL           ;SAVE LOGARITHM
          SRL  H            ;0  -> H -> CY
          RR   L            ;CY -> L -> CY
          PUSH HL           ;PUSH SQUARE ROOT
          POP DE
          POP DE
          POP HL            ;GET LOGARITHM
          ADD HL,DE         ;COMPUTE SQUARE
          ADD HL,DE         ;ADD IN SQUARE ROOT
          CALL ANTILOG      ;CONVERT BACK TO BINARY
          RET
```

TEST-DATA

INPUT	IDEAL RESULT	ACTUAL RESULT	ERROR
14	733	734	0.08%
27	3788	3776	0.3%
50	17678	17664	0.07%
63	31503	31488	0.05%
61	59049	59008	0.07%

BILD 6: Man kann Logarithmen-Rechnung in Routinen wie dieser benutzen um Zahlen zu Potenzieren, sogar wenn der Exponent gebrochen ist.

Einer von zwei Wegen die Genauigkeit zu erhoehen ist, weiterhin mit 16 Bit zu arbeiten, den Exponenten aber nur mit 4 Bit darzustellen. Dann stehen 12 Bit fuer die Mantisse zur Verfuegung. Das aber wiederrum begrenzt den moeglichen Zahlenbereich der Zwischenergebnisse (wegen des 4 Bit Exponenten).
Der andere Weg ist, den Exponenten weiterhin bei 8 Bit zu belassen, dafuer dann aber die Mantisse auf 24 Bit zu verlaengern. Das macht dann aber die Benutzung eines zusaetzlichen 8-Bit Registers erforderlich. Gleichgueltig welcher Weg beschritten wird, er bedeutet eine Aenderung in der Art wie die Log.-Tabelle angesprochen wird. Ein Beispiel :
Ein geaenderter Algorithmus im Sinne des zweiten Weges, wuerde die Mantisse so erzeugen. Zuerst wird der verbleibende Rest aus der Schiebeaktion als 16-Bit Zahl ausgedrueckt. Die oberen 8 Bit dienen nun zur Adressierung der 16-Bit Log.-Tabelle. Sobald nun der 16-Bit Logarithmenwert vorliegt, werden die unteren 8 Bit linear interpoliert und hinzuaddiert. Hierfuer

wird die Interpolations-Methode aus Bild 2 benutzt. Das hinzufuegen der unteren 8 Bit, wie genannt, verringert den Interpolation-Fehler einer nur 256 Eintraege umfassenden 16-Bit Tabelle.

Sollten in irgend einer Applikation einmal gebrochene Zahlen von noeten sein, so kann der Algorithmus dahingehend geaendert werden, dass alle 16-Bit Eingaben als 8-Bit Integer (high Byte) und 8-Bit gebrochener Anteil (low Byte) aufgefasst werden. Grundsaetzlich muss hierfuer der Initialisierungszaehler des Algorithmus geaendert werden, dieser Schleifenzaehler bestimmt den Exponenten. Wird die genannte Aenderung durchgefuehrt, dann liefert eine Zahl, deren 'most significant binary one' des high Byte im low Byte liegt, einen negativen Exponenten.

```
RMS:  CALL    COMPUTELOG
      ADD     HL,HL       ;PERFORM SQUARE
      LD      BC,LG313    ;BC GETS LOG OF 313, COULD
                          ;EASILY BE ANOTHER PARAM.
      OR      A           ;CLEAR CARRY
      SBC     HL,BC       ;PERFORM DIVISION
      CALL    ANTILOG     ;CONVERT BACK TO BINARY
      PUSH    HL          ;SAVE FIRST TERM
      PUSH    DE          ;PUSH TERM B
      POP     HL
      CALL    COMPUTELOG
      ADD     HL,HL       ;PERFORM SQUARE
      LD      BC,LG462    ;BC GETS LOG OF 462
      OR      A           ;RESET CARRY
      SBC     HL,BC       ;DO DIVISIN BY 462
      CALL    ANTILOG     ;CONVERT BACK TO BINARY
      POP     DE          ;GET FIRST TERM
      ADD     HL,DE       ;PERFORM ADDITION
      CALL    COMPUTELOG  ;COMPUTE LOG OF THE SUM
      SRL     H           ;0 --> H --> CY, PERFORM
                          ;SQUARE ROOT
      RR      L           ;CY --> L --> CY
      CALL    ANTILOG     ;CONVERT BACK TO BINARY
      RET                 ;READY
LG313 EQU     084AH       ;LOG OF 313
LG462 EQU     08DAH       ;LOG OF 462
```

Test-Data

A	B	IDEAL RESULT	ACTUAL RESULT
1000	2000	108.9	106
782	418	48.8	48
219	87	13.0	13

Bild 7 : Ein Beispiel fuer die Moeglichkeiten, die in der Logarithmenrechnung stecken. Die gezeigte Routine berechnet die Quadratwurzel der Summe von Quadraten.

Manchmal erscheint eine Gleichung durch Logarithmenrechnung nicht loesbar zu sein. In der Regel ist dies aber nur ein Umformungsproblem. Man denke sich ein Problem, welches die Sinus-Funktion benutzt. Weiterhin sei eine Routine benoetigt, die eine 16-Bit Integerzahl mit dem Cosinus eines Winkels multipliziert (beide als Variable vorliegend). Der Winkel sei dabei auf dem Bereich

$$-90^\circ < 0 <= +90^\circ$$

begrenzt. Die Gleichung

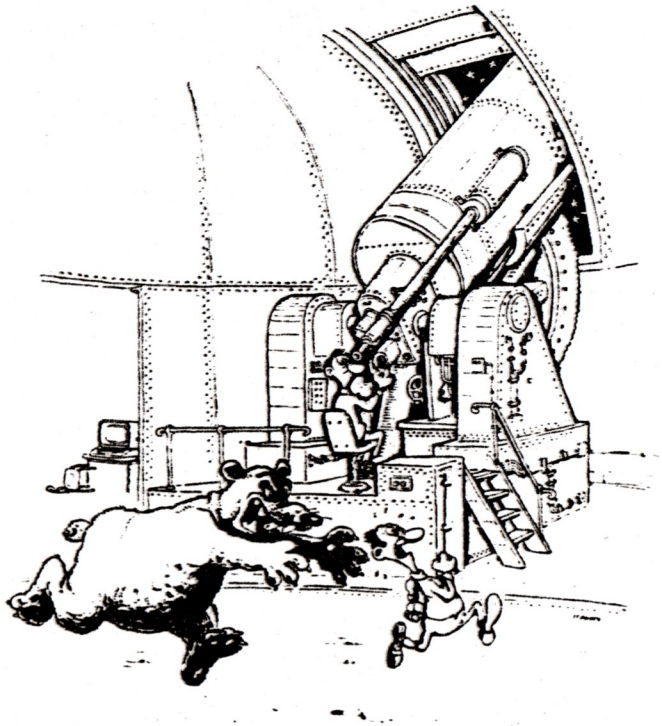
```
V = X * COS(TETA)
```

kann nicht direkt geloest werden, da log(cos(90)) nicht definiert ist. Umformen hilft hier weiter :

```
V = X * (1 + COS(TETA)) - X
```

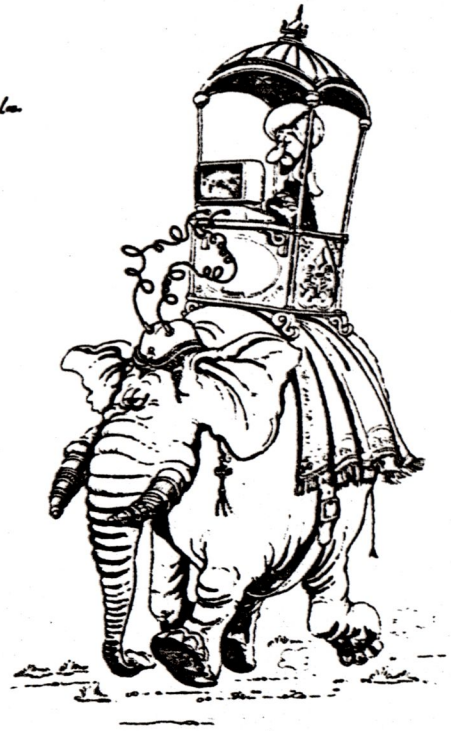
In dieser Form ist das Problem fuer unseren Algorithmus verdaulich. Eine passende Routine ist nun relativ schnell codiert. Jedoch wird eine dritte Tabelle benoetigt, in der die Werte fuer log(1+cos(teta)) abgelegt sind.

ENDE



Ich wünsche allen Clubmitgliedern, die Spaß am Softwaretüfteln haben, viel Vergnügen bei der Lektüre.

Euer Karl Meiller



Ich hab' doch nur vergessen einzugeben, „suche das Sternzeichen Großer Bär".

Floppytrick's

Ergänzung
Die im letzten Info veröffentlichte 80/40-Spurumschaltung wurde inzwischen von mir in das neue Model 4P von Günther Wagner eingebaut. Dabei stellte sich heraus, daß die Impulsbreite der durch die Schaltung zusätzlich erzeugten Stepimpulse für die verwendeten Shugartlaufwerke nicht ausreichten. Dieses Problem ließ sich jedoch durch Erhöhen der Kapazität des 56 nF-Kondensators aber leicht beheben. Ein genauer Wert ist hier nicht anzugeben, es muß nach der "Try and Error"-Methode vorgegangen werden. Die Schaltung funktioniert nun schon längere Zeit problemlos!

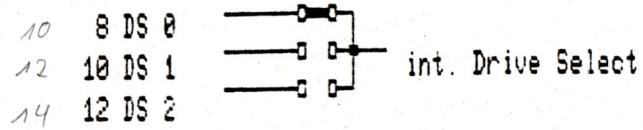
Noch eine Umschaltmöglichkeit
Für User die jeweils mindestens ein 40'er und ein 80'er Laufwerk ihr Eigen nennen, gibt es auch eine Möglichkeit Systemdisketten in beiden Spurzahlen zu booten. Die Möglichkeit besteht darin, die Drive-Select-Leitungen (DS) umzuschalten.
Die Auswahl des jeweils angesprochenen Laufwerks geschieht durch die Leitungen 10, 12 und 14 (Drive 0, 1 und 2) des Shugart-Bus. Beim Model 1 wird die Leitung 32 in Systemen die nur singlesided Laufwerke besitzen, zur Anwahl des Drives 3 benutzt. Sind Doppelkopflaufwerke im System, dient die Leitung zur Ansteuerung des "Hinterkopfes" (Backside select BS). Bei Model 3/4/4P-Geräten wird Laufwerk 3 durch die Leitung 6 selektiert und 32 dient als BS-Signal.

Belegung des Shugart-Bus:
alle ungeraden Pins führen Masse!

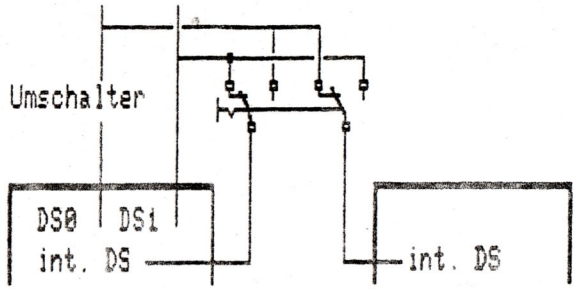
2	=	NC	nicht angeschlossen
4	=	NC	" "
6	=	DS 3 (NC)	Drive select 3 (im Model 1 NC)
8	=	Index Pulse	
10	=	DS 0	" " 0
12	=	DS 1	" " 1
14	=	DS 2	" " 2
16	=	Motor ON	
18	=	Direction Select	
20	=	Step	
22	=	Write Data	
24	=	Write Gate	
26	=	Track Zero	
28	=	Write Protect	
30	=	Read Data	
32	=	BS (DS 3)	Backside select (im Model 1 auch DS 3)
34	=	NC	nicht angeschlossen

Im Normalfall gehen alle diese Signale an alle angeschlossenen Laufwerke. Welches Laufwerk auf welche Adresse (0,1,2,3) reagiert, wird erst auf den Drives durch sog. Jumper oder kleine Schalter eingestellt. Das heißt, daß ein Laufwerk, welches als Drive 0 benutzt wurde, durch Umstecken einer Brücke oder Umschalten eines Schalters auch als Drive 1,2 oder 3 verwendet werden kann!

