

Januar 1985  
-----

Hallo Freunde,

wieder mal ist es soweit - das neue Info kommt ins Haus geschneit. Ich hoffe, Ihr habt wieder alle recht viel Freude daran. Nun es steht ja auch ne Menge drin:

So fängt z.B. der Harald mit seinem Cobol-Kurs an. Ihr könnt nachlesen, wie man einen Computer modernisiert. Es gibt den 2. Teil von BASIC Faster & Better. Ein Beitrag über die Kleinbuchstaben ist dabei. Der Gerald Dreyer zeigt Euch, wie man die Taktfrequenz beim Genie erhöht. Wie man einen Joystick anschließt sagt Euch der Jens. Mehrere Seiten stammen vom Hartmut mit einigen Programmen etc. Natürlich wieder dabei: Die Adventure-Ecke. Der Matthias schreibt über die Computerszene in der DDR. Noch vieles mehr steht drin in diesem Info, doch schaut selbst.

Ja - und da so ein Info über 4-6 Wochen entsteht, geht es natürlich wieder ein wenig durcheinander. Wieder wurde (auch zu Recht) bemängelt, daß die Abheft-Möglichkeit des Infos schlecht ist. Ok - das stimmt. Aber durch die Verkleinerung und das doppelseitige Kopieren (1 Blatt = 4 DIN A4 Seiten) sowie eben der langen Bearbeitungszeit ist es mir nicht möglich, eine befriedigendere Lösung zu finden. Wer unbedingt das Info im Ordner archiviert haben will, kann ja Klarsichthüllen verwenden.

Das nächste Clubinfo erscheint auf alle Fälle mindestens eine Woche vor unserem Clubtreffen. Wer Beiträge liefern kann, sollte diese bitte nach Möglichkeit bis Mitte/Ende Februar einsenden. Die Beiträge können kopierfertige Vorlagen sein oder Textfiles, die ich auf mein Scripsit überziehe oder auch handschriftliches (wenn es nicht anders geht). Ich hoffe, daß ich für das 6. Info wieder recht viel Material bekomme.

Übrigens - das beiliegende Schreiben mit Anmeldung vom Jürgen liegt extra deshalb lose bei, damit es Jedem auffällt und damit es Jeder so bald wie möglich dem Jürgen zusendet (auch die, die am Clubtreffen nicht teilnehmen!).

Aber nun genug mit dem Prolog - viel Spaß wünscht Euch Euer



Günther Wagner

Es wurde Zeit, die Programmbibliothek abzugeben. Ich bin froh, daß sich der Hartmut bereit erklärt hat, diese zu übernehmen. Ich weiß, daß die Bibliothek bei Ihm in den besten Händen ist. Unterstützt Ihn bitte so gut wie möglich und laßt Ihm vor allem genügend Zeit für die Neuorganisation der Bibliothek. Erleichtert Ihm die Arbeit wo es geht und denkt daran - auch er macht das ausschließlich in seiner Freizeit - also Geduld haben wenn es mal länger dauert.

Ich möchte mich bei all denen entschuldigen, die bei mir noch Programme bestellten und diese von mir nicht mehr erhalten haben. Es ging leider nicht anders. Ihr könnt Sie dann ab ca. März beim Hartmut anfordern. Ich weiß - einige werden auf mich sauer sein - das kann ich leider nicht ändern.

Ein wenig geärgert hat mich auch etwas. Ich mußte 18 Mitgliedern ein Schreiben zusenden, in dem ich Sie an die Bezahlung des Jahresbeitrages erinnern mußte. Diese Arbeit hätte ich mir gerne erspart.

Die Mitgliederanzahl beträgt über 30. Es ist also noch gefährlicher geworden, daß ich etwas übersehe oder etwas nicht schreibe usw. Bitte erinnert mich daran, wenn ich etwas vergessen habe. DANKE.

Durch die relativ hohe Mitgliederzahl ist es auch nicht mehr möglich, daß ich jeden Brief ausführlich beantworte. Oft bin ich nun der Ansicht, daß der Brief durch das Info ausreichend beantwortet wird, oder daß eine Beantwortung des Briefes nicht unbedingt erforderlich ist.

