

Club SO - Info Nr. 6

Hallo liebe Mitglieder

Ihr werdet zu Recht erstaunt sein über die neue Aufmachung unseres Clubinfos.

Warum habe ich das gemacht? Ganz einfach - ich habe endlich die Vorschläge einiger Clubmitglieder aufgegriffen. Ich glaube, daß durch die Querrichtung und die neue Heftung das Info besser zu lesen ist. Ferner gibt es ab sofort eine durchgehende Seitennummerierung und ein Inhaltsverzeichnis. Ich hoffe, daß Ihr nun mit dem Info noch besser zurecht kommt. Ihr könnt mir ja mal Euer Meinung dazu mitteilen.

Ein ganz dickes Lob verbunden mit einem großen Dankeschön möchte ich loswerden an die Mitglieder, die Beiträge zum Clubinfo liefern oder sonst im Club konstruktiv mitarbeiten. Nur durch diese Mitglieder ist das Info in dieser Aufmachung überhaupt erst möglich.

Ich möchte deshalb auf dieser 1. Seite auch weiterhin um rege Mitarbeit seitens der Mitglieder bitten. Klar ist, daß viele Mitglieder keine Beiträge liefern können - aber die, die was beisteuern können, sollten dies bitte auch tun.

Was dieses Info bringt steht im Inhaltsverzeichnis. Ich möchte Euch nun noch über die aktuelle Mitgliederzahl informieren. Wir haben 41 Mitglieder (ist doch toll!) + 3 Mitglieder mit einem großen Fragezeichen (von Ihnen wurde der Jahresbeitrag 85 noch nicht bezahlt - Austrittserklärung liegt auch nicht vor!). Was wir mit den 3 letztgenannten 'Mitgliedern' machen (alle mindestens zweimal angemahnt) entscheiden wir auf dem Clubtreffen).

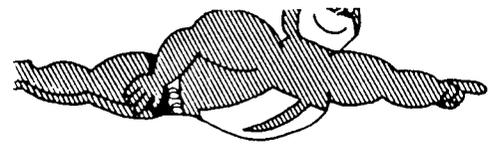
Genug für das Vorwort - viele Grüße

E u e r

Günther Wagner

INHALTSVERZEICHNIS:

Vorwort.....	Seite 1
Inhaltsverzeichnis.....	Seite 2
Rücktritt von Günther Wagner.....	Seite 3
Aktuelle Mitgliederliste.....	Seite 4
Clubtreffen.....	Seite 5-6
Verkausangebote, Systemänderung Hartmut Obermann.....	Seite 6
Gesucht, Treffen in München.....	Seite 7
Fragen.....	Seite 8
Die neuen Club-Mitglieder.....	Seite 9-10
Persönliche Vorstellungen.....	Seite 11-12
Cobol-Kurs (2. Teil).....	Seite 13-17
Funktionsweise des Druckerpuffers.....	Seite 18
Datenfernübertragung (Akkustikkoppler selbst bauen?)...Seite	19-22
Angebot Tandy-Akkustikkoppler.....	Seite 22
Monitor im Genie.....	Seite 23
Ausgabeumschaltung in Basic (PRINT - LPRINT).....	Seite 24
Erweiterung für TASMOS (Ausführung von NEWDOS-Kommand.)..Seite	25-26
Kurzbeschreibung des Komtek 1.....	Seite 27
Die Geschichte vom armen Hacker (aus alter Chronik).....	Seite 28
Joystick-Anschluß (2. Möglichkeit).....	Seite 29-30
LPRINT alles (Ausgabe aller Drucker-Codes).....	Seite 31-32
Der geknackte TRS-80 (int. Aufbau des Basic-Interpreter)..Seite	33-36
Adventure-Ecke: Abenteuer (aus HC Mai 84).....	Seite 37-38
Genie-Minitext (ein Text-Programm).....	Seite 39-40
String-Anzeige mit Volldampf.....	Seite 41-44
Bücher, POKE & PEEK.....	Seite 45
Basic Faster & Better (3. Teil mit Fehlerberichtigung)...Seite	46-60
Programmbibliothek abschaffen (ein Vorschlag).....	Seite 61
Anmerkungen zum o.g. Vorschlag.....	Seite 62
Tape Transfer (LMOFFSET-Erweiterung für Model III).....	Seite 63-64
Tricks mit Strings.....	Seite 65-66
Schnelleres Arbeiten mit EDTASM.....	Seite 67
Hardware-Projekt (Bus-System).....	Seite 67
Konvertierung TRS-80-Level-2-Basic nach MBasic.....	Seite 68
Schlusswort.....	Seite 68



Achtung:

Bereits im letzten Info habe ich angedeutet, daß ich ev. mein 'Amt' abgeben möchte.

Nun - ich habe mich endgültig entschlossen, daß ich die Leitung des CLUB 80 nicht weiter ausüben kann. Zwar hat mich die Abgabe der Clubbibliothek zeitlich sehr erleichtert, aber dennoch ist die Arbeit für mich noch zu viel. Ab September habe ich keine Zeit mehr. Für alle nochmals zur Erinnerung: Ich bin Student und komme ab September in die entscheidende Phase meines Studiums, das 7. und 8. Semester mit den Abschlußprüfungen und einer sehr zeitintensiven Diplom-Arbeit. Da bleibt einfach nicht genügend Freizeit für den Club. Die berufliche (und auch private) Zukunft muss einfach auch mal Vorrang haben.

Daher mein fester Entschluß, mein Amt abzugeben. Das heißt, ich suche Jemanden (oder auch mehrere), die meinen Tätigkeitsbereich übernehmen. Dies wären Beantwortung der Anfragen von Interessenten; Erledigung des anfallenden Briefverkehrs; Erstellung und Versand des Clubinfos und zuletzt die Kassenführung. Wie gesagt - Übernahme durch eine oder auch mehrere Personen.

Der Club hat über 40 Mitglieder und bestimmt findet sich Jemand, der weitermacht. Wer Interesse hat, der soll sich bei mir melden. Ich hoffe, daß ich nach dem Clubtreffen meinem Nachfolger (meinen Nachfolgern) alles übergeben kann (vielleicht nimmt mein Nachfolger ja auch am Clubtreffen teil). Ob der (die) Nachfolger von den Mitgliedern offiziell gewählt wird (werden), wird auf dem Clubtreffen besprochen. Sollte sich wider Erwarten niemand bereit erklären, meinen Tätigkeitsbereich zu übernehmen, müßte der Club aufgelöst werden. Ich hoffe (und bitte alle Mitglieder), daß es so weit nicht kommt.

Ich bitte alle Mitglieder um Verständnis für meine Entscheidung, die mir wirklich schwer gefallen ist, da ich nicht wegen Amtsmüdigkeit aufhöre, sondern wirklich aus oben genannten Gründen.

Euer Günther

Übersetzungen eines Blechidioten

Fremde Sprachen sind Ihnen im wahrsten Sinne des Wortes noch fremd, jenen Intelligenzmaschinen. Das menschliche »Mini-Hirn« wird hier noch gebraucht - welch tröstliche Gewißheit!

Text englisch (Matthäus 26, 41):

»Der Geist ist willig,
aber das Fleisch ist schwach.«

Übersetzung des Computers Ins Deutsche:
»DER ALKOHOL IST BRAUCHBAR,
DER BRATEN IST SCHLECHT.«

Text lateinisch (mors certa hora incerta):

»Der Tod ist bestimmt,
aber die Stunde ist ungewiß.«

Übersetzung des Computers Ins Deutsche:

»TODSICHER IST DIE UHRZEIT UNGENAU.«



- Alber Herbert, Al Innenstr. 20, 7732 Niedereschach, 07721/7102
 Baldes Hans, Johann-Strauss-Str. 6, 8025 Unterhaching, 089/6115179
 Boecker Dieter, Lehmweg 4, 2930 Varrel 1, 04451/7640
 Boeckling Ulrich, Am Sonnenhang 11, 5414 Vallendar, 0261/69522
 Bozek Hans Juergen, Gut Fachenfelde, 2093 Stelle, -
 Buskowiak Thomas, Eschersheimer Landstr. 257, 6000 Frankfurt 1, 069/5601621
 Dreyer Gerald, Am Speiergarten 8, 6200 Wiesbaden-Bierstadt, 06121/508218
 Gradjewski Werner, Zedernweg 29, 4220 Dinslaken, 02134/54573
 Hailup Matthias, Jungesellenstr. 15, 4600 Dortmund, -
 Held Manfred, Stinnerstr. 22, 8835 Pleinfeld, 09144/6563
 Hermann Klaus, Gartenstr. 22, 7401 Pliezhausen, 07127/70024
 Hummel Anton, Schubertstr. 2, 7612 Haslach, 07832/8289
 Jablotschkin Rainer, Thiekamp 29, 4780 Lippstadt B, -
 Kasper Dieter, Zeppelinstr. 9, 8952 Marktoberdorf, 08342/1630
 Koenig Hans J., Heibelstr. 25, 2080 Pinneberg, 04101/209444
 Konrad Josef, Anzenruber 35, 8038 Groebenzell, 08142/8494
 Kuhn Eckehard, Im Dorf 14, 7443 Frickenhausen 1, 07022/45417
 Marx Andreas, Mecklenburgring 48, 6600 Saarbruecken, 0681/812983
 May Holger, Marienstr. 9, 5768 Sundern 2, 02935/1668
 Neueder Jens, Panoramastr. 21, 7178 Michelbach/Bilz, 0791/42877
 Neukam Dieter, Garstedter Weg 18, 2087 Boenningstedt, 040/5566986
 Obermann Hartmut, Schwalbacher Str. 6, 6209 Heidenrod/Kemel, 06124/3913
 Perschbach Patrick, Waldstr. 52, 5000 Koeln 91, 872118
 Piller Walter, Rohrenstr. 8, CH-8835 Feusisberg, 01/7847418
 Preuss Lothar, Lautshof 13, 2940 Wilhelmshaven, 04421/84247 (dienstl. 804-1)
 Rank Heinrich, Fruehlingstr. 2, 8080 Fuerstenfeldbruck, 08141/3791
 Schmidt Peter, Drei-Aehrenstr. 18, 7800 Freiburg, \Rightarrow Beitrag 85 noch nicht bezahlt
 Schrewe Christian, Fliederweg 32, 4000 Duesseldorf 31, 0203/740897
 Schroeder Gerald, Am Schuetzenplatz 14, 2105 Seevetal 1, 04105/2602
 Sickmann Bernhard, Fleigenweg 2, 4430 Steinfurt 2, 02552/60344
 Smerling Frank, Tangstedter Str. 5, 2080 Pinneberg, 04101/207284
 Sopp Arnulf, Wakenitzstr. 8, 2400 Luebeck 1, 0451-791926
 Stephan Hans-Martin, Am Glasesch 9a (Postf. 1207), 4506 Hagen a.TW., 05401/99585
 Stevens Peter, Postfach 6327, 7800 Freiburg, 0761/35384
 Trapp Harald, Kranichstr. 46, 4270 Dorsten 1, -
 Troesch Eberhard, Altenessener Str. 414, 4300 Essen 12, 0201/342324
 Trost Robert, Aalscholverstraat 9, 6921 WR Duiven (Holland), 08367/4736 \Rightarrow Beitrag 85 fehl.
 Voigtlaender Holm, Haselnussweg 30, 6940 Weinheim, 06201/65241
 Wagner Alexander, Theresienstr. 21c, 8224 Chieming, 08664/1500
 Wagner Guenther, Gartenstr. 4, 8201 Neubuern, 08035/3361
 Wies Jean-Claude, Harthweg 9, 6600 Saarbruecken, 0681/582513
 Wright Donald, Apothekenweg 5a, 8056 Neufahrn, \Rightarrow
 Wucherer Juergen, Brauneeggerstr. 14, 7750 Konstanz, 07531/29145
 Zwickel Walter, Lengfelden 123, A - 5101 Bergheim, 0043662/51130

CLUBTREFFEN

Noch einmal viele Zeilen zum Clubtreffen (ergänzend zum 4. und 5. Info).

Es findet statt vom 23.-24. März 1984 im Hotel "Darmstädter Hof", Fam. Bauer, Ludwigstraße 1, 6107 Reinheim (Odenwald).

Reinheim liegt ca. 15 km (Luftlinie) südöstlich von Darmstadt am Kreuzungspunkt der B38 und B426. In der Nähe sind die Autobahnen A680 und A5.

Folgende Mitglieder nehmen meines Wissens teil:

Josef Konrad	Klaus Hermann	Heinrich Rank
Holm Voigtländer	Günther Wagner	Dieter Kasper
Gerald Schröder	Ulrich Böckling	Martin Held
Hans König	Peter Stevens	Jürgen Wucherer
Christian Schrewe	Gerald Dreyer	Hartmut Obermann
Jens Neueder	Thomas Buskowiak	Annulf Sopp
Alexander Wagner	Walter Zwickel mit Frau	
Frank Smerling	Walter Piller mit Frau	
	Wolfgang Beckhausen mit Frau (neues Mitg.)	

Es sind also derzeit 23 Mitglieder + 3 Frauen - es freut mich riesig, daß so viele Mitglieder teils mehrstündige Anfahrtszeiten auf sich nehmen (manche würden noch gerne kommen, können aber aus Termingründen nicht).

Wenn sich noch wer anmelden möchte, so soll er sich bei mir (Günther) melden. Ich kläre dann mit dem Jürgen Wucherer ab, ob noch ein Zimmer frei ist. An dieser Stelle möchte ich dem Jürgen danken, der mir sehr viel Arbeit abgenommen hat. Er hat das Hotel organisiert und auch sämtlichen schriftlichen Kram wie Anmeldebestätigung etc. erledigt.

Am Samstag werden wir so gegen 17 Uhr mit dem 'offiziellen' Teil des Treffens beginnen. Die Zeit davor dient zum Kennenlernen, zum Fachsimpeln, zum Programm-Tausch etc. Es gibt ziemlich viel zu besprechen. Bitte bereitet Euch darauf vor, daß ziemlich viel geredet wird, aber ich glaube, es läßt sich nicht vermeiden. Der offizielle Teil endet am Sonntag so gegen 11 Uhr, da ja die meisten nach dem Mittagessen die meist längere Heimfahrt antreten werden.

Nicht versäumen möchte ich, nochmals einige Punkte anzuschneiden, die sicherlich auf dem Treffen besprochen werden:

- Programmbibliothek (weiter wie geplant? Punkte abschaffen? ev. weitere Ideen hierzu)
- Hardware-Projekt(e) siehe auch dieses Info
- Änderungen an der Clubsatzung wie:
Ausschliessen von Mitgliedern; Thema Programmbibliothek; Ergänzungen zu (6); Fragen der Vorstandschaft bzw. der weiteren Clubleitung; feste Einrichtung einer Jährl. Mitgliederversammlung; usw.
- 80-Micro
- und vieles mehr

Alle Mitglieder, die am Treffen nicht teilnehmen, können sog. Eingaben machen, das heißt, daß Sie Anträge, Vorschläge etc. bei mir schriftlich einreichen können. Diese werden dann auf dem Clubtreffen behandelt.

Zur Beschlussfähigkeit des Clubtreffens:

Dieser Punkt ist in der Clubsatzung (noch) nicht geregelt. Damit das Clubtreffen einen echten Sinn hat, muß es voll beschlußfähig sein. Damit alle anderen Mitglieder über das Clubtreffen und über das, was besprochen wird, auch informiert sind, wird auf dem Treffen ausführlich Protokoll geführt. Dieses Protokoll wird im 7. Info veröffentlicht. Lediglich Entscheidungen, die eine größere finanzielle Belastung von Mitgliedern nach sich ziehen, werden wie gewohnt über das nächste Clubtreffen beantragt und beschlossen. (Mitglieder, die am Treffen nicht teilnehmen, können schriftliche Einwände gegen die von mir oben vorgeschlagene Beschlußfähigkeit vorbringen. Liegen mehrere Einwände vor, so wird die Beschlußfähigkeit des Clubtreffens ganz aufgehoben und alles über das 7. Info beschlossen!).

So - ich hoffe, der letzte Absatz war im Sinne aller Mitglieder. Ich bitte alle Mitglieder um geistige Vorarbeit und um Ausarbeitung von Vorschlägen, damit wir auf dem Treffen zügig alle Punkte abarbeiten können. Alles klar?

Unser Clubmitglied Mister Cobol (Harald Trapp) würde sich freuen, wenn er folgende Geräte verkaufen könnte:

- 1 Laufwerk 40 Spuren/ ss/ dd
Typ BASF 6106 ca. 1 Jahr alt
Preis: 150 DM
- 1 Magnetkartenleser Fabrikat Burroughs
incl. 100 Karten a 10,4 (Neupreis 1200 DM)
Preis: 80 DM (originalverpackt)

Ein Angebot von einer Privatperson habe ich bekommen. Herr Dr. Peter Hampel, Ringstr. 10, 8057 Eching möchte gerne folgende Geräte verkaufen:

- Model 100 Bk mit Cassettenlaufwerk (originalverpackt) 885 DM
- Model 4P 64k mit 2 Laufwerken + Programme (neuwertig) 3885 DM
- Model 2000 128k 2 Laufwerke, Colour Monitor B/W 9000 DM
Monitor, Graphicsboard + Programme (neuwertig)
- 4-Farben-Tintenstrahldrucker CGP-220 (orig.verpackt) 1300 DM

Systemänderung

Seit kurzem besitze ich ein neues Diskettenlaufwerk und einen Doubler. Das Laufwerk ist zwar für Double Sided ausgelegt, arbeitet zur Zeit jedoch nur Single Sided. Das bedeutet, daß ich ab sofort auch SS/DD Diskettenformate (40 Tracks) lesen und schreiben kann.

Weiterhin war meine Information im letzten Info scheinbar missverständlich. Ich kann selbstverständlich auch Datendisketten und nicht nur Systemdisk's lesen.

Karlmut Obermann

WANTED * GEBUCHT * WANTED * GESUCHT * WANTED * GESUCHT * WANTED

A N Ich suche Jemanden, der mit mir eventuell ein Laufwerk tauschen will. Ich besitze ein TEC FB5 (11 Monate alt), sowie ein TEAC FD55A (neu). Beim Kauf des TEAC-drives hat man mir zugesagt, das wäre baugleich mit meinem TEC. Doch das stimmt nicht. Es sind zwar beide Drives SS/SD und auch slimline, aber äußerlich und in der Anordnung der Stecker sind sie verschieden. Nun möchte ich gerne 2 gleiche Laufwerke haben (welche ist mir egal). Beide Laufwerke funktionieren einwandfrei. (Walter Zwickel)

U C Ich suche auch eine Kopie des Manuals zum TEAC FD55A!

H T Nächste Bitte: Vielleicht kann mir einer der Besitzer der HRG1B mal kurzzeitig die Unterlagen überlassen, damit ich klären kann, ob auch am KOMTEK eine Anschlussmöglichkeit dafür besteht. (Walter Zwickel)

W A N T Wir wären interessiert an Bauanleitungen oder billigen Bausätzen ('saubere' Bezugsquellen) für ein Modem. (Andreas Marx und Jean Claude Wies)

E D Ich suche Bauanleitungen für Interfaces für Schreibmaschinen. (Andreas Marx)

* D R I N G E N D gesucht wird ein Nachfolger, der die Geschäfte von Günther Wagner übernimmt.

* GESUCHT * WANTED * GESUCHT * WANTED * GESUCHT * WANTED * GESUCHT

H I N W E I S =====

Wenn ich irgendetwas vergessen habe,
wenn irgendwer noch irgendetwas bekommen müßte,
wenn ...

dann entschuldigt das bitte und seid so nett, daß Ihr mich kurz benachrichtigt. Günther Wagner

T R E F F E N I N M Ü N C H E N =====

In München treffen sich relativ regelmäßig der Josef Konrad, der Dieter Kasner und der Günther Wagner. Wer an einer Teilnahme interessiert ist, sollte sich bitte bei Günther Wagner melden.

Geplant ist für den Herbst ein Treffen in München, zu dem nach Möglichkeit mehrere Mitglieder aus dem Münchner Einzugsbereich kommen sollten/könnten. Dies wäre dann ein 'kleines' Clubtreffen wie es auch in anderen Gebieten aufgebaut werden könnte/sollte. Überlegt Euch diesen Punkt doch bitte mal!

FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ?

R A Wer hat ein Super-Sripsit, daß unter NEWDOS oder LDOS auf dem Model III läuft ? (Günther Wagner)

G E N Wer kann mir sagen, wie ich auf meinem Model III (dt. Ausführung) unter NEWDOS die Umlaute ausdrucken kann? (Günther Wagner)

? F Wer kann mir die Anweisung zu dem Programm 'TEBAST', veröffentlicht in c't, zur Verfügung stellen ? (Hans-Martin Stephan)

R A G E Wer hat Erfahrung mit Schreibmaschinen als Drucker (ev. Probleme mit Textsystemen?) (Andreas Marx)

N ? Zum 3. Info Peek & Poke eine Frage: 100 IF PEEK(14312)>=127 THEN PRINT "DRUCKER ANSCHALTEN": GOTO 100

F R A G E N Gilt dies nur für den TRSB0 III/Epson RX-B0 ? Bei meiner Ausrüstung (Genie I und Star DP 510) klappt das nicht. Wie lautet die Zeile bei meinem Computer? (Werner Grajewski)

N Wer hat die Auflösungen zu den Adventures:
? 1) Haunted House
? 2) Pyramid 2000
? 3) Peake Tu (Patrick Perschbach)

F R A G E N Haben sich Clubmitglieder bereits mit 'Super-Tape' (siehe auch c't 1985 Heft 1) befaßt? Liegt ein Super-Tape für den Genie I vor ? (Werner Grajewski)

N Wer hat Erfahrung mit Eprom-Bänken im Genie oder TRSB0 ? (Harald Trapp)

? F Wer kann gebrauchtes oder neues B"-Laufwerk single oder double side besorgen? (Harald Trapp)

R A G E N Wer kann mir sagen, wie ich das Disk-Basic bzw. NEWDOS mit neuen BASIC- bzw. System-Kommandos erweitern kann? (Harald Trapp)

? F Wer hat Service Manual bzw. Schaltpläne für den Genie? (Harald Trapp)

R A G E N Wer kann mir einen Tip geben, wie die Umschaltung beim 'Superdoubler' (LNW-Doubler) funktioniert? Welche Daten müssen in welche Adressen, um von Single nach Double Density und von Double nach Single Density umzuschalten? (Manfred Held)

? F Ist es richtig, daß das Zusatz-ROM beim Genie, das auf den Adressen 3000H bis 37FFH liegt, gleichzeitig mit dem Floppycontroller, der die Adressen 37E0H bis 37FFH belegt, eingeschaltet ist ? (Manfred Held)

A G E N Wer hat: Aktien-Programme ? Programme, die mit dem Bauen zu tun haben ? Dateiverwaltungs-Programme ? (Günther Wagner)

? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN ? FRAGEN

Neu im Club!

Kuhn Eckehard
Im Dorf 14
7443 Frickenhausen 1
07022/45417

Clubmitglied seit : 03.01.85
Punktestand = 45
System- und Drivekonfiguration :
TRS-80 Model 1 (Level 2/ 48K)/ Cassetten
bevorzugtes Betriebssystem : -

Buskowiak Thomas
Eschersheimer Landstr. 257
6000 Frankfurt 1
069/5601621

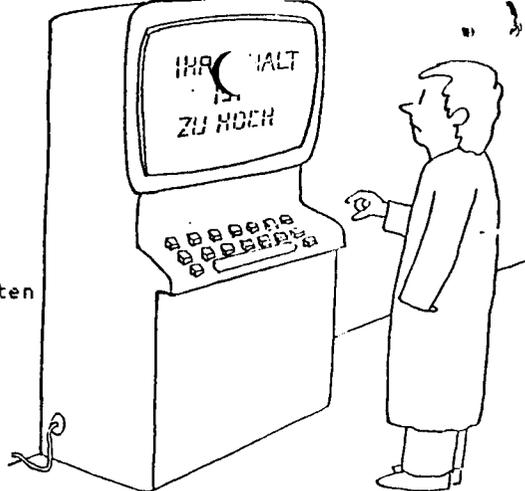
Clubmitglied seit : 03.01.85
Punktestand = 45
System- und Drivekonfiguration :
TRS-80 Model 3/ Drucker DMP 120/ 2 Laufwerke (je 40 Spuren/ ss/
dd)
bevorzugtes Betriebssystem : TRSDOS und NEWDOS

Held Manfred
Stirnerstr. 22
8835 Pleinfeld
09144/6563

Clubmitglied seit : 16.01.85
Punktestand = 45
System- und Drivekonfiguration :
Genie II/ Expander (Eigenbau)/ Drucker DRH 80/ 2 Laufwerke (je
40 Spuren/ ds/ dd)
bevorzugtes Betriebssystem : Newdos 80 V2

Baldes Hans
Johann-Strauss-Str. 6
8025 Unterhaching
089/6115179

Clubmitglied seit : 16.01.85
Punktestand = 45
System- und Drivekonfiguration :
Komtek 1S mit 32K/ Drucker Epson RX 80/ Recorder/ Monitor
bevorzugtes Betriebssystem : -



Alber Herbert
Alemannenstr. 20
7732 Niedereschach
07721/7102

Clubmitglied seit : 10.02.85
Punktestand = 45
System- und Drivekonfiguration :
Video-Genie I/ Drucker Speedy 100-80/ Basiccode-Adapter/ 1 Laufwerk
(40 Spuren/ ds/ sd)
bevorzugtes Betriebssystem : NEWDOS 80

Smerling Frank
Tanstedter Str. 5
2080 Pinneberg
04101/207284

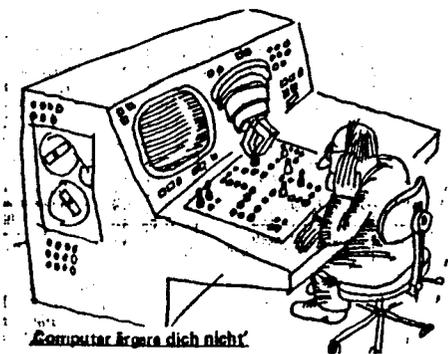
Clubmitglied seit : 14.02.85
Punktestand = 45
System- und Drivekonfiguration :
TRS-80 Model III/ Drucker RX-80 FT/ Modem/ 2 Laufwerke (je 40 Sp
uren/ ss/ dd)
bevorzugtes Betriebssystem : NEWDOS 80

Preuss Lothar
Lautshof 13
2940 Wilhelmshaven
04421/84247 (dienstl. 804-1)

Clubmitglied seit : 15.02.85
Punktestand = 45
System- und Drivekonfiguration :
Video-Genie I/ Tandy Line Printer VII/ GP 100/ 2 Laufwerke (je 4
0 Spuren/ ss/ sd und dd)
bevorzugtes Betriebssystem : TRS-DOS und G-DOS

Gesegnete Mahlzeit

Zwei Hungerige sitzen im Automaten-Restaurant. Während der eine sich sofort über seine Mahlzeit hermacht, fährt der andere skeptisch mit dem Löffel durch die Suppe. Schließlich fördert er etwas zutage und meint daraufhin ärgerlich zu seinem Tischnachbarn:
»Früher fand man nur Haare in der Suppe. Heute sind es - Schrauben.«



Das Bankgeheimnis

Es war im Hochsommer, und der Mann hinter dem Bankschalter hatte die dunkle Sonnenblende tief herabgezogen. Die Kundin, die jetzt an den Schalter trat, konnte ihn daher nicht sehen - wohl aber er sie. Der Mann begrüßte die Kundin und schob das Zahlbrett heraus. Sie blickte mißtrauisch auf das Brett, legte ihren Scheck darauf und zog hastig die Hand zurück. Der Scheck verschwand, und das Brett kam mit einigen Banknoten wieder hervor. Die Dame blickte sich um, als fürchte sie einen Lauscher. Dann beugte sie sich vor und sagte: »Ich weiß, du bist nur ein Automat. Trotzdem, vielen Dank!«

Es stellt sich vor: Werner Grajewski

Am 14.09.1951 in Duisburg geboren; bin ich nach dem Realschulabschluss 1968 in die Finanzverwaltung eingetreten. Seit 1982 bin ich bei der Großbetriebsprüfungsstelle in Essen tätig. Beruflich befasse ich mich seit 1980 mit Computern. Sharp PC 1211, Sharp PC 1500, Sharp PC 1251, Epson HX-20 und Genie I war die Reihenfolge der Geräte; mit denen ich mich befaßt habe bzw. noch befasse. Die Geräte habe ich bisher nahezu ausschließlich zur Unterstützung meiner beruflichen Tätigkeit eingesetzt.