Prinzipschaltbild:



Entfernt man nun die Jumper von zwei Laufwerken und verschaltet sie nach untenstehendem Schaltbild, kann man durch Umschalten wählen, welches Laufwerk welche Adresse bekommt! Natürlich kann man die Schaltung auch mit 3 Laufwerken verwenden, man benutzt einfach DS 2 statt DS 1. Damit bleibt Laufwerk 1 immer unverändert und Drive 0 und 2 tauschen ihre Plätze im System.



Probleme mit dem Model 3
Bei original Model 3/4/4P - Geräten hat Tandy übrigens einen Fallstrick für Leute eingebaut, die ihr System selbst mit Doppelkopflaufwerken aufrüsten wollen. Die Radio Shack - Leute haben nämlich aus den Floppysteckern all die Pins herausgelassen, die bei den SS - Originallaufwerken nicht gebraucht werden. So fehlen im Stecker für Drive 0 die Pins 6, 12, 14 und 32. Dadurch können mit diesen Kabeln normalerweise keine DS - Laufwerke eingebaut werden, man muß sich neue Kabel besorgen. Mit folgendem Trick kann man sich eine Menge Geld sparen, da man die alten Kabel weiterverwenden kann!

Man entferne das Floppykabel und stecke alle Anschlüsse (auch den am Floppycontroller!!!) um 180° gedreht wieder auf. Das funktioniert, weil alle ungeraden Stifte auf einer Seite liegen und normalerweise Masse führen. Werden alle Stecker um 180° gedreht, sind auf der "Signalseite" nun alle Pins vorhanden und das Fehlen von 4 Masseleitungen (von insg. 17) ist nicht tragisch.

Speedjustage
Nochmals zur Justierung der Laufwerksgeschwindigkeit und damit zu LWT und der Anmerkung von Bernd Retzlaff zu meinen Änderungsvorschlägen in diesem Programm. LWT kann meiner Meinung nach immer nur eine Hilfe sein, ohne das Gerät öffnen zu müssen, die Laufwerksgeschwindigkeit festzustellen. Zur genauen Justage bedient man sich am besten des auf dem "Schwungrad" des Laufwerks angebrachten Strob ops.

Dazu wird das Laufwerk mit einer durch Netzspannung betriebenen Lichtquelle beleuchtet und das Justierpoti so lange gedreht, bis sich die aufgemalten Striche scheinbar nicht mehr bewegen! Vorsicht, auf der Scheibe sind meist zwei Einteilungen für 50 und 60 Hz Netzfrequenz aufgetragen. Also nicht die falsche benutzen!

So, das waren vorerst alle Tricks und Tips, die ich zur Zeit auf Lager habe. **Zum Schluß eine Bitte:** "Kramt doch mal in eurer Trickkiste, sicher fördert ihr etwas Interessantes zu Tage, was sich zu veröffentlichen lohnt. Auch wenn es auf den ersten Blick recht banal aussieht, kann man anderen Mitgliedern damit einen großen Schritt weiterhelfen!" In diesem Sinne viel Glück, Euer

Hartmut Obermann

An alle HRG-1b Benutzer !

Als ich mir vor ca. zwei Jahren die Grafik-Erweiterung der Fa. RB-Elektronik kaufte, hatte ich noch keine Vorstellung von dem, was noch im Kaufpreis enthalten war. Ich spreche von den technischen Unzulaenglichkeiten der Karte und den Loesungsvorschlaegen !.

Angefangen hat es damit, dass das Laufwerk 0 gelegentlich beim Nachladen irgentwelcher System-Module nicht so recht wusste 'was Sache war' und recht geraeuschvoll einmal ueber die Diskette und zurueck steppte, bis die richtige Spur eingestellt war. Eine Anfrage bei RB-Elektronik bestaetigte meinen Verdacht: Es stimmte etwas nicht. So erfuhr ich dann, dass beim ansprechen der Grafik-Karte der Disk-Kontroller gestoert wird und beim naechsten Diskzugriff ersteinmal nach Spur 0 laeuft und dann die angeforderte Spur anfaehrt. Das Problem sollte sich gemaess Auskunft durch umsetzen der $\overline{MREQ}$-Leitung auf $\overline{M1}$ beheben lassen (???). Den Grund konnte man mir leider nicht erklaeren. Hier tappte der Meister selbst im Dunklen. Leider funktionierte der Trick bei meinem Rechner nicht, warum auch immer. So hatte ich den Schwarzen Peter zurueck und durfte nun in eigener Sache nach dem Grund suchen. Der war dann auch bald gefunden : Falsches Hardware-Design!

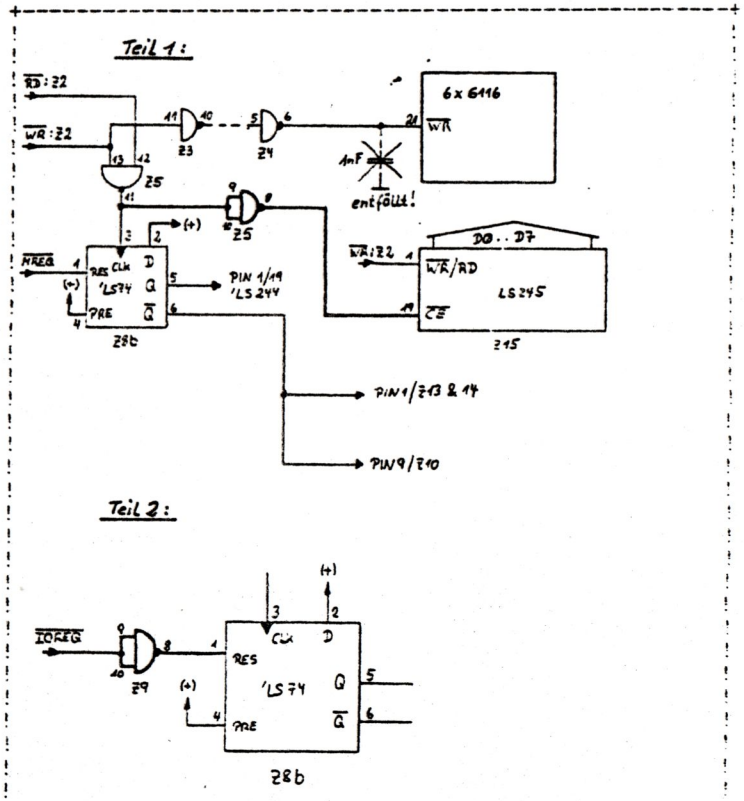
Sobald der Rechner auf die Grafik-Karte zugreifen will, wird das Flip-Flop Z8b umgeschaltet und dadurch die Ausleselogik der Karte gesperrt. Gleichzeitig wird der Datenbus-Treiber Z15 freigeschaltet. Ueber PIN1 von Z15 wird bei einem Schreibzugriff ausserdem noch die Richtung umgeschaltet. Die Uebertragungsrichtung nach dem Zugriff auf die Karte ist in Richtung Datenbus (!!!) und wird erst mit dem naechsten $\overline{MREQ}$-Zyklus aufgehoben. Bis dies geschieht, liegt ein Buskonflikt vor mit den oben genannten Folgen.
Der PIN19 von Z15 darf deshalb nicht an PIN6 von Z8b angeschlossen sein !. Dieser Pin gehoert an PIN11 Z5 oder an PIN3 Z8b. Hier liegt die NOR-Verknuepfung der $\overline{RD}$ & $\overline{WR}$ Clocks von Z2 an. Mit diesem Signal erfolgt die Freigabe von Z15 immer dann, wenn dies auch erforderlich. Ansonsten ist Z15 immer im TRI-State Zustand und stoert den Datenbus nicht mehr.
Allerdings muss das Signal zuvor invertiert werden, ansonsten passiert genau das Umgekehrte. Den benoetigten Inverter kann Z5 oder Z9 liefern. Hier sind noch einige frei.

Ausserdem sollte noch der Kondensator am PIN21 'WR' der 6116 RAM's entfernt werden. Er sollte die Probleme wohl etwas 'verrunden', verursacht aber bei abgeschalteter Grafik und '#CLS' Stoerungen auf dem Monitor. Wer ganz sauber arbeiten will, kann noch das $\overline{MREQ}$-Signal von PIN1 Z8b 'RES' abklemmen und stattdessen das $\overline{IOREQ}$-Sig. ueber einen Inverter benutzen. Dann wird die Kartenlogik mit Beendigung eines jeden Portzugriffs zurueckgeschaltet und nicht erst mit dem naechsten RAM-Zugriff.

Damit vermindert sich die Groesse der 'Schneeflocken' auf dem Monitor etwas.

Wem dies zu aufwendig ist, kann auch alles lassen wie es ist. Man sollte es sich dann allerdings verkneifen, die Grafik auf Diskette abspeichern zu wollen. Bedingt durch die 'Verwirrung' des Controlers kann das angesprochene Laufwerk (bei bereits gut gefuellter Diskette) beschaedigt werden, da der Kopfschlitten wiederholt gegen den Anschlag gefahren wird.

Viel Spass Kurt Müller



Auszug aus dem Schaltplan der HRG1B mit den Aenderungen

256K RAM für Z80-Systeme

Helmut Bernhardt

'64K-Byte RAM sind die obere Grenze für den vom Z80 adressierbaren Speicher'. Diese Behauptung ist nur dann richtig, wenn die direkte Adressierung gemeint ist. Durch Banking d.h. Umschalten eines Teilbereichs des 64K-Adreßraumes auf parallele Speicher läßt sich der vom Z80 nutzbare Speicher beliebig ausbauen; 1M-Byte-RAM-Karten werden heute für fast alle Z80-Systeme angeboten.

Gegenüber solchen Karten hat die im folgenden beschriebene Speichererweiterung den Vorteil, daß sie für wenig Geld zu realisieren ist und weder an einen bestimmten Busstandard noch gar einen speziellen Computer gebunden ist. Einzige Voraussetzung ist ein Z80-Prozessor.

Der Nachteil des Projektes besteht darin, daß man sich auf den CPU-Board ein bißchen auskennen muß und daß es ohne Lötkolben doch nicht geht.

Das Grunprinzip der Speichererweiterung ist ein Ersetzen der vorhandenen 4164-RAMs (oder auch 4116-RAMs) durch die pinkompatiblen 41256-Typen und eine mit wenigen Standard-ICs aufgebaute Banking-Logik, mit der ein gezielter Zugriff auf die vielen K-Byte möglich ist.

Die Banking-Logik

Vom Z80 aus gesehen ist der direkt adressierbare Speicherraum von 64K-Byte in zwei Blocks von 32K-Byte Größe unterteilt. Die obere Hälfte (Adressen 8000H-FFFFH) ist dabei nicht umschaltbar und deshalb vom Z80 immer erreichbar (Common-Bereich). Interrupt-Service-Routinen, der Stack und alle Routinen, auf die ständig zugegriffen werden muß, müssen hier liegen.

Die untere Hälfte ist in mehreren parallelen 32K-Blocks vorhanden. Welcher dieser Blocks im Adreßraum 0000H-7FFFH für den Z80 jeweils verfügbar sein soll, läßt sich durch Ausgabe der Nummer des Blocks (Bank Nr.) an einen Port (Latch) einstellen. Der Block ist dann solange als Speicher verfügbar, bis eine andere Bank Nr. ausgegeben wird. Durch RESET* wird beim Einschalten des Computers automatisch immer die Bank 0 eingestellt, um bei der Systeminitialisierung definierte Verhältnisse vorzugeben.

Von der Aufteilung der 256K (512K, 1024K) RAM her gesehen sind die Verhältnisse etwas anders zu sehen. Die unteren 64K sind nach dem Einschalten des Computers diejenigen, auf die der Z80 zugreifen kann. Alle anderen 32K-Blocks können anstelle des ursprünglichen 32K-Blocks in die untere Hälfte des Adreßraumes des Z80 gelegt werden, wenn eine der Bank Nummern 1-7 (15, 31) ausgegeben wird.

Das wird dadurch erreicht, daß durch die Latchausgänge Q1 und Q2 die Adressen A16 und A17 vorgegeben werden, die einen der vier 64K-Blocks innerhalb eines 256K-Blocks anwählen, Q3 und Q4 codieren einen der bis zu vier 256K-Blocks und Q0 und A15 des Systembus werden dafür herangezogen, die Adresse A15' für den Speicher zu liefern, um dort die untere oder obere Hälfte eines 64K-Blocks zu adressieren.

Die Herleitung von A15' über OR-Verknüpfung von A15 und Q0 sorgt dafür, daß immer dann, wenn A15 high ist, die obere Hälfte des 64K-Blocks angewählt wird (Common-Bereich), während dann, wenn A15 low ist, der Pegel des Latchausgangs Q0 den Pegel von A15' bestimmt und so zur richtigen Anwahl der eingestellten Bank beiträgt.