Über die Leitung des CLUB 80, also die sog. Vorstandschaft, müssen wir auch auf dem Clubtreffen ausführlich sprechen. Ich 'trete' sozusagen von meinem (damals als Gründer des CLUB 80 selbst erwählten) Amt zurück. Ich muß offiziell entlastet werden (Kasse). Aus wievielen Personen (1 oder mehrere) sich die neue Vorstandschaft zusammensetzt und wie die Wahl vor sich geht ist Thema des Clubtreffens. Ich für meine Person muß Euch mitteilen, daß es durchaus sein kann, daß ich mich als Kandidat für die Clubleitung nicht mehr zur Verfügung stelle. Das 7. und 8. Semester mit der Diplomarbeit sind sehr sehr wichtig und sehr zeitintensiv.

Es sollte sich also jeder für sich überlegen, ob er im Club ein Amt übernehmen kann. Noch einmal ganz klar: Auf dem Clubtreffen wird die neue Clubleitung bestimmt und jeder steht zur Diskussion bzw. kann Vorstand werden!

*Es fragte den Besucher:  
"Wann spricht der Hr. Grogg  
so durch die Nase?"  
Um sein neues Gefäß zu  
schonen -"*

*„Wetten, daß ich jedes Getränk mit verbundenen Augen erkenne?“ gibt der Partygast an. Der Test beginnt sofort. Man reicht ihm ein Glas mit einer Flüssigkeit. Er nimmt einen Schluck und schimpft:  
„Pfui Teufel, das ist ja Benzini!“  
„Richtig“, sagt der Gastgeber, „aber welche Marke?“*

### KOMTEK I - DATENBLATT

Das Grundgerät (KT 1001/2) bietet folgende Ausstattungsmerkmale:

- + 2-80 CPU 1,93 MHz zukünftig 1,78 MHz
- + 12 K Basic Level II-Interpreter im ROM
- + 16 K RAM (erweiterbar bis auf 64 K)
- + echte Schreibmaschinentastatur (ASC II) mit
  - Tastenentprellung
  - Groß-, Kleinschreibung (echte Unterläggen!)
  - Wiederholfunktion
  - Blockgraphik auf der Tastatur
  - Umschaltung 64/32 Zeichen pro Zeile per Tastatur
  - 64 Zeichen pro Zeile, 16 Zeilen
- + Kassettenrecorderanschluß für alle handelsüblichen Recorder mit Start über Remote durch KOMTEK I
- + Anschlüsse für Fernseher oder Monitor
- + Programmierbarer Tongenerator
- + Echtzeituhr
- + Centronics-Schnittstelle zum Anschluß eines Druckers
- + Vier Sensor-Eingänge für Temperaturfühler, Lichtschranken, Alarmer usw.
- + Sechs programmierbare Zeitschalter zur sekundengenauen Fernsteuerung von z.B. Lampen, Alarmanlagen, Heizungen, Elektroisenbahnen usw.
- + voll Software-kompatibel mit TRS-80\*, daher Zugriff auf eines der größten Software-Programme der Welt
- + Bedienungs- und Basic Level II-Handbuch in Deutsch

### Optionen, Einbau im Gerät

- + Floppy-Controller, anschließbar bis zu 4 Laufwerke
- + 32 und 64 K RAM
- + Farbe (automatisch 4 Farbkombinationen um alle Schwarz-Weiß-Programme in Farbe wiederzugeben, 16 Farben programmierbar)
- + High-RES Graphik voraussichtlich bis Mitte September 83 verfügbar
- + RS-232 c-Interface ca. Mitte September verfügbar
- + Analog/Digital-Wandler in Vorbereitung (1)
- + Pufferung der Speicherinhalte durch Batterie bei Netzausfall in Vorbereitung
- + ROM-Bank zum schnellen Zugriff auf Anwenderprogramme in Vorbereitung (1)

(1) außerhalb des Gerätes anzuschließen  
\* eingetragenes Warenzeichen der Tandy Corporation



ERGEBNIS DER CLUBSATZUNGSÄNDERUNG (PUNKT 7)

\*\*\*\*\*

Allen Mitgliedern dürfte ja bekannt sein, daß der Jahresbeitrag seit 1985 DM 30 beträgt - somit ist also mein Antrag vom letzten Info durchgegangen.