Einige Bücher und VHS-Kurse versetzten mich in die Lage BASIC-Programme im Bereich Steuerrecht zu entwickeln. Dies führte soweit, daß ich mit einem Kollegen bereits 2 Bücher geschrieben habe. Im Februar beginnen die Arbeiten für das 3. Buch. Die Bücher und Programme sind zwar für den Epson HX-20 geschrieben; jedoch habe ich auch eine Version auf dem Genie I erstellt. Die Programme kann ich dem Club laut Vereinbarung mit dem 2. Autor nicht kostenlos zur Verfügung stellen. Soweit jedoch Interesse bei einem Clubmitglied in Sachen Steuerrecht vorliegt, kann ich die Programme oder Bücher zu Sonderkonditionen besorgen.

Das Genie I kenne ich nur ungenügend und kann auch nur in BASIC Programme erstellen. Dies hat mich veranlaßt, Mitglied im CLUB 80 und in einem weiteren Club zu werden.

„Was, Sie haben 30.000 Mark Schulden und wollen meine Tochter heiraten?“ - „Allerdings, oder wissen Sie einen anderen Ausweg?“



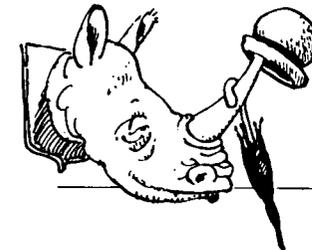
Mein Name ist Dieter Kasper, bin 22 Jahre alt und habe soeben in diesen Tagen das 3te Semester Mathematik an der TU-München hinter mich gebracht. Seit kurzem besitze ich auch eine Floppy-Station (2x80 ds/dd unter Newdos80/2) für mein Genie 1 '83er Modell; auf einen Drucker glaube ich noch verzichten zu können bis sich ein günstiges Angebot ergibt, während eine Erhöhung der Taktfrequenz mir sehr willkommen wäre.

Meine Begeisterung zum Computer wurde eigentlich erst in der Kollegstufe durch die bekannten cbm 4xxx geweckt. Auch im Bewußtsein der Nützlichkeit für das spätere Berufsleben habe ich dann die Investition für den listigen Rechenknecht gewagt. Mein Interesse liegt in der Anwendung und Erstellung von Programmen (Hardware ist so ne Sache) und so versuche ich mich zur Zeit mit Pascal, Fortran und Assembler etwas näher anzufreunden.

Ich selbst erhoffe mir von "alten, erfahrenen Hasen" hilfreiche Tips, wenn ich mal irgendwo nicht weiter weiß und das liebe Gerät absolut nicht das tut, was ich erwarte. Meinerseits wünsche ich, daß ich Anderen in mathematischen, physikalischen und chemischen Problemen etwas weiter helfen kann.

Persönliche Vorstellung von - Manfred Held -

Ich heiße Manfred Held, bin 30 Jahre alt und von Beruf Ausbilder fuer Informatioelektroniker bei der DB AW - Nuernberg. Nachdem ich einsah, daß ich das Betriebssystem in Hex-dez. fuer meinen Computer Marke "Eigenbau" nicht selbstschreiben konnte, habe ich mir 1982 einen Genie II gekauft. Die Erweiterungen habe ich selbst gebaut. (Faedeltechnik) Mein System besteht aus - Genie II, 48k Speicher, 2 Laufwerken DS/DD, Epromer (aus Elcomp 10/82), IHR 80, SD 4035. - Derzeit baue ich einen neuen Genie (Richtung Genie III S). Mein bevorzugtes Betriebssystem ist Newdos 2.0 Durch den CLUB 80 erhoffe ich einen Erfahrungsaustausch und Tips.



Hallo Clubmitglieder ,

ich freue mich nun Mitglied im Club 80 zu sein und möchte mich kurz vorstellen .

Ich heiße Hans Jürgen Bozek , bin am Nikolaustag 1954 in einem kleinen Vorort von Hamburg geboren und heute als HF-Techniker tätig .

Schon früh kam ich durch meinen Beruf mit der Digital-Technik und später mit der Micro-Technik in Berührung und beschäftige mich daher seit etwa vier Jahren mit Computern .

Nach dem CT 65 und dem ZX 81 kaufte ich mir vor zwei Jahren den Komtek 1 (Kurzbeschreibung anbei) .

Nach anfänglichen Schwierigkeiten mit dem TRS 80 - Nachbau verbringe ich nun einen großen Teil meiner Freizeit mit der Herstellung von Verwaltungsprogrammen .

Trotz fehlender Unterstützung meines Komtek-Lieferanten ist es mir endlich auch noch gelungen NEWDOS 80 auf der Maschine zum Laufen zu bringen und somit in weitere Dimensionen meines Hobbys vorzudringen .

Mit freundlichen Grüßen

Kölner Erfolgsmodell



Hans Jürgen Bozek

Auch für Tünnes und Schäl, die beiden Kölner Originale, bleibt die neue Zeit der Elektronik nicht ohne Folgen: Während Schäl arbeitet wie jeder anständige Mensch, geht Tünnes spazieren. Schäl, der genau weiß, daß sein Freund über keinen roten Heller verfügt, wundert sich darüber und stellt ihn zur Rede. Tünnes erklärt ihm, er habe sich einen Klein-Computer angeschafft, der ihm jetzt wie ein Sekretär alle Arbeit abnehme. Schäl empört sich: »Do Jeck, det Ding koss doch enen Haufen Jeld. Wie willst du dat dann bezahlen?« Meint Tünnes zuversichtlich: »Dat Problem is er jo jrad am löse!«

Nachdem wir uns im letzten Teil hauptsaechlich mit allgemeinen Hinweisen und anschliessend mit der ersten von vier DIVISIONS beschaeftigt haben, soll es nun hier an's Eingemachte gehen.

Zunaechste eine kleine Wiederholung:

```
1----67A---B----- usw. ... 72
IDENTIFICATION DIVISION.
*****
PROGRAM-ID. WAHL.
AUTHOR. HARALD TRAPP.
INSTALLATION. KRANICHSTR 46 4270 DORSTEN 1.
DATE-WRITTEN. 10/12/83.
*DATE-COMPILED. 10/24/83.
SECURITY. KEINE.
*REMARKS. AUSWAHLMENUE FUER STAMMDATENPFLEGE.
*
*
```

Das war also die erste DIVISION die wir im ersten Teil besprochen haben. Sind die Bedeutung der Spalten 1 bis 6, der Spalte 7, der Zone A und der Zone B noch in Erinnerung ?? (sonst schnell noch einmal nachschauen)

Es geht also um die Programmeinleitung. - Der Stern (*) in der Spalte 7 bedeutet, dass es sich hier um Bemerkungen handelt. Die Befehlswoerter, die teilweise hinter den Sternchen stehen, sind im Standard COBOL enthalten - nicht so im COBOL fuer TANDY und aehnliche. Aus diesem Grunde und zum Zweck der vollstaendigen Information stehen diese Kommandos hinter den bewaerten Sternchen.

An dieser Stelle sei bemerkt, dass sich diese Programmeinleitung auf ein von mir entwickeltes Programmpaket (ADRESSEN TEXT) bezieht. Im Zuge dieses kleinen Kurs werde ich auf einige Punkte dieses Programmpakets eingehen und erlaeuern. ADRESSEN TEXT ist ueberigens fuer alle Interessierte aus der CLUB80 Programmbibliothek abrufbar, das komplette Paket besteht aus vier Disketten, der Objekt-, Daten- und Reorganisations-Diskette, sowie einer Diskette mit den COBOL-Sources.

Nach diesem Kurs besteht fuer alle Interessierten die Moeslichkeit das Programm zu verbessern und zu erweitern.

So, nun aber genug der Werbung fuer ADRESSEN TEXT, kommen wir zur zweiten von vier DIVISIONS, der ENVIRONMENT DIVISION.

Frei uebersetzt koennte man dazu sagen: 'Maschinen/Geraete Erklaeung'. Damit liegen wir gar nicht so falsch. Tatsaechlich werden hier allen physischen Dateien (und Peripherien) logische Name zugeordnet und die Dateien naeher erklart.

Diese Dateien koennen z.B. Files auf dem Laufwerk (1) oder dem Laufwerk (2) sein. Aber auch der Drucker wird vom COBOL wie eine (sequentielle) Datei behandelt.

Weiterhin gibt die ENVIRONMENT DIVISION Auskunft ueber den SOURCE und den OBJECT Computer. ~~Auskunft~~. In unserem Falle ist das immer 'unser' Rechner, nicht aber so bei Grossanlagen.

Als letzte Information erhaelt die ENVIRONMENT DIVISION Angaben ueber den logischen Aufbau unserer Dateien, also sequentieller Aufbau, Random oder Key-Index (dazu spaeter mehr).

```
1----67A---B----- usw... 72
ENVIRONMENT DIVISION.
*****
CONFIGURATION SECTION.
SOURCE-COMPUTER. GENIE.
OBJECT=COMPUTER. GENIE.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT DATUM ASSIGN TO RAANDOM, "HPTDATUM/DAT:1"
ORGANISATION IS RELATIVE
ACCESS MODE IS RANDOM
RELATIVE KEY IS SATZ.
SELECT DATEN ASSIGN TO RANDOM, "STAMMDAT/DAT:1"
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEI IS E-NUMMER
ALTERNATE RECORD IS E-NAME.
*
```

Nur keine Angst vor vielen Fachausdruecken - es ist wirklich nicht so schwer wie es am Anfang immer aussieht.

Als erstes eine Erklaeung:

Beim Cobol-Programm gibt es einen gewissen Aufbau. Die vielfach genannten DIVISIONS geben die grobe Struktur. Untergeordnet den DIVISIONS sind die SECTIONS, von denen kann es eine unbestimmte Anzahl geben. Darunter geordnet bestimmen die PARAGRAPHEN die weitere Struktur und darunter sind die einzelnen Befehls-Woerter.

Nach einem kurzen Blick auf die ENVIRONMENT DIVISION wird man¹ also feststellen, dass diese Division aus zwei SECTION's besteht. Die erste (CONFIGURATION-SECTION) gibt Auskunft ueber die verwendeten Computer. Die zweite SECTION (INPUT-OUTPUT SECTION) bestimmt unser 'File-Handling'.

Die INPUT-OUTPUT SECTION unterteilt sich in die FILE-CONTROL und die I-O CONTROL. (Um letztere brauchen wir uns nicht zu kuemmern, da Sie nur ganz selten benutzt wird, die Funktion soll aber nicht verschwiegen werden: Um Hauptspeicherplatz zu sparen kann man ein und den gleichen Speicherbereich fuer mehrere Dateien gebrauchen, man reserviert also z.B. 200 Bytes fuer Stammdaten und kann diese Bytes auch gleichzeitig fuer irgendeine andere Datei benutzen).

Aber wie gesagt, fuer uns nicht interessant. Bleibt also noch die FILE-CONTROL.

Hier wird erklart, welche logischen Datei-Namen den physischen Dateien zugeordnet werden. In unserem Beispiel sind das: DATUM und DATEN.

SELECT DATUM ASSIGN TO RANDDM bedeutet im Klartext: ordne dem Namen DATUM die DISKETTENDATEI (RANDDM) "HPTDATUM/DAT" im Schacht 1 zu. (siehe Beispiel)

Also: SELECT bedeutet zuordnen
ASSIGN TO RANDDM bedeutet 'zu einem Diskettenlaufwerk'

'HPTDATUM/DAT:1' gibt den Namen der phys. Datei und die Laufwerksnummer an.

ORGANIZATION IS RELATIVE, diese Aussage ist wohl ganz klar, es handelt sich bei der genannten Datei um eine RELATIV-DATEI.

ACCESS MODE IS RANDOM heisst in diesem Zusammenhang, dass ueber die Datei ueber eine bestimmte Satznummer direkt auf einen bestimmten Satz zugegriffen werden kann, die Bezeichnung der Variablen der Satznummer wird in der naechsten Zeile des ObjectProgramms genannt:

RELATIVE KEY IS SATZ, der Schluessel, also die Satznummer wird durch die Variable SATZ bestimmt. (Energische Basic-Programmierer: Variablenbezeichnungen nicht mit z.B. \$ (Dollar) oder @ Variablennamen koennen bis zu 32(!) Stellen lang sein.

SELECT DATUM..... usw. heisst also nichts weiter als das ich eine Datei habe in der das Datum gespeichert wird, meine Satznummer steht in der Variablen SATZ, es handelt sich um eine RELATIVE DATEI auf dem Laufwerk (1) mit dem Namen 'HPTDATUM/DAT', und immer dann, wenn ich diese Datei ansprechen will, kann ich das ueber den logischen Namen DATUM machen. (z.B. Schliessen der Datei 'HPTDATUM/DAT': CLOSE DATUM)

Alles klar ?? Ich glaube schon.... weiter geht's:

SELECT DATUM ASSIGN TO RANDOM ist wohl jetzt nicht mehr schwierig, es verhaelt sich alles so, wie oben beschrieben.

Aber jetzt: ORGANIZATION IS INDEXED

Key-Index Dateien werden dem reinen Basic Programmierer noch nicht untergekommen sein. Diese Dateiform hat fantastische Moeglichkeiten (und das nicht nur zur Stammdatenverwaltung). Vom BASIC her kennt man nur sequentielle und Random (Relative) Dateien (also direkter Zugriff auf eine bestimmte Information ueber die Satznummer) Bei einer Stammdatenverwaltung wuerde das z.B. heissen, dass ich dem Meier den Satz (und damit auch die Nummer) (1) zuweise, dem Karl die Satz'Nummer' (2) usw.

Moechte ich nach einer bestimmten Zeit alle Daten vom Meier abrufen, so habe ich mit hundert % iger Sicherheit die Nummer vergessen, oder ich sehe in einer Liste nach (die dann aber auch bereits die gewuenschte Informationen enthalten kann), und somit wird der Computer schon wieder unnuetz.

Nicht so bei Key-Index Dateien. Bei dieser Form der Datenspeicherung wird den eigentlichen Daten ein Index v orgestellt. Also eine Art Inhaltsverzeichnis - das ist noch nicht unbedingt etwas besonderes. Stellen wir uns ein umfangreiches Nachschlagewerk vor. Wir suchen nach einem bestimmten Begriff und schlagen das Stichwortverzeichnis auf. Dieses Verzeichnis ist meist immer alphabetisch geordnet.

Übertragen wir das nun einmal auf unsere Stammdaten-Datei. Wir haben z.B. drei verschiedene Meier gespeichert. Wir wissen also, das wir in unserem Stichwortverzeichnis nach mindestens dem Begriff: 'MEIE' suchen muessen. Wir erhalten nun die Information:
Meier, Alfred Band 10, Seite 376

Meier, Gustav Band 13, Seite 561 usw.

Nun wissen wir, wo wir unsere 'Meier' finden koennen, wir suchen aber den Meier, der am 17. Juni Geburtstag hat. Also eine Verknuepfung (quasi eine kleine Datenbank). Nun, so etwas laesst sich alles (wenn auch mit Einschränkungen) mit einer Key-Index Verwaltung vornehmen.

Kommen wir auf den Boden der Tatsachen zurueck. In Verbindung zu meinem ADRESSEN TEXT bedeutet Key-Index Verwaltung, dass ich Daten zu einer bestimmten Person wahlweise ueber das erste Verzeichnis (Index-1), die Nummer abrufen kann (Nummer ist nicht unbedingt gleich die Satznummer, unsere Nummer kann auch alphanumerisch sein), oder aber einen Zugriff direkt ueber den Namen habe (ich gebe also direkt den Namen: 'Meier' ein und erhalte die gewuenschte Information). Man kann sogar, wenn man z.B. nur 'Mei' eingibt, von dieser Position aus Blaettern (also sequentiell suchen....).

Um auf die SELECT DATEN ASSIGN Anweisung zurueckzukommen:

Es wird die Aussage gemacht, das alle Information dem logischen Namen DATEN, der physischen DATEI 'STAMMDAT/DAT:1' zugeordnet wird, die Datei INDEX-organisiert ist, der Zugriff direkt erfolgt und zwar wahlweise ueber den INDEX E-NUMMER oder den INDEX E-NAME.

E-NUMMER und E-NAME sind hierbei Variablen, E-NUMMER bedeutet hier Eingabe-Nummer und E-NAME Eingabe-Name. Anders ausgedrueckt, mein erster Zugriffsschluessel ist eine (wahlweise alphanumerische) Nummer, der zweite Schluessel ist der Name.

Der Vollstaendigkeit halber sei noch erwaehnt, dass ich einer Datei bis zu 8(!) Schluessel zuordnen kann (also z.B. den Wohnort, Geburtstag usw.) Fuer doppelte Schluessel (es koennen ja mehrere gespeicherte Personen in ein und dergleichen Stadt wohnen) gibt es eine spezielle Klausel (... WITH DUPLICATES).

Man kann natuerlich auch reif/ sequentielle Dateien behandeln:
SELECT DATEI ASSIGN TO RANDOM, 'BRABRA/DAT:0'
ORGANIZATION IS SEQUENTIAL
ACCESS MODE IS SEQUENTIAL.

Man kann aber auch relative Dateien sequentiell, random oder dynamic behandeln (also sequentiell durcharbeiten, z.B. fuer Stammdatenlisten, random (direkter Zugriff ueber Satznummer) bearbeiten, oder aber sequentiell und random -dynamic bearbeiten)

Im Klartext heisst das, das ich eine durch OPEN geoffnete RELATIVE DATEI ohne Ruecksicht in ein und demselben Programm sequentiell, mit Direktzugriff oder sogar beidem bearbeiten, sprich auf den Datensatz zugreifen kann.

SELECT DATEI ASSIGN TO RANDOM, 'BRABRA/DAT:1'
ORGANIZATION IS RELATIVE
ACCESS MODE IS SEQUENTIAL, RELATIVE KEY IS xxxxx
oder RANDOM, RELATIVE KEY IS zzzzzz
oder DYNAMIC.

(bei RELATIV-DATEIEN muss bei sequentiellem oder direktem

DATENFERNUEBERTRAGUNG (DFUE)

Frage:

Akustikkoppler selbst bauen oder lieber kaufen ??

Ein Bericht von Harald Trapp

Datenfernuebertragung als Schlagwort - es ist schon mehrmals im Club-Info gefallen. Bereits im Info Nummer vier habe ich einen Bericht ueber den Einsatz des Akustikkopplers der Firma EPSON abegeben. Inzwischen wurden auch viele verschiedene Anweisungen einer fuer Datenfernuebertragung noetigen V.24 Schnittstelle gemacht (Interface und Software von RB electronic ca. 240 DM) und nicht zuletzt eine Eigenbauversion von unserem Mitglied W. Zwickel.

Tja, was braucht man fuer DFUE ?? Es wurde schon gesagt, man benoetigt einen Akustikkoppler (oder einen MODEM), eine serielle Schnittstelle und Kommunikationssoftware.

Zunaechst zum Modem:

Ein Modem wurde dem Computeranwender von der Post zur Verfuegung bestellt, es kostet 80 Dm im Monat (fuer 300 Baud Uebertragungsrates = ca. 35 Zeichen/sec), eine einmalige Anschlussgebuehr von 80 DM wird zusaetzlich faellig und gekoppelt wird ein Modem nur mit Rechnern mit FTZ-Zulassungsnummer-. Diese Loesung faellt also fuer den zukuenftigen DFUE-isten flach (Erwaehnt werden soll allerdings noch, dass mit einem galvanisch gekoppelten Modem ein fast stoerungsfreier Betrieb moeglich ist, man kann nicht nur Rechner anwaehlen, sondern sich sogar anwaehlen lassen. Das heisst, der Rechner ist in Betrieb und kann jederzeit von aussen durch irgendeinen Benutzer angewaehlt werden (wie die ueblichen Mailboxen)).

Fuer Club 80 Mitglieder bbt also nur noch der Akustikkoppler, doch dazu spaeter mehr. Was benoetigt man noch ?? Eine Schnittstelle, ueber die die Daten in den Computer 'rein' und aus den Computer 'raus' koennen. Diese Schnittstelle funktioniert aehnlich einer Druckerschnittstelle, die Daten werden bei einer Centronics-Druckerschnittstelle jedoch jeweils 8 bit parallel abgegeben, bei einer Schnittstelle fuer Akustikkoppler (und auch andere Anwendungen) bit-seriell. Man nennt so eine genormte Schnittstelle 'V.24' oder 'RS 232'. (Unterschiede liegen hier bei der Steckerebelegung od. Ausfuehrung, denn obwohl genormt, Ausnahmen bestaetigen die Regel).

So eine Schnittstelle kann man entweder kaufen, z.B. rb electronic (mit entsprechender Kommunikationssoftware) oder aber selber bauen (z.B. in Faedeltechnik, z.B. die Schnittstelle von unserem Mitglied W. Zwickel). Wird die Schnittstelle von W. Zwickel noch mit V.24 Treibern, einer symmetrischen Speisung von ca. +/- 12 Volt versehen und eine Software fuer diese Schnittstelle geschrieben, so hat man fuer vergleichsweise sehr wenig Geld eine mindestens ebenbuertige Schnittstelle zur Verfuegung.

Hat man die Schnittstelle und die Software, dann fehlt eigentlich nur noch der Akustikkoppler, und, damit auch alles gesetzlich geregelt ist, sollte der Akustikkoppler eine FTZ-Nummer besitzen. Diese Koppler kosten leider einiges an Geld. Obwohl die Preise fuer Koppler gefallen sind, so liegen

sie doch zwischen 300 DM und 800 DM.

Man kann so etwas auch selber bauen (ohne FTZ-Nummer aber mit Risiko und hier beginnt meine Geschichte!)

Durch zahlreiche Beitraege in der Fachpresse und nicht zuletzt durch den Einsatz von Akustikkopplern im Betrieb wurde mein Interesse geweckt. Nachdem ich vom Betrieb aus mit mehreren Boxen Kontakt aufgenommen habe, wollte ich dies auch von zu Hause aus. Also wurde eine V.24 Schnittstelle besorgt (die von rb electronic, die ich immer wieder nur loben kann) und, da das Geld nun ziemlich verbraucht war, das Modem Sonderheft mit einer Bauanleitung fuer einen Akustikkoppler.

Da ich mich schon sehr lange mit Elektronik beschaefigte und auch entsprechend labormaessig mit Oszilloskop und aenlichem ausgeruestet bin, sollte mir der Nachbau nicht schwerfallen.

Also wurde ruck-zuck eine Platinenfolie vom Layout angefertigt, eine CU-Platte belichtet, gesetzt und gebohrt. Der erste Schritt war getan.

Bauteile wurden aus einem nahegelegenen Shop zu recht humanen Preisen gekauft, Standardwiderstaende mit 10 bis 20% Toleranz, hochwertige Folienkondensatoren und gepruefte Transistoren im Metallgehaeuse und zuletzt noch ein paar Schalterchen.

Die Platine wurde bestueckt, ein Steckernetzteil mit 12 Volt anschlossen, als Schallwandler dienten zwei Miniatur-Lautsprecher. Ein munteres Pfeifen war nach dem Anlegen der Versorgungsspannung zu hoeren, legte man das Schalterchen um, so wurde es sogar noch hoeher. Mir war klar, der Modulator funktionierte.