Da nun aber bei einer Adresse im Common-Bereich (A15=1) nicht nur die obere Hälfte irgendeines durch Q1 bis Q4 codierten 64K-Blocks im RAM sondern ausschließlich die obere Hälfte des unteren 64K-Blocks angewählt werden soll, wird durch den Pegel von A15 an ST8*-Eingang des 74LS157-Multiplexers (Abb.1) Low-Pegel ausgegeben.

Der Rest der Schaltung erfüllt die Aufgabe, zu erkennen, ob Bank Nr.1, die den gleichen 32K-Block im RAM wie der Common-Bereich belegen würde, eingestellt ist, und daraufhin die höchste verfügbare Bank zu adressieren. Das geschieht dadurch, daß bei eingestellter Bank Nr.1 das Bitmuster an den Latchausgängen zu einen High-Pegel am Select-Eingang des Multiplexers führt, der dann die Pegel der B-Eingänge, die über 4k7 an +5V gelegt sind, durchschaltet. Diese Sicherung des Common-Bereichs führt dazu, daß unter den Bank-Nummern 1 und 7 (15, 31) der gleiche 32K-Block adressiert wird. Die Bank 7 (bei 256K), 15 (bei 512K) bzw. 31 (bei 1024K) sollte also nicht benutzt werden.

Mehr als 256K RAM lassen sich dann realisieren, wenn im Computer mehrere Reihen mit 8 Sockeln für RAMs vorhanden sind, oder indem die RAM-ICs übereinandergelötet werden, wobei dann Pin 15 (CAS*) des aufgelöteten ICs hochgebogen werden muß. Durch die Drahtbrücken A-E muß die Schaltung dem Speicherausbau angepaßt werden. Je nach Speichergröße sind folgende Drahtbrücken zu legen:

256K	1 Reihe RAMs	Brücken A und D
512K	2 Reihen RAMs	" B " E
1024K	4 Reihen RAMs	" C " E

Diese Drahtbrücken sorgen dafür, daß je nach Speicherausbau der durch Bank Nr.1 angewählte 32K-Block identisch ist mit dem durch die höchste mögliche Bank-Nummer freizugebenden Block.

Bei mehr als 256K RAM müssen die einzelnen 256K-Blocks durch die Signale CASEN1* bis CASEN4* (bzw. deren Verknüpfung mit CAS* über OR-Gatter) freigegeben werden. Bei nur 256K RAM braucht die CAS*-Freigabe, wie sie vorher für 64K vorgelegen hat, nicht geändert zu werden.

Wenn 512K RAM realisiert worden sind, muß auf dem Banking-Board die Leiterbahn zu Pin13 des 74LS139 durchtrennt werden, an diesen Pin ist stattdessen GND zu legen, um zu verhindern, daß bei Ausgabe der Bank-Nr.1 nicht die nicht vorhandene Bank31 anstelle der Bank15 eingestellt wird.

Das Refreshing

Ein weiteres Problem stellt neben der Adressierung innerhalb der 41256-RAMs noch das Refreshing dar. Gegenüber den meisten 4164-Typen mit 128 Refresh-Zyklen (nur einige Ausnahmen) benötigen 41256-RAMs grundsätzlich 256 Refresh-Zyklen. Der Z80 liefert bei seinem Hidden Refresh aber nur eine 7-Bit-Refresh-Adresse.

Dieses Problem ist aber von Michael Hungershausen (c't 9/84, S.22) schon gelöst worden und soll deshalb hier nicht weiter abgehandelt werden. Der Vollständigkeit halber ist in Abb.1 eine der beiden vorgestellten Varianten für die Herleitung einer 8-Bit-Refresh-Afresse wiedergegeben.

Der Adreßmultiplexer

Ebenfalls in Verbindung mit dem Refreshing ist folgendes Problem zu sehen. Durch den Zyklus von 128 Refresh-Adressen ist es nötig, daß beim Adreß-Multiplex die Adressen A0 bis A6 zusammen durchgeschaltet werden. Anstelle von A7 kann aber auch irgend eine höhere Adresse zusammen mit A0-A6 durchgeschaltet werden. Bei Computern, in denen ursprünglich 4116-RAMs verwendet wurden, und die dann später auf 4164-RAMs umgerüstet wurden, muß sogar davon ausgegangen werden, daß dies der Fall ist, da für die interne Adressierung von 4116-RAMs die Adressen A0-A6 und A7-A13 abwechselnd durchgeschaltet werden.

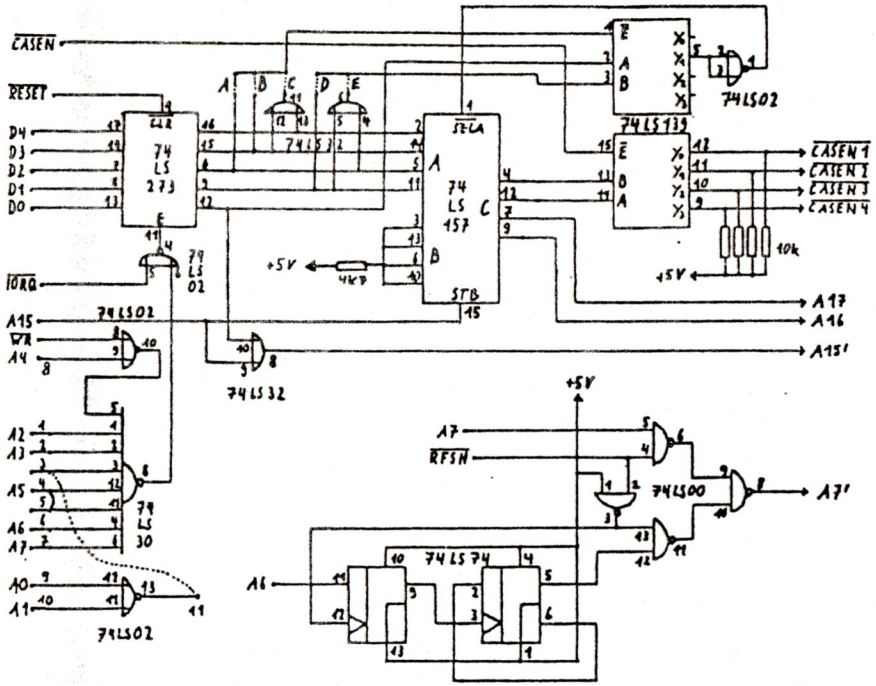
Es muß bei den Adreßmultiplexern also ermittelt werden, ob A7 mit A0-A6 zusammen durchgeschaltet wird. Wenn dies nicht der Fall ist, muß ermittelt werden, wo A7 anliegt und wo eine höhere Adresse mit A0-A6 zusammen durchgeschaltet wird. Beide Signale sind von den Pins der Multipexer abzutrennen; an den Pin, wo vorher die höhere Adresse anlag, ist A7' (Abb.1) zu legen und die dort abgetrennte Adreßleitung ist dahin zu führen, wo vorher A7 anlag.

Außerden muß durch ein huckepack-aufgelötetes 74LS157 (auf einen der schon vorhandenen Multiplexer; bei 74LS157 können die Pins 1,8,15 und 16 direkt angelötet werden) noch das Multiplexen von A16 und A17 möglich gemacht werden. Der Ausgang MA8 (Abb.2) wird (gegebenenfalls über einen 33-Ohm-Widerstand) an die Pins 1 der RAMs geführt.

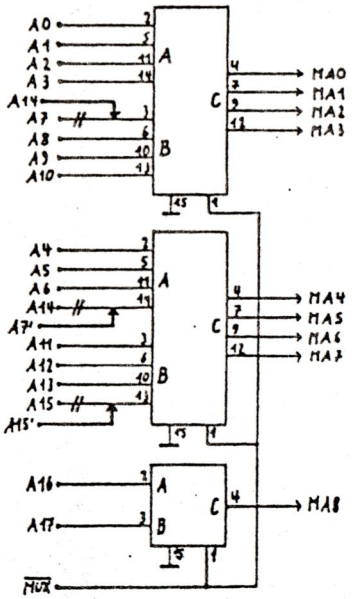
Anstelle von A15 muß das Signal A15' (Abb.1) an die Adreßmultiplexer geführt werden.

Bei Computern mit 4116-RAMs

müssen alle Speicherchips entfernt werden, und bei Vorhandensein getrennter Daten-Treiber für verschiedene 16K-Blöcke muß auch die Freigabe der Treiber nicht benutzter RAM-Reihen durch Anlegen von +5V über einen 4K7-Widerstand an den E*-Eingang der Treiber verhindert werden.



1) Schaltung des Banking-Boards und Generierung einer 8-Bit Refresh-Adresse



2) Änderung und Erweiterung des Adreßmultiplexers für die RAM-Adressen am hypothetischen Beispiel eines Computers, der schon von 4116-RAMs auf 4164-RAMs umgerüstet worden ist. Die Belegung der A-Eingänge mit den niederen Adressen ist nicht zwingend (siehe mc-CP/M-Computer). Von einem zusätzlich nötigen 74LS157 wird nur 1/4 benötigt.

Da diese Speicherchips mit -5V an Pin 1, +12V an Pin 8 und +5V an Pin 9 versorgt werden, müssen diese Leitungen von den Versorgungsspannungen abgetrennt werden. Die Versorgungsspannung +5V wird an die Pins 8 gelegt. Bei den die Pins 1 und die Pins 9 verbindenden Leitungen werden alle Abblockkondensatoren entfernt. An die Pins 9 wird MA7 und an die Pins 1 wird MA8 der Adreßmultiplexer gelegt.

Das CAS*-Signal an Pin 15 der 4116-RAMs ist ein mit einem 16K-Freigabesignal (meistens durch OR, 74LS32) verknüpftes CAS*. Anstelle des 16K-Freigabesignals muß ein low aktives (bei OR-Verknüpfung) Freigabesignal, das den gesamten als RAM vorgesehenen Adreßraum des Z80 umfaßt (nur ROM und memory-mapped I/O ausgeblendet) oder bei Speichererweiterung auf mehr als 256K eines der Signale CASEN1* - CASEN4* verwendet werden.

Banking der oberen 32K

ist nur dann sinnvoll, wenn es sich um einen BASIC-in-ROM-Computer, der nicht CP/M-fähig ist, handelt, und läßt sich durch Invertieren von A15 (durch z.B. 74LS04) für die Schaltung in Abb. 1 erreichen.

Einbau des Banking-Boards

Die Wahl der Portadresse für das Latch zum Einstellen der Bank Nr. wurde flexibel gehalten. Die Portadresse kann sich aus maximal 7, minimal 5 high aktiven und maximal 3, minimal 1 low aktiven Adreßbits zusammensetzen.

Low aktive Adreßleitungen werden an die Punkte 8 bis 10 angeschlossen. Der Punkt 8 muß auf alle Fälle belegt werden. Wenn dann nur noch eine zusätzliche Adreßleitung low sein soll, muß diese an beide Punkte (9 und 10) gelegt werden. Wenn die Punkte 9 und 10 belegt werden, muß Punkt 11 mit einem der Punkte 1-7 verbunden werden. Die high aktiven Adreßleitungen werden an die restlichen Punkte (der Punkte 1-7) gelegt. Nicht belegte Punkte (der Punkte 1-7) werden mit belegten Punkten verbunden. Selbstverständlich müssen A6 und A7 auch noch an die dafür vorgesehenen Punkte (n und m) neben den 74LS00 angeschlossen werden.

Die Adreßdecodierung sei am Beispiel der Portadresse ECH genauer beschrieben: ECH hat folgendes Bitmuster

A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	0	1	1	0	0

Es müssen dann A0, A1 und A4 als low aktive Adreßleitungen an die Punkte 8, 9 und 10 gelegt werden und wegen der Belegung der Punkte 9 und 10 muß Punkt 11 mit z.B. Punkt 3 verbunden werden. Die Adreßleitungen A2, A3, A5, A6 und A7 werden dann z.B. an die Punkte 1, 2, 4, 6 und 7 angeschlossen. Der unbenutzte Punkt 5 muß dann mit z.B. Punkt 4 verbunden werden.