Hier aber nun das genaue Abstimmungsergebnis:

18 abgegebene Stimmen - davon  
16 Stimmen für Änderung der Clubsatzung  
1 Stimme gegen Änderung der Clubsatzung  
1 Stimmenthaltung

Damit wurde die notwendige 3/4-Mehrheit eindeutig erreicht.

ANMERKUNG: Auf unserem Clubtreffen müssen wir unsere Clubsatzung unbedingt in einer überarbeiteten Form neu beschließen. Es muß z.B. der Bereich Programmbibliothek geändert werden. Es ist daran zu denken, die Frage der Vorstandschaft usw. und andere Punkte in die Satzung mitaufzunehmen.

DES WEGEN sollte sich mal jeder Gedanken machen, was seiner Meinung nach in die Satzung aufgenommen bzw. ergänzt gehört.

\*\*\*\*\*

ZUM FRAGEBOGEN:

Die Idee des Verkaufs von Programmen und Anleitungen über den CLUB 80 war eine schlechte Idee von mir. Schon bei der Veröffentlichung war mir das eigentlich klar. Aber es hat trotzdem nicht geschadet, dieses Thema anzusprechen. Auf dem Clubtreffen werden wir darüber noch mal kurz sprechen.

Der Vorschlag mit LOAD 80 fand hingegen breite Zustimmung. Wir werden auf dem Clubtreffen darüber ausführlich sprechen und entscheiden, ob wir dies offiziell über den CLUB machen (der Jahresbeitrag müßte nochmals erhöht werden), oder ob sich die Personen, die mitmachen wollen, die Kosten anteilig aufteilen (genügend Interessenten sind da, so daß der Betrag für ein Jahr nicht über 30 DM zu liegen kommt). Die rechtliche Frage ist auf alle Fälle geklärt: Solange wir LOAD 80 nicht an andere verkaufen, also nur innerhalb des CLUB 80 kopieren, solange ist dies rechtlich in Ordnung.

Das Kopierkonto wird eingeführt. Der Hartmut (als Leiter der Programmbibliothek) wird dies auf seine Weise regeln, ich (falls nötig) ev. auf eine andere.

Das Thema Programmbibliothek wurde bereits an anderer Stelle besprochen.

Ich danke für die Zusendung der Fragebogen - insgesamt erreichten mich 16 - es war eine große Hilfe für mich.

# Clubtreffen

Einiges gibt es zum Thema Clubtreffen zu sagen. Zunächst einmal: Es findet statt

vom 23.-24. März 1985

im Großraum Frankfurt. Um die Unterkunft kümmert sich der Jürgen (siehe unbedingt beiliegendes Schreiben mit Anmeldeformular). ALLE Clubmitglieder - auch die, die nicht zum Treffen kommen - senden bitte dem Jürgen so bald wie möglich - jedoch bis spätestens 31.1., das Anmeldeformular zurück.

Zum Quartier sei noch gesagt, daß sich der Jürgen selbstverständlich um günstigere Quartiere bemüht, aber den höheren Preis angesetzt hat.

Gut fände ich es, wenn Mitglieder Fahrgemeinschaften bilden. Zum einen wird die Fahrt wesentlich billiger. Ein zweiter - viel wichtigerer Punkt jedoch ist: Wir haben viele Schüler und Studenten ohne eigenes Fahrzeug im Club. Diese würden sich bestimmt freuen, wenn Sie bei jemandem mitfahren könnten. Anhand der Karte aus dem 4. Info und der Veröffentlichung der neuen Adressen in diesem Info können alle Mitglieder ersehen, 'wer mit wem' möglich wäre. (Übrigens: Der Jens wurde von mir falsch eingetragen; er wohnt in der Nähe von Stuttgart).