Nun aber fluss ans Abgleichen (hierzu dienen zwei Trimmer). Zwei Trimmer fuer Modulator, zwei fuer den Demodulator, jeweils Sende und Empfangskanal. Der Abgleich wurde mit dem Oszilloskop vorgenommen, Kanal A auf 980 Hz, Kanal B auf 1650 Hz. Mit einem aus dem Betrieb geliehenen Koppler wurde mein Abgleich kontrolliert, den Koppler am linken Ohr, den Lautsprecher am rechten Ohr wurden die erzeugten Toene verglichen. Sie stimmten (wie solls auch anders sein) nicht ganz ueberein - woher nun die dritte Hand zum Drehen an dem Trimmer ?? Irrendwie gings schliesslich doch und ich habe beide Frequenzen auf Schwebungsnull abgleichen koennen.

Doch was waren das vorher fuer Experimente ?? Mit den angegebenen Bauteilwerten war die untere Frequenz ueberhaupt nicht zu erreichen - statt 56 Ko musste ein Widerstand von 82 Ko eingesetzt werden (ob man hier noch von Toleranzen sprechen kann??). Der Widerstand kam in die Platine und somit auch der gewuenschte Ton aus dem Lautsprecherchen.

Das war der zweite Schritt - schon sehr erfolgreich. Nun musste ueberprueft werden, ob der Modulator auch von 980 Hz auf 1180 und von 1650 auf 1850 Hz schaltete. Ein Widerstand nach +12 Volt schaltete den Modulator um, ein Vergleich mit dem geliehenen Koppler - schauerhaft schaurige Uebereinstimmung der zweiten Frequenz, diese war nicht ueber Trimmer zu beeinflussen, die zwei verbliebenen Trimmer waren fuer die PLL, also fuer das 'Einrasten' des Demodulatorteils und nicht fuer die Frequenz zustaendig. Ich redete mir ein, dass diese Abweichungen im Rahmen der Toleranz lagen, kurz ueberprueft, und bestaetigt.

Nur noch zwei Trimmerchen (PLL) und es kann fast aussehen. Ueberpruefen des Demodulators. Dazu habe ich eine kleine Schaltung fuer einen V.24 Treiber aufgebaut, zwei IC's fuer die Signale, eins zum Invertieren der Signale und eins zur Spannungswandlung, +/- Spannung aus nur einer Stromversorgung.

Die V.24 Schnittstelle des Kopplers wurde mit der V.24 Schnittstelle des Rechners verbunden. - Ein Schraubendreher wurde in die Tastatur geklemmt damit immer ein Signal ueber die Rechner-Schnittstelle an den Koppler gesandt wurde.

Die Schalterchen des Kopplers stellte ich beide auf Kanal A, legte beide Lautsprecher aufeinander (akustische Kopplung) und los ging's mit vielen wirren Zeichen auf dem Bildschirm. Ein paar vorsichtige Drehungen am Trimmer und einwandfrei erschien der 'auf der Tastatur festgeklemmte' Buchstabe auf dem Schirm. Meine Freude war gross, schnell ein Schluck Mineralwasser (!) fuer den klaren Kopf, die Flasche zurueck auf den Tisch und schon war es passiert, ein Kuddelmuddel auf dem Bildschirm, dann wieder das 'festgeklemmte Zeichen'. Ein vorsichtiges leichtes Husten bestaetigte meinen Verdacht: sehr stoeranfaellig auf Umweltgerauesche, na das wird sich schon nochgeben (beruhigte ich mich).

Umschalten auf Kanal B, schnell noch mal die gleiche Prozedur. Aber nein, es klappte nicht. Ja wieso denn nicht ?! (Verblueffung), noch einmal auf Kanal A, alles ok, und Kanal B ?

Nichts !!, Selbst ein lautes und unwirschiges Husten in das Lautsprecher-Mikrofon. Ja holt der Geier

Um meinen Frust nicht weiter zu beschreiben: es las einfach an der mangelnden Empfindlichkeit des Lautsprecher-Mikrofons auf dem zweiten Kanal. Ich habe dann die Lautsprecher gegen Hoer- und Sprechkapsel (von der Post) vertauscht. Das Ergebnis war noch mieser, jetzt ging gar nichts mehr. Das Kohlemikrofon war viel zu traese. Also wurde wieder der Lautsprecher als Lautsprecher und die dynamische Hoerkapsel der Post als Mikrofon eingesetzt.

Meine Freude war gross - endlich klappte alles auf beiden Kanalen, die Zeichen kamen einwandfrei auf dem Bildschirm an.

Schnell wurden die Schallwandler in Kappen von Spraydosen (von Teppich-Schampoo), eingebettet und in viel Schaumstoff verpackt.

Telefon her, ca. 20 Minuten vergebliches Waehlen, dann endlich eine freie Mailbox. Ja was kam da fuer ein Kuddelmuddel ueber den Bildschirm. Um es kurz zu machen, nach einer Woche frustrierenden Versuchen habe ich den Koppler aufgegeben. Das Ding war einfach zu unempfindlich fuer die Datenuebertragung und zu empfindlich auf Umgebungsgerauesche. Selbst die Neon-Roehren in meinem Arbeitszimmer verursachten Stoerungen, kein Laut durfte zu hoeren sein. Jedes Druucken einer Taste verursachte eine Stoerung.

Abschirmte Kabel brachten ebensowenig wie ein Auswechseln und zweites Aufbauen der Schaltung. Nun, ich habe mir einen Akustikkoppler der Firma TANDY gekauft, Kosten ca. 400 DM, Ausgegeben habe ich fuer den Selbstbaukoppler ca. 120 DM, mit V.24 Schnittstelle, Steckernetzteil, alle moeslichen Schallwandler und was weis ich) - haette ich mir sofort den TANDY Koppler gekauft, so haette er mich nach der Rechnung nur

280 DM gekostet, so aber 520 DM. (auch wenn die Rechnung nicht ganz stimmt). Hinzu kommen die vielen Abende (und Naechte) des Probierens, (die Telefonrechnung habe ich noch nicht bekommen, aber mir graut es schon jetzt).

Wenn man aber ganz sachlich ueberlest, so sollte einem Jedem, der diese Schaltung nachbauen will klar sein, dass ein Bausatz von ca 40 bis 50 DM nicht das gleiche leisten kann wie ein Geraet von ca. 400 DM. Das wird einem auch klar, wenn man mal in den 'Bauch' eines gekauften Kopplers sieht. An Jedermann die Warnung also: Diesen Koppler nicht nachbauen.

Die Filtereigenschaften dieses Kopplers sind sehr gering - ausserdem wird ueberhaupt nicht beachtet, dass die Post, um die Sprachverstaendlichkeit zu erhoehen, die Frequenzen anhebt. Das muss der Koppler ebenso ausgleichen koennen wie die unterschiedlichen Empfindlichkeiten des Mikrofons auf verschiedene Frequenzen. Es muesste also fuer jeden Kanal noch ein zusaetzlicher Trimmer fuer die Lautstaerke vorhanden sein usw.

Ich habe allerdings mit dem TANDY Koppler keine Probleme und kann diesen sehr empfehlen - und wenn einer doch Lust hat den Koppler zu bauen: Mit Piezos und Electret-Mikro habe ich es noch nicht probiert, vielleicht fruchtet das ? Und an alle diejenigen, die diesen Koppler zur Zufriedenheit bauen koennen (oder bauen werden), meinen herzlichsten Glueckwunsch. Ich jedenfalls bin kuriert - und wenn einer mit mir ueber Akustikkoppler in Kontakt treten will (was mich sehr freuen wuerde), und die Uebertragung ist unter allen Sa..., dann liegt es entweder an einer schlechten Sprechleitung oder an diesem bewussten Koppler der trotz allem seinen Einsatz gefunden hat.

Den Akustikkoppler von Tandy kann ich Euch fuer 349,00 DM (anstatt knapp 400 DM) anbieten. Das Angebot geht allerdings nur bis 18. März!

Wer dieses Angebot wahrnehmen will, der schickt mir einen Scheck ueber 349,00 DM wenn ich ihm den Akustikkoppler beim Clubtreffen uebergeben kann, andernfalls einen Scheck ueber 354,00 DM (Porto!).

Günther Wagner
Gartenstraße 4
8201 Neubuern

```

00010 ;*****
00020 ;* PROGRAMMNAME : DOSCAL *
00030 ;* FILENAME : DOSCALL/ASM *
00040 ;* DATUM : 12.05.84 *
00050 ;* *
00060 ;* ERSTELLT VON : JOSEF KONRAD *
00070 ;* ANZENGRUBERSTRASSE 35 *
00080 ;* B038 GROEBENZELL *
00090 ;* *
00100 ;* BESCHREIBUNG : ERWEITERUNG FUER TASMOM ZUM *
00110 ;* AUFRUFEN VON NEWDOS80-ROUTINEN NACH EINGABE *
00120 ;* DES KOMMANDOS 'U'. *
00130 ;* FUER DIE ANGEGEBENE ORG-ANWEISUNG MUSS TASMOM *
00140 ;* IM BEREICH A000H-BFFFH ABGESPEICHERT SEIN. *
00150 ;* DIE DOS-KOMMANDOS DUERFEN MAXIMAL 32 ZEICHEN *
00160 ;* LANG SEIN. *
00170 ;* *
00180 ;* LITERATUR 80 MICRO 6,7/82/12 *
00190 ;* *
00200 ;*****
00210 ;
00220 ;
9F75 00270 ORG 9F75H
0020 00280 BUFF DEFS 20H
9F95 44 00290 MESS1 DEFM 'DOS COMMAND ?'
9FA2 57 00300 MESS2 DEFM 'WEITER <ENTER>'
9FB1 00310 MESEND EQU $
00312 ;
00314 ;
9FB1 E5 00320 DOSCAL PUSH HL
9FB2 C5 00330 PUSH BC
9FB3 D5 00340 PUSH DE
9FB4 CD5CA9 00350 CALL WL ;DISPLAY 'U'
9FB7 01000A 00360 LD BC,0A00H
9FBA CD6000 00370 CALL 60H ;DELAY
9FBD CDABA9 00380 CALL CLS
9FC0 21959F 00390 LD HL,MESS1
9FC3 11403F 00400 LD DE,3F40H
9FC6 010D00 00410 LD BC,MESS2-MESS1
9FC9 EDB0 00420 LDIR
9FCB 214F3F 00430 LD HL,3F40H+MESS2-MESS1+2
9FCE 22B40F 00440 LD (4020),HL ;SETZE CURSOR
9FD1 21759F 00450 LD HL,BUFF
9FD4 0620 00460 LD B,20H
9FD6 CDD905 00470 CALL 05D9H ;INLINE
9FD9 CD1944 00480 CALL 4419H ;DOS COMMAND AUSFUEHREN
9FDC 3807 00490 JR C,END ;FALLS KEIN FEHLER
9FDE 2805 00500 JR Z,END ;FALLS KEIN FEHLER
9FE0 CBFF 00510 SET 7,A
9FE2 CD0944 00520 CALL 4409H ;AUSGABE FEHLERMELDUNG
9FE5 21A29F 00530 END LD HL,MESS2
9FEB 11C03F 00540 LD DE,3FC0H
9FEB 010F00 00550 LD BC,MESEND-MESS2
9FEE EDB0 00560 LDIR
9FF0 21D13F 00570 LD HL,3FC0H+MESEND-MESS2+2
9FF3 22B40F 00580 LD (4020),HL ;SETZE CURSOR
9FF6 CDE3A0 00590 LOOP1 CALL KYINP ;TASMOM TASTATURABFRAGE
9FF9 FE0D 00600 CP 0DH
9FFB 20F9 00610 JR NZ,LOOP1
9FFD D1 00620 POP DE
9FFE C1 00630 POP BC
9FFF E1 00640 POP HL

```

```

00642 ;
00644 ;** HIER BEGINNT TASMOM **
00646 ;
A95C 00650 WL EQU 0A95CH
A9AB 00660 CLS EQU 0A9ABH
A0E3 00670 KYINP EQU 0A0E3H
00680 ;
00690 ; PATCH IN USER ROUTINE SPRUNGADRESSE
00700 ;
00710 ;
A0DE 00720 ORG 0A0DEH
A0DE B19F 00730 DEFW DOSCAL
9FB1 00740 END DOSCAL
00000 TOTAL ERRORS
31334 TEXT AREA BYTES LEFT

```

```

BUFF 9F75 00280 00450
CLS A9AB 00660 00380
DOSCAL 9FB1 00320 00730 00740
END 9FE5 00530 00490 00500
KYINP A0E3 00670 00590
LOOP1 9FF6 00590 00610
MESEND 9FB1 00310 00550 00570
MESS1 9F95 00290 00390 00410 00430
MESS2 9FA2 00300 00410 00430 00530 00550 00570
WL A95C 00650 00350

```



Das Programm DOSCAL ermöglicht die Ausführung von NEWDOS80-Kommandos (z.B. DIR <dn>) von TASMOM aus durch Eingabe des Kommandos 'U'.

Nach dem Abspeichern auf Diskette wird das Programm von TASMOM aus mit L D <ENTER> DOSCALL/CIM zum TASMOM-Programm 'gelinkt' und dann das erweiterte Programm mit W D 9F95 BFFF A000 TASMOM/CMD abgespeichert (gilt wenn TASMOM im Bereich A000-BFFF liegt; andernfalls muß man die entsprechenden Adressen und die ORG-Anweisung ändern bzw. TASMOM in den oben genannten Bereich verschieben; 'linken', dann wieder in den gewünschten Bereich verschieben und dann mit den entsprechenden Adreßeingaben abspeichern).

Richard's Regel zum Eigentum:
 1) Wenn man Dinge nur lange genug aufbewahrt, kann man sie danach auch sicher wegwerfen.
 2) Wirft man hingegen etwas weg, so benötigt man es in dem Augenblick, in dem es nicht mehr greifbar ist!

Ich möchte als einer der wenigen Komtek-Besitzer auf diesem Planeten der Bitte von Hartmut nachkommen und den Computer kurz beschreiben .

Es handelt sich bei dem KOMTEK 1 um einen mechanisch recht gut gelungenen TRS 80 - Nachbau aus Hongkong , der vor etwa zwei Jahren für ca 1000,- DM zu bekommen war .

In der Grundausstattung 16k-RAM (12k-ROM Basic Level II), die leicht intern auf 32k-RAM , einfach durch Einstecken der RAM in vorhandene Sockel zu erweitern ist .

Groß/Kleinschreibung , Umlaute , Tastenentprellung und Autorepeat stehen nach Aufrufen von 'SUPER-BASIC' zur Verfügung . Auch belegt das 'SUPER BASIC' jede Taste mit einem Graphiczeichen , das , wenn es nicht in "" gesetzt wird einem Basic-Befehl entspricht . (Also : Hochpfeil"↑" = / Hochpfeil ↓ = CLOSE)

Wenn man erst einmal im Kopf hat , was sich unter der jeweiligen Taste versteckt , dann kann man sehr viel Zeit beim Programmieren sparen .

Der KOMTEK 1 verfügt über einen eingebauten und programmierbaren Tongenerator incl. Lautsprecher , der mit Hilfe eines Maschinenprogramms auch ohne Anschluß eines-Cassettenrecorders o.ä. den Computer zum 'Sprechen' bringt .

Echtzeituhr und sechs programmierbare Zeitschalter mit 'Chinch-Ausgängen' sowie vier programmierbare Sensoreingänge fordern die Bastler unter uns heraus . Ohne zusätzliche Hardware können direkt Relais oder Transistoren geschaltet werden , die dann z.B. eine elektrische Eisenbahn , die Zentralheizung oder etwa eine Messdaten erfassung steuern .

Weiterhin verfügt das Gerät über einen 50 Pin-Erweiterungsbus , einen Disk-Driverport (Shugart - nicht betriebsbereit) und ein Druckerinterface .

Die Nachrüstung mit einem Floppy-Controller ist nicht problemlos , da im Gerät kein Platz für einen Controller ist und die Schnittstelle nicht mit TRS/GENIE hardwaremäßig kompatibel ist .

Eine genaue Dokumentation oder gar ein Schaltbild für das Gerät konnte ich bis heute nicht auftreiben . Das hat zur Folge , daß ich weder TRSDOS noch NEWDOS 40 sondern nur NEWDOS 80 zum Laufen bekommen habe (wahrscheinlich wegen Speicherdoppelbelegung) .

Meine Erfahrungen beziehen sich fast ausschließlich auf das recht komfortable Basic und da ich weder den TRS 80 noch das GENIE genau kenne , kann ich keine evtl. Unterschiede aufzeigen .

Kurz aber noch einige Nachteile des Komtek 1 .

Er verfügt nicht über Funktionstasten und die Reset-Taste befindet sich genau über der Return(Enter)-Taste . Dokumentation über Speicherbelegung oder Schaltbilder sind schwer oder garnicht zu bekommen . Der Erweiterungsbus ist nicht an TRS/GENIE- Zubehör anzuschließen .

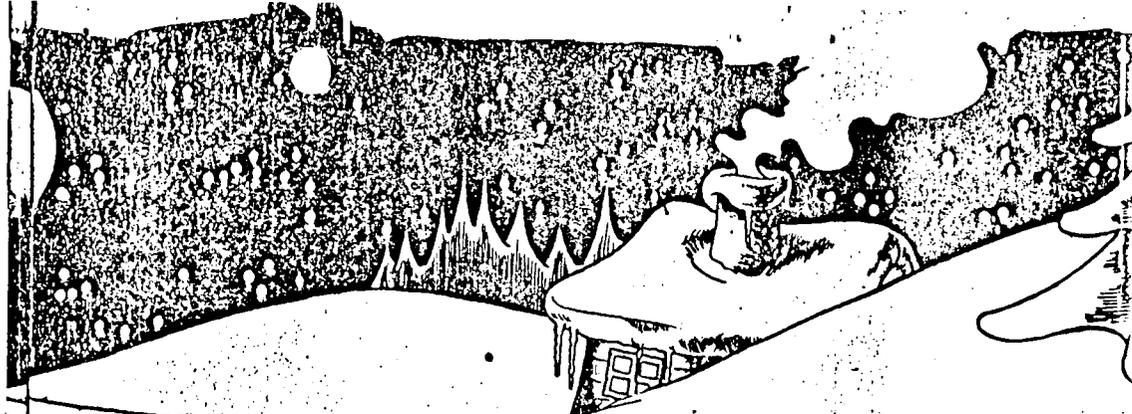
Das Gerät ist aber mit TRS/GENIE voll softwarekompatibel .

Kaufen kann man den KOMTEK 1 noch beim 'COMPUTER STUDIO BRAUN-SCHWEIG) für ca 650,- DM .

Speziellen Fragen zum Komtek stehe ich gern zur Verfügung .

Gruß

Hans Jürgen Bozek



Die Geschichte vom armen Hacker

Irgendwo hinter dem Dorf Oberwiesenau hausten in einer kleinwinzigen Hütte gar arme Leute, und der Computer, den sie benutzten war alt und langsam, und die Datensette savete gar kärglich, und zuweilen hatten sie keine einzige lauffähige Kople und waren weit schlimmer noch dran als die Mäuse im Wald.

Nun waren einmal wieder die seligen Weihnachten gekommen, und Mutter und Vater waren ins Kirchdorf hinuntergegangen, um dort in der Christmette den Ileben Gott um einen brauchbaren 16-biter zu bitten. Die Kindlein, die sie hatten, es waren ihrer sieben, die blieben zuhause, saßen an ihren Joysticks und meinten, es müsse heut nacht doch noch ein goldbekränzter Space-Invader daher klingeln und ihnen einen Sack voller RAM-Packs, Cartridges und neuer Spielprogramme bringen, auf daß sie nicht gar so arm und hilflos dasäßen und unter Umständen nach einem Buche greifen müßten mitten in der Heiligen Nacht. Aber draußen regte sich nichts, nur die Sterne gingen um das Haus und spiegelten sich im Schnee, und die nahen Tannen standen weiß und ehrwürdig.

Als es Mitternacht worden und es die Kinder schläferte und sie am liebsten geweint hätten vor vergeblichem Warten, sieh, da klopfete es und ein Grunzen ertönte an der Tür. Da taten sie denn stracks und fröhlich die Türe auf. Doch draußen stand kein Space-Invader, auch kein Gremlin mit Rucksack, sondern nur ein armer Hacker.

Man hätte die Rippen an ihm zählen können, so verhämt sah er aus: Man hatte ihm sein Modem beschlagnahmt und dadurch jede Lebensgrundlage genommen. So blickte er müde und verzweifelt drein und hatte gar wenig Freude mehr an der Welt.

Das arme Wesen erbarmte die Kindlein sehr. Sie nahmen den Hacker mit in die Wohnung, holten den allerletzten Kasten Pilsener aus dem Keller, räumten den Kühlschrank aus, soweit Nahrhaftes in ihm zu finden, tischten dem armen Hacker auf und nötigten ihn, vor dem Bildschirm Platz zu nehmen.

Dem Hacker, der vor Zeiten Besseres gewöhnt gewesen, mutete die Hardware der Kleinen gar kümmerlich an doch sah er ihre gute Absicht, und ihre Mildtätigkeit rührte ihn.

Nachdem der Kasten geleeret und die Happen zwischen den gelichteten Beißern des Hackers verschwunden waren, ging ein Leuchten über seine müden Augen. Er streckte die Beine aus und hielt ein erquickliches Nickerchen.

Als nun aber die Kleinen zur Ruhe kamen und merketen, daß sie den armen Eltern das letzte Weihnachtsmahl genommen und verschenkt hatten, da erschranken sie gar sehr und hatten große Angst vor der Rückkunft der Eltern.

Der Mond schien über der Alm hinter dem Haus und der silber glänzende Schnee machte ein gar friedliches Bild.

Als nach einer guten Weile der Hacker ausgeschlafen, da reckte und streckte er sich, zwinkerte den Kindern zu und wandte sich zum Gehn: Er band sich einen dicken Wollschal um, stülpte eine Pudelmütze über und stapfte in den Schnee hinaus.

Und siehe da, als ihm die Kinder nachblickten, wie er die mondbeschiedene Alm hinanstieg und ab und zu den Schnee von den Schultern schüttelte, da blinkte und blitzte es bisweilen wie Sternschnuppen. Und als die armen Kleinen hinaus traten und den Spuren des Hackers folgten, da fanden sie lauter kleine Eproms, sieben an der Zahl, im Schnee liegen.

Sie hoben sie auf, trugen sie in die Wohnung, löteten sie in ihre Rechner und sahen, daß jedes Eprom mit der köstlichsten Software beschossen war, welche die Kleinen ja kennengelernt hatten.

Und als die Eltern von der Christmette heinkehrten und die geheimnisvollen Vorgänge in dieser Nacht erfuhren, da zürneten sie nicht, sondern freuten sich mit den Kindern.

(Aus einer alten Chronik)

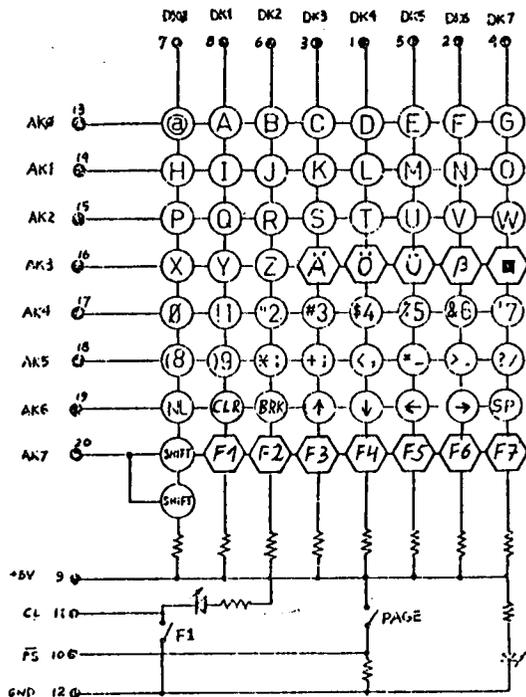
NOCHMALS: JOYSTICK - ANSCHLUSS

Im Clubinfo Nr. 5 beschreibt Jens den Anschluß eines Joysticks an das Genie bzw. den Tandy. Der Trick, über den Port Ø zu gehen, wobei auf der Adreßleitung eben nichts liegt, ist geradezu hinterlistig. Respekt! Die Nachteile nennt Jens allerdings selber: Der Busstecker ist besetzt (wenn man nicht löten will), der Port Ø ist es auch. Es gibt eine Abhilfe, bei der man allerdings ums Löten nicht herumkommt:

Unsere Tastatur ist ein Aggregat aus lauter simplen Tastern. Sie erzeugen einen Kurzen zwischen je zwei Leitungen der Tastaturmatrix. Nichts spricht dagegen, ihn von einem oder mehreren externen Tastern erzeugen zu lassen, z. B. von denen im Joystick.

Dazu müssen die Leitungen, die für das Leerzeichen und die Pfeiltasten zuständig sind (DK3 - DK7 sowie für alle gemeinsam AK6) von außen über eine Buchse zugänglich gemacht werden. Man findet sie an den Lötstellen des Flachmankabels, das die Tastatur mit der CPU-Platine verbindet. So ist das zumindest beim Genie. Die Unterschiede zum TRS-80 dürften kaum mehr als optisch sein.

Das "Who's who" bzw. "Where's what" auf der Kabelflunder habe ich jetzt nicht mehr auswendig drauf. Meinen Computer zerlegen und nachsehen möchte ich nun auch nicht. Die Numerierung der AK- und DK-Leitungen in der Abbildung von 1-20 hilft wohl weiter.