Das Signal CASEN* ist das CAS*-Freigabe-Signal für einen mit 4164-RAMs bestückten Speicher. Wenn vor dem Umrüsten auf 41256-RAMs noch 4116-RAMs vorhanden waren, muß ein solches Signal noch hergeleitet werden. CASEN* muß bei allen Adressen, die weder ROM noch memory-mapped I/O ansprechen, low aktiv sein. Bei TRS 80 und Kompatiblen ist dies ein Signal 16-64K*, das durch Invertieren des Signals 0-16K* und - leider nicht überall durchgeführt - OR-Verknüpfung mit MERQ* hergeleitet wird.

Die Freigabe des Datenbus-Transceivers

für die RAMs und eventuell auch ROMs kann in einigen Computern größere Schwierigkeiten bereiten. Am einfachsten gestaltet sich die Lösung dieses Problems, wenn nur 8 Stück 4164-RAMs vorhanden sind und der zugehörige Datenbus-Transceiver ausschließlich die RAMs bedient. In diesem Fall kann die Freigabe des Transceivers so, wie sie war, belassen werden.

Auch dann, wenn vorher mehrere Reihen 4116-RAMs vorhanden waren und später durch Umrüsten auf 4164-RAMs in einer Reihe und Freilassen aller anderen Sockel die Freigabe eines für alle RAMs gemeinsamen Datenbus-Transceivers auf die 4164-RAMs angepaßt wurde, braucht bei Verwendung nun wieder aller Sockel die Freigabe des Transceivers nicht mehr geändert zu werden.

Bei Vorhandensein je eines Transceivers für eine Reihe RAMs (ursprünglich 4116) und nun Bestückung von 2 oder 4 Reihen mit 41256-RAMs müssen zur Freigabe der Transceiver die Signale CASEN1* bis CASEN4* des Banking-Boards verwendet werden.

Wenn für ROM und RAM bisher ein gemeinsamer Datenbus-Transceiver bestanden hat, so kann dessen Freigabe nur bei bisheriger Bestückung mit 4164-RAMs beibehalten werden. In Fällen, wo für ROM und einen 16K-RAM-Block (z.B. ältere GENIE-Modelle) ein gemeinsamer Datenbus-Treiber besteht, kann nur diese eine Reihe RAM-Sockel verwendet werden und die Freigabe muß die vollen 64K abzüglich von Adreßbereichen für memory-mapped I/O umfassen. Wenn in solchen Fällen eine Umrüstung auf 4164-RAMs schon durchgeführt worden ist, ist die Freigabe des Treibers bereits den Anforderungen für den Betrieb von 41256-RAMs mit Banking gerecht.

Anschluß des Banking-Boards

Signal	Anschlußpunkt Banking-Board	TRS 80 IC	Typ	Pin	GENIE I/II IC	Typ	Pin	Kontek 1 IC	Typ	Pin	mc-CP/M-Computer IC	Typ	Pin
D0	g	76	367	11	19	367	6	42	2732	9		Z80	14
D1	f	"	"	13	"	"	10	"	"	10		"	15
D2	e	75	"	13	"	"	4	"	"	11		"	12
D3	d	"	"	5	"	"	13	"	"	13		"	8
D4	c	"	"	7	"	"	11	"	"	14		"	7
A0	8	55	"	17	17	"	9	"	"	8		"	30
A1	9	"	"	13	"	"	7	"	"	7		"	31
A2	1	22	"	11	"	"	5	"	"	6		"	32
A3	2	"	"	13	"	"	3	"	"	5		"	33
A4	10	39	"	7	4	"	9	"	"	4		"	34
A5	4,5	"	"	9	"	"	7	"	"	3		"	35
A6	6,m	"	"	5	"	"	11	"	"	2		"	36
A7	7,n	"	"	11	"	"	5	"	"	1		"	37
A15	h	38	"	9	6	"	9	33	Z80	5		"	5
RESET* (SYSRES*)	a	37	02	1	5	Z80	26 #1	"	"	26 #1		"	26
WR* (OUT*)	l	23	32	3	16	367	11	"	"	22		"	22
IORQ* (OUT*)	k	"	"	"	2	"	9	"	"	20		"	20
CASEN*	b	74	00	10	36	04	2 #2	43'	04	2 #2	32	1	#7
RFSH*	o	40	Z80	28	16	367	3	33	Z80	28	Z80	28	
A7'	p	51	157	14	24	157	14 #3	30	157	5 #4	157	13	#8
A15'	q	52	"	6	23	"	6 #5	"	"	6 #5	"	14	#8
A16	r	51'	"	2 #6	24'	"	2 #6	34'	"	2 #6	"	3	#6
A17	s	"	"	3 #6	"	"	3 #6	"	"	3 #6		2	#6
CASEN1*	w	----------			----------			54	32	12	----------		
CASEN2*	v	----------			----------			2	2	1	----------		
CASEN3*	u	----------			----------			----------			----------		
CASEN4*	t	----------			----------			----------			----------		
+5V	j	52	157	16	23	157	16	30	157	16		Z80	11
GND	i	"	"	8	"	"	8	"	2	8		"	29

#1 Anstelle des reinen RESET*-Signals ist eine AND-Verknüpfung von RESET* und NMI* (Z80, Pin17) zweckmäßiger, um bei jedem Booten Bank 0 einzustellen.

#2 Anstelle dieses invertierten 0-16K*-Signals ist es sicherer, dieses Signal über ein OR-Gatter 74LS32 mit MERQ* (GENIE: IC16, 74LS367, Pin 5 / Kontek 1: IC33, Z80, Pin 19) zu verknüpfen.

#3 Das beim Umrüsten auf 4164-RAMs an diesen Pin gelegte Signal ist an Pin 13 von Z24, 74LS157 zu legen. Die dorthin führende Leiterbahn ist zu durchtrennen.

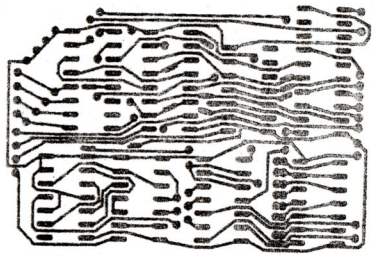
#4 Das beim Umrüsten auf 4164-RAMs an diesen Pin gelegte Signal ist an Pin 3 von IC 34, 74LS157 zu legen. Die dorthin führende Leiterbahn ist zu durchtrennen.

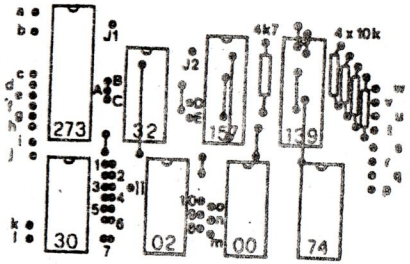
#5 Die an diesen Pin führende Leiterbahn ist zu durchtrennen.

#6 Dieses IC 74LS157 wird mit den Pins 1, 8, 15 und 16 huckepack auf einen anderen 74LS157-Multiplexer aufgelötet. Die Nummerierung bezieht sich auf das IC darunter.

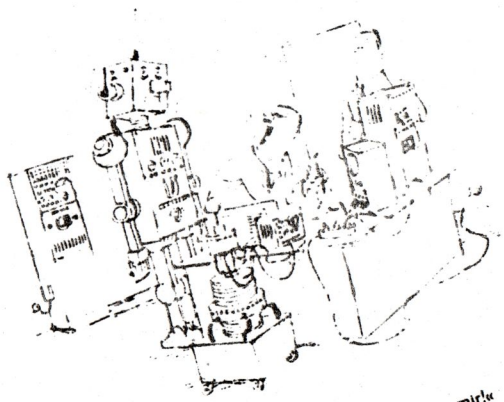
#7 74LS32 auf Höhe des EPROMs

#8 74LS157 beim u1-Kondensator





3) Layout, Bestückung, Drahtbrücken, Jumper und externe Anschlüsse



„Verdammt noch mal, sag nicht immer PAPA zu mir!«

»Ich werd' verrückt: Das Ding sagt: COGITO ERGO SUM!*«
* Ich denke, also bin ich

1M-Byte RAM im GENIE III

Einbauanleitung für das 256K-(1M-)Banking-Board

Wegen nichtz vorhandener Hardware-Dokumentation werden hier für den GENIE III alle wichtigen Informationen zum Umrüstzen auf 256K, 512K oder 1M RAM zusammengestellt.

Das Banking-Board wird dabei in geeigneter Weise unter den CPU-Board angebracht und die Verbindungen zwischen den Boards werden mit dünnen isolierten Drahtlitzen hergestellt. Bei den meisten Leitungen können die Enden der Litzen direkt auf der Lötseiten an die entsprechenden Pins der ICs gelötet werden.

Verbindungen zu Pins von Huckepack-ICs müssen durch Bohrlöcher, die an geeigneter Stelle im CPU-Board angelegt werden, geführt werden. Wenn man das Board gegen das Licht hält, sieht man freie Stellen, wo auf beiden Seiten keine Leiterbahnen verlaufen.

Es sollte darauf geachtet werden, daß die Drahtverbindungen möglichst kurz sind.

Im einzelnen sind folgende Maßnahmen durchzuführen:

1) Sämtliche 4116-RAMs werden durch 41256-RAMs ersetzt

2) Auf der Bestückungsseite wird die breite unter Z8 (74LS00) hindurchführende Leiterbahn (+12V) durchtrennt.

3) Auf der Lötseite wird die außen an den ROM-Sockeln vorbeiführende und mit "-5V" beschriftete Leiterbahn durchtrennt.

4) Auf der Lötseite wird die Verbindung zwischen Pin16 von Z12 (74LS161) zu Pin9 des RAMs daneben durchtrennt.

5) Auf der Lötseite wird die breite Leiterbahn (neben der mit "-5V" beschrifteten Leiterbahn) zwischen dem Widerstandsarray und dem Kondensator, neben dem auf der Bestückungsseite "D0" steht, durchtrennt.

7) Zwischen Pin8 des obersten linken (bestückungsseitig gesehen) RAMs und dem äußersten linken Pin des Widerstandsarays wird mit einem nicht zu dünnen, isolierten Draht eine Verbindung hergestellt.

8) Im Bereich der RAMs werden alle auf dem Bestückungsplan eingekreisten Kondensatoren ausgelötet.

9) Auf der Bestückungsseite werden die 4 nebeneinander verlaufenden Leiterbahnen von den Pins 4-7 von Z17 (74LS139) zu den Pins 2, 5, 10 und 13 von Z21 (74LS32) durchtrennt.

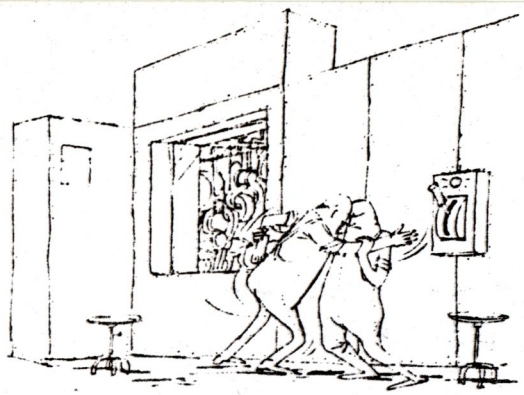
10) Auf der Lötseite wird die zu Pin3 von Z23 (74LS157) führende Leitung (A7) durchtrennt.

11) Auf Z23 (74LS157) wird ein zusätzliches IC 74LS157 mit den Pins 1, 8, 15 und 16 huckepack-aufgelötet (gleiche Orientierung); alle anderen Pins werden waagerecht abgebogen.
An die Pins 2 und 3 des Huckepack-ICs werden A16 und A17 des Banking-Boards geführt. Pin 4 wird über einen 33-Ohm-Widerstand mit den Pins 1 der RAMs (vorher -5V) verbunden.
An Pin5 wird A7' und an Pin6 wird A15' des Banking-Boards geführt. Pin7 wird über einen 33-Ohm-Widerstand mit den Pins 9 der RAMs (vorher +5V) verbunden.

12) Das Signal CASEN* für das Banking-Board wird von Pin1 von Z17 (74LS139) abgenommen.

13) RESET* kann an Pin26 der Z80-CPU abgenommen werden.
Wegen der Nutzung des NMI*-Signals zum Booten des DOS ist es aber zweckmäßig, anstelle des reinen RESET* für das Banking-Board lieber ein SYSRES*-Signal zu erzeugen. Dafür wird auf Z22 (74LS00) ein 74LS08 mit den Pins 7 und 14 huckepack-aufgelötet. Alle anderen Pins werden hochgebogen. Pin1 dieses ICs wird mit Pin17 des Z80 und Pin2 mit Pin26 des Z80 verbunden. An Pin3 wird das Signal SYSRES* abgenommen, das anstelle von RESET* an das Banking-Board geführt wird.