Damit es allen klar ist: Dieses Clubtreffen ist voll beschlußfähig. Alles, was auf dem Clubtreffen beschlossen wird, hat Gültigkeit. So natürlich auch Änderungen der Clubsatzung, Beschlüsse über die Finanzen, 80micro, etc. Im 7. Clubinfo wird ausführlich über das Clubtreffen berichtet, damit auch die anderen Mitglieder erfahren, was gelaufen ist.

Damit sich jeder auf das Treffen thematisch vorbereiten kann, bitte ich nun alle Mitglieder um folgendes: Überlegt Euch, was zu besprechen ist. Macht Euch Gedanken über unsere derzeitige Clubsatzung. Welche Probleme, Themen, Fragen sollten angesprochen werden. Was machen wir bei der 80micro....

Bitte schreibt mir, was alles im offiziellen Teil besprochen werden sollte. Ich veröffentliche dies alles im 6. Info, damit sich jeder schon auf das Treffen vorbereiten kann.

Bitte beachtet auch die Hinweise auf das Clubtreffen, die in diesem Info noch an anderen Stellen zu finden sind (z.B. Vorstandschaft).

Mehr fällt mir jetzt zu diesem Thema nicht ein - ich hoffe nun, nichts wichtiges vergessen zu haben. Ich rechne bei derzeit 33 Mitgliedern mit mindestens 15 Teilnehmern, glaube aber, daß es 20-25 sein werden.

Anmerkung: Von folgenden Mitgliedern ist mir bekannt, daß Sie mit eigenen Autos zum Treffen kommen: Hans König, Jens, Jürgen, Walter

Hermann Klaus  
Gartenstr. 22  
7401 Pliezhausen  
07127/70024

Clubmitglied seit : 02.11.84  
Punktstand = 35  
System- und Drivekonfiguration :  
Tandy Model 4P (bis vor kurzem Model I) / Epson MX 80 F/T/ 2 Laufwerke (Je 40 Spuren/ ss/ dd)  
bevorzugtes Betriebssystem : TRSDOS 6.1.1 D (Model I)

Schrewe Christian  
Fliederweg 32  
4000 Duesseldorf 31  
0203/740897

Clubmitglied seit : 30.10.84  
Punktstand = 35  
System- und Drivekonfiguration :  
TRS-80 Model III/ ? Laufwerke (40 Spuren/ ss/ dd)  
bevorzugtes Betriebssystem : TRSDOS 1.3 und NEWDOS 80

Bozek Hans Juergen  
Gut Fachenfelde  
2093 Stelle  
-

Clubmitglied seit : 11.11.84  
Punktstand = 30  
System- und Drivekonfiguration :  
Komtek 1/ 1 Laufwerk (40 Spuren/ ss/ sd)  
bevorzugtes Betriebssystem : TRSDOS/ NEWDOS 40+

Sopp Arnulf  
Wakenitzstr. 8  
2400 Luebeck 1  
0451-791926

Clubmitglied seit : 28.11.84  
Punktstand = 30  
System- und Drivekonfiguration :  
Genie I (mit div. Ein- und Umbauten) / HRG1B/ Speed-Up/ INT-Timer (regelbar) / div. Sondertasten / 3 Laufwerke (2 \* 80 Spuren/ ds/ dd/ und 1 \* 40 Spuren/ ss/ dd)  
bevorzugtes Betriebssystem : H-DOS

„Gegen Ihr Leiden ist nicht viel zu machen“, sagt der Arzt zum Patienten. „Sie haben es erbt!“  
„Na schön, Herr Doktor, dann schicken Sie die Rechnung bitte an meinen Vater!“

Perschbach Patrick  
Waldstr. 52  
5000 Koeln 91  
872118

Clubmitglied seit : 24.11.84  
Punktstand = 30  
System- und Drivekonfiguration :  
TRS-80 Model I (Level 2) / Datasette/ Expansions Interface/ Monitor/ Laufwerke (?)  
bevorzugtes Betriebssystem : TRSDOS