Ein Test gibt letzte Sicherheit: Wenn das Gehäuse geöffnet ist, kann man das Flachbandkabel von der Tastatur abziehen. Vorsicht! Es ist sehr steif, und die Lötstellen könnten brechen. Nun werden mit einem Stück Draht nacheinander je zwei Leitungen verbunden. Bei durchschnittlich jedem zweiten Versuch sieht man etwas auf dem Bildschirm. Beim Vergleich mit der Abbildung ergibt sich, welche zwei Leitungen man erwisch hat.

Schließlich sind alle identifiziert. An die betreffenden Kabelenden kann jetzt ein zusätzliches Kabel angelötet werden, das zu einer Buchse führen soll. Es eignet sich jede Buchse, die wenigstens sechspolig ist. Die Rückseite des Genie ist schon recht voll, aber an den Seiten des Gehäuses findet sich noch genug Platz zum Einbau.

Beim beschriebenen Kabeltest nach der Zufallsmethode sind auf dem Bildschirm sicherlich Umlaute, das ß und das Cursorzeichen erschienen (jedenfalls beim Genie). Demzufolge sind sie per Tastendruck darstellbar; es existieren bloß keine Tasten. Glücklicherweise sind sie aber auf der Tastaturplatte vorgesehen. Über der obersten Tastenreihe sind Löcher vorhanden, in die sich diese Tasten (bei TCS erhältlich) einlöten lassen. Dazu muß allerdings am Gehäuseoberteil ein Stück ausgesägt werden, damit sie Platz haben und von außen erreichbar sind.

Diese Zeichen liegen auf AK3. Dort hat die Tastaturmatrix von DK3 - DK7 eine Lücke. In der Abbildung sind an diesen Stellen die zusätzlichen Tasten mit sechseckigen Symbolen dargestellt. Eine weitere Lücke gibt es in der Shift-Reihe AK7. Nur DKØ ist mit den Shifttasten erreichbar. Die Kreuzungspunkte von AK7 mit DK1 - DK7 bieten sich für Funktionstasten an (in der Abbildung ebenfalls sechseckig): gewöhnliche Klingelknöpfe für'n Groschen, die noch gut unterhalb des eingebauten Recorders (Genie I) Platz haben. Z. Zt. entsteht für H-DOS die erforderliche F-Software.

Ein weiterer Gedanke kommt dem Bastler zwangsläufig, wenn er an der Tastatur herumlötet: Wieso nicht gleich alle Leitungen auf eine Buchse legen? Dann kann eine komplette zweite Tastatur angeschlossen werden. Das Konzept des "remote keyboard" hat seine Vorzüge. Bei meiner Maschine gibt es eine solche Buchse zusätzlich zum Joystick-Anschluß. Auch ein separater Zehner- oder gar Hexblock ist mit entsprechenden parallel liegenden Zusatztasten kein Problem.

Und eine letzte Änderung ist sinnvoll, wenn der Computer eh' schon offen ist: Die Tasten Y und Z können vertauscht werden, um die Tastatur wenigstens ein bißchen einzudeutschen. Dazu werden die Leitungen DK1 und DK2 direkt an diesen beiden Tasten durchtrennt (man findet sie nach der oben beschriebenen Methode). Über eine Drahtbrücke erhält nun die eine Taste den DK-Anschluß der anderen. Zuletzt werden noch die Tastenköpfe mit einem Schraubenzieher vorsichtig abgehobelt und vertauscht.

Die Löterei am eingeschalteten Computer kostet zwar Nerven, ist aber nicht gefährlich, wenn man ein paar Regeln beachtet. Der LötKolben sollte einen Trenntrafo haben oder bei Berührung der Platine ausgeschaltet sein. Außerdem kann man von Zeit zu Zeit die Heizung oder den Schutzkontakt der Steckdose anfassen (bitte nicht danebengreifen!), um selber nicht statisch aufgeladen zu sein. Offen gestanden habe ich selber dergleichen nie beachtet, und dennoch ging immer alles glatt.

Eine kurze vergleichende Betrachtung von Jens' und meiner Lösung sei noch nachgetragen. Wer den Computer noch "pur" hat, d. h. ohne Peripheriegeräte wie Floppy usw., die auf den Busstecker kommen (und wer sich zutraut, auch Maschinenprogramme auf die Port-I/O-Methode umzustellen), der ist mit Jens' Anschluß bestens bedient. Er spart eine Stunde Arbeit, Nerven und Lötzinn. Wer den Bus aber frei haben, die HRG 1b betreiben, bei der Gelegenheit auch gleich die weiteren Verbesserungen anbringen und seine Software unverändert lassen möchte, dem sei die hier beschriebene Methode empfohlen.

Viel Spaß beim Fummeln - wie auch immer!

Arnulf Sopp

"Unser" Microsoft-BASIC wird allgemein gelobt. Insgesamt wohl zu Recht. Einiges Ärgerliche steckt aber drin. Dazu gehört das leidige Problem, daß mit der LPRINT-Anweisung nicht alle Codes auf den Drucker ausgegeben werden können. CHR\$(0) führt nur zur Abfrage des Druckerstatus (den man in BASIC so nicht einmal erfahren kann), CHR\$(10) wird in CHR\$(13) umgewandelt (LF wird zu CR), alle senkrechten Tabulationen (VT und FF) werden zu einer Folge von Zeilenvorschüben (LF).

In der Steinzeit, als die Dampfdrucker noch kaum mehr als die Schriftzeichen beherrschten, mag das sinnvoll und einzig richtig gewesen sein. Moderne Drucker erfordern jedoch, daß als Steuerzeichen alles, also auch z. B. die 0 ausgegeben werden kann. Wenn beispielsweise in der ESC-Sequenz zum Ausdruck hochauflösender Graphik die Anzahl der folgenden Dotspalten angegeben werden muß (Gemini, DP 510, ITOH, NEC usw.), ist es unabdingbar, daß restlos alle Zeichen von 0-255 ohne Abwandlung den Interpreter verlassen. Das ist ohne Änderungen des Interpreters nur mit einer Krücke möglich: Das Genie macht es mit OUT 253, code, der Tandy mit POKE 14312,code.

Hierfür sind zwei Segmente des Interpreters verantwortlich. Die Umwandlung von LF in CR geschieht ab 03A6. Dort wird geprüft, ob es sich um CHR\$(10) handelt. Falls ja, wird der Code im Akku in 0D (CHR\$(13)) verwandelt. Ein paar Bytes zuvor wird FF (CHR\$(12)) verwandelt: Der Interpreter springt nach 0386, wo lauter LFs daraus werden. Das geschieht durch den Aufruf einer weiteren Routine, die ich aber hier nicht näher erläutern will, um den Rahmen dieses Beitrags nicht zu sprengen. Im Druckertreiber schließlich, wo LPRINT letztendlich immer landet, wird LPRINT CHR\$(0) in die Statusabfrage umgewandelt, die übrigen o. g. Codes werden dort ebenfalls undefiniert.

Eine wirksame Abhilfe ist sehr simpel. Nahezu alle Bearbeitungsroutinen der BASIC-Befehle landen früher oder später im frei programmierbaren RAM oberhalb 3FFF. In der Regel wird dort ein sog. DOS-Vektor angesprochen, denn je nach angeschlossener Peripherie werden die Befehle unterschiedlich behandelt. Ein einleuchtendes Beispiel wäre derUSR-Befehl, der in Level 2 eine andere Syntax hat als unter Disk-BASIC. Der Plattenquirl lädt in diesen unteren RAM-Bereich bestimmte Werte, die die Bearbeitungsadressen der Befehle bei angeschlossener Disco darstellen.

Hier können wir eingreifen. Die im folgenden beschriebenen Manipulationen laufen nur unter Level 2. Da meine Quirle z. Z. in Reparatur sind, kann ich eine Version für Disk-BASIC oder beides nicht anbieten. Mit G-DOS bzw. NEWDOS-80 oder H-DOS müßte es auch klappen, denn diese Betriebssysteme lassen die Adresse im unteren RAM in Frieden, wo das hier vorgestellte Programm eingreift.

In diese Adresse, nämlich 41C1, wird der Sprungbefehl in die eigene Bearbeitungsroutine eingeschrieben: JP F000. Der RET-Befehl, der zuvor dort stand, ist damit überschrieben. Diese eine Routine ist für alle oben bezeichneten Fälle ausreichend, denn bei der Auswertung eines CHR\$-Ausdrucks wird immer dorthin verzweigt, bevor der Druckertreiber und die anderen genannten Teile des Interpreters angesprochen werden. Das bedeutet, daß unser Programm ein kleiner Treiber für Codes bis CHR\$(13) sein muß. Er funktioniert folgendermaßen:

Am Anfang steht natürlich die Abfrage, ob es sich überhaupt um eine Druckerausgabe handelt. Die Bildschirmausgabe bedient sich nämlich derselben ROM-routine. Hierzu werden zunächst mit PUSH AF das zu druckende Zeichen im Akku und die Flags gerettet. Jetzt wird in 409C nachgesehen, ob dort als Druckerflag der Wert 1 abgelegt ist. Wenn ja, muß der Akku nach DEC A auf 0 stehen und die Z-Bedingung erfüllt sein. In diesem Falle erfolgt ein Sprung zur weiteren Bearbeitung, andernfalls der Rücksprung zum BASIC-Interpreter.

Wenn der Drucker angesprochen war (1 in 409C), muß nun geprüft werden, ob das zu druckende Zeichen eines der kritischen <CHR\$(14) ist (CP OEH). Falls nein, erfolgt der sofortige Rücksprung zur weiteren Bearbeitung des LPRINT-Befehls im Interpreter. War es aber einer dieser Codes, wird er ohne weitere Umschweife auf den Drucker ausgegeben (OUT (ODFH),A; für Tandy: LD (37E8H),A).

Da wir uns hier in einer Unterprogrammebene befinden, liegt die Rückkehradresse auf dem Stack. An die alte Adresse wollen wir aber nicht zurückkehren, denn die Druckerausgabe hat ja nun bereits stattgefunden. Also wird sie mit POP BC kurzerhand vom Stack "heruntergeholt" (in Wirklichkeit wird der Stapelzeiger bzw. Stackpointer auf die nächsten 2 Bytes auf dem Stack gerichtet). Es folgt noch einmal POP BC. Das Registerpaar BC ist nämlich im Interpreter vor dem Anspringen unserer Routine auf den Stack gerettet worden. Da wir nicht zur alten Routine zurückkehren, wo der POP stattgefunden hätte, müssen wir ihn eben selber vollziehen. Mit RET geht es schließlich zurück in den Interpreter, wo der nächste BASIC-Befehl bearbeitet werden kann.

Den Effekt dieser kleinen Routine kann man mit einem kleinen BASIC-Einzeiler (hier der Deutlichkeit wegen als Zweizeiler) überprüfen:

```
10 LPRINTCHR$(27)"K"CHR$(0)CHR$(1);
20 FORI=0TO255:LPRINTCHR$(I);:NEXT
```

Direkt darunter steht gleich das Resultat nach RUN. In Zeile 10 wird auf dem Drucker die Einzelpunktgraphik für 256 Punkte initialisiert (wer den Gemini hat, versteht diese Zeile; ich möchte sie jetzt nicht erläutern). Wie man sieht, wird CHR\$(0) mit LPRINT ausgegeben, was bisher nicht möglich war. In Zeile 20 folgen nun alle Codes von 0 - 255. Wie man an dem regelmäßigen Muster erkennt, wurde sie alle unverändert ausgedruckt, und zwar auch mit LPRINT.

Einen Haken hat die Routine: Auch der LLIST-Befehl passiert den DOS-Vektor an 409C. Er verträgt sich mit dieser Erweiterung offenbar nicht, wie das Listing des BASIC-Testprogramms zeigt:

```
10 LPRINTCHR$(27)"K"CHR$(0)CHR$(1);20 FORI=0TO255:LPRINTCHR$(I);:NEXT
```

Der Zeilenvorschub vor der BASIC-Zeile 20 ist unterblieben. Ohne Floppies ist mir jedoch die EDTASM-Cassettreie zu albern, deshalb möchte ich darauf verzichten, eine Version anzubieten, die auch dieses Problem noch löst. Vielleicht bietet jemand von euch im nächsten Info ein entsprechendes Programm an!?

Arnulf Sopp

F000	00100	ORG	0F000H	
F000 F5	00110	CHECK	PUSH	AF ;Akku und Flags retten
F001 3A9C40	00120	LD	A, (409CH)	;Ausgabeflag laden
F004 3D	00130	DEC	A	;Akku -1
F005 2B02	00140	JR	Z,LPRINT	;0, falls Druckerausgabe
F007 F1	00150	POP	AF	;nicht Dr., Reg. restaur.
F008 C9	00160	RET		;und normal weitermachen
F009 F1	00170	LPRINT	POP	AF ;Akku restaurieren
F00A FE0E	00180	CP	OEH	;einer der krit. Codes?
F00C D0	00190	RET	NC	;norm. weiter, falls nein
F00D D3DF	00200	OUT	(ODFH),A	;Code auf Dr. ausgeben
F00F C1	00210	POP	BC	;Stack korrigieren (CALL)
F010 C1	00220	POP	BC	;BC restaurieren
F011 C9	00230	RET		;erledigt
	00240			
41C1	00250	ORG	41C1H	;DOS-Vektor
41C1 C300F0	00260	JP	CHECK	;dorthin umleiten
	00270			
0000	00280	END		
00000				Fehler ges.

Luidger Röckrath

Der geknackte TRS-80

Über den internen Aufbau von Basic-Interpretern legen die Hersteller verbreiteter Tischcomputer meist den Mantel des Schweigens. So erfordert es schon einige Arbeit, diese Nuß zu knacken. Die Ergebnisse dieser Arbeit (über 500 Adressen und einiges mehr) sind im folgenden dargestellt.

Im Franzis-Sonderheft „Mikrocomputer-Anwendungen“ erschienen für mehrere 6502-Computer RAM/ROM-Listings [1], d. h. Aufstellungen aller wichtigen Adressen sowohl für Programmspeicher als auch für Datenspeicher. Dies ist erforderlich, weil die meisten Hersteller sich über diese Dinge ausschweigen, obwohl sie zur optimalen Ausnutzung des Computers für den Anwender aus verschiedenen Gründen notwendig sind [2].

Erstens für den Basic-Programmierer, der Programme für andere Computer adaptieren muß und dazu z. B. die Tastaturadressen benötigt, aber auch für Programmierer, die es entgegen allen Hindernissen nicht sein lassen können, ihren Computer in Maschinensprache zu programmieren, sind besonders die Adressen der Ein-/Ausgaberroutinen von Interesse. Und schließlich für jene Unverbesserlichen, die meinen, ihr Basic sei noch nicht gut genug und könnte durch einige zusätzliche Befehle sinnreich erweitert werden.

Das RAM/ROM-Listing für den TRS-80 enthält für alle drei Gruppen etwas: für den Basic-Programmierer alle wichtigen I/O-Adressen (einschließlich aller Tastaturadressen), für die passionierten Maschinenprogrammierer alle wichtigen Unterprogramme, die sie zum Anpassen fremder oder zum Entwickeln eigener Programme benötigen. Um es dieser Gruppe besonders leicht zu machen, enthält die Tabelle eine genaue Beschreibung, was das Unterprogramm eigentlich tut und welche Register verändert werden. Die letzte Gruppe sei auf den vorletzten Abschnitt dieses Artikels verwiesen, in dem ausführlich die zahlreichen Mög-

lichkeiten des Eingriffs in den Basic-Interpreter erläutert sind.

Einige grundsätzliche Bemerkungen

Wer schon mal in der Tabelle gestöbert hat, der wird sich gewundert haben, daß nur die Beeinflussung der 8080-Register angegeben ist, obwohl der TRS-80 einen Z80 als Zentraleinheit beinhaltet, der doch erheblich mehr Register hat. Des Rätsels Lösung ist ebenso einfach wie enttäuschend: Der TRS-80 arbeitet mit einem 8080-Basic. Man hat sich nur die Mühe gemacht, die Sprünge, wenn möglich, in relative Sprünge umzuwandeln. Lediglich die systemspezifische Ein-/Ausgabe ist im Z80-Code programmiert; dort wird auch einmal das X-Indexregister benutzt, aber zuvor gerettet. So stehen der ganze Zweitregistersatz und die Indexregister für Erweiterungen zur Verfügung.

Das Level-2-Basic des TRS-80 ist ein Microsoft-Basic, d. h. der Programmtext wird nicht so abgespeichert wie er eingegeben wurde, sondern zuvor in einen Zwischencode (1 Byte pro Befehl, zwischen hex 80 und FA) umgewandelt. Diese Zwischencodes sind in [3] auf Seite E/1 aufgelistet. Bei der Programmausführung muß jetzt also nicht mehr das ganze Befehlswort erkannt und dann aus einer Tabelle eine Adresse herausgesucht werden, sondern der Zwischencode dient als Zeiger auf eine Tabelle, die die Ansprungsadressen der Befehle und Funktionen enthält. So nimmt die Interpretation des Programmes nicht mehr so viel Zeit in Anspruch wie z. B. beim Level-1-Basic. Nach der gleichen Methode

arbeiten die meisten handelsüblichen Interpreter (PET, Apple usw.).

Unterschiedliche Versionen

Das RAM/ROM-Listing wurde von einer Version (V.2) des Level-2-Basic erstellt, die Mitte 1980 verkauft wurde. Bei dieser Version sind der READ/DATA-Fehler [4] und das Prellen der Eingabetastatur ausgemerzt. Zum Vergleich wurde eine ältere Version (V.1) herangezogen, welche die beiden oben genannten Fehler noch hat. Die Unterschiede der V.2 gegenüber der V.1 seien im folgenden dargestellt:

1. Der Tastendruck SHIFT → wird nicht mehr erkannt aufgrund einer Änderung in der Tastaturdecodierungstabelle. (SHIFT → hat nun die Funktion einer CONTROL-Taste.)
2. Zugunsten der Entprellung wurden die Texte „RADIO SHACK LEVEL 2 BASIC“ und „MEMORY SIZE“ zu „RS L2 BASIC“ und „MEM SIZE“ gekürzt. Dies hat noch eine Änderung der Zeiger zur Folge.
3. Um beim Laden von der Kassette die Fehlerträchtigkeit zu erniedrigen, wurden zwei Schleifen geringfügig verlängert.
4. Einige kleine Änderungen am PRINT-Befehl, die bewirken, daß
 - a) mehrere @ in einer PRINT-Anweisung möglich sind,
 - b) nicht mehr zwischen @ und SHIFT @ unterschieden wird,
 - c) das Argument von TAB nicht mehr modulo 64, sondern modulo 128 vorbehandelt wird.
5. Der READ/DATA-Fehler wurde behoben.
6. Durch eine Änderung bei CLOAD ist es nicht mehr möglich, eine Kassettenrecorder-Nummer anzugeben (CLOAD #-1, „A“ führt zu Fehlermeldung).

Wie von Radio Shack bestätigt, existieren auch andere Versionen, bei denen z. B. nur der READ/Data-Fehler ausgemerzt ist. Bei der Programmentwicklung sollten diese Unterschiede beach-

tet werden, um die Kompatibilität zu wahren.

Eingriff in den Interpreter

Der Level-2-Interpreter ist zwar in zwei ROMs abgespeichert, und somit nicht zu ändern, aber an sehr vielen Stellen verzweigt die Programmausführung ins RAM. Dort kann leicht in den Programmablauf eingegriffen werden.

1. Alle wichtigen Ein-/Ausgaberroutinen werden über die DCBs angesprungen, die die Treiberadressen enthalten. Ändert man diese Adressen, so können leicht eigene Ein-/Ausgaberroutinen mit dem Basic-Interpreter in Verbindung treten [5], [6], [7].

2. Wichtige Unterprogramme des Basic werden über die RSTs 8 bis 20 (hex) angesprungen, deren Vektoren im RAM liegen. In praktisch allen Programmsegmenten des Basic-Interpreters werden diese Unterprogramme benötigt. Eine Analyse des entsprechenden Programmleites zeigt schnell, wie man zweckmäßig eingreifen kann. Dann wird nur noch der Vektor geändert, nach Ansprung des entsprechenden RST geprüft, ob das richtige Programmsegment aufruft und schon kann die eigene Erweiterung folgen [8].

3. Eine einfache und fast unerschöpfliche Möglichkeit des Eingriffs sind die zahlreichen Disk-Basic-Befehle und Funktionen. Diese werden erkannt; sodann wird ins RAM verzweigt, wo Sprünge zu den eigentli-

chen Programmsegmenten zur Ausführung dieser Befehle und Funktionen stehen. Diese Sprünge zu ändern und eigene neue Befehle zu implementieren, ist kein Problem. Wie [9] zeigt, muß dies noch nicht einmal dazu führen, daß die Zusammenarbeit mit dem Disk-Basic gestört ist.

4. Zur Erweiterung vorhandener Befehle des Level-2-Basic durch das Disk-Basic (z. B. PRINT #, was im Disk-Basic zum Schreiben auf Diskette dient), rufen diese Befehle Unterprogramme im RAM auf. Normalerweise sind diese Adressen mit Return-Befehlen (C9) belegt, aber sie können leicht durch entsprechende Sprünge ersetzt werden. Auch hier analysiert man zweckmäßigerweise das zu erweiternde Programmsegment und wird bald herausfinden, wie man günstig eingreifen kann. Man muß beachten, daß eine Adresse unter Umständen aus verschiedenen Programmteilen aufgerufen wird. Nun schnell den TRS-80 angeschaltet und an die Arbeit! Wem das hier alles noch zu abstrakt ist, der möge sich einmal [7], [8] und [9] anschauen, drei Beispiele für die ersten drei Methoden. Sollte es jemandem an Ideen erman- geln, auch dem kann geholfen werden. [10] und [11] dürften für die nächsten Feierabende ausreichen.

Das RAM/ROM-Listing

Das RAM/ROM-Listing enthält über 500 ROM-, RAM- und I/O-Adressen. Bei wichtigen Unterprogrammen ist

angegeben, welche Register verändert werden (xx). Ist in dieser Spalte ein konkreter Zahlenwert angegeben, so bedeutet dies, daß nach der Rückkehr von diesem Unterprogramm das entsprechende Register immer diesen Wert beinhaltet. Das Register HL dient bei der Ausführung von Basic-Programmen als Zeiger auf den Programmtext. Hat es diese Funktionen, so ist ein P in der entsprechenden Spalte zu sehen.

Literatur

- [1] Martin, R.: Smode, D.: ROM und RAM in PET und CBM. Franzis-Sonderheft „Mikrocomputer-Anwendungen“.
- [2] Feichtinger, H.: Meine Meinung... FUNKSCHAU 1979, Heft 24, Seite 1418.
- [3] Radio Shack: Level II Basic Reference Manual.
- [4] Neske, W.: TRS-80 - ein Computer mit kleinem Fehler. Franzis-Sonderheft „Programme für Kleincomputer und Taschenrechner“.
- [5] Duddek, G.: Ein preiswerter Drucker für den TRS-80. Franzis-Sonderheft „Hobbycomputer 2“.
- [6] Hofer, R.: TRS-80-Treiberprogramm für Schreibmaschinen-Drucker FUNKSCHAU 1980, Heft 7, Seite 91.
- [7] Röckrath, L.: Repeatfunktion für den TRS-80 (in Vorbereitung).
- [8] Röckrath, L.: RESTORE N für den TRS-80 (in Vorbereitung).
- [9] Röckrath, L.: Double-Precision-Funktionen (in Vorbereitung).
- [10] Feichtinger, H.: Super-Basic für AIM-65 und PC-100. FUNKSCHAU 1980 Heft 23, Seite 103.
- [11] Klein, R. D.: S-100-System mit Basic und schnellem Kassettentransfer. Franzis-Sonderheft „Hobbycomputer 1“.