14) Die Signale D0-D5, A0-A7 und A15 werden von den im Auszug aus dem Bestückungsplan bezeichneten Pins der ICs Z25 (74LS245), sowie Z20 und Z16 (74LS244) abgenommen.



Fluch der Automation: »Weg da! Bei der letzten Störung vor vier Wochen hast du gesagt, daß ich den nächsten Handgriff tun darf!«

15) IORQ* wird von Pin3 von Z19 (74LS367), WR* von Pin13 und RFSH* von Pin5 von Z19 (74LS367) abgenommen.

16) +5V und GND können von beliebigen ICs in der Nähe der Stelle, wo das Banking-Board montiert wird, abgenommen werden.

```
  +5V: Z16, 20, 25 und 30  jeweils Pin 20
       Z17, 19, 23 und 26  jeweils Pin 16
       Z13, 14, 18, 21, 22 und 24  jeweils Pin 14
  GND: Z16, 20, 25 und 30  jeweils Pin 10
       Z17, 19, 23 und 26  jeweils Pin 8
       Z13, 14, 18, 21, 22 und 24  jeweils Pin 7
```

17) Die Signale CASEN1* bis CASEN4* werden (in dieser Reihenfolge) an die Pins 2, 10, 5 und 13 von Z21 (74LS32) gelegt. Wenn nur 256K RAM realisiert werden sollen, brauchen diese Signale nicht benutzt zu werden. Dann wird nur die linke Reihe RAM-Sockel mit 41256-RAMs bestückt und eine Verbindung zwischen Pin2 von Z21 (74LS32) und Pin1 von Z17 (74LS139) hergestellt.
Wenn 512K RAM gewünscht sind, werden die beiden linken RAM-Reihen bestückt und CASEN1* an Pin2 und CASEN2* an Pin10 von Z21 (74LS32) geführt.

18) Wenn das Banking in den oberen 32K des Z80-Adreßraumes erfolgen soll, muß ein invertiertes A15 an das Banking-Board geführt werden. Dafür kann ein freies NAND-Gatter von Z8 (74LS00) verwendet werden. A15 (von Pin9 von Z20, 74LS244) wird an die Pins 12 und 13 von Z8 gelegt und an Pin11 von Z8 kann das invertierte A15 abgenommen werden.

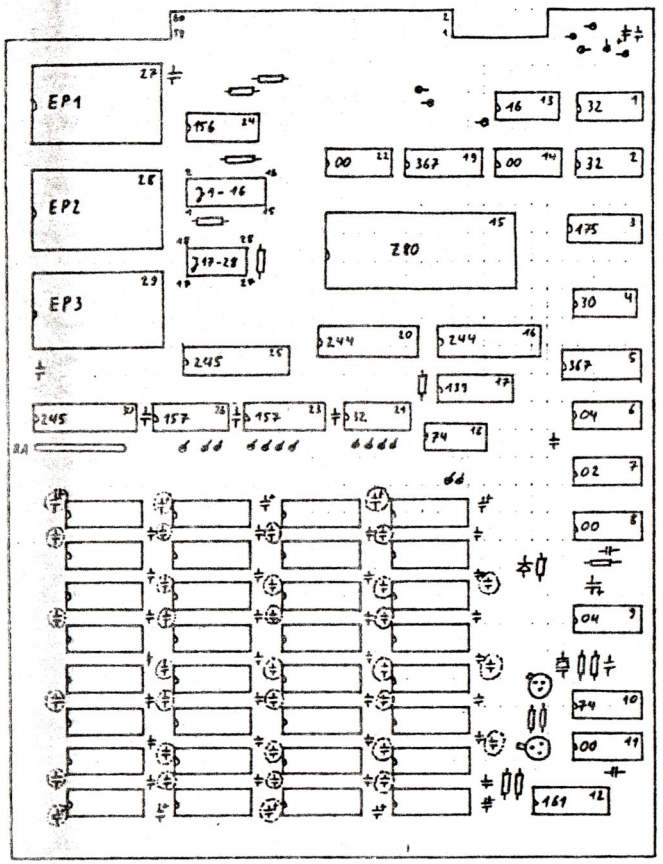
Um das Banking sowohl unter GDOS als auch unter CP/M (und wenn es das mal geben sollte: CP/M+) nutzen zu können, ist es zweckmäßig, zwischen A15 selbst und den invertierten A15 mit einem Umschalter wählen zu können.

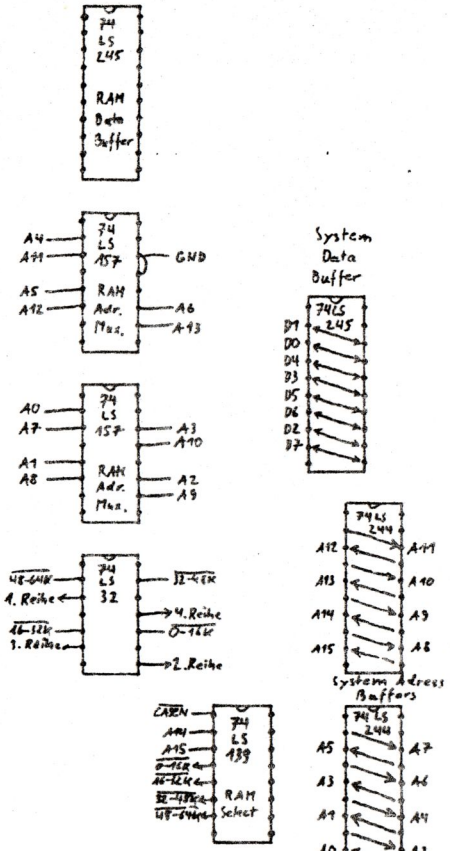
Da im GENIE III die Ports E0H bis FFH belegt sind, muß für das Banking-Board eine andere Portadresse gewählt werden. Geeignet ist die Portadresse DFH; dafür sind dann die Adreßleitungen an folgende Punkte anzuschließen:

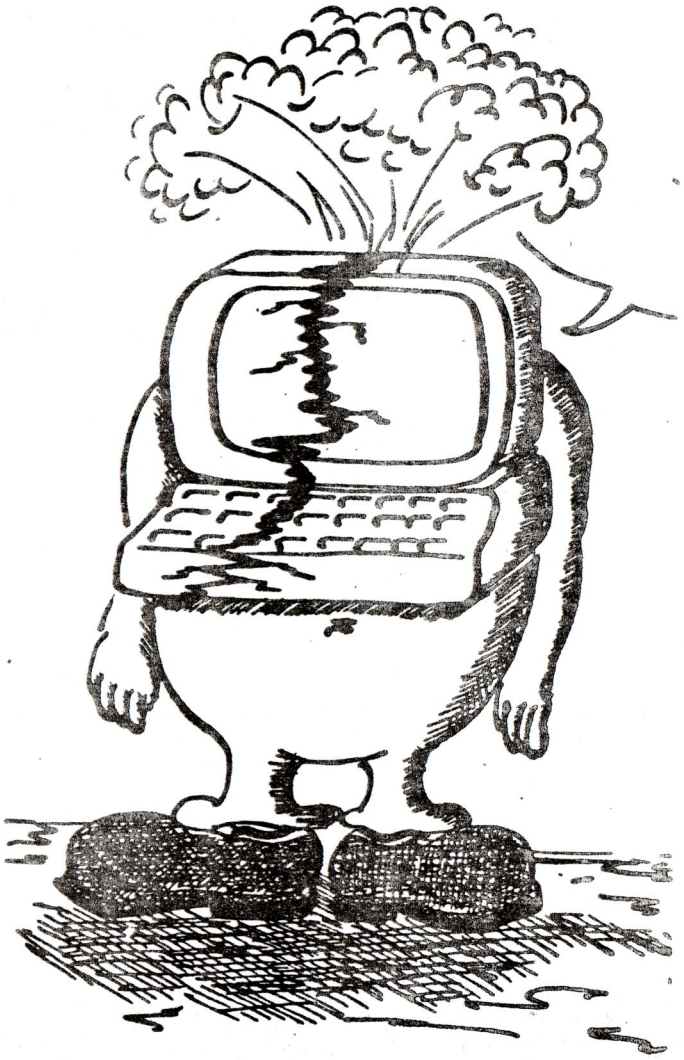
```
A7  an  7 und n
A6  an  6 und m
A5  an  8
A4  an  5
A3  an  4
A2  an  3
A1  an  2
A0  an  1
```

Der Punkt 11 wird mit keinem der Punkte 1-7 verbunden. Die Punkte 9 und 10 werden nicht angeschlossen.

Anmerkung
Die Einbauanleitung wurde aus einer Untersuchung des CPU-Boards eines GENIE III abgeleitet und noch nicht durchgeführt. Wegen der Analogie zum GENIE I ist aber nicht zu erwarten, daß die Erweiterung versagt.







Bitte beachtet dazu auch die
Seiten 69 - 75
dieses INFO's

CLUB 80 - ECB - BUS - SYSTEM

Im letzten INFO wurde die Idee geboren den TRS-80 als Selbstbau in ECB-Bus-Technik aufzubauen. Daraufhin meldeten sich bis jetzt schon sieben Mitglieder, die an diesem Projekt interessiert sind.

Wir möchten nun in dieser INFO ein Konzept vorstellen, daß mit seinen Möglichkeiten sicher viele von Euch anspricht.

ECB-Bus ---- Bus Erweiterung

In Verbindung mit dem aus der INFO schon länger bekannten ECB-Bus-Adapter und dem Aufbau des Grundbausteins können alle vorhandenen, sowie die für später geplanten, Zusatzplatinen an den schon vorhandenen Computer angeschlossen werden.
Der Grundbaustein besteht aus folgenden Teilgruppen:

--- 19 Zoll-Gehäuse
--- ECB-Bus-Platine, 96-polig mit 10 Karten-Steckplätzen
--- Stromversorgung, +12V/2A...-12V/0.4A...+5V/4A

ECB-Bus ---- Selbstbau TRS 80

Als weiteres besteht die Möglichkeit einen kompletten Rechner in ECB-Bus-Technik aufzubauen. Dazu ist auch der Aufbau des oben beschriebenen Grundbausteins erforderlich. Hinzu kommen noch 4 Platinen (der Computer).

SB1T80 ... CPU-Karte mit 64K-RAM, CP/M-fähig, doppelter Takt Bankinglogig
SB2T80 ... 80-Zeichenkarte mit 6845 und HRG 384 x 192 (RB-Kompatibel) (SoftwareKompatibel zum NEWDOS und ähnliche)
SB3T80 ... Tastatur-Karte mit Druckerausgabeadresse für Tandy
SB4T80 ... DD - FloppyKontrolerkarte mit Datenseperator

Die Karten SB3T80 und SB4T80 sind schon lauffähig. Die beiden anderen Karten sind noch in Entwicklung.

ECB-Bus ---- Zusatzplatinen

Die Zusatzplatinen passen für beide Bautypen unseres Projektes. Sie werden im einzelnen in den kommenden INFO's vorgestellt. Als lauffähige Karten sind in der Sammlung folgende Zusatzplatinen zum Nachbauen:

ZK01 ... Druckerkarte für Druckausgabe Video-Genie
ZK02 ... Universelle I/O-Karte, auf jede der 256 möglichen Adressen einstellbar
ZK03 ... Epromprogrammierkarte
ZK04 ... RS 232
ZK05 ... 2k zusätzliche RAM/Eprom

EK01 ... Meßwertaufnahme (Erweiterungskarte zu ZK02)
EK02 ... Laststeuerung (Erweiterungskarte zu ZK02)

Für später sind:
... Soundkarte mit 6581 Soundchip, ... Farbgrafikkarte 256 x 192, ... Digitalisierer, ... Plotter, ... Messen/Steuern/Regeln als Zusatzkarten geplant.
Auch die Karten aus der c't sind verwendbar. Sicher habt Ihr selbst auch Vorschläge für eine Zusatzkarte. Bitte meldet Euch dazu!!!

ECB-Bus ---- Allgemeines

Unser Ziel ist es, erst einmal für alle Interessierten den Grundbaustein aufzubauen. Damit haben schon einmal alle die Möglichkeit die Zusatzkarten zu nutzen (den Bus auszuprobieren). Wie oben schon angeführt benötigt man dazu noch die ECB-Bus-Adapterkarte und den schon vorhandenen Computer.
Anschliesend wird dann der ECB-Bus-Computer aufgebaut. Dies ist der 2. Bauabschnitt unserer Aktion. So kann sich jeder selbst entscheiden, wie weit er mitbauen möchte.
Wir bieten in jedem Fall beim Aufbau und der eventuellen Fehlersuche unsere Unterstützung an.
Desweiteren haben wir uns überlegt, Euch mehrere "Aktionsstufen" anzubieten. Dadurch hat jeder von Euch die Möglichkeit an dieses System zu kommen.
Wir bieten also an:

--- Kompletter Selbstaufbau in eigener Regie
--- Kompletter Selbstaufbau in einer Gruppe (zu zweit oder zu dritt) bzw. mit teilweiser Unterstützung.
--- nur mechanischer Aufbau. Die Platinen werden von uns mitaufgebaut und geprüft es bleibt nur noch der Zusammenbau.
--- übernahme eines fertigen Gerätes (Ausstattung nach Wunsch)

ECB-Bus ---- Der Bus

Das wichtigste an diesem System ist der Bus. Wir haben den ECB-Bus aus der c't übernommen und auf unsere Belange angepasst. Da schon einige Bus-Typen vorhanden sind haben wir mit Rücksicht auf die Kompatiblität folgende Belegung gewählt.

	a		b		c
+5V	1	+5V	1	+5V	1
D5	2	MRD -	2	D0	2
D6	3	MWR -	3	D7	3
D3	4	IN -	4	D2	4
D4	5	OUT -	5	A0	5
A2	6	reserviert CP/M	6	A3	6
A4	7	"	7	A1	7
A5	8	"	8	A8	8
A6	9	"	9	A7	9
WAIT -	10		10		10
BUSRQ -	11		11	IEI	11
A18	12		12	A19	12
+12V	13		13	-12V	13
	14		14	D1	14
-5V	15		15	-15V	15
2 PHI	16		16	IEO	16
A17	17		17	A11	17
A14	18		18	A10	18
+15V	19		19	A16	19
M1 -	20		20	NMI -	20
	21		21	INT -	21
	22		22	WR -	22
BAI	23		23	GDT	23
VCMOS	24		24	RD -	24
BAO	25		25	HALT -	25
	26		26	PWRCL -	26
IORQ -	27		27	A12	27
RFSH -	28		28	A15	28
A13	29		29	PHI	29
A9	30		30	MREQ -	30
BUSAK -	31		31	RESET -	31
GND	32	GND	32	GND	32

Die noch nicht bezeichneten Stifte sind frei bzw. aus Kompatiblitäts Gründen freigehalten.
Die Signale CAS und RAS fehlen wegen bevorzugtem CMOS-RAM Einsatz bzw. Refresch-Kontroler auf der RAM-Karte.

ECB-Bus ---- Kosten

Wie schon angeführt ist der erste Schritt, das Aufbauen des Grundbausteins, für alle gleich, egal ob Ihr Euch dann für die Bus-Erweiterung oder den Selbstbau-TRS-80 entscheidet.
Aus diesem Grund empfiehlt sich eine Sammelbestellung. Ein weiterer Vorteil der Sammelbestellung wäre der kostengünstigere Einkauf durch den Mengenrabatt, der bei entsprechenden Stückzahlen gewährt wird. Wir hoffen mindestens die magische Zahl 10 zu erreichen. Dazu sollten wir allerdings ein paar Leute mehr sein. Vielleicht bekommen wir noch weitere Interessierte für unser Projekt, wenn sich auch Nichtmitglieder daran beteiligen. Fragt bitte in Eurem Freundes- und Bekanntenkreis nach weiteren Mitbauern. Es lohnt sich für Euch auf jeden Fall, denn dadurch könnt Ihr Euren Geldbeutel schonen.
Nun endlich zu dem Preis unseres Projektes. Für die Grundeinheit haben wir einen Betrag von DM 250,-- als oberste Grenze errechnet. Für diesen Preis bekommt Ihr dann das 19 Zoll-Gehäuse, die Bus-Platine, 10 Stück 96-polige Federleisten und die Stromversorgung. Des weiteren haben wir für die vier Computerplatinen (SB1T80 - SB2T80) eine Preisvorstellung von ca. DM 550,--. Da die Halbleiterpreise ständigen Schwankungen unterlegen sind, möchten wir uns hier noch nicht festlegen. Wir haben aber die Hoffnung, daß der Preisverfall weiterhin anhält und wir unter die Preisvorstellung kommen werden. Wir werden den Computerteil sowieso kartenweise aufbauen, sodaß der Geldbetrag nicht aufeinmal fällig wird sondern nach und nach mit Vorstellung der einzelnen Karte in der INFO.
Noch eine Anmerkung zum Preis - liegen wir unter unserem Preisvorschlag, bekommt Ihr Euer Restgeld wieder ausbezahlt oder Ihr könnt es stehen lassen für die nächste Sammelbestellung. Natürlich erwarte ich im Gegenzug von Euch, daß Ihr mir bei eventueller überschreitung unserer Preisvorstellungen den dann noch fehlenden Betrag auf meinem Konto wieder gutschreibt. Ist ja Ehrensache. Wir werden aber die Preise so wählen, daß dies nicht der Fall sein wird.

ECB-Bus ---- Bestellung

Um am Kostengünstigsten zu arbeiten, bitten wir alle Interessenten sich mit Hilfe des im INFO beiliegenden Antwortbogens anzumelden.
Anmeldeschluß ist der 20. August 1986.
Bis zu diesem Termin möchte ich alle Anmeldungen zusammenhaben.
Anmeldungen sind nur gültig, wenn bis zum Anmeldeschluß auch der entsprechende Betrag auf meinem Postgirokonto eingegangen ist.
Ich werde dann die Sammelbestellung der Bauteile in die Wege leiten, sodaß bis zum nächsten INFO -dort erscheint dann die Aufbauanleitung- die Bauteile bei Euch ankommen.

Das Anmeldeformular findet Ihr am Ende des INFO's.

ECB-Bus ---- Kontaktpersonen

Zum Abschluß möchte ich Euch nun noch die Kontaktadressen bekanntgeben. Wenn Ihr noch weitere Informationen benötigt, meldet Euch bitte bei uns. Die vollständige Adresse findet Ihr in der Adressenliste.
Ulrich Böckling, Bernd Drohwälder, Eckehard Kuhn
und Jens Neueder.

Jens Neueder

ARD und ZDF im Spiegel der Zuschauer

Drei Buchstaben, die jeden Abend pünktlich um 20 Uhr Millionen Bundesbürger in die „gute Stube" flimmern, reizen offensichtlich ungemein die Phantasie der Betrachter: „Aktion Ruhendes Deutschland" oder „Alpträume Rühren Daher" deuteten Fernsehzuschauer das Kürzel ARD in einem Ideenwettbewerb, zu dem das Ulk-Magazin „mad" aufgerufen hatte. „Allgemeines Ratloses Durcheinander", „Anbieter Reizloser Dumpfkost", „Armutszeugnis Resignierter Dünnbrettbohrer" und „Alles Reichlich Daneben", hieß es da in den Zuschriften.

ARD: Anschalten, Reingucken, Draufhauen

Noch drastischer antworteten die Leser auf die Frage, was für ein Programm die ARD biete: „Abführmittel, Rezeptfrei Dargeboten" textete ein Teilnehmer der Aktion bissig. „Abfall, Ramsch, Dutzendware", „Altmüll, Richtig Durchgequirlt", „Alter, Ranziger Durchschnittskäse" oder „Achtung, Reine Dosenkost", höhnten andere Leser.

Die Frage, was ARD bei den Zuschauern bewirke, reizte eine Teilnehmerin zur Aufforderung: „Anschalten, Reingucken, Draufhauen!" Weniger gewalttätig lauteten andere Zuschriften: „Alles Rennt Davon", „Anschauen Reduziert Denkvermögen" oder „Alle Rufen Dallas!"

Aber auch den Kollegen vom ZDF sollte es nicht besser ergehen. Ihr Sender wurde mit nicht weniger kernigen Deutungen bedacht. Darunter waren „Zankapfel Der Familie", „Zerstörung Deines Feierabends", „Zwangsdroge Der Frührentner", „Zweites Deutsches Fiasko", „Zimmerfang Durch Flimmerzwang", „Zimmermanns Detektiv-Fortbildung" und „Zensur Durch Flachmänner". Bei dem Ideenwettbewerb gingen natürlich auch einige positiv klingende Deutungen ein. Die waren aber nicht so originell.

ANGEBOTE GENIE16

Genie 16 C statt 3495,-DM nur 3195,-DM
Genie 16 TC statt 4495,-DM nur 4095,-DM
Genie 16 XC statt 5995,-DM nur 5495,-DM
Genie 16 TXC statt 6495,-DM nur 5895,-DM

Info anfordern bei:

Peter Spieß
Offsetdruck + EDV-Zubehör
Trugenhofenerstr. 27
8859 Rennertshofen 1
Tel: 08434/454

Floppy-Laufwerk

EPSON SMD-100

zu verkaufen. 40 Tks., SS, DD, 3,5" Shugart-Bus, direkt an Controller Genie 1, 2, TRS80 M.1, M.3 ansteckbar; incl. 4 Disketten (NEWDOS auf Disk) und Manual. NAGELNEU !
Preis: einhundertsechzig Mark (VB)

256K RAM, gebanktes DOS

RAM-Speicher satt, alle DOS-Overlays im Speicher, kein Nachladen von Diskette mehr, dazu noch zusätzliche 128K RAM für Anwenderdaten

fertig bestückte Platine zum Selbsteinbau ohne 41256-RAMs mit Software zum Selbstkostenpreis von 50,-DM; unbestückte, ungebohrte Platine mit Software für 15,-DM

Bitte Diskette mit Lieblings-DOS und 3,-DM in Briefmarken für Rückporto beilegen.

Helmut Bernhardt, Hafenstr.7, 2305 Heikendorf

Wer kann mir weiterhelfen !

So richtig angefangen hat es mit meinen Schwierigkeiten, als ich versucht habe, einen groesseren File (34K) mit dem Editor des FORTRAN-Packetes von Tandy weiterzubearbeiten. Zuerst einmal fiel mir auf, dass der Editor von sich aus Page-Marken in den File eingebaut hatte. Grund: Da er mit der Zeilen-Nummer 100 beginnt und in 100ter-Schritten weiterzaehlt, war es zu einem Zeilennummernueberlauf gekommen. In diesem Fall beginnt er eine neue Seite, wiederrum mit 100 beginnend, bis zum naechsten Ueberlauf. So weit so gut, waere da nicht im Verlauf meines weiteren tuns geaendert, umnummeriert, geloescht und nach irgendwelchen Strings gesucht worden. Da jeder Abend auch einmal ein Ende hat, so kommt auch das Retten des eigenen Werkes. Zu guter letzt, mal sehen ob soweit alles stimmt, schnell noch eine Probeuebersetzung mit M80. Die Welt waere um einiges schoener, gaebe es Murphy nicht. Jede Menge Fehlermeldungen ueber fehlende Labeldefinitionen. Das aber konnte nicht sein, denn die waren nachweislich zuvor noch gesichtet worden. Also alles zurueck, E80 wieder aufrufen und den File nochmals geladen. Und tatsaechlich, die Label PEEKW, PEEKB, POKEW und POKEB - um diese handelte es sich naemlich - waren verschwunden, samt dem zugehoerigen Assemblertext !?!. Um es noch genauer zu sagen, eine Seite Text = 58 Zeilen fehlten.

Um dem Problem naeher zu kommen, habe ich dann folgendes getan: Den gesammten Speicher mittels CLEAR geloescht und danach E80 und den File erneut geladen. Dammit die Sache in Schwung kommt anschliessend neu durchnummeriert und ein wenig nach einem String suchen lassen. Zwischen jedem dieser Schritte erfolgte jeweils ueber DEBUG eine Speicherkontrolle (deswegen das CLEAR). Dabei ist mir dann aufgefallen, dass mit jeder Aktion E80 den freien RAM-Speicher weiter vollgeschrieben hat (???). Da der Stack am oberen Speicherende liegt,mussten sich Text und Stack dann unweigerlich einmal begegnen. Sofern man es nicht zu sehr uebertreibt wird scheinbar nur der Text bzw. die angelegten Vermerke ueber Aenderungen und der Gleichen veraendert. Man kann es auch so weit treiben, dass der Editor regelgerecht abstuerzt (so dann im weiteren geschehen). Warum der Indexfile nicht angelegt wird, laesst sich hiermit noch nicht erklaeren, denn dieser fehlt auch dann, wenn man den File sofort nach dem Laden wieder abspeichert. An dieser Stelle muss ich noch dazu sagen, dass dieser Fehler nicht generell auftritt, was die Sache nur Unverstaendlicher macht, leider.