ROM- und RAM-Adressen

ROM-Adressen Level-II-Basic-Interpreter Initialisierung und Ein-/Ausgabe					A	F	BC	DE	HL
0000	0000	00000	00000	RST 0, Basic-Kaltstart					
0008	0008	00008	00008	RST 08, Basic-UP s. 1C96					
0010	0010	00016	00016	RST 10, Basic-UP s. 1D78					
0018	0018	00024	00024	RST 18, Basic-UP s. 1C80					
0020	0020	00032	00032	RST 20, Basic-UP s. 25D9					
0028	0028	00040	00040	RST 28, wird bei Drücken der Break-Taste angespr.					
002B	002B	00043	00043	INCH1-Ansprung (über DCB), Tastaturabfrage: ASCII-Code gedruckter Taste in A, wenn keine Taste gedrückt ist, A=0			xxxx	4015	
0030	0030	00048	00048	RST 30, unbenutzt					
0033	0033	00051	00051	OUTCH-Ansprung (über DCB), Ausgabe des Akkuinhaltes auf den Bildschirm			xx	401D	
0038	0038	00056	00056	RST 38H, unbenutzt					
003B	003B	00059	00059	PRINT-Ansprung (über DCB), Ausgabe des Akkuinhaltes auf den Drucker			xxxx	4025	
0049	004F	00073	00079	INCH2, wie INCH1, es wird aber erwartet, bis eine Taste gedrückt wird			xxxx	4015	

					AF	BC	DE	HL
0050	005F	00080	00095	Tabelle für die Tastaturdecodierung				
0060	0065	00096	00101	DELAY, Zeitschleife 14,6 µs × BC	0044	0000		
0066	0074	00102	00116	Reset-Ansprung (NMI-Vektor). Wenn Floppy angeschlossen, Basic-Kaltstart, sonst Warmstart				
0075	0104	00117	00260	Basic-Initialisierung:				
0075	008D	00117	00141	DCBs, RST-Vektoren und I/O-Buffer einrichten				
008E	00AB	00142	00171	Disk-Basic-Zeiger initialisieren (werden beim Laden des Disk-Basic geändert)				
00AC	00F8	00172	00248	MEM SIZE anfordern oder selbst Speicherende suchen				
00F9	0104	00249	00260	NEW „R/S L2 BASIC“ ausdrucken und Sprung zur Hauptschleife				
0105	010D	00261	00269	Text „MEM SIZE“				
010E	011B	00270	00283	Text „R/S L2 BASIC(CR)“				
011C	012C	00284	00300	Tastaturentprellung				
012D	0131	00301	00305	L3-Error				
0132	0132	00306	00306	POINT A=00 } Ansprung der drei Grafikbefehle				
0135	0135	00309	00309	SET A=80 } und Setzen des Flags (A).				
0138	0138	00312	00312	RESET A=01 }				
013A	017D	00314	00381	Aus den Koordinaten werden die Adresse im Bildschirmram und die Maske errechnet				
017E	019C	00382	00412	In Abhängigkeit von dem Flag wird ein POINT, SET oder RESET ausgeführt				
019D	01C8	00413	00456	INKES-Funktion				
01BC	01C8	00444	00456	Leerstring nach X				
01C9	01D2	00457	00466	CLS-Befehl, Bildschirm wird gelöscht	1Fxx			
01D3	01D8	00467	00472	RANDOM-Befehl (als zufällige Größe wird das R-Register benutzt)				
01D9	01F7	00473	00503	Impuls auf Kassette ausgeben	xxxx	00		FC00
01F8	01FD	00504	00509	Kassettenrecorder abschalten	xxxx			
01FE	021D	00510	00541	Kassettenrecordernummer decodieren und Kassettenrecorder einschalten	xxxx	xxxx	xxxx	P
0215	021D	00533	00541	Kassettenrecorder einschalten	xxxx			
021E	0220	00542	00544	Bit 7 von Port (FF) zurücksetzen	xxxx			FC00
0221	022B	00545	00555	{403D} & H v L nach {403D} und zum Port FF	xxxx			
022C	0234	00556	00564	Stern in rechter, oberer Ecke des Bildschirms umschalten	xxxx			
0235	0260	00565	00608	Byte von Kassette lesen (wird im Akku übergeben)	xxxx			
0241	0260	00577	00608	Bit (b) von Kassette lesen (2 · A + b ÷ A)	xxxx			FC00
0251	0283	00609	00643	Byte (im Akku) zweimal auf Kassette aufzeichnen				
0264	0283	00612	00643	Byte (im Akku) auf Kassette aufzeichnen				
0264	0292	00644	00658	Kassette einschalten, 255 mal 0 und A5 aufzeichnen	xxxx	xxxx	xxxx	P
0293	02A8	00659	00680	Kassette einschalten, auf A5 warten und Sternchen auf Bildschirm setzen	2Axx	xxxx	xxxx	P
02A9	0329	00681	00809	SYSTEM-Befehl				
02B2	02B2	00650	00690	Ansprung				
02CE	0313	00718	00787	Objektfile von Kassette laden				
0314	031C	00788	00796	Wort (in HL) von Kassette laden (L, H)	xxxx			xxxx
031D	0329	00797	00809	Ansprung des Objektfile oder irgendeiner anderen Adresse				
032A	0347	00810	00839	Ausgabe des Akkuinhaltes auf Bildschirm (00), Drucker (01) oder Kassette (80) in Abhängigkeit von (409C)	xx			
033A	0347	00826	00839	OUTCH2, wie OUTCH, zusätzlich Cursorposition nach (40A6)	xx			
0348	0357	00840	00855	Position des Cursors in der Bildschirmzeile nach A	xxxx			
0358	0360	00856	00864	INCH3, wie INCH1	xxxx			
0361	0383	00865	00899	INLINE, Eingabe von max. 240 Zeichen in den I/O-Buffer mit allen Cursorfunktionen. Wenn Break gedrückt wird, Rückkehr mit gesetztem Carry-Flag, bei Drücken von Clear wird der Bildschirm gelöscht und die Eingabe beginnt von vorne. HL enthält die Bufferanfangsadresse -1	xxxx		401D	xxxx
0384	038A	00900	00906	INCH4, wie INCH2	xxxx			
038B	039B	00907	00923	CR auf Drucker ausgeben, wenn Druckkopf nicht in Position 0, (409C) auf 0 setzen	xxxx			
039C	03C1	00924	00961	Zeichen in A auf Drucker ausgeben, Zeichenzähler (409B) incrementieren und bei 0A, 0C und 0D auf 0 setzen				
03C2	03E2	00962	00994	Routine zum Aufruf der Ein-/Ausgabe über die DCBs. (DCB-Typ in B, DCB-Adresse in DF.)				
03E3	0457	00995	01111	Tastaturabfrage und Decodierung (Ansprung nur über INCH1-4, da nur dann die Register gerettet werden, siehe dort)				
0458	058C	01112	01420	Bildschirmausgabe (Ansprung nur über DCB, da nur dann Register gerettet werden und die Cursoradresse übergeben wird, siehe OUTCH1)				
04B8	058C	01208	01420	Bildschirm-Steuerbefehle				

					AF	BC	DE	HL
04B8	04BC	01208	01212	Cursor ON				(0E)
04BD	04BF	01213	01215	Cursor OFF				(0F)
04C0	04CD	01216	01229	Cursor HOME (64 cpl)				(1C)
04CE	04D9	01230	01241	Cursor BACKSPACE				(08)
04DA	04EB	01242	01259	Cursor ↑				(18)
04E7	04E8	01255	01259	Cursor →				(1A)
04EC	04F5	01260	01269	Cursor ↓				(19)
04F1	04F5	01265	01269	Cursor ←				(1B)
04F6	0505	01270	01285	32cpl				(17)
0506	0540	01286	01344	Auswertung der Steuerbefehle				
0541	0563	01345	01379	Zeichen auf Bildschirm setzen und Scroll, wenn nötig				
0564	0572	01380	01394	New-Line	(0A-0D)			
0573	058C	01395	01421	Clear to end of line				(1E)
057C	058C	01404	0142L	Clear to end of frame				(1F)
058D	05D0	01421	01488	Drucker-Treiber				
05D1	05D8	01489	01496	Drucker bereit?	xxxx			
				J,Z=1.				
05D9	0673	01497	01651	Unterprogramm für INLINE wie INLINE, maximale Anzahl der Zeichen in B, Bufferanfangsadresse in HL Bei Rückkehr Anzahl der eingegebenen Zeichen in B	xxxx	xxxx	401D	
0674	06CB	01652	01739	Wenn Break gedrückt ist, Sprung zur Basic- Initialisierung, sonst Testen ob Floppy vorhanden Wenn ja, DOS-Laden und starten				
06CC	06CC	01740	01740	Adresse für Rücksprung von SYSTEM-Programmen zum Basic (Basic-Warmst.)				
06D2	0707	01746	01799	RST-Vektoren, DCBs usw. (werden bei Initialisierung ins RAM übertragen (4000-4035))				
Arithmetik								
0708	07F7	01800	02039	X + 1 ÷ X	xxxx	xxxx	xxxx	xxxx
0710	07F7	01808	02039	(HL...) + X ÷ X	xxxx	xxxx	xxxx	xxxx
0713	07F7	01811	02039	BCDE - X ÷ X	xxxx	xxxx	xxxx	xxxx
0716	07F7	01814	02039	BCDE + X ÷ X	xxxx	xxxx	xxxx	xxxx
0778	077C	01912	01916	0 ÷ X	0044			
07A8	07B1	01960	01969	Rundung				
07B2	07B6	01970	01974	OV-Error				
07B7	07C2	01975	01986	Festkommaaddition: (HL...) + CDE ÷ CDE	xxxx	xx	xxxx	+2
07D7	07F7	02007	02039	CDE um A-Bitpositionen nach rechts schieben	00xx	xxxx	xxxx	xx
07F8	07FB	02040	02043	Konstante 1!				
07FC	0808	02044	02056	Konstanten für LOG-Reihe				
0809	0896	02057	02189	LOG(X) ÷ X	xxxx	xxxx	xxxx	xxxx
0841	0896	02057	02189	X · ln2 ÷ X	xxxx	xxxx	xxxx	xxxx
0847	0896	02119	02189	X · BCDE ÷ X	xxxx	xxxx	xxxx	xxxx
0897	0906	02199	02311	X / 10 ÷ X	xxxx	xxxx	xxxx	xxxx
08A2	0906	02211	02310	BCDE / X ÷ X	xxxx	xxxx	xxxx	xxxx
0907	093D	02311	02365	Vorbereitung der Exponenten und Vorzeichen bei Multiplikation und Division				
093E	0954	02366	02388	X · 10 ÷ X	xxxx	xxxx	xxxx	xxxx
0955	0963	02389	02403	Text X (für Single und Double Precision)	xxxx			
				Wenn X = 0, Z.P				
				Wenn X < 0, C.S				
				Wenn X > 0, keine				
0964	0976	00404	02422	Akku in Fließkommazahl (in X) umformen	xxxx	xxxx	xxxx	xxxx
0977	0989	02423	02441	ABS(X) ÷ X	xxxx	xxxx	xxxx	xxxx
0982	0989	02434	02441	- X ÷ X	xxxx			4123
098A	0993	02442	02451	SGN(X) ÷ X	xxxx			xxxx
0994	09A3	02452	02467	Test X, auch für Integer (siehe oben 0955) bei String in X TM-Error	xxxx			xxxx
09A4	09B0	02468	02480	X ÷ (SP)				xxxx
09B1	09BE	02481	02494	(HL...) ÷ BCDE ÷ X	xxxx	xxxx		+4
09B4	09BE	02484	02494	BCDE ÷ X	xxxx			xxxx
09DF	09CA	02495	02506	X ÷ BCDE	xxxx	xxxx		4125
09C2	09CA	02498	02506	(HL...) ÷ BCDE	xxxx	xxxx		+4
09CB	09DE	02507	02526	X ÷ (HL...)	xxxx	00	4125	+4
09D2	09DE	02514	02528	(HL...HL+(40AF)) ÷ (DE...)	xxxx	00	+4	+4

Fortsetzung folgt

Im letzten Info habe ich angedroht, Maschinenspracheprogramme vorzustellen, falls Interesse dafür besteht. Das ist der Fall, also los!

Als ich in die abgebildete junge Dame noch schrecklich verliebt war, digitalisierte ich sie auf eine Art und Weise, die ich vielleicht in einem anderen Info beschreiben werde und gab ihren traurigen Rest dem Computer ein. Das Resultat seht ihr am Ende dieses Artikels. Die einzelnen Graphik-Codes hatte ich den vorbereiteten Stringvariablen mit einem speziellen Programm einverleibt, das den Anfang des hier gelisteten Programms darstellte (deshalb fängt es erst bei Zeile 520 an). Die Anzeige geschah zunächst mit der kreuzbraven BASIC-Befehlsfolge

```
FOR I = 0 TO 4 : PRINT A$(I); : NEXT
```

Es sollte aber gerne schneller gehen. Das war nämlich gerade in der Zeit, als ich die ersten zaghaften Versuche in der Maschinensprache unternahm. Das Ergebnis ist das Listing im Listing ab Zeile 10001. Da es ein paar Merkwürdigkeiten enthält, auf die ich noch eingehen werde, folgt hier zunächst eine vereinfachte Fassung, die man so allerdings nicht in einer Stringvariablen unterbringen könnte. Vergewissert euch dann bei Unklarheiten an diesem Listing:

```

CALL      2687
LD        A,5
LD        DE,15360
LOOP:    LD      BC,205
        LDIR
        LD      BC,13
        ADD    HL,BC
        DEC    A
        JR    NZ,LOOP
        JP    73

```

Diese Fassung hat außerdem den Vorteil, syntaktisch richtig zu sein, was man von meinem damaligen Machwerk nicht sagen kann. Jeden Versuch, es einem Editor-Assembler so einzugeben, würde er mit einer ganzen Latte von Fehlermeldungen quittieren. Ich habe es aber absichtlich nicht in seiner Gestalt und seiner Programmlogik nachträglich geschönt, weil ich mich nicht geniere, zu zeigen, daß ich einmal ziemlich unbeholfen anfing.

Zunächst aber zum BASIC-Programm. Es beginnt mit der Zuweisung der Stringvariablen, die die Graphik bereits fertig enthalten. Was im Listing wie BASIC-Befehle aussieht, sind die einzelnen Graphikzeichen. Deren Codes werden nämlich vom Interpreter wegen des gesetzten Bits 7 als BASIC-Tokens mißverstanden. Ähnlich ist in Zeile 10000, wo das Maschinenprogramm zur schnellen Anzeige in der Stringvariablen P\$ steht. Das Programm, das theoretisch alle Codes von 0 - 255 enthalten kann, wird bei Werten über 80h (128d) ebenfalls als Folge von BASIC-Befehlswörtern angezeigt. U. a. sehen wir dort übrigens den "Befehl" IsA (der gar keiner ist), über den ich bei anderer Gelegenheit schreiben werde.

In den Zeilen 1010 - 1030 wird schließlich mit Hilfe der VARPTR-Funktion dem Interpreter in der bekannten Weise mitgeteilt, wo er das Maschinensprache-Unterprogramm findet (P) und wo das erste Byte der Graphik steht (X). Mit demUSR-Befehl wird das Programm nun angesprochen.

Es hat keinen Sinn, diesen BASIC-Text eintippen zu wollen. Zwischen Anführungszeichen werden die Befehlswörter nur als Folge sinnloser Buchstaben übernommen, nicht als Tokens.

Und jetzt zum Maschinenprogramm. Daß man dasUSR-Argument (hier X) als Parameter an das Maschinenprogramm über die ROM-Routine ab 0A7Fh (2687d) übergeben kann, wonach es im Registerpaar HL bereitliegt, steht im BASIC-Manual. Hier wird die Routine mit CALL aufgerufen; das ist so ziemlich dasselbe wie GOSUB in BASIC. Nun wird der Zähler für die fünf Stringvariablen eingerichtet. Der Akkumulator (oder einfach Akku), eines der Register der CPU, wird mit dem Befehl LD A,5 mit dem Wert 5 geladen. In BASIC entspricht das der LET-Anweisung.

Und jetzt wird's ein bißchen komplizierter. Es gibt einen sehr leistungsfähigen Maschinenbefehl, der einen zusammenhängenden Block von Daten (Bytes) an eine andere Stelle transferieren kann: LDIR. Er verlangt drei Vorgaben: Das Registerpaar BC (die Register B und C) enthält die Anzahl der zu übertragenden Bytes. In HL steht die Adresse des ersten Bytes in diesem Block. DE hält die Adresse, an die der Block bewegt werden soll. In unserem Fall ist BC = 205 (so viele Bytes enthält je eine Stringvariable), HL hält den Wert aus der BASIC-Zeile 1020 (erstes Byte von A\$(0)) und DE ist = 3C00h (15360d) - das ist der Anfang des Bildschirmspeichers, also die linke obere Ecke der Bildröhre. Mit LDIR wird nun der Inhalt von A\$(0) auf die ersten 205 Stellen des Bildschirms übertragen.

Der aufmerksame Leser hat bemerkt, daß die obigen Zahlen nicht im Listing erscheinen, und daß die zusätzlichen Befehle INC DE und INC B dort stehen. Das hat eine Bewandnis, die eigentlich schon einen gesonderten Artikel füllen könnte. Aber ich will es kurz machen: Wenn der Interpreter im Programmtext eine 0 antrifft, versteht er sie als Markierung für das Zeilenende. Die Rede ist wohl gemerkt von ASCII 0, nicht von der Ziffer "0" (= ASCII 30h = 48d). Die vier folgenden Bytes versteht er dann als Zeiger auf die übernächste Zeile und die Nummer der folgenden Zeile. Enthält demnach die Zeile 10000 im Maschinenprogramm eine logische 0, dann wird der Rest vollkommen falsch verstanden.

Deshalb durfte ich DE nicht mit 3C00h (15360d) laden. Ich nahm stattdessen 3BFFh (15359d) und erhöhte mit dem folgenden Befehl INC DE auf 3C00h. Dann wird der Inhalt von E ebenfalls = 0, aber diese Null erscheint nicht im Programm. Dasselbe passierte mit dem Zähler BC: Sein MSB, also das Register B, wurde zunächst mit FFh (255d) geladen und anschließend durch den Befehl INC B auf 0 inkrementiert. Denn für die CPU ist 255 + 1 = 0. So war am Ende der Inhalt von BC = 00CDh (205d), die Anzahl der Bytes pro Stringvariablen.

Nachdem nun die Zeiger auf die Quelle (HL) und das Ziel (DE) und der Zähler (BC) die korrekten Werte enthalten, wird mit LDIR das Teilbild in A\$(0) schlagartig angezeigt. Zwischen diesem Teilbild und der nächsten Portion in A\$(1) liegt jedoch noch allerhand BASIC, das übersprungen werden muß: Da sind die Abführungszeichen von A\$(0) (die im Listing nicht erscheinen), die Null als Markierung für das Zeilenende, der Zeiger auf die übernächste Zeile, die Nummer der folgenden Zeile und der Text 'A\$(1)="'. Macht zusammen 13 Bytes, die dem Zeiger HL hinzuaddiert werden müssen. Das geschieht mit ADD HL,BC. Aber da ist das alte Problem: Das Registerpaar BC muß mit 0000h (13d) geladen werden. Die 0 als MSB von BC ist aber mitten in einer BASIC-Zeile nicht zulässig, wie oben beschrieben. Da zieht wieder der alte Trick: Durch INC B wird aus FF00h (65293d) der Wert 0000h (13d). Jetzt kann auch A\$(1) - zack! - auf den Bildschirm übertragen werden.

Das Programm muß natürlich auch wissen, wann es fertig ist. Gleich beim zweiten Schritt haben wir den Akku als Zähler der Variablen mit dem Wert 5 geladen. In Zeile 10021 wird er dekrementiert, also um 1 vermindert. Ist er schon auf 0 geschrumpft? Falls ja, dann ist das Zero-Flag im Register F gesetzt. Die CPU setzt dort nämlich bei sehr vielen Befehlen ganz bestimmte Flags (bzw. löscht sie, je nach dem), wenn die ihnen zugordne-

ten Bedingungen erfüllt sind. Bei diesen Flags handelt es sich übrigens um ganz gewöhnliche Bits eines ganz gewöhnlichen Bytes in diesem Register. Der Befehl JR NZ,LOOP (jump relatively on non-zero) wird nur befolgt, wenn das Z-Flag noch nicht gesetzt ist. In diesem Falle springt das Programm zurück an den Beginn der Schleife, der willkürlich den Namen LOOP erhielt. Ich hätte ihn auch SEPPEL taufen können.

Irgendwann ist der Akku = 0, die 5 Stringvariablen sind auf den Bildschirm übertragen. Dieser komplizierte Vorgang, über den ihr seit drei Minuten lest, hat einen winzigen Bruchteil einer Sekunde gedauert. Tatsächlich steht die Graphik schlagartig auf dem Bildschirm, nachdem nach RUN die NEW LINE-Taste gedrückt wurde.

Damit sind wir aber noch nicht ganz am Ende. USR stellt einen Unterprogrammaufruf dar, ähnlich GOSUB in BASIC. Doch wo ist RETURN? In Assembler heißt der entsprechende Befehl RET. Nahezu alle einzelnen Abschnitte des Interpreters werden mit einem RET abgeschlossen, denn er ist ein gigantisches Sammelsurium von Unterprogrammen, die von irgendwoher mit CALL aufgerufen werden. Der Befehl JP in Zeile 10024 entspricht einem GOTO. An der Adresse 0049h (73d) befindet sich ein Unterprogramm, das in einer Endlosschleife hängenbleibt, bis irgendeine Taste gedrückt wird. Kommt endlich der erlösende Tastendruck, weil der User Brigitte nicht mehr sehen mag, dann wird der Befehl RET erreicht und READY angezeigt.

Bevor ihr erlöst seid, noch ein Wort zu den komischen Dezimalzahlen links: Die verschiedenen Assembler-Befehle setzen sich aus einem oder mehreren Bytes zusammen. Diese Zahlen enthalten deren Werte. Sie könnten in dieser Form beispielsweise in DATA-Zeilen gepackt werden.

Dieses war sozusagen mein Jungfernprogramm, die erste Maschinenroutine, die ich je schrieb (auch wenn es vielleicht die 5. Version ist). Mit diesem Problem, eine Graphik anzuzeigen, fing alles an. Ich wünsche den Anfängern unter euch viele Erfolgserlebnisse. Ihr habt meine ganze Sympathie.

Arnulf Sopp



```

520 A$(0)="USINGUSINGUSINGUSINGUSINGUSINGNEXTSET
      OUTOUTCLEARDEF      FIELDRESTORE CLSPUTTOUSINGUSINGU
SINGUSINGLIST USINGUSINGLPRINTCMDIFFOR      CLS
      DEFPRINTUSINGUSINGUSINGTOLISTRESTORERESTORE      R
530 A$(1)="      OUTTAB (USINGUSINGUSINGUSINGUSINGUSING
USINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSING
XTSAVEUSINGUSINGUSINGTONEWUSINGDEFDEF USINGRESUMERESET
      DATARUNLPRINTUSINGUSINGUSINGUSINGUSINGUSINGUSING
540 A$(2)=" RESETNEWTAB (DELETECLOADCLS RESETLPRINTUSINGUSINGUSI
NGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSING
INGLPRINTUSINGNEWUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGFORRES
TORE      OUTTAB (TAB (TAB (CLEARTAB (TAB (USINGUS
550 A$(3)="USINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSING
ECONTUSINGFNUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGUSINGT
OCLEARERROR      NAMEERRORLINEAUTOUSINGUSINGUSINGUSINGUSINGU
SINGUSINGUSINGUSINGTAB (TAB (FNUSINGUSINGUSINGUSINGUSINGU
560 A$(4)=" RESETUSINGREMINPUTRSETUSINGUSINGUSINGUSINGUSINGUSINGUSING
DE
FDEFTRON CLS      DEF SETSETFOR SETLETINPUTNEXTLPRINTUSINGUS
INGUSINGUSINGUSINGUSINGUSINGRESUMEFORNEWUSINGUSINGUSINGUSINGUSI
NGTO DEFCSAVEUSINGDEF RESETLPRINTUSINGUSINGTROFF      RE
1000 P$="
>IsA:
+IsAMK$DEF
IsA = FIXERRI
1005 REM
      *** DIE ZEILEN 520 - 1000 DUERFEN NICHT ***
      *** EDITIERT WERDEN. SIE ENTHALTEN IN ***
      *** A$(0-4) DIE TEILBILDER ZUR ERZEUGUNG ***
      *** DER GRAPHIK, IN P$ DAS MASCH.-PRG. ***

1010 P=PEEK (VARPTR (P$)+1)+256*PEEK (VARPTR (P$)+2);FM=P/256;FOKE16527.
INT (PM):FOKE16526. (FM-INT (PM))*256
1020 X=PEEK (VARPTR (A$(0))+1)+256*PEEK (VARPTR (A$(0))+2)
1030 U=USR (X)
10000 END'

```

```

*** ASSEMBLER-LISTING IN DEZIMAL ***

*** DAS PROGRAMM (P$ IN BASIC) ***
*** IST FREI RELOZIERBAR (KEINE ***
*** ABSOLUTEN ADRESSEN). ***

```

```

10001 205,127,10      CALL 2687      ;$-ANFANG -> HL
10004 62,5            LD A,5        ;5 TEILBILDER
10006 17,255,59      LD DE,15359   ;ANF. VIDEO-RAM
10009 13             INC DE        ;KEINE 0 IN P$
10010 1,205,255      LOOP: LD BC,65485 ;LEN(A$(I))=205
10013 4              INC B        ;KEINE 0 IN P$
10014 237,176        LDIR         ;BLOCKTRANSFER
10016 1,13,255       LD BC,65293   ;HL = HL + 13
10019 4              INC B        ;KEINE 0 IN P$
10020 9              ADD HL,BC     ;FORTS. HL + 13
10021 61             DEC A        ;NOCH EIN $ ?
10022 32,242         JR NZ,LOOP    ;JA, WIEDERH.
10024 195,73,0       JP 73        ;ENDE BEI BE-
                                      ;LIEBIGER TASTE

```

Arnulf Sopp

BÜCHER

- Heiko Reuardt: Basic: Alles über PEEK und POKE
Eine Software-Sammlung in Basic
erschienen im Franzis Verlag

Der Patrick Perschbach hat folgende Bücher zu verleihen:

- 57 BASIC-Programme (dt.)
- Advanced Level II BASIC
- Computer Programming in BASIC for everyone
- Announcing Computer Games for Business, School and Home
- Programming is Style
- Business Programs Applications

Der Günther Wagner hat folgendes Buch zu verleihen:

- TRS-80 und VG ROM-Listing für Level II (Luidger Röckrath)

So, jetzt erst mal wieder eine aktuelle Aufstellung meiner Programmanleitungen und Handbücher.

- Bedienungshandbuch Disketten-Betriebssystem
- Bedienungshandbuch G-Dos
- Bugout-Maschinensprache-Monitor
- Visicalc
- Datapro
- ZBasic 2.2
- Scipsit Textverarbeitung
- Tiny-Pascal Supersoft
- GEAF Grafik-Editor
- Musimp
- Air-Traffic-Controller

Jean Claude Wies
(und Andreas Marx)

Dann noch einiges für die Peek und Poke Ecke
 Poke 16448 Echtzeit-Uhr Zähler
 Poke 16449-51 stellt Sekunden, Minuten und Stunden der Uhr
 kann mit Peek auch zur Abfrage verwendet werden
 Poke 16452-54 dasselbe mit Jahr, Tag und Monat
 Poke 16526-27 Startadresse für System-Programme USR(n)
 über Peek auch brauchbar zur Ermittlung des
 DEFUSR-Befehls zu gebrauchen.
 Ann: Einfach Peek(16527)*256+Peek(16526), dann
 den Wert in Hex-Umrechnen und als DEFUSR=&Hxxxx
 angeben. Dieses Problem tritt öfter bei Tape-
 Programmen auf, die keinen DEFUSR Befehl
 enthalten.