Und nun meiner Frage an die Mitglieder : Wem ist dieser Fehler bekannt und kennt den Zap oder andere Massnahmen ?. In diesem Zusammenhang noch einen Zusatz. Ich habe ausserdem einen E85 (= E80), bei dem allerdings der verfuegbare Textspeicher kleiner ist (um ca. 5K). Da ich nur eine defekte und nicht lauffaehige Version besitze, konnte ich dem nicht weiter nachgehen. Eventuell ist bei dieser Variante der Fehler beseitigt. Kann hierzu ebenfalls Jemand etwas sagen ?.

Mein zweites Problem betrifft den M80-Assembler. Als ich versuchte, mit der INCLUDE-Funktion zu arbeiten weigerte er sich, diese zu kennen. Und dass, obwohl er das laut Handbuch muesste. Hiervon ebenfalls betroffen sind die Alias-Namen $INCLUDE und MACLIB. Hat Microsoft bei der Anpassung fuer das Model-I diese moeglicherweise heimlich unterschlagen ?. Es ist mir jedenfalls nicht gelungen, den M80 zu ueberreden. Moeglicherweise gibt es einen anderen Befehl, der das Gleiche bewerkstelligt. Wem ist hierzu etwas bekannt. Gibt es womoeglich auch Versionen, denen diese Befehle noch nicht abhanden gekommen sind oder gilt dies generell ?. Wenn dem so ist, waere das schade, aber nicht zu aendern.

Wer mich zu diesen Themen anrufen will, die Nummer steht ganz hinten im Info unter Mueller, K., ich bin aber erst nach 19.00h zu erreichen. Schriftlich ist's womoeglich billiger und man kann sich dann hinterher alles nochmals in Ruhe durchlesen (Schrecken oder Euphorie und so !).

Das duerfte ersteinmal genug sein. Bis zum naechsten Info habe ich vielleicht das naechste Packet an 'Dahlschlaegen' 'an Land' gezogen. Bis dahin Tschuess

Kurt Müller

Paß auf! Ich laß mir heute morgen von dir keine Frechheiten bieten.

Buchbesprechung

Immer wieder habe ich die traurige Erfahrung gemacht, daß der Preis eines Buches leider nur sehr wenig über seine Qualität aussagt. Dies ist vor allem auch und nicht zuletzt bei der Spezies der Computerbücher so! So stehen in meinem Bücherschrank eine ganze Menge sehr teurer aber wertloser Schinken herum und sicher geht es anderen Computerfreaks nicht viel anders! Informationsaustausch tut hier Not! Schließlich brauchen nicht mehrere Clubmitglieder mit teurem Geld die gleiche Erfahrung zu bezahlen, nämlich daß dieses oder jenes Buch nichts taugt. Aus diesem Grunde habe ich mir vorgenommen, in lockerer Folge einige neuere Publikationen (die alten hat ja sowieso fast jeder!) vorzustellen und meine Meinung zu ihnen zu äußern. Sehr freuen würde ich mich, wenn noch weitere Mitglieder sich zu Veröffentlichungen, egal ob positiv oder negativ, auslassen würden. Sicher kann so manche müde Mark gespart und/oder besser angelegt werden!

CP/M Bücher

Immer mehr unserer Mitlieder können auf ihren Maschinen (Model 1/3 mit entspr. Hardwareänderung und Model 4/4P sowieso) das Betriebssystem CP/M fahren. CP/M ist aber nicht zu vergleichen mit den sonstigen Diskettenbetriebssystemen, die es für den TRS-80 gibt und so tun sich die Benutzer von TRS-, NewDOS usw. relativ schwer damit. Natürlich gibt es auch über diesen Bereich der Computerei genügend Literatur, aber welches Buch soll man sich anschaffen?

CP/M kompakt (Plate, Franzis-Verlag)
Noch nicht mal 10,00 (in Worten "zehn"!) Mark kostet ein Kurzhandbuch für CP/M, das Büchlein "CP/M kompakt"! Trotzdem enthält es alle für den CP/M-Benutzer wichtigen Informationen. Angefangen bei den Control-Sequenzen (z.B.: CTRL P, CTRL S usw.) über die resistenten (DIR, USER, usw.) bis zu den transienten (PIP, STAT usw.) Befehlen ist hier alles fein säuberlich und sehr übersichtlich aufgelistet. Im zweiten Teil des nur 87 Seiten starken, im Taschenbuchformat gehaltenen Büchleins werden alle BIOS- und BDOS-Aufrufe mit genauer Angabe der zu übergebenden Parameter angegeben, sodaß auch Assembler-Programmierer von der Publikation profitieren können. Zu empfehlen ist "CP/M kompakt" allen CP/M-Usern, die sich einen schnellen Überblick über die verfügbaren CP/M-Befehle verschaffen wollen, ohne gleich einen ganzen Berg von Handbüchern durchzuwälzen. Zum gründlichen Einarbeiten in das "meistverbreitete Betriebssystem für Microcomputer" ist das Buch aber nicht geeignet!

Dazu gibt es andere Literatur. Z.B.:

Betriebssystem CP/M (Plate, Franzis-Verlag),
dieses Buch kostet immerhin schon fast 50,-- DM. Es behandelt CP/M von Grund auf (es werden erst einmal die Begriffe "Monitor" und "Betriebssystem" erklärt) und enthält auch Musterprogramme (BIOS, BDOS). Es wendet sich mehr an den Programmierer (Assembler) als an den Anwender, auch wenn im letzten Drittel des Buches die CP/M-Befehle kurz erklärt sind. Leider ist das Buch zu sehr auf den mc-CP/M- Computer zugeschnitten und bringt dem Tandy-Benutzer kaum etwas, ist also kaum empfehlenswert!

CP/M Handbuch (R. Zaks, SYBEX),
auch ein Buch in der Preislage eines halben Hunderters, ist da schon besser geeignet einen CP/M-Neuling in die Bedienung desselben einzuführen. Nach einer allgemeinen Einführung in die Grundlagen (von "was ist eine Floppy und wie wird sie in den Drive eingelegt" bis "Organisation von Dateien" ist da alles zu finden) werden die CP/M-Befehle ausführlich erklärt. Dieser Teil umfaßt ca. 60% des Buches, die verbleibenden Seiten bringen alle Befehle nochmals in einer Kurzbeschreibung, sodaß das Buch auch als Nachschlagewerk dienen kann! Im großen und ganzen ist das Buch sehr brauchbar und empfehlenswert, was bei diesem renommierten Autor und dem Verlag auch kaum anders zu erwarten war! Aber auch hier müssen Abstriche gemacht werden, da sich das Buch praktisch nur an den Anwender wendet, den Programmierer aber praktisch ganz vergißt.

CP/M - Anatomie eines Betriebssystems (J.D. Dennon, M u. T),
ist zwar die teuerste, dieses Buch kostet immerhin beinahe 70,-- DM, aber auch die beste der hier vorgestellen Publikationen über CP/M. Es ist praktisch als Lehrgang aufgebaut und führt den Leser systematisch in den Umgang mit CP/M. Dabei werden im ersten Teil nicht nur die Befehle angesprochen, sondern auch die Programmiersprachen M- und C-BASIC kurz gestreift. Danach wird ausführlich auf den Assembler eingegangen und dabei CP/M systematisch auseinandergenommen. Das geht so weit, daß am Schluß ein Programm zur Rettung gelöschter Dateien geschrieben werden kann. Das Buch ist sehr unterhaltsam und kurzweilig geschrieben und immer wieder mit vielen Beispielen und Fragen zur Selbstkontrolle gespickt. Leider fehlt dafür ein Zusammenfassungsteil, so daß man das Buch praktisch nicht als Nachschlagewerk benutzen kann. Ein "Einstieg in der Mitte" ist jedoch durchaus möglich, so daß der nicht an BASIC und sonstigem interessierte Assemblerprogrammierer ohne weiteres ein paar Kapitel auslassen kann.

Meiner Meinung nach ist eine Kombination der Bücher "CP/M kompakt" und "CP/M - Anatomie eines Betriebssystems" genau die richtige Mischung, sowohl für CP/M-Neulinge als auch für Leute, die tiefer ins System einsteigen wollen! Euer

Hartmut Ebermann

===> Unsere Programm-Diskothek <===

HALLO BOYS !

(Girls habe ich im Mitgliederverzeichnis nicht entdecken können. Sollte sich dennoch eins unter einem männlichen Pseudonym darin verstecken, so sei sie hiermit extra gegrüßt!)

Um 14 Programme wurde unsere Diskothek in diesem Monat bereichert! Den fleißigen Abtippern und Einsendern sei auch hier noch einmal herzlich gedankt! Und selbstverständlich sollen sie auch extra aufgeführt werden.
Es waren dies:
Hans-Martin Stephan (8 Programme)
Josef Konrad (4 Programme)
Klaus Hermann (1 Programmpaket)

Welche Programme das sind, seht Ihr in nachstehender Liste.
Ich habe deren Aufmachung wieder etwas verbessert. Wenn Ihr nichts einwendet, wird auch der nächste Programm-Katalog in dieser Form erscheinen (und, wie ich hoffe, bald!)
Allerdings dann nach Programm-Gattungen (Utilities, Spiele usw.) geordnet und innerhalb derselben alphabetisch nach Programm-Namen.
Etwas Schwierigkeiten bereitet mir immer die sog. "Benotung" der Programme. Es ist fast unmöglich, hierbei gerecht zu sein. Da ich der Gerechtigkeit sowieso nicht zum Siege verhelfen kann, werde ich ihre Truppen wohl in Zukunft in der Kaserne lassen und damit eine Pioniertat für die allg. Abrüstung tun. Sagt mir, ob Ihr damit einverstanden seid. (Schweigen gilt als Zustimmung'!)

Wenn Ihr gestattet, noch eine Anmerkung in eigener Sache: Meine Programme "LIESDIR" und "LIESKOMM", die ich im letzten Heft veröffentlichte, enthielten einige Bugs. Wer es bemerkt und mir trotzdem nicht mitgeteilt hat, den muß ich unangemessener Zurückhaltung "zeihen"!
Ich habe beide Programme inzwischen nicht nur korrigiert, sondern auch erweitert, so daß man File-Kommentare jetzt bequem speichern und auch wieder ändern kann. So habt Ihr hiermit ein Werkzeug zur Kommentierung eines Disketten-Inhaltes auf einem Blatt Papier, das man mit in die Diskettenhülle stecken kann, so daß man nicht erst eine "Datenbank" umständlich befragen muß. Ich mache es seit langem so und finde es sehr praktisch...

MIT "SUUM GEQUIEKE" (LAT.) UND "GREETINGS A LOT" (ENGL.)
VERBLEIB' ICH WIE IMMER "DER DISKO-KA-JOT" !

Neue Programme seit letztem INFO Nr.13 vom Mai 1986

CREF/CMD UTL ProgrNr 0335 Teile: 06 Note 0 Eing. 6 /7 /86
========= f.Comp Quelle:
Verfasser KONRAD, Josef eingetippt von K., J.
Cross-Reference-Programm fuer 64- sowie 80-Zeichen-Ausgabe mit ALCOR-FASCAL-Source: auch mit Ausblenden von PASCAL-Kommentaren Disk-Nr. 9

EXT/CMD UTL ProgrNr 0336 Teile: 02 Note 1 Eing. 6 /7 /86
======= f.Comp alle Quelle:
Verfasser KONRAD, Josef eingetippt von K., J.
zum Aendern der Extension von Files.
Aufruf: EXT,#,ALT,NEU (# = Laufwerk-Nr.)
Disk-Nr. 9

FORMDRUK/CMD UTL ProgrNr 0337 Teile: 02 Note 2 Eing. 6 /7 /86
============ f.Comp Quelle:
Verfasser KONRAD, Josef eingetippt von K., J.
Formatiertes Drucken von ASCII-Files (mit ALCOR-Source).
Disk-Nr. 9

PASFORMK/CMD UTL ProgrNr 0338 Teile: 04 Note 1 Eing. 6 /7 /86
============ f.Comp Quelle:
Verfasser KONRAD, Josef eingetippt von K., J.
Zur Umwandlung in Kleinbuchstaben (mit Source-Code).
Disk-Nr. 10

CASLABEL/BAS UTL ProgrNr 0339 Teile: 01 Note 1 Eing. 6 /16/86
============ f.Comp Quelle: 80 MICRO
Verfasser eingetippt von
Druckt Etiketten fuer Kassetten
(Einsender: Hans-Martin STEPHAN)
Disk-Nr. 10

FRAME/CMD UTL ProgrNr 0344 Teile: 05 Note 2 Eing. 6 /21/86
============ f.Comp Quelle: 80MICRO 9/83
Verfasser BOWKER & KaJot Mb eingetippt von KaJot Mueh-tein
BASIC-Erweiterung durch einen neuen Befehl, z.B. "FRAME".
Dieser zeichnet einen Rahmen (FRAME) um den Bildschirm.
Source-Code, DO-File u. zwei BASIC-Beispiele Disk-Nr. 10

SVC/BAS UTL ProgrNr 0320 Teile: 03 Note 3 Eing. 6 /23/86
============ f.Comp Model 4 Quelle: 80MICRO 4/86
Verfasser H.Brothers eingetippt von Klaus Hermann
Das Programm ruft SVC (SuperVisorCall) aus einem BASIC-Program auf. Hierzu liegt ein DEMO-Programm vor.
Informationen hierzu s.INFO Juli 86 (& File) Disk-Nr. 10

biorhyth/bas SPL ProgrNr 0340 Teile: 01 Note 1 Eing. 6 /16/86
========== f.Comp Quelle: SIEHE UNTEN
Verfasser Stuebs eingetippt von Hans-M.Stephan
Die ziemlich bekannte (Pseudo-)Weissagung mit "Stimmungskurven" fuer Geist, Gemuet und Koerper...
Aus: Anwenderprogr.f.TRS80 u.GENIE.-Hofacker Disk-Nr. 6

HALLO/BAS SPL ProgrNr 0341 Teile: 01 Note 1 Eing. 6 /16/86
======== f.Comp Quelle: S.UNTEN
Verfasser eingetippt von Hans-M. Stephan
Der Computer stellt dumme Fragen und gibt seine Kommentare zu den schlauen Antworten...
(aus: BASIC-Computer-Spiele, Bd.1 (SYBEX) Disk-Nr. 10

REDBARON/BAS SPL ProgrNr 0345 Teile: 01 Note 2 Eing. 6 /16/86
========== f.Comp Quelle: S.UNTEN
Verfasser STUEBS eingetippt von Hans-M.Stephan
Bei diesem Spiel muss man aufpassen, dass dem Kameraden neben einem nichts passiert.
Aus: Anwender-Programme f.TRS80 und GENIE Disk-Nr. 10

MONDLNDG/BAS SPL ProgrNr 0346 Teile: 01 Note 1 Eing. 6 /16/86
========== f.Comp Quelle: MC 8/83
Verfasser eingetippt von Hans-M.Stephan
Man muss den Treibstoff-Vorrat richtig einteilen, um bei der Landung nicht zu zerschellen!
Disk-Nr. 10

AUTOKOST/BAS KFM ProgrNr 0347 Teile: 01 Note 3 Eing. 6 /16/86
========== f.Comp Quelle: C'T 3/84
Verfasser eingetippt von Hans-M.Stephan
Laufende detaillierte Erfassung und Ueberwachung der Kfz-Kosten (mit Datenbank, versteht sich).
Disk-Nr. 10

ENTSCHDG/BAS MTH ProgrNr 0343 Teile: 01 Note 4 Eing. 6 /16/86
========== f.Comp Quelle: SIEHE UNTEN
Verfasser STUEBS eingetippt von Hans-M.Stephan
Ein gutes Optimierungs-Programm im Frage-und-Antwort-Stil, so dass man nicht merkt, dass Mathe (LIN.OPTIMIERUNG) dahinter steckt.-Aus "Anw.Progr.f.TRS80 u.GENIE" Disk-Nr. 10

BOOKINV/BAS DAT ProgrNr 0342 Teile: 01 Note 3 Eing. 6 /16/86
========== f.Comp Quelle: SIEHE UNTEN
Verfasser eingetippt von
Der Name sagt es: Buecherverzeichnis.
Aus: 80MICRO Magazine, Nov.1982; eingesandt von H.M.Stephan
Disk-Nr. 10

- I M P R E S S U M -

1. Vorsitzende: Peter STEVENS, Postfach 56, 4600 Dortmund 1, ☎ 0231 /593883

2. Vorsitzende: Hartmut OBERMANN, Schwalbacher Straße 6, 6209 Heidenrod 1, ☎ 06124 /3913

Hardwarekoordinator: Eckehard KUHN, Im Dorf 14, 7443 Frickenhausen 1, ☎ 07022 /45417

Diskothekar: Klaus-Jürgen MÜHLENBEIN, Am Mönchgarten 28, 6940 Weinheim -Lü., ☎ 06201 /55052

Redaktion: Jens NEUEDER, Panoramastraße 21, 7178 Michelbach /Bilz, ☎ 0791 /42877

Redakteure dieser Ausgabe: Helmut Bernhardt, Klaus-J. Mühlenbein, Jens Neueder, Gerald Schröder, Günther Wagner, Klaus Hermann, Kurt Müller, Hartmut Obermann, Arnulf Sopp

Bankverbindung des CLUB 80
Postgirokonto: Peter STEVENS, Sonderkonto CLUB 80
Konto-Nummer: 285 491 - 465
Postgiroamt: Dortmund
BLZ: 440 100 46

Das INFO erscheint zweimonatlich.

Es erfolgt keine Zensur oder Kontrolle der jeweiligen eingeschickten Infobeiträge durch die Redaktion.

HEFT 14 Juli 1986

Schluß

Hallo Club-80er,

wieder ist ein neues INFO fertiggestellt. Ganz besonders freut es mich, daß wir in diesem INFO keine Zeitungsartikel haben, und trotzdem unser bisher dickstes INFO ausgeben. Hierfür meinen Dank an unsere fleißigen Autoren.

Auch möchte ich mich für die guten Kritiken bedanken, die ich für meine Redaktionsarbeit bekam. Wie ich mit einem kurzen Blick in die Fragebögen feststellte. leider war es mir nicht möglich, eine ausreichende Auswertung vorzunehmen, da die Mehrzahl der Fragebögen noch fehlt. Ich hoffe, daß sie bis zur nächsten INFO noch eintrudeln. Bemängelt wurde die unleserliche Kontaktadresse. Dies werde ich mit Erscheinen der nächsten INFO abgestellt haben. Wenn das mit der e.V. -Eintragung klappt, möchte ich sowieso ein neues Deckblatt entwerfen.
Auch die manchmal schlechten listingabdrucke wurden wieder angeführt. Dies liegt aber an den schlechten Vorlagen, die ich von Euch bekommen habe. Ich verweise nochmals auf die "redaktionellen Richtlinien".

Da wir nach Möglichkeit wieder ein SONDERHEFT für Euch herausbringen wollen, möchte ich Euch um Eure Mitarbeit bitten. Gesucht werden Artikel zu folgenden Themen:
Modem (Hard und Soft), Grafik M4/4p, ROM
Sicher habt Ihr schon einiges dazu in Euren "Archiven".

Des weiteren ist Mithilfe bei den Kurzbefehlslisten für BASIC und DOS erwünscht. In der Regel sind es Übersetzungen schon vorhandener Kurzanleitungen, sowie das Schreiben eines "Erfahrungsberichtes" über einen Befehl. Wie zum Beispiel in diesem INFO auf Seite 6 dargestellt. Bitte meldet Euch zur Koordination der Arbeit bei der Redaktion.

Ich wünsche Euch viel Freude an der neuesten INFO.

Bis zum nächsten INFO verabschiedet sich

Euer

J. Neuder

Der Computer der Partnervermittlung hat Sie aber etwas – nun ja – anders beschrieben . . .

CLUB 80 - ECB - BUS - SYSTEM

ANMELDUNG

Wie schon in der Systembeschreibung angeführt habt Ihr nun hiermit die Möglichkeit Euch an diesem Club-Projekt zu beteiligen. Diejenigen, die mitbauen wollen bitte ich, die Anmeldung auszufüllen. Auch der "Nichttechniker" findet durch unsere "Aktionsstufen" die Möglichkeit, kostengünstig an einen ECB-Bus-Rechner zu gelangen bzw. seinen Rechner um einen ECB-Bus zu erweitern.

Allgemeine Information an die Organisatoren

Um Sammelbestellungen zu koordinieren brauchen wir von den Interessierten natürlich einige Informationen. Auch wollen wir wissen, für welche Geräteversion Ihr Euch entscheidet. Wem im Moment nur die ECB-Bus-Erweiterung anspricht, hat natürlich auch später noch die Möglichkeit zum Computer aufzurüsten. Die entsprechenden Unterlagen werden ja im INFO veröffentlicht. Sicher fährt man aber günstiger, wenn man gleich an den geplanten Sammelbestellungen teil nimmt.

Ich interessiere mich für:

____ ECB-Bus ---- Bus Erweiterung
____ ECB-Bus ---- Selbstbau TRS 80

Ich wähle folgende "Aktionsstufe":

____ --- Kompletter Selbstaufbau in eigener Regie
____ --- Kompletter Selbstaufbau in einer Gruppe (2-er oder 3-er) bzw. mit teilweiser Unterstützung.
____ --- nur mechanischer Zusammenbau der fertigen Baugruppen. Die Platinen werden von uns mitaufgebaut und geprüft.
____ --- übernahme eines fertigen Gerätes (mit Wunschausstattung)
____ --- Ich bin nur theoretisch an dem Projekt interessiert und würde gegebenenfalls bei Problemlösungen helfen.

Vorschläge zu den Zusatzplatinen

Ich würde mich für folgende weitere, außer den schon vorhandenen oder geplanten, Zusatzplatinen interessieren:

_____
_____
_____
_____

ECB-Bus ----- Bestellung

Ich bestelle _____ Grundbausteine für je DM 250,--
und habe DM _____ auf angeführtem PostscheckKonto überwiesen.

.....
Ort Datum Unterschrift

Achtung:

Die Bestellung hat nur Gültigkeit, wenn bis zum 20. August 1986 der entsprechende Betrag Eurer Bestellung auf dem

PostgiroKonto	17 63 33 -700
beim Postgiroamt	Stuttgart
BLZ	600 100 70
Kontoinhaber	Jens NEUEDER
	Panoramastr. 21, 7178 Michelbach/Bilz

unter der Bezeichnung Grundbaustein eingegangen ist !!

Ich werde dann die Sammelbestellung in die Wege leiten.

Die Ausschreibung erfolgt im Genie/TRS80 User-Club Bremerhaven und im Club-80 gleichzeitig.

Aufgabenstellung:

Zu erstellen ist ein BASIC-Programm mit maximal 33 Zeilen. Die Thematik ist freigestellt. Das Programm muß auf den Rechnern Genie I, II, IIs, (III), oder TRS80 Mod. 1, Mod. 3 lauffähig sein und sollte unter der Herrschaft von NEWDOS80 oder G-DOS stehen. Je mehr "Spezialitäten" eingebaut werden, umso höher fällt die Bewertung aus (z.B. Diskettenroutinen, ansprechen der HRG, Druckerausgabe, Maschinenroutinen, etc.). Das Programm muß natürlich mit einer entsprechenden Betriebsanleitung ausführlich dokumentiert werden. Abtippen aus Zeitschriften (auch aus Clubinfos) oder einsenden von kommerziellen Programmen ist verboten. Im Namen aller Teilnehmer bitte ich um Fairness !!!

Zu gewinnen gibt's:

1. Preis: elektronische Schreibmaschine BROTHER EP-20

2. Preis: Ein 10er Pack Disketten

3. Preis: Buch "Start in die Computergrafik"

4.-10. Preis: je eine Leerdiskette

Die Preise wurden gestiftet von der Firma:

Peter Spieß
Offsetdruck + EDV-Zubehör
Trugenhofenerstr. 27
8859 Rennertshofen 1
Tel: 08434/454

Einsendeschluß ist der 30.09.86
(Datum des Poststempels)

Der Rechtsweg ist ausgeschlossen ! Die Gewinner werden schriftlich benachrichtigt. Die Bekanntgabe erfolgt dann im jeweiligen Clubinfo. Bitte die Diskettenformatierung angeben ! Die Disk's werden nach der Auswertung an den Einsender zurückgeschickt. Einsendeadresse:

Genie/TRS80
User-Club
Peter Spieß
Trugenhofenerstr. 27
8859 Rennertshofen 1