Wieder darf ich im Vorwort einige Worte über dieses Thema verlieren. Zunächst ist anscheinend manchen Mitgliedern noch nicht klar, wer diese Übersetzung macht. Nun - derjenige heißt Günther Wagner.

Da auch dieses Mal die Resonanz wieder positiv war, erscheinen auch wieder 12 Seiten. Damit ist das Kapitel 4 abgeschlossen.

Zusätzlich sind noch 2 Seiten des Anhangs abgedruckt, die für den Einen oder Anderen wichtig sind.

A C H T U N G *** W I C H T I G :

In der letzten Ausgabe haben sich Fehler bzw. Ungenauigkeiten eingeschlichen:

Seite 21, letzter Absatz:
... mit der Adresse aus dem HL-Register in die Adresse aus dem DE-Register. Dann werden das HL- und DE-Register erhöht, das BC-Register wird erniedrigt.....

Seite 21, 3. Absatz:
anstatt 'DATA-Blöcke' werden 'Daten-Blöcke' verschoben

Zum VW-Verwalter, Seite 26:
Klar dürfte sein, daß nach 'START' eine Routine folgen muß.

Seite 16, letzter Absatz:
Statt 'veränderliche' sollte es besser 'verschiebliche' heißen.

Ich bitte um Entschuldigung für diese Fehler. Wer weitere Fehler findet, soll mir diese bitte mitteilen. Viel Spaß mit den neuen 12 Seiten - ich hoffe, Sie bringen Euch was!

Günther Wagner

Scotts erstes Gesetz:

Egal was schiefeht - man sieht es ihm zuerst nicht an.

Scotts zweites Gesetz:

Wenn man einen Fehler gefunden und endlich korrigiert hat, stellt sich heraus, daß die erste Version richtig war.

Folgerung:

Nachdem sich die Korrektur plötzlich als falsch herausstellte, ist es unmöglich, den Originalzustand wiederherzustellen.

USR Routine Pointer Addresses

DOS VERSION	0	1	2	3	4	5	6	7	8	9
TRSDOS 2.3 Radio Shack Model I	23415 5B77	23417 5B79	23419 5B7B	23421 5B7D	23423 5B7F	23425 5B81	23427 5B83	23429 5B85	23431 5B87	23433 5B89
TRSDOS 2.0 Radio Shack Model 2	11050 2B2A	11052 2B2C	11054 2B2E	11056 2B30	11058 2B32	11060 2B34	11062 2B36	11064 2B38	11066 2B3A	11068 2B3C
TRSDOS 1.2 Radio Shack Model III	22586 583A	22588 583C	22590 583E	22592 5840	22594 5842	22596 5844	22598 5846	22600 5848	22602 584A	22604 584C
TRSDOS 1.3 Radio Shack Model III	22632 5868	22634 586A	22636 586C	22638 586E	22640 5870	22642 5872	22644 5874	22646 5876	22648 5878	22650 587A
NEWDOS 2.1 Apparat	23316 5B14	23318 5B16	23320 5B18	23322 5B1A	23324 5B1C	23326 5B1E	23328 5B20	23330 5B22	23332 5B24	23334 5B26
NEWDOS 1.0 Apparat	22330 573A	22332 573C	22334 573E	22336 5740	22338 5742	22340 5744	22342 5746	22344 5748	22346 574A	22348 574C
DOS PLUS 3.3D Micro Systems Software	23483 58BB	23485 58BD	23487 58BF	23489 58C1	23491 58C3	23493 58C5	23495 58C7	23497 58C9	23499 58CB	23501 58CD
LDOS 5.0.1 Lobo Drives, Int'l	23415 5B77	23417 5B79	23419 5B7B	23421 5B7D	23423 5B7F	23425 5B81	23427 5B83	23429 5B85	23431 5B87	23433 5B89
ULTRADOS 4.2 Level IV Products	20992 5200	20994 5202	20996 5204	20998 5206	21000 5208	21002 520A	21004 520C	21006 520E	21008 5210	21010 5212
DBLDOS 4.23 Percom	23316 5B14	23318 5B16	23320 5B18	23322 5B1A	23324 5B1C	23326 5B1E	23328 5B20	23330 5B22	23332 5B24	23334 5B26

Disk Buffer Memory Locations

DISK OPERATING SYSTEM	VERSION	MODEL	1	2	3	4	5	6
TRSDOS Radio Shack	2.3	1	26335 66DF	26625 6801	26915 6923	27205 6A45	27495 6B67	27785 6C89
TRSDOS Radio Shack	2.0	2	27779 6C83	28613 6FCS	29447 7307	30281 7649	31115 798B	31949 7CCD
TRSDOS Radio Shack	1.2	3	25812 64D4	26172 663C	26532 67A4	26892 690C	27252 6A74	27612 6BDC
TRSDOS Radio Shack	1.3	3	26232 6678	26592 67E0	26952 6948	27312 6AB0	27672 6C18	28032 6D80
NEWDOS Apparat	2.1	1	25973 6575	26263 6697	26553 67B9	26843 68DB	27133 69FD	27423 6B1F
NEWDOS/80 Apparat	1.0	1	26347 66EB	26648 6818	26949 6945	27250 6A72	27551 6B9F	27852 6QCC
DOS PLUS Micro Systems Software	3.3D	1	28053 6D95	28599 6FB7	29145 71D9	29691 73FB	30237 761D	30783 783F
DOS PLUS - TBASIC Micro Systems Software	3.3D	1	25450 636A	25996 658C	26542 67AE	27088 69D0	27634 6BF2	28180 6E14
DOS PLUS Micro Systems Software	3.3	3	28039 6D87	28585 6FA9	29131 71CB	29677 73ED	30223 760F	30769 7831
LDOS Lobo Drives, Int'l	5.0.1	1	27237 6A65	27783 6C87	28329 6EA9	28875 70CB	29421 72ED	29967 750F
ULTRADOS Level IV Products	4.2	1	25531 63BB	25821 64DD	26111 65FF	26401 6721	26691 6843	26981 6965
DBLDOS Percom	4.23	1	25973 6575	26263 6697	26553 67B9	26843 68DB	27133 69FD	27423 6B1F

Beachten Sie, daß nur der gegenwärtige Inhalt jeder Variablen gezeigt wird. Der String XY\$, welcher die Zahl 100 im MKI\$-Format (2 Bytes) enthält, wird für uns automatisch umgewandelt. Enthalten Strings nichtdarstellbare Character (ASCII-Werte kleiner 32 oder größer 191), so werden hierfür Blanks ausgegeben. Anführungszeichen am Beginn und Ende der Strings werden angegeben, damit man Blanks am Anfang und Ende eines Strings auch sieht. Da die Variable G% in Zeile 20 durch die Programm-Logik nicht definiert wurde, wird Sie auch vom AVA nicht angezeigt.

```

65000 PRINT "AKTIVE EINFACHE VARIABLEN:"
65002 ZD%=0: ZZ%=0: ZZ$="": ZZ$(3)="": ZZ$(0)=PEEK(16633):
ZZ$(1)=PEEK(16634)
65004 GOSUB 65110
65006 IF ZZ$(0)=PEEK(16635) AND ZZ$(1)=PEEK(16636) THEN 65030
ELSE GOSUB 65130
65007 GOSUB 65140: GOTO 65006
65030 PRINT "AKTIVE FELDER:"
65032 ZZ$(0)=PEEK(16635): ZZ$(1)=PEEK(16636)
65034 GOSUB 65110
65036 IF ZZ$(0)=PEEK(16637) AND ZZ$(1)=PEEK(16638) THEN RETURN
ELSE GOSUB 65130: GOSUB 65100: GOSUB 65100: GOSUB 65100:
GOSUB 65110: ZD%=ZZ$(3): ZZ%=0
65038 IF ZZ%=ZD% THEN 65040 ELSE GOSUB 65100: GOSUB 65110:
ZZ$(1)=ZZ$(0): GOSUB 65100: GOSUB 65110: ZZ$(8+ZZ%)=0:
ZZ$(5+ZZ%)=CVI(ZZ$(1)+ZZ$(0)): ZZ%=ZZ%+1: GOTO 65038
65040 ZZ$=LEFT$(ZZ$,2): ZZ$(3)="(": FOR ZZ%=ZD% TO 1 STEP -1:
ZZ$(3)=ZZ$(3)+STR$(ZZ$(7+ZZ%))
65041 IF ZZ%>1 THEN ZZ$(3)=ZZ$(3)+"," ELSE ZZ$(3)=ZZ$(3)+")"
65042 NEXT
65050 GOSUB 65140
65051 ZZ$(7+ZD%)=ZZ$(7+ZD%)+1: IF ZZ$(7+ZD%)<ZZ$(4+ZD%) THEN
65040
65052 IF ZD%=1 THEN 65070 ELSE ZZ$(7+ZD%)=0
65053 ZZ$(6+ZD%)=ZZ$(6+ZD%)+1: IF ZZ$(6+ZD%)<ZZ$(3+ZD%) THEN
65040
65054 IF ZD%=2 THEN 65070 ELSE ZZ$(6+ZD%)=0
65055 ZZ$(5+ZD%)=ZZ$(5+ZD%)+1: IF ZZ$(5+ZD%)<ZZ$(2+ZD%) THEN
65040 ELSE 65070
65060 GOTO 65040
65070 GOSUB 65100: GOSUB 65110: GOTO 65036
65100 ZZ$(0)=ZZ$(0)+1: IF ZZ$(0)=256 THEN ZZ$(0)=0:
ZZ$(1)=ZZ$(1)+1
65101 RETURN
65110 ZZ$(4)=CVI(CHR$(ZZ$(0))+CHR$(ZZ$(1))): ZZ$(3)=PEEK(ZZ$(
4)): ZZ$(0)=CHR$(ZZ$(3)): RETURN
65120 FOR ZZ%=1 TO ZZ$(2): GOSUB 65100: GOSUB 65110: ZZ$(1)=
ZZ$(1)+ZZ$(0): NEXT: IF ZZ$(3)="" THEN GOSUB 65100: GOSUB
65110
65121 IF INSTR("ZZ$ZZ$ZD%",ZZ$) THEN ZZ$=""
65122 RETURN
65130 ZZ$(2)=ZZ$(3): GOSUB 65100: GOSUB 65110: ZZ$=ZZ$(0):
GOSUB 65100: GOSUB 65110: ZZ$=ZZ$(0)+ZZ$: RETURN

```

```

65140 ZZ$(1)="" : ON (INSTR(" 2 3 4 8",STR$(ZZ$(2)))-1)/2+1
GOSUB 65144: 65146: 65160: 65162
65142 RETURN
65144 ZZ$=ZZ$+"%": GOSUB 65120: IF ZZ$="" THEN RETURN ELSE
PRINT ZZ$: ZZ$(3) TAB(20) CVI(ZZ$(1)): RETURN
65146 ZZ$=ZZ$+"$": GOSUB 65120: IF ZZ$="" THEN RETURN ELSE
PRINT ZZ$: ZZ$(3) TAB(20)
65148 PRINT CHR$(34): ZZ$(2)=CHR$(ZZ$(0))+CHR$(ZZ$(1))+CHR$(
ZZ$(2)): ZZ%=ASC(ZZ$(1)): ZZ$(0)=ASC(MID$(ZZ$(1),2)):
ZZ$(1)=ASC(MID$(ZZ$(1),3)): ZZ$(1)="" : ZZ$(2)=ZZ%
65150 IF ZZ%>0 THEN 65156 ELSE PRINT CHR$(34): ZZ$(0)=ASC(ZZ$(
2)): ZZ$(1)=ASC(MID$(ZZ$(2),2))
65152 IF ZZ$(2)=2 THEN PRINT "CVI(": ZZ$: ZZ$(3): ")": TAB(20)
CVI(ZZ$(1)) ELSE IF ZZ$(2)=4 THEN PRINT "CVS(": ZZ$: ZZ$(
3): ")": TAB(20) CVS(ZZ$(1)) ELSE IF ZZ$(2)=8 THEN PRINT
"CVD(": ZZ$: ZZ$(3): ")": TAB(20) CVD(ZZ$(1))
65154 ZZ$(2)=ASC(MID$(ZZ$(2),3)): GOSUB 65110: RETURN
65156 GOSUB 65110: GOSUB 65100: ZZ$(1)=ZZ$(1)+ZZ$(0): IF ZZ$(3)
<32 OR ZZ$(3)>191 THEN PRINT ".": ELSE PRINT ZZ$(0):
65158 ZZ%=ZZ%-1: GOTO 65150
65160 ZZ$=ZZ$+"!": GOSUB 65120: IF ZZ$="" THEN RETURN ELSE
PRINT ZZ$: ZZ$(3) TAB(20) CVS(ZZ$(1)): RETURN
65162 ZZ$=ZZ$+"#": GOSUB 65120: IF ZZ$="" THEN RETURN ELSE
PRINT ZZ$: ZZ$(3) TAB(20) CVD(ZZ$(1)): RETURN

```

ANMERKUNGEN ZUM AVA

1. Durch Zusammenfassung von Zeilen und ausschließlicher Verwendung von Variablen beginnend mit 'ZZ' oder 'ZD', hat die Lesbarkeit dieser Routine gelitten. Speichersparnis und eine kurze Variablenliste sind das Ergebnis. Außerdem wird der Übergang zu anderen Variablennamen erleichtert. Für ev. Änderungen ist hier eine Liste der verwendeten Variablen:

```

ZZ% Vorübergehender Zähler und Arbeitsspeicher.
ZZ$(0) LSB der laufenden Adresse.
ZZ$(1) MSB der laufenden Adresse.
ZZ$(2) Schreibt Kennzahl 2,3,4 o. 8 für %, $, ! o. #-Variablen. Auch vorübergehender Speicher für Stringlänge.
ZZ$(3) Inhalt der laufenden Speicheradresse (0-255).
ZZ$(4) Laufende Speicheradresse im Variablen-Speicher.
ZZ$(5) 1. Dimension des laufenden Feldes.
ZZ$(6) 2. Dimension des laufenden Feldes falls vorhanden.
ZZ$(7) 3. Dimension des laufenden Feldes falls vorhanden.
ZZ$(8) Zähler für die 1. Dimension.
ZZ$(9) Zähler für die 2. Dimension.
ZZ$(10) Zähler für die 3. Dimension.
ZZ$ Laufender Variablen-Name.
ZZ$(0) Inhalt der laufenden Speicheradresse (CHR$-Format)
ZZ$(1) Laufender Inhalt der Variablen- bzw String-Pointer
ZZ$(2) vorübergehender Speicher der laufenden Adresse während dem String-Aufbau.
ZZ$(3) vorüberg. Variablen-Ausdruck für Anzeige mit Felder.
ZD% Dimension des laufenden Feldes (1-, 2- o. 3-fach).

```

2. Sie können ein 'GOSUB 65030' ausführen falls Sie nur Felder analysieren wollen. Sie können in Zeile 65030 ein 'RETURN' schreiben; falls Sie nur einfache Variablen analysieren wollen. Die Zeilen 65030 bis 65070 werden nicht benötigt; falls Sie nur einfache Variablen anzeigen wollen.
3. In der Zeile 65002 laden wir die Startadresse für einfache Variablen in den Speicher. Der Zeiger steht in der Speicherstelle 16633 und 16634. Wir sind mit den einfachen Variablen fertig; wenn wir die Adressen aus den Speicherstellen 16635 und 16636 erreicht haben (Zeile 65006). Hier beginnt nun der Speicherbereich für die Felder. Beachten Sie; daß wir die Startadresse in der Zeile 65032 nochmals laden. Somit wird erst ein direktes 'GOSUB 65030' (siehe Punkt 2) möglich. Wenn wir die Adressen der Speicherstellen 16637 und 16638 erreicht haben; sind wir mit den Feldern fertig.
4. Im Unterprogramm 65100 wird unsere Adresse erhöht. Dieser Teil ist bei vielen Anwendungen sinnvoll; wo es nötig ist; den Speicher Byte für Byte zu lesen. Wir addieren 1 zum LSB der Adresse. Wenn LSB 256 erreicht; wird es auf 0 zurückgesetzt und MSB wird um 1 erhöht.
5. Das Unterprogramm 65110 wandelt LSB und MSB in eine Adresse mit Integer-Format zurück. Dann wird mit PEEK der Wert der laufenden Adresse eingelesen und umgewandelt; so daß schließlich CHR\$ des PEEK-Wertes abgespeichert wird. Dieses Unterprogramm dient der Programmier-Bequemlichkeit und der Leistungsfähigkeit des Speichers; die Ausführungsgeschwindigkeit wird dafür langsamer.
6. Das Unterprogramm 65120 erstellt einen String; welcher die laufenden Variablen an der laufenden Adresse beinhaltet.
7. In der Zeile 65121 wird überprüft; ob der Variablenname ein Teil des AVA ist. Wollen Sie irgendwelche Variablen-Namen umgehen bzw. nicht analysieren; so fügen Sie einfach diese Variablenamen in diese Liste ein oder machen hier eine Änderung; so daß nur noch die von Ihnen bestimmten Variablen angezeigt werden. Falls eine Variable in der Liste steht; wird ZZ\$ in einen Null-String umgewandelt.
8. Das Unterprogramm 65130 erstellt den Variablen-Namen.
9. Im Unterprogramm 65140 steht die Logik für den korrekten Aufruf des jeweiligen Unterprogramms für Integer; String; einfache Dichte und doppelte Dichte.
10. Falls Sie keine Anzeige der CVI-, CVS- und CVD-Umwandlungen für 2-, 4- und 8-Byte Strings wollen; so können Sie die Zeile 65152 löschen.
11. Falls Sie sich eine LPRINT-Version des AVA anfertigen; müssen Sie unter Umständen den Wert '191' in Zeile 65156 in einen niedrigeren Wert verändern; z.B. 128. Manche Drucker verwenden ASCII-Werte über 128 für spezielle Kontrollcodes.

DAS SUPER-KOPIERFELD

Viele besondere Effekte und schnelle Techniken beinhalten im Prinzip nicht mehr als das Bewegen, oder genauer bezeichnet 'Kopieren', eines Datenblockes von einer Speicherstelle zu einer anderen. Mit speziellen Hilfsroutinen in Z-80 Maschinensprache können wir diese Funktion sehr schnell ausführen. Wir geben einfach an; von welcher Adresse wieviele Bytes nach welcher Adresse übertragen werden sollen.

Zum Kopieren von Daten gibt es mit LDIR und LDDR in Z-80 zwei besonders sinnvolle Anweisungen. Um die Arbeitsweise besser erklären zu können; wollen wir einmal annehmen; daß wir einen Block mit 16 Bytes im Speicher haben. Wir wollen Sie mit 0 beginnend durchnummerieren. Natürlich können Sie an jeder Stelle zwischen 0 und 65535 stehen. Weiterhin nehmen wir an; daß die ersten 4 Bytes dieses Speicherblockes das Wort 'DATA' beinhalten.

```
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
  D A T A ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```

Um das Wort 'DATA' auf die Speicherstelle 6 zu kopieren; wird das LDIR-Kommando zunächst das 'D' an die Stelle 6 kopieren; dann 'A' nach 7; 'T' nach 8 und 'A' nach 9. Nun sieht unser Speicherblock so aus:

```
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
  D A T A ? ? D A T A ? ? ? ? ? ? ? ?
```

Wir haben soeben 4 Bytes von der Speicherstelle 0 nach 6 kopiert.

Mit der LDDR-Anweisung können wir das selbe Ergebnis erreichen; nur beginnt LDDR mit dem am Ende stehenden 'A' in 'DATA' und arbeitet sich bis zum 'D' zurück. Zuerst wird 'A' von der Speicherstelle 3 nach 9 kopiert; dann 'T' von 2 nach 8; 'A' von 1 nach 7 und 'D' von 0 nach 6.

Welche dieser beiden Methoden verwendet wird ist egal; solange sich die Ursprungsadresse und die Bestimmungsadresse nicht überschneiden. Aber nehmen wir mal an; Sie wollen 4 Bytes von Stelle 0 nach 1 bewegen. Wir beginnen wieder mit dem ursprünglichen Speicherinhalt. Die LDIR-Anweisung kopiert das 'D' von 0 nach 1. Nun wird der Inhalt der Speicherstelle 1 auf die Speicherstelle 2 kopiert. In der Speicherstelle 1 steht aber nun nicht mehr das 'A' aus 'DATA' sondern das bereits kopierte 'D'. Unser Ergebnis sieht so aus:

```
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
  D D D D D ? ? ? ? ? ? ? ? ? ? ? ?
```

Hier müssen wir also die LDDR-Anweisung benutzen; die ja das Kopieren mit dem letzten Byte beginnt. Um das Wort 'DATA' um 1 Stelle nach vorne zu kopieren lassen wir die LDDR-Anweisung 4 Bytes von Position 3 nach 4 kopieren.

Ausgehend vom ursprünglichen Speicherinhalt erhalten wir:

```
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
D D A T A ? ? ? ? ? ? ? ? ? ? ? ?
```

In der Z-80 Maschinensprache arbeiten die LDIR- und die LDDR-Anweisung mit den Inhalten von 3 Registern: HL, DE und BC (wer nicht Z-80 versteht, soll sich HL, DE und BC einfach als 3 Integer-Variablen in BASIC vorstellen). Das HL-Register bezeichnet die 'von'-Adresse, das DE-Register bezeichnet die 'nach'-Adresse und das BC-Register gibt an, wie oft ein Byte von einer Adresse zu einer anderen kopiert werden soll. Die LDIR-Anweisung erhöht die 'von'- und 'nach'-Adresse nach jedem kopierten Byte. Die LDDR-Anweisung erniedrigt die 'von'- und 'nach'-Adresse nach jedem kopierten Byte. Das BC-Register wird bei LDIR und LDDR nach jedem kopierten Byte erniedrigt. Wenn BC 0 erreicht, ist die Kopieranweisung beendet.

Wir können diese schnellen Kopiermöglichkeiten sehr gut in ein 'Super-Kopierfeld' umsetzen. Wir laden einfach die benötigten Z-80 Codes in ein Integerfeld von 8 Elementen. Mit DEFUSR legen wir das erste Element des Feldes als Adresse derselben derUSR-Routine fest und mit derUSR-Routine führen wir das Kopieren aus.

Dies sind die für das 'Super-Kopierfeld' benötigten Z-80 Codes:

```
Element 0: 8448
Element 1: 'von'-Adresse
Element 2: 4352
Element 3: 'nach'-Adresse
Element 4: 256
Element 5: Anzahl der zu kopierenden Bytes
Element 6: -20243 für LDIR oder -18195 für LDDR
Element 7: 201
```

Die Elemente 0, 2, 4 und 7 können bereits vorab mit den richtigen Elementen geladen werden, da diese ja immer die gleichen sind. Sie können auch das Element 6 bereits mit -20243 laden, falls Sie keine überlappenden Bereiche haben bzw. wenn Sie keine LDDR-Anweisung benötigen.

Zur Demonstration einiger Kopieraktionen spielen wir mit dem Bildschirmspeicher der die Adressen 15360 bis 16383 belegt. Geben Sie hierzu das folgende Programm ein:

```
10 DEFINT A-Z: J=0: A$=""
20 US(0)=8448: US(2)=4352: US(4)=256: US(7)=201
30 CLS: PRINT "KOPIERFELD DEMO"
40 PRINT @64, "VON      ";; INPUT US(1)
50 PRINT "NACH      ";; INPUT US(3)
60 PRINT "ANZAHL BYTES: ";; INPUT US(5)
70 PRINT "I=LDIR, D=LDDR ";; INPUT A$
80 IF A$="D" THEN US(6)=-18195 ELSE US(6)=-20243
90 DEFUSR=VARPTR(US(0)): J=USR(0)
100 GOTO 40
```

Bevor Sie Ihr Demo-Programm starten speichern Sie es ab und nehmen als Vorsichtsmaßnahme alle Disketten aus den Laufwerken. Falls Sie nämlich zufällig falsche Zahlen eingeben, können Sie Daten auf Speicherstellen kopieren, die BASIC- oder DOS-Zeiger enthalten. Bei der Kopier-Routine ist es wichtig zu wissen, wohin die Daten kopiert werden und wieviele Daten kopiert werden. Wenn Sie die folgenden Beispiele sorgfältig nachvollziehen können keine Fehler auftreten.

Beispiel 1: Wir wollen die obere Bildschirmhälfte auf die untere kopieren. Starten Sie das Programm:

```
von = 15360
nach = 15872
Anzahl Bytes = 512
'I' für LDIR
```

Beispiel 2: Der Titel 'KOPIERFELD DEMO' soll von Position 0 nach 32 kopiert werden:

```
von = 15360
nach = 15392
Anzahl Bytes = 15
'I' für LDIR
```

Beispiel 3: Die ersten 512 Bytes des ROM sollten auf der unteren Bildschirmhälfte ausgegeben werden:

```
von = 0
nach = 15872
Anzahl Bytes = 512
'I' für LDIR
```

Beispiel 4: Wir tauschen ein Verschieben der eben aus dem ROM kopierten Daten um 1 nach rechts vor:

```
von = 1
nach = 15872
Anzahl Bytes = 512
'I' für LDIR
```

Beispiel 5: Wir wollen 'KOPIERFELD DEMO' überlappend kopieren um 11 Positionen nach rechts was uns als Ergebnis 'KOPIERFELD KOPIERFELD DEMO' liefert.

```
von = 15375
nach = 15386
Anzahl Bytes = 16
'D' für LDDR
```

Beispiel 6: Der Bildschirm wird mit 'K' gefüllt (in der Annahme, daß an der Bildschirmposition 0 ein 'K' steht):

```
von = 15360
nach = 15361
Anzahl Bytes = 1023
'I' für LDIR
```

Viele andere Beispiele sind möglich. Geben Sie für die Anzahl der zu kopierenden Bytes nie eine 0 ein. Das ist sehr wichtig, denn eine 0 als Parameter für BC bedeutet bei LDIR und LDDR, daß 65536 mal kopiert wird. Ein katastrophaler Speicherinhalt ist die Folge.

Die folgende Tabelle gibt Ihnen eine brauchbare Auskunft über die möglichen Ausführungsarten mit dem 'Super-Kopierfeld' und gibt an, wie die Elemente 1, 3, 5 und 6 zu belegen sind:

NICHT ÜBERLAPPENDES KOPIEREN NACH OBEN UND UNTEN

 Element 1 (HL) = 'von'-Adresse
 Element 3 (DE) = 'nach'-Adresse
 Element 5 (BC) = Anzahl der zu kopierenden Bytes
 Element 6 (LDIR) = -20243

ÜBERLAPPENDES KOPIEREN NACH OBEN

 Element 1 (HL) = höchste Adresse des zu kopierenden Blockes
 Element 3 (DE) = höchste Adresse des Ziels
 Element 5 (BC) = Anzahl der zu kopierenden Bytes
 Element 6 (LDDR) = -18195

ÜBERLAPPENDES KOPIEREN NACH UNTEN

 Element 1 (HL) = 'von'-Adresse
 Element 3 (DE) = 'nach'-Adresse (kleiner wie 'von'-Adresse)
 Element 5 (BC) = Anzahl der zu kopierenden Bytes
 Element 6 (LDIR) = -20243

VERVIELFÄLTIGUNG EINER BYTE-FOLGE NACH OBEN

 Element 1 (HL) = Adresse des 1. Byte der Byte-Folge
 Element 3 (DE) = Adresse des 1. Byte des 1. Duplikat
 Element 5 (BC) = Die Anzahl der Duplikate (ohne Original) ist zu multiplizieren mit der Länge der Byte-Folge.
 Element 6 (LDIR) = -20243

VERVIELFÄLTIGUNG EINER BYTE-FOLGE NACH UNTEN

 Element 1 (HL) = Adresse des letzten Byte der Byte-Folge
 Element 3 (DE) = Adresse des ersten Byte der Byte-Folge - 1
 Element 5 (BC) = Die Anzahl der Duplikate (ohne Original) ist zu multiplizieren mit der Länge der Byte-Folge.
 Element 6 (LDDR) = -18195

Einige Beispiele für die Möglichkeiten mit dem 'Super-Kopierfeld':

1. Einfügen und Löschen auf dem Bildschirm
2. Auf- und abrollen des Bildschirms oder Teile davon. In oder von geschützten Speicherbereich kopieren.
3. Sichern des Bildschirms in einem geschützten Speicherbereich um in später wieder laden zu können.
4. Daten in geschützten Speicher kopieren, so daß diese an andere Programmierer übergeben werden können.
5. Einfügen und Löschen von Feld-Elementen.
6. Kopieren von Daten aus einem Random-Disk-Buffer direkt (ohne Feld) in den Bildschirmspeicher. Abspeichern des Bildschirms auf Diskette, wobei jeweils 256 Bytes in den Buffer kopiert werden, um dann mit 'PUT' auf Diskette abgespeichert zu werden.
7. Verschieben einer veränderlichen USR-Routine von einer Speicheradresse zu einer anderen.
8. Schnelles Laden numerischer Feldelemente von Diskette und schnelles Übertragen numerischer Felder auf Diskette.
9. Speicher bereinigen oder sich öfters wiederholende Byte-Folgen kopieren. Graphische Effekte.
10. Sofortiges Kopieren von Feldelementen.
11. Daten oder USR-Routinen direkt aus dem Disketten-Buffer in den geschützten Speicher kopieren.

Wie Sie sehen, ist das Super-Kopierfeld sehr nützlich und sehr schnell. Auf einige Besonderheiten kommen wir noch in anderen Teilen dieses Kurses zu sprechen.

EINE USR-ROUTINE ZUM DATEN KOPIEREN

Diese Routine kopiert sofort einen Speicherblock von einer Adresse zu einer anderen. Diese Kopier-USR-Routine kann bis auf folgende Unterschiede die gleichen Funktionen wie das 'Super-Kopierfeld' ausführen:

1. Sie können der USR-Routine die Angaben der 'von' und 'nach' Adresse sowie die Anzahl der zu kopierenden Bytes als einfachen BASIC-Ausdruck übergeben. Damit wird die Programmierung bequemer.

2. Die USR-Routine verwaltet Jede Kopierrichtung, auch überlap-pendes Kopieren nach oben oder unten. Sie brauchen nicht entscheiden, ob LDIR oder LDDR verwendet werden soll.
3. Das 'Super-Kopierfeld' benötigt 16 Bytes, die Kopier-USR-Routine 88 Bytes. Dennoch wird die Kopier-USR-Routine in den meisten Fällen günstiger sein, da das BASIC-Programm weniger Operationen durchführen muß und eine einfache Übergabe der notwendigen Werte möglich ist.
4. Die USR-Routine verwendet den 'Vielwert-Verwalter'. Daher müssen Sie zunächst entscheiden, welche USR-Nummer (0-9) Sie verwenden und Sie müssen in Abhängigkeit von Ihrem DOS zwei Bytes abändern.

Um einen Aufruf der USR-Routine von BASIC aus erklären zu können, nehmen wir an, daß wir die obere Hälfte des Bildschirms auf die untere kopieren wollen. In der Annahme, daß die Routine geladen ist und als USR0 definiert worden ist, lautet unsere Anweisung:

J = USR(15360) OR USR(15872) OR USR(512)

Um den Inhalt der 1. Bildschirmzeile um eine Position nach rechts wandern zu lassen schreiben Sie:

J = USR(15360) OR USR(15361) OR USR(63)

Um die 1. Zeile um eine Position nach links wandern zu lassen:

J = USR(15361) OR USR(15360) OR USR(63)

Um jeden beliebigen Teil des Bildschirms nach oben rollen zu lassen, können Sie schreiben:

J = USR(15360+LI+64) OR USR(15360+LI) OR USR(64*(LV-1))
PRINT @ 15360+LI+(LV-1)*64, CHR\$(30);

Dabei beinhaltet LI% den Anfang des zu rollenden Teils als PRINT @ - Position und LV% die Anzahl der zu rollenden Zeilen.

Wie Sie wahrscheinlich schon gemerkt haben, rufen Sie die Routinen mit einem Ausdruck folgenden Formats auf:

J% = USR(F%) OR USR(T%) OR USR(B%)

Hierbei bedeuten die Integer-Variablen folgendes:

- J% ist eine Schein-Variable.
- F% ist eine Integer-Variable, Konstante oder Ausdruck für die 'von'-Adresse.
- T% ist eine Integer-Variable, Konstante oder Ausdruck für die 'nach'-Adresse.
- B% beinhaltet die Anzahl der zu kopierenden Bytes.
WICHTIG: B% darf nicht 0 sein!

```

00000 ;KOPIER-USR-ROUTINE
00001 ;
F000 00100          ORG      0F000H      ;ANFANG (VERÄNDERLICH)
00110 ;
00120 ;DIE FOLGENDE LOGIK NIMMT DIE 3 ARGUMENTE AN
00130 ;
F000 CD7F0A 00140          CALL    0A7FH      ;ÜBERGEBE WERT AN HL
F003 00      00150          NOP
F004 DD2A145B 00160         LD      IX,(05B14H) ;IX BEINHÄLTET DEFUSR-ADR.
F008 DD7531 00170         LD      (IX+49),L
F008 DD7432 00180         LD      (IX+50),H
F00E DD340A 00190         INC     (IX+10)
F011 DD340A 00200         INC     (IX+10) ;ERHÖHE ZÄHLER UM 2
F014 DD340D 00210         INC     (IX+13)
F017 DD340D 00220         INC     (IX+13) ;ERHÖHE ZÄHLER 2 UM 2
F01A DD7E0A 00230         LD      A,(IX+10)
F01D 0631 00240         LD      B,49
F01F 90      00250         SUB     B ;A = ÜBERGEBENE WERTE * 2
F020 DD4630 00260         LD      B,(IX+48) ;B = VERBLIEBENE WERTE * 2
F023 90      00270         SUB     B
F024 2801 00280         JR      Z,PASS1 ;WENN 0, KEINE WEITEREN W.
F026 C9      00290         RET
F027 DD360A31 00300        PASS1  LD      (IX+10),49
F02B DD360D32 00310        LD      (IX+13),50 ;ZÄHLER ZURÜCKSETZEN
F02F 1806 00320         JR      START
F031 0000 00330         DEFW   0 ;SPEICHER 'VON'-ADRESSE
F033 0000 00340         DEFW   0 ;SPEICHER 'NACH'-ADRESSE.
F035 0000 00350         DEFW   0 ;SPEICHER FÜR BYTE-ANZAHL
00351 ;
00352 ;DIE FOLGENDE LOGIK FÜHRT KOPIEREN AUS
00353 ;
F037 E5      00360        START  PUSH   HL
F038 C1      00370         POP    BC
F039 DD6E31 00380         LD      L,(IX+49)
F03C DD6632 00390         LD      H,(IX+50)
F03F E5      00400         PUSH   HL
F040 DD5E33 00410         LD      E,(IX+51)
F043 DD5634 00420         LD      D,(IX+52)
F046 B7      00430         OR     A
F047 ED52 00440         SBC    HL,DE
F049 E1      00450         POP    HL
F04A 3803 00460         JR      C,MOVEUP
F04C EDB0 00470         LDIR
F04E C9      00480         RET
F04F 09      00490        MOVEUP ADD    HL,BC ;RÜCKSPRUNG INS BASIC
F050 2B      00500         DEC    HL
F051 EB      00510         EX     DE,HL
F052 09      00520         ADD   HL,BC
F053 2B      00530         DEC    HL
F054 EB      00540         EX     DE,HL
F055 EDB8 00550         LDDR
F057 C9      00560         RET
F04F 09      00570         END
00000 TOTAL ERRORS
    
```

'Super-Feld'-Format, 44 Elemente:

```
32717 10 10973 23316 30173 -8911 12916 13533 -8950
2612 13533 -8947 3380 32477 1546 -28623 18141 -28624
296 -8759 2614 -8911 3382 6194 6 0 0
-6912 -8767 12654 26333 -6862 24285 -8909 13398 -4681
-7854 824 -20243 2505 -5333 11017 -4629 -13896
```

POKE-Format, 88 Bytes:

```
205 127 10 0 221 42 20 91 221 117 49 221 116 50 221 52
10 221 52 10 221 52 12 221 52 13 221 126 10 6 49 144
221 70 48 144 40 1 201 221 54 10 49 221 54 13 50 24
6 0 0 0 0 0 229 193 221 110 49 221 102 50 229
221 94 51 221 86 52 183 237 82 225 56 3 237 176 201 9
43 235 9 43 235 237 184 201
```

10 DEFINT A-Z: J=0

```
30 'LADEN DER KOPIER-USR-ROUTINE IN EIN SUPER-DATENFELD
31 DATA 32717,10,10973,23316,30173,-8911,12916,13533,-8950,
2612,13533,-8947,3380,32477,1546,-28623
32 DATA 18141,-28642,296,-8759,2614,-8911,3382,6194,6,0,0,
-6912,-8767,12654,26333,-6862
33 DATA 24285,-8909,13398,-4681,-7854,824,-20243,2505,-5333,
11017,-4629,-13896
34 DIM UX(43): FOR X=0 TO 43: READ UX(X): NEXT
```

```
100 CLS: PRINT "KOPIER-USR-ROUTINE - DEMO UND UTILITY"
110 PRINT @120, "KOPIERE VON: "; INPUT MF%
120 PRINT "KOPIERE NACH: "; INPUT MT%
130 PRINT "ANZAHL DER BYTES: "; INPUT NB%
131 IF NB%=0 THEN 130
140 DEFUSR=VARPTR(UX(0))
150 J=USR(MF%) OR USR(MT%) OR USR(NB%)
160 GOTO 110
```

Die Kopier-USR-Routine wird im 'Super-Datenfeld'-Format, im POKE-Format und als Assemblerlisting gezeigt. Die vorgestellte Version läuft als USR0 unter NEWDOS 2.1. Um sie als eine andere USR-Routine (USR1-USR9) unter NEWDOS 2.1 benutzen zu können, oder wenn Sie ein anderes DOS verwenden, so schlagen Sie im Anhang 2 nach und befolgen nachstehende Anleitungen:

1. Bei der Ausführung als 'Super-Datenfeld' ersetzen Sie das 4. Element, 23316, mit dem entsprechenden Integerwert aus dem Anhang 2. Wenn Sie z.B. TRSDOS 2.3 benutzen, und die Kopier-USR-Routine als USR6 ausführen wollen, so finden Sie im Anhang 2 den Wert 5883 was dezimal 23427 entspricht. Das 4. Element lautet also 23427.

- Falls Sie das POKE-Format verwenden, so ersetzen Sie das 7. und 8. Byte, 20 und 91, mit den benötigten Bytes aus dem Anhang 2. Wenn Sie z.B. NEWDOS 2.1 benutzen, und die Routine als USR9 ausführen wollen, so finden Sie im Anhang 2 den Wert 5826. Das 7. Byte wird 26 (dezimal 38) und das 8. Byte 58 (dezimal 91).
- Beim Assemblerlisting ist der Wert 5814 in Zeile 160 durch den benötigten Hexadezimalwert zu ersetzen.

Das obenstehende Demo-Programm zur Kopier-USR-Routine fragt nach der 'von'- und 'nach'-Adresse und nach der Anzahl der zu kopierenden Bytes. Die Routine wird aus DATA-Zeilen in ein 'Super-Datenfeld' geladen, so daß Sie keinen Speicher schützen müssen. Denken Sie daran, daß Sie in Zeile 31 den Wert 23316 ändern müssen, falls Sie ein anderes DOS verwenden.

Ein weiteres Demo im Zusammenhang mit der Kopier-USR-Routine erhalten Sie, wenn Sie die Zeilen 100 bis 160 löschen und nachstehende Zeilen einfügen. Das Demo liefert eine schnelle, sichtbare 'Reise' durch den Speicher und zeigt sehr schön die gewaltigen Möglichkeiten dieser Routine. Die letzte Bildschirmzeile zeigt die gegenwärtige Adresse in Abständen von 64, während der Speicherinhalt in der oberen Hälfte des Bildschirms rollt.

```
100 CLS: A=0: DEFUSR=VARPTR(UX(0))
110 FOR X=-1 TO 32766 STEP 64: A=X+1: GOSUB 200: NEXT
120 FOR X=-32768 TO 0 STEP 64: A=X: GOSUB 200: NEXT
130 END
200 PRINT @990,A: J=USR(A) OR USR(15360) OR USR(960): RETURN
```

Vorschlag zur Programmbibliothek: a b s c h a f f e n

Im Clubinfo Nr. 5 erklärt Hartmut im Zusammenhang mit der Programmbibliothek, er werde auf weitere Vorschläge eingehen. Hartmut, mein Vorschlag ist so radikal, daß Du hoffentlich nicht zu Deinem Wort stehst, ohne beim Clubtreffen ein Votum dazu einzuholen.

Daß unsere Softwaresammlung juristisch heikel ist, wurde schon des längen und breiten diskutiert. Ob sie überhaupt sinnvoll ist, danach hat bisher komischerweise niemand gefragt. Im folgenden will ich Denkanstöße zu dieser Frage geben. Jeder mag sie für sich selber beantworten. Für mich persönlich ist die Antwort klar: Nein.

Nach der Reinigung der Bibliothek von Fremdprogrammen bleibt voraussichtlich ein schmaler Rest von Brauchbarem. Ihn mit dem bisherigen Aufwand von einem Club-Amtsleiter verwalten zu lassen, entbehrt nicht einer gewissen Komik. Die Gesamtmenge, in Bytes gemessen, bleibt zwar vielleicht verwaltungswürdig, aber bei allem Respekt vor den Selbstdenkern: So richtigen Schmiß wie die ganz großen Raubkopien in der bisherigen Sammlung hat doch kaum ein Eigenbauprogramm.

Das alleine hätte mich aber noch nicht bewegt, mich mit meiner Meinung beim Fest des Clubs möglicherweise gleich am Anfang meiner Mitgliedschaft unbeliebt zu machen. Der wesentliche Grund ist ein ganz anderer:

Jeder von uns startet mit 30 Punkten. Wer aus Zeitgründen, oder weil er nun mal nicht so drauf ist, keine eigene Software liefern kann (oder nicht, wie manche Mitglieder, die Chuzpe besitzt, irgendwelchen Daddelkram einzureichen), ist für den Fest seines Clubdaseins von der Teilhabe ausgeschlossen. Was hat das noch mit einem Club zu tun?

Und umgekehrt: Mein H-DOS, falls man es mit 5 Pkt. bewerten sollte, soll mir nur 25 Bonuspunkte einbringen? Ein paar Videospiele für 200 Arbeitsstunden? Da verschenke ich es lieber. Und genau das ist die Methode in anderen Clubs: Software kostet nichts. Man tauscht Listen, äußert Wünsche, fragt nicht nach Copyrights oder danach, wieviele Bytes in welche Richtung gehen und hat so in kurzer Zeit dasselbe, was alle haben: Alles!

Da es die Programmbibliothek aber noch (?) gibt, will ich mich nicht gegen die Clubregeln stellen und meine Programme nicht unter Umgehung der Clubsammlung anbieten. Dabei ist es mir vollkommen egal, wer noch alles von meinem Freizeitspaß an der Tastatur profitiert. Jegliches Monopoly-Denken liegt mir fern. Aber da sind nun mal die Clubregeln. Noch lange?

Arnulf Sopp, Tel. 0451-791926

Anmerkungen zur Programmbibliothek: (bzw. zum Vorschlag) von A. Sopp dieselbe abzuschaffen)

Im vorstehenden Artikel macht unser neues Mitglied A. Sopp den Vorschlag, die Programmbibliothek abzuschaffen. Meiner Meinung nach tut er das etwas vorschnell.

Hier nun ein paar Punkte, die klar für die Weiterführung der Programmsammlung sprechen und die die Beibehaltung des Punktesystems zumindest sinnvoll erscheinen lassen.

1. Auch eine von allen Raubkopien (sehr oft Spiele, die sowieso fast jeder besitzt) befreite Programmsammlung enthält noch sehr viele interessante Programme. Der Schmiß ist, das kann ich dem Arnulf versichern, nicht raus. Im Gegenteil, da die sieben Pacman-Versionen jetzt verschwunden sind, fallen die Selbstentwicklungen (die nicht überall zu haben sind) und die abgetippten Listings (die abzutippen sich doch die meisten aus Zeitmangel scheuen) umso deutlicher ins suchende Auge.

2. Selbstverständlich bleibt jedem die Möglichkeit, privat zu Tauschen mit wem es ihm beliebt. Dabei sind aber zwei Punkte, die jeder bedenken sollte:

Wenn ich innerhalb oder außerhalb des Clubs tausche, kann ich jederzeit an einen "Fänger" geraten! Das ist einem unserer Mitglieder ja schon passiert, und zwar beim Tausch mit einem anderen Clubmitglied. Die Sache birgt also ein gewisses Risiko.

Wer mit allen Clubmitgliedern (inzwischen über 40) persönlich tauschen möchte, muß sich auf einen erheblichen Briefverkehr einrichten. Allein die Durchsicht aller Tauschlisten würde eine Menge Zeit in Anspruch nehmen.

Die Vorteile einer Clubbibliothek dagegen, sind:

Nur eine Tauschliste für alle Programme und keine Angst vor einem Gerichtsverfahren wegen Verletzung von Rechten anderer!!!

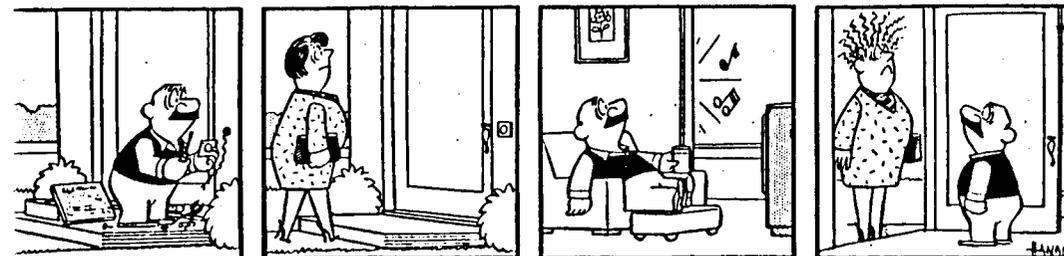
3. Damit die Programmbibliothek jedoch vollständig ist und möglichst viele wertvolle Programme enthält, muß ein Anreiz dafür her, daß alle Mitglieder die von Ihnen erstellten oder eingegebenen Programme auch einsenden. Eine mögliche Lösung dieses Problems besteht in der Beibehaltung der Punkteregeung. Eventuell gibt es auch andere, vielleicht sogar bessere Lösungen für dieses Problem und alle Mitglieder sind dazu aufgefordert, sich eine solche zu überlegen und sie beim Clubtreffen vorzubringen.

4. Es gibt hier und da Mitglieder (die gibt es in jedem Club oder Verein) die aus dem CLUB 80 möglichst viel herausholen wollen, ohne etwas in ihn zu investieren (der Clubbeitrag deckt gerade so die Kosten für das Clubinfo). Solche Mitglieder sollen nicht etwa dem Club ferngehalten werden sondern sie sollen die Clubkasse aufbessern, indem sie sich die für gewünschte Programme fehlenden Programme erkaufen (1.- DM pro Punkt). Dadurch können auch sie einen erheblichen Beitrag zur Clubexistenz beitragen.

Computerfreunde, die im Selbstprogrammieren nicht so fit sind, können Listings abtippen und diese einsenden. Sie ersparen damit anderen viel Zeit und sind durchaus nicht vom Clubleben ausgeschlossen, wie der Arnulf befürchtet.

Ich hoffe, diese kurze Klarlegung einiger, vielleicht für den Neuling undurchsichtiger Punkte, macht deutlich, daß auch wir (der Günther und ich) nicht an Monopolydenken leiden sondern von der Notwendigkeit der Clubbibliothek überzeugt sind.

Hartmut Obermann



Tape Transfer

by J.L. Kissel



Model III NEW-DOS80 2.0 users can't load tape-based source files because NEWDOS's EDTASM doesn't support Model III tape input/output (I/O). While the DOS's documentation describes how to load a source file from tape, zap 30 states that NEW-DOS supports tape I/O for the Model I only.

I decided to rectify this. Rather than try to add Model III tape I/O capability to NEWDOS80 2.0's EDTASM, however, I modified the LMOFFSET program so it would load source files from tape to disk. You can do so by adding the LMOFFSET zaps indicated in the Figure. The Program Listing shows the source code of the patch.

To add the zaps yourself, copy LMOFFSET and save it under a new file name. (The zaps destroy LMOFFSET's ability to handle system tapes and support tape-to-disk loading only.) Then use SuperZap to make the changes listed in the Figure. □

For instructions on how to assemble source code, consult your editor/assembler manual or "An Idiot's Guide to Assembly Language," Parts I and II (80 Micro, May 1981, p. 168, and June 1981, p. 112).

Contact J.L. Kissel at 287 London Road, Isleworth, Middlesex, England.

The Key Box



Model III
NEWDOS80 2.0

At 00,58 change: 3E FF CD 57 54 CD 3p 58 to: 3E 00 CD 7B 54 CD 7B 54	9D 53 C3 E7 53 30 00 CD 7B 54 10 P9 CD 7B 54 C3 D5 54 C3	At 01,6F change: C3 50 55 21 to: C3 FC 56 21	At 03,A1 change: F0 CD 20 44 C2 57 58 D9 21 90 5E CD 33 57 30 FB FE 01 20 0A FE 02 CA 44 57 CD 5C 57 18 0B CD 5C 57 05 05 CD 60 57 CD 60 57 CD 60 57 10 FB 18 0B 7E FE 20 D8 FE FE DA 48 58 23 5E 23 56 23 19 AF C9 CD 5C 57 CD 5C 57 D9	At 00,78 change: 58 CD 35 02 FE 78 CA CA 54 FE 3C 28 05 CD 92 54 18 EC 3E 01 CD 7B 54 CD 35 02 47 C6 02 CD to: 50 06 FF CD 35 02 CD 7B 54 FE 1A CA 32 54 10 P3 CD 35 02 CD 7B 54 FE 1A CA 32 54 18 10	78 B7 to: F8 21 00 52 11 00 53 06 00 CD 20 44 D9 21 9F 5E 22 74 53 2A 9D 5E 23 23 22 90 5E ED 5B 9D 5E 1B ED 53 9D 5E 7A B3 CA 45 57 0E 00 06 01 2A 74 53 11 00 52 ED 00 22 74 53 D9 CD 3C 44 C2 9D 5E D9 18 D9 D9 CD 28 44 C3 59 57 B7	At 00,87 change: 58 CD 35 02 FE 3C CA F8 53 FE 78 CA CA 54 F5 3E 49 32 3D 3C 3E FD CD 57 54 F1 C3 to: 58 ED 5B 9D 5E 13 ED 53
---	--	---	---	--	--	---

Figure. Zaps to LMOFFSET. Zap locations appear in sector and offset format (e.g., XX,YY means sector XX, offset YY).

Program Listing. Patch, the LMOFFSET modification program.

```

00100 ;
00110 ; LMOFFSET PATCHES
00120 ;
00130 ;TO ALLOW LMOFFSET TO READ A EDTASM TAPE AND
00140 ;WRITE IT TO A DISK FILE FOR USE BY APPARAT
00150 ;DISK BASED EDTASM
00160 ;
00170 ;DISK FORMAT MAY BE COMPATIBLE WITH OTHER EDTASM
00180 ;
00190 ; ORG 53C6H ;START OF PATCH
00200 ;CASSETTE IS ON AND SYNC HEADER HAS BEEN READ
00210 LD A,0H ;ZERO THE
    
```

Listing continued

```

573B D9 00900 EXX ;FOR PROPER FCB INPO
573C CD3C44 00910 CALL 443CB ;WRITE A SECTOR
573F C29D5E 00920 JP NZ,5E9DH ;IF DISK ERROR
5742 D9 00930 EXX ;BACK TO TRANSFER POINTERS
5743 18D9 00940 JR WRDSK1 ;LOOP BACK FOR NEXT SECTOR
5745 D9 00950 WRDSK2 EXX ;ORIGINAL REGS
5746 CD2844 00960 CALL 4428H ;CLOSE FILE
5749 C35957 00970 JP 5759H ;TO DONE MESSAGE
0000 00980 END
00000 TOTAL ERRORS
32543 TEXT AREA BYTES LEFT
    
```

```

53C8 CD7054 00220 CALL 547BH ;BLOCK COUNT
53CD CD7B54 00230 CP NZ,547BH ;I.E. NUMBER OF SECTORS TO WRITE
53D1 00240 ;
00245 ;
00250 ;FIRST BYTE OF TAPE IS NOW IN THE A REG
00251 ;
00260 ; CP 0D3H ;CHECK IF EDTASM
53D1 FED3 00260 JR Z,53E1H ;TAPE AND JUMP IF IT IS
53D3 280C 00270 ;
00271 ;
00272 ;WRITE FIRST BYTE TO BUFFER
00273 ;
00274 ; ORG 53E1H
53E1 00275 CALL 547BH ;FIRST BYTE TO BUFFER
53E1 CD7B54 00280 ;
00290 ;53E4 WILL CHECK FOR THE UP ARROW KEY ABORT FUNCTION
00300 ;
00310 ; ORG 53E7H
53E7 00310 LD B,255 ;SET UP LOOP COUNT
53E7 06FF RDBLK1 LD B,255 ;READ A BYTE FROM TAPE
53E9 CD3502 00340 RDBLK2 CALL 0235H ;WRITE BYTE TO BUFFER
53EC CD7B54 00350 CALL 547BH ;CHECK IF
53EF FE1A 00360 CP 1AH ;END OF FILE
53F1 CA3254 00370 JP Z,TAPPIN ;LOOP FOR 255 BYTES
53F4 10F3 00380 DJNZ RDBLK2 ;GET 256'S BYTE
53F6 CD3502 00390 CALL 0235H
53F9 CD7B54 00400 CALL 547BH ;AND SAVE IT
53FC FE1A 00410 CP 1AH ;CHECK IF
53FE CA3254 00420 JP Z,TAPPIN ;END OF FILE
5401 1818 00430 JR 541BH ;ELSE CONTINUE TO READ TAPE
00440 ;
00450 ;541B TWINKLES THE STARS AND CHECKS
00460 ;FOR CANCELED FUNCTION
00470 ;
00480 ; ORG 5426H
5426 00480 LD DE,(5E9DH) ;AND
5426 ED5B9D5E 00490 ;
542A 13 00500 INC DE ;INCREMENT
542B ED539D5E 00510 LD (5E9DH),DE ;IT
542E C3E753 00520 JP RDBLK1 ;CONTINUE READING TAPE
5432 3E00 00530 TAPPIN LD A,0H ;ZERO
5434 CD7B54 00540 CALL 547BH ;THE REMAINING
5437 10F9 00550 DJNZ TAPPIN ;BYTES OF THE BLOCK
5439 CD7B54 00560 CALL 547BH ;
543C C3D554 00570 JP 54D5H ;TO TURN OFF TAPE
00580 ;
00590 ;CASSETTE IF OFF INTERRUPTS ARE ON
00600 ;
00610 ; NOW ASK FOR DESTINATION FILESPEC
00620 ;
54D9 00620 ORG 54D9H
54D9 C3FC56 00630 JP 56FCH ;TO FILESPEC MESSAGE
00640 ;
00650 ;DISK FILE IS NOW OPEN STARTING WRITING TO DISK
00660 ;
5704 00670 ORG 5704H
5704 210052 00680 LD HL,5200H ;START OF FCB'S BUFFER
5707 110053 00690 LD DE,5300H ;FCB ITSELF
00700 ;
570A 0600 00700 LD B,0H ;256 BYTE RECORDS
570C CD2044 00710 CALL 4420H ;OPEN FILE
570F D9 00720 EXX ;SAVE ASSOCIATED FCB INFO
5710 219F5E 00730 LD HL,5E9FH ;FIRST BYTE OF TAPE BUFFER
5713 227453 00740 LD (5374H),HL ;PGM POINTER TO TAPE BUFFER
5716 2A9D5E 00750 LD HL,(5E9DH) ;AJUST
5719 23 00760 INC HL ;BLOCK
571A 23 00770 INC HL ;COUNT
571B 229D5E 00780 LD (5E9DH),HL ;TO +1 OF TRUE VALUE
571E ED5B9D5E 00790 WRDSK1 LD DE,(5E9DH) ;CCKECK
00800 ;
5722 1B 00800 DEC DE ;IP
5723 ED539D5E 00810 LD (5E9DH),DE ;SECTOR
5727 7A 00810 LD A,D ;COUNT
5728 B3 00820 OR E ;COMPLETED
5729 CA4557 00830 JP Z,WRDSK2 ;ELSE SET FOR
572C 0E00 00840 LD C,0H ;256 BUTE TRANSFER
572E 0601 00850 LD B,1H ;POINT FIRST BYTE TO TRANSFER
5730 2A7453 00860 LD HL,(5374H) ;FCB BUFFER FIRST BYTE
5733 110052 00870 LD DE,5200H ;MOVE A SECTORS WORTH OF DATA
5736 EDB0 00880 LDIR ;STORE UPDATED POINTER TO BYTE
5738 227453 00890 LD (5374H),HL
    
```

Ein Städter kommt in ein kleines Bergdorf und fragt einen Einheimischen:
"Ich suche einen jungen Mann, der hier wohnt und 20 Jahre alt, groß, blond, kräftig, mit nem Ohr - hamens Oskar!"
"Hm", überlegt der Einheimische, "und wie heißt das andere Ohr?"

Der Doktor hat gesagt, ich brauche dringend Luftveränderung, erzähl Frau McGregor, mein lieber Mann, Da hast du aber Glück, meinst du auf der Gasse Scholte... der Wind hat nämlich gerade geblasen!

Ich begreife nicht, wie du nur so faul sein kannst", schimpft der Vater mit seinem Sohn. "Für mich ist die Arbeit immer ein wahres Vergnügen!"
"Aber, Papa, wir sind doch nicht nur zum Vergnügen auf die Welt gekommen!"

Tricks mit Strings

Ohne String-Bearbeitung wäre ein Computer nur eine simple Rechenmaschine. Komfortable Befehle in puncto Strings zeichnen deshalb eine gute Programmiersprache aus. Die meisten Basic-Dialekte beschränken sich auf das Notwendigste, den Rest muß man selbst schreiben.

Listing 1 ist sozusagen die Grundübung. Im String X\$ soll das Zeichen C\$ gesucht werden, und zwar ab Position P. Wurde das Zeichen gefunden, soll P diese Position als Wert erhalten, ansonsten wird P=0. Das Unterprogramm ab Zeile 100 führt den Job aus. Warum verwenden wir kein FOR-NEXT? Nun, wir wären gezwungen, die FOR-Schleife beim ersten »Finden« zu verlassen, und das ist von Übel. Der Interpreter legt die Schleifenparameter (Index-Name, Step und ähnliches) auf dem Stack ab. Erst wenn die Schleife ordnungsgemäß über NEXT verlassen wird, wird der Stack bereinigt. Im anderen Falle bleibt der Stackpointer an der alten Position. Beim nächsten Aufruf des Unterprogramms wird eine neue Schleife eröffnet, wieder gehen ihre Parameter auf den Stack und so weiter. Wiederholt man das Spielchen oft genug, ist der Stack mit Schleifen-Trümmern gefüllt und der Computer steigt aus. Listing 2 demonstriert, wie man in einem String einen anderen, Substring genannt, sucht. Haben Sie diese Routine in ein Programm eingebaut, brauchen Sie Listing 1 nicht mehr, denn ein Zeichen ist nichts weiter als ein Substring der Länge eins. Es passiert nichts weiter, als daß aus dem Zielstring Substrings gebildet und diese gegen den Suchstring verglichen werden. Eine andere Methode wäre, erst das erste Zeichen zu suchen. Wenn das gefunden ist, prüft man das zweite auf Übereinstimmung, dann das dritte und so weiter. Dieses Verfahren ist typisch für Assembler.

Strings verändert man durch Einfügen, Löschen und Ersetzen von Substrings. Auch hier gilt, daß für alle Funktionen ein Unterprogramm ausreicht. Löschen beispielsweise kann als Ersetzen durch einen Nullstring realisiert werden, aber das ist nicht unbedingt der optimale Weg. Listing 3 zeigt das Einfügen für diesen Fall: In X\$ soll C\$ nach Position P eingefügt werden. Zeile 100 ist alles, was dafür erforderlich ist. Fast genau so funktioniert das Löschen nach Listing 4. In X\$ sollen L Zeichen ab Position P gelöscht werden. Das Ersetzen nach Listing 5 ist dann lediglich Löschen mit Einfügen kombiniert. Auch dazu reicht eine Zeile. Listing 6 wird in der Praxis häufiger gebraucht werden: In X\$ soll ab Position P C\$ gesucht und falls gefunden, durch E\$ ersetzt werden. Dafür reicht eine Kombination von Listing 2 mit Listing 5. Und jetzt die Tricks:

Listing 7 zeigt ein Menü, wie es häufig vorkommt. Ich könnte schreiben

```
IF A$="S" THEN...
IF A$="D" THEN...
IF A$="E" THEN...
```

Dafür reicht aber eine Zeile (hier 50), die darauf beruht, daß ein logischer Ausdruck wie »A\$="S"« den Wert -1 annimmt, wenn die Bedingung zutrifft, und den Wert 0 erhält, wenn sie nicht erfüllt ist. Ist zum Beispiel in Zeile 50 der 3. Ausdruck wahr, dann ergibt $-1-3=+3$. Alle anderen Ausdrücke bekommen den Wert 0. Sollten Sie einen Computer besitzen, der logisch wahr als +1 meldet (zum Beispiel Apple), dann brauchen Sie nur die negativen Vorzeichen wegzulassen. Wenn Sie schreiben »IF A <> =0 THEN...« ist das zwar richtig, aber »IF A THEN...« tut's oft auch. Basic prüft dann, ob A logisch wahr ist. Zeile 45 testet, ob in Kleinschrift eingegeben wurde, und wandelt gegebenenfalls in »uppercase«. Das beruht darauf, daß Kleinbuchstaben im ASCII-Code um 32 versetzt sind. Die Zeile 50 kann bei langen Menüs Ausmaße annehmen, die in Tipparbeit ausartet. Das sollte vermieden werden. Eine Lösung zeigt Listing 8. Alle erlaubten Antworten stehen in X\$. In diesem wird nach bekanntem Muster das Zeichen A\$ gesucht und anhand seiner Position verzweigt.

Listing 9 schließlich können Sie dann in die Menü-Routine einbauen, wenn die Antworten länger als 1 Zeichen sind. Aber auch dann einsetzen, wenn Sie Speicher sparen wollen. Mit einem String-Array kann man zwar bequemer eine Beziehung zum Beispiel zwischen Monatsnummer und -namen herstellen, jeder Eintrag kostet aber zusätzlich 3 Byte. Solange alle Einträge gleich lang sind, ist die Sache einfach. Die kleine Formel in Zeile 30 reicht.

Etwas komplizierter wird die Sache, wenn Sie im Menü verschiedenen lange Texte zulassen. Auch das wird mit den Monatsnamen, jetzt in Listing 10, demonstriert. Der Trick: Die Namen müssen mit einem Delimiter (hier der *) getrennt sein. In den Zeilen 50 bis 60 wird nach dem n-ten »*(=Nummer) gesucht. Dessen Position zuzüglich 1 ist der Beginn des Substrings und wird in P1 gerettet.

Ab Zeile 80 wird nun das nächste »*« gesucht, dessen Position findet sich in P. Folglich ist $P-P1-1$ die Länge des Substrings, $P1+1$ war der Start, Zeile 90 kann mit dem Ausdruck beginnen.

Zeiger verbogen

Jede Variable, die Basic im Programm antrifft, wird in eine Tabelle (Vartab) in der Reihenfolge des Auftretens eingetragen. Wird einer Variablen ein neuer Wert zugewiesen, muß sie der Interpreter in Vartab suchen und dort den neuen Wert eintragen. Bei numerischen Variablen ist das kein Problem, zur Abspeicherung steht immer ein konstant großer Platz zur Verfügung. Nur bei Strings wird es schwierig, da die Länge variieren kann. Um nicht jedesmal die Tabelle zu verschieben, arbeitet Basic mit einem Trick: In Vartab wird die Länge des Strings (1 Byte) und seine Adresse (2 Byte) eingetragen. Der String selbst steht außerhalb der Tabelle im Stringpool, die Adresse sagt wo. Beim TRS-80 gibt es die Funktion VARPTR (Variable Pointer). Beim Apple und anderen gibt es zwei Bytes, die die Adresse der zuletzt benutzten Variablen halten. In Listing 11 wird in den Zeilen 25 bis 30 VARPTR simuliert. Zeile 50 holt dann erst das Längen-Byte (L) und schließlich noch die Adresse (A). In der folgenden Schleife wird dann A\$ ausgedruckt. Mit »PRINT A\$« ginge das einfacher, aber es sollte nur das System bewiesen werden.

Man kann in die 3 Bytes alles mögliche POKen, zum Beispiel die Werte eines anderen Strings. Damit lassen sich Strings sehr schnell tauschen und sortieren. Oder man läßt das Längen-Byte mit der Länge einer Bildschirmzeile und die

Adreßbytes mit deren Adresse. »LPRINT A\$« gibt dann die Schirmzeile auf dem Drucker aus. Experimentieren Sie selbst!

(Peter Wollschläger/hg)

```
10 INPUT X$,C$,P
20 GOSUB 100:PRINT P
30 GOTO 10
100 L=LEN(X$):P=P-1
110 P=P+1
120 IF MID$(X$,P,1)=C$ THEN RETURN
130 IF P<L THEN 110
140 P=0:RETURN
```

Listing 1. So sucht man ein Zeichen in einem String

```
10 INPUT X$,C$,P
20 GOSUB 100:PRINT P
30 GOTO 10
100 L1=LEN(C$):L=LEN(X$)-L1+1:P=P-1
110 P=P+1
120 IF MID$(X$,P,L1)=C$ THEN RETURN
130 IF P<L THEN 110
140 P=0:RETURN
```

Listing 2. Schon besser, auch Worte werden gefunden

```
10 INPUT X$,C$,P
20 GOSUB 100:PRINT X$
30 GOTO 10
100 X$=LEFT$(X$,P)+C$+RIGHT$(X$,LEN(X$)-P)
110 RETURN
```

Listing 3. Einfügen von Strings in einen String

```
10 INPUT X$,P,L
20 GOSUB 100:PRINT X$
30 GOTO 10
100 X$=LEFT$(X$,P-1)+RIGHT$(X$,LEN(X$)-P-L+1)
110 RETURN
```

Listing 4. Teil eines Strings löschen

```
10 INPUT X$,P,L,C$
20 GOSUB 100:PRINT X$
30 GOTO 10
100 X$=LEFT$(X$,P-1)+C$+RIGHT$(X$,LEN(X$)-P-L+1)
110 RETURN
```

Listing 5. Löschen + Einfügen = Ersetzen

```
10 INPUT X$,C$,F,E$
20 GOSUB 100:PRINT X$
30 GOTO 10
100 L1=LEN(C$):L=LEN(X$)-L1+1:P=P-1
110 P=P+1
120 IF MID$(X$,P,L1)=C$ THEN 150
130 IF P<L THEN 110
140 P=0:RETURN
150 X$=LEFT$(X$,P-1)+E$+RIGHT$(X$,LEN(X$)-P-L+1)
160 RETURN
```

Listing 6. Ersetzen mit Komfort: Der alte String darf genannt werden

```
10 PRINT"<S> SICHERN"
20 PRINT"<D> DRUCKEN"
30 PRINT"<E> ENDE"
40 INPUT A$
45 IF A$="Z" THEN A$=CHR$(ASC(A$)-32)
50 ON (A$="S")*-1+(A$="D")*-2+(A$="E")*-3 GOSUB 100,200,300
60 GOTO 10
100 PRINT "ROUTINE SICHERN":RETURN
200 PRINT "ROUTINE DRUCKEN":RETURN
300 PRINT "ROUTINE ENDE ":ENDE
```

Listing 7. »ON GOSUB« funktioniert auch mit Strings

```
10 PRINT"<S> SICHERN"
20 PRINT"<D> DRUCKEN"
30 PRINT"<E> ENDE"
40 INPUT A$
45 IF A$="Z" THEN A$=CHR$(ASC(A$)-32)
50 X$="SDE":L=3:P=0
51 P=P+1:IF MID$(X$,P,1)<>A$ AND P<=L THEN 51
52 ON P GOSUB 100,200,300
60 GOTO 10
100 PRINT "ROUTINE SICHERN":RETURN
200 PRINT "ROUTINE DRUCKEN":RETURN
300 PRINT "ROUTINE ENDE ":ENDE
```

Listing 8. Das Menü in einem String versteckt

```
10 D$="JANFEBMARAPRMAIJUNJULAUUGSEPOKTNOV
DEZ"
20 INPUT "MONATS-NR. ";M
30 PRINT MID$(D$,3*M-2,3)
40 GOTO 20
```

Listing 9. Eine Reihe von Substrings spart Speicher

```
10 D$="/JANUAR/FEBRUAR/MARZ/APRIL/MAI/J
UNI/JULI/AUGUST"
20 D$=D$+" /SEPTEMBER/OCTOBER/NOVEMBER/DE
ZEMBER/"
30 INPUT "MONATS-NR. ";M
40 P=0:P1=0
50 P=P+1:IF MID$(D$,P,1)<>"/" THEN 50
60 P1=P+1:IF P1<>M THEN 50
70 P1=P
80 P=P+1:IF MID$(D$,P,1)<>"/" THEN 80
90 PRINT MID$(D$,P1+1,P-P1-1)
100 GOTO 30
```

Listing 10. String-Reihe mit Delimiter

```
10 A$="ABC"
20 POKE 6,PEEK(131):POKE 7,PEEK(132)
30 VP=PEEK(6)+256*PEEK(7)
50 L=PEEK(VP):A=PEEK(VP+1)+256*PEEK(VP+2)
60 FOR I=A TO A+L-1
70 PRINT CHR$(PEEK(I));
80 NEXT I
```

Listing 11. So simuliert man VARPTR, hier auf dem Apple

 Ein Tip, mit dem EDTASM unter Newdos80 schneller und eleganter zu arbeiten:
 Ihr kennt alle die Prozedur: Source-file laden, assembleieren, Source-file save, go to dos, object-code laden, test-run.. Wenn das Programm nicht so tut, wie es soll, beginnt das ganze Spielchen von vorne.
 Ich mache es so: Textfile laden, assemblieren, goto dos, exec object-code, wenn ein Problem auftaucht, mit 123 in den Debugger, dann G6FA0, das ist ein Warmstart des Assemblers, wobei der Textbuffer erhalten bleibt. Der Cursor ist allerdings unsichtbar, aber es geht für kurze Edit oder Einfügen von Zeilen auch so sehr gut.
 EDTASM belegt den Platz: 5200-76FF
 Deshalb muß die ORG-Anweisung oberhalb von HA000 liegen, um auch für den Source-Code noch genügend Platz zu lassen. Nach Fertigstellung des Programmes kann man die ORG ja wieder hingeben, wo man will.
 Viel Spaß
 W.Zwiczke

Der Mann mit der rauchenden Lötglöbde melzt:
 Für das Clutreffen überlege ich, ob ich ein Selbstbauprojekt zur Sprache bringen soll. Ich denke da an ein Bus-System, wie bei Apple od. IBM-PC.... Da könnte man dann alles, was es an I/O Spielchen gibt anstecken. (Sprachausgabe, Joystick, Erinnerung, RS232, FID, D/A-Wandler, Relaiskarte,
 Der Bus-Controller wäre auf jeden Fall voll gepuffert und könnte alle I/O Adressen von 1-254 betreiben.
 Jede Karte hat für sich die Adressdecodierung, damit das System universell werden können.
 Zur Größe der Steckkarten:
 Als Format stelle ich mir Europa-Größe vor, da man darauf sicher alles unterbringt. Es gibt auch in dieser Größe diverse Fachzeitschriften jede Menge Bauvorschläge, die man 'abklupfern' kann.
 Vielleicht habt Ihr, liebe Freunde einen guten Vorschlag für die Pin-Belegung des Bus-Systems. Es muß auf jeden Fall 8 Datenbits, die unteren 8 Adressbits, den Control-bus sowie die Stromversorgung (+5V +12V) beinhalten.
 Wenn wir alle 16 Adressbits herausführen, so können wir auch Speichererweiterungen, Eprom-Banks, High-res. Grafik, Floppy-Controller..... über das Bus-System realisieren.
 Wie denkt Ihr darüber? Lasst es mich wissen. Wer Zeit für ein Briefchen hat, den ersuche ich schon vor dem Clutreffen mal kurz ein paar Zeilen mir zukommen zu lassen.

 # Walter Zwiczke
 # Lengfelden 123
 # A-5101 BERGHEIM



Konvertierung TRS-80 Level-2-Basic nach MBASIC

Das Level-2-Basic der TRS-80-Computer besitzt zum größten Teil alle Befehle, Funktionen und Operatoren von MBASIC. Daher ist es meist kein Problem, die TRS-80-Basic-Programme auf anderen Computern unter MBASIC laufen zu lassen. Eine Schwierigkeit liegt jedoch im Cursor-Positionierungsbefehl des TRS-80. Alle Cursor-Positionen sind bei ihm von der linken oberen bis zur rechten unteren Ecke von 0 bis 1023 durchnumeriert (beim 16 x 64-Bildschirm). Man positioniert den Cursor nun an eine bestimmte Bildschirmposition durch den Befehl PRINT@ZAHL, wobei Zahl eine der Positionen 0 bis 1023 ist. Bei den meisten anderen Computern wird eine Bildschirmposition aber durch Angabe der Zeile und Spalte erreicht.

Die lästige Umrechnung der TRS-80-Zahlen (0 bis 1023) in Zeilen und Spaltenangaben erledigt folgendes kleine Unterprogramm:

```
(In der Variablen ZAHL ist die TRS-80-Position zu übergeben.)
ZEILE = INT(ZAHL/64)
SPALTE = ZAHL-64 * ZEILE
```

ZEILE und SPALTE nehmen die Werte von 0 bis 15 beziehungsweise 0 bis 63 an. Zahlen auf Ihrem Computer die Zeilen und Spalten von 1 bis 16 (1 bis 24) beziehungsweise 1 bis 64 (1 bis 80) sind die Variablen ZEILE, SPALTE am Ende des Unterprogramms einfach um Eins zu erhöhen. Die TRS-80-Programme mit dem 16 x 64-Bildschirm laufen so auch auf 24 x 80-Bildschirmen, wobei natürlich nur 16 Zeilen und 64 Spalten benutzt werden.

Da es noch relativ umständlich ist, bei jeder Cursor-Positionierung obiges Unterprogramm aufzurufen, hier noch ein Tip, wie man es sich, eventuell nicht auf allen Computern, noch einfacher machen kann:

Da MBASIC selbst keinen Cursor-Positionierungsbefehl besitzt, muß auf jedem Computer eine systemspezifische Positionierungsroutine benutzt werden. Bei manchen Computern ist es möglich, diese Routine in einer benutzerdefinierten MBASIC-Funktion (DEF FN...) unterzubringen.

Ein Beispiel dazu: Der Cursor werde zum Beispiel durch PRINT CHR\$(27)+CHR\$(22)+CHR\$(Zeile)+CHR\$(SPALTE) positioniert.

Eine entsprechende Funktion, die direkt die TRS-80-Zahlen in Zeilen/Spalten umrechnet und gleichzeitig den Cursor positioniert, sähe dann folgendermaßen aus
 DEF FN\$(ZAHL) = CHR\$(27)+CHR\$(22)+CHR\$(INT(Zahl:64))+CHR\$(Zahl-64*INT(Zahl:64))

Damit fallen alle Umrechnungen des TRS-80-Formats weg. Statt PRINT@Zahl im TRS-80-Basic schreibt man jetzt in MBASIC: PRINT FN\$(ZAHL)

Ob es möglich ist, eine Funktion dieser Art zu definieren, hängt jedoch von der Art der Cursor-Positionierung des jeweiligen Computers ab

Und nun viel Spaß beim Übertragen der TRS-80-Programme auf MBASIC, ohne lästige Rechnereien!

(J.Voß/bo)

H.L.R.R.A.

Wieder einmal habe ich es geschafft - wieder einmal kann ich mit dieser letzten Seite ein Info abschließen. Dieses Mal habe ich es wieder mit Freude gemacht, da ich momentan etwas mehr Zeit habe. Jetzt zum Schluß hinaus mußte ich mich zwar nochmal ganz schön beeilen - aber ich glaube, das Info kommt noch rechtzeitig an.

Nochmal Danke bei allen Mitgliedern, die an der Entstehung dieses Infos beteiligt waren.

Ich freue mich schon riesig auf unser Clutreffen - obwohl das Wochenende ziemlich anstrengend werden wird.

Manche werden vielleicht einige Antworten zu Vorschlägen oder Fragen zum letzten Info vermissen - hier habe ich dieses Mal so gut wie keine Reaktionen von Euch erhalten, und kann aus diesem Grund auch nichts schreiben.

So - nun nur noch Info kopieren lassen, zusammensetzen und heften und dann rein in die Kuverts, Adresse und Briefmarke drauf und ab damit zur Deutschen Bundespost.

Tschau bis bald

Euer Günther Wagner