Boeckling Ulrich  
Am Sonnenhang 11  
5414 Vallendar  
0261/69522

Clubmitglied seit : 04.12.84  
Punktstand = 30  
System- und Drivekonfiguration :  
TRS-80 Model I (Level 2) / Expansions Interface/ HRG1B/ Drucker Itoh 8510 A/ 2 Laufwerke (Je 80 Spuren/ ds/ dd)  
bevorzugtes Betriebssystem : NEWDOS V2

Jablotschkin Rainer  
Thiekamp 29  
4780 Lippstadt 8  
-

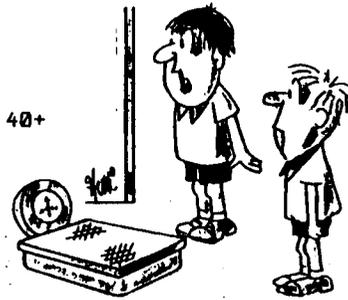
Clubmitglied seit : 27.11.84  
Punktstand = 30  
System- und Drivekonfiguration :  
Genie II (48K) / HRG1B/ Drucker Star DP 510/ 2 Laufwerke (Je 40 Spuren/ ss/ sd)  
bevorzugtes Betriebssystem : G-DOS 2.2

Robert L.A. Trost  
Aalscholverstraat 9  
6921 WR Duiven (Holland)  
08367/4736

Clubmitglied seit : Dez. 84  
Punktstand = 30  
System- und Drivekonfiguration :  
noch nicht bekannt  
bevorzugtes Betriebssystem :

Der Liebestöter

Zwei wohlgeformte Büroblondinen schauen zu, wie die neue EDV-Anlage installiert wird.  
»Schau, unser neuer Computer, meint die eine mit kaum verhelter Wut.  
»Ihm.«  
»Ersetzt vierzig Männer!«  
»Ihm.«  
»Mistding, das!«



„Ich weiß auch nicht, was das für ein Ding ist. Ich weiß nur, daß Mutti immer furchtbar wütend wird, wenn sie draufsteht!“

Boecker Dieter  
Lehmweg 4  
2930 Varnel 1  
04451/7640

Clubmitglied seit : 30.12.84  
Punktestand = 30  
System- und Drivekonfiguration :  
Genie I/ DMP 100/ Line Printer VII Laufwerk (ss/dd)  
bevorzugtes Betriebssystem : G-DOS

Grajewski Werner  
Zedernweg 29  
4220 Dinslaken  
02134/54573

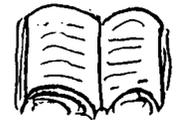
Clubmitglied seit : 02.01.85  
Punktestand = 30  
System- und Drivekonfiguration :  
Genie I (64K)/ Star DP 510/ Cassette/ Laufwerke vermutlich ab F  
eb.85 verfuegbar  
bevorzugtes Betriebssystem : -

### CLUBAUSTRITT :

Weiser Wolfgang  
Behaimstr. 12  
1000 Berlin 10  
(er hat einen Colson - Computer mit 6502 - CPU)

☹ du siehst sehr schlecht aus, lieber Freund!  
„Ich habe auch die ganze Woche kein Auge mehr zugetan. Wenn ich nämlich bis morgen keine zwanzigtausend Mark auftreibe, muß ich Konkurs anmelden!“  
„Aber, mein Lieber, warum hast du das nicht eher gesagt?“  
„Was, du kannst mir wirklich zwanzigtausend Mark leihen?“  
„Nein, das natürlich nicht! Aber ich kenne ein ausgezeichnetes Mittel gegen Schlaflosigkeit!“

## neue Bücher



Nr. 0011: TRS-80 PROGRAMS  
Tom Rugg und Phil Feldman --- Dilithium Press, Beaverton, USA  
32 BASIC-Programme (Erziehung, Anwendung, Spiele, Graphic, Mathematik und Verschiedenes) fuer Level II.  
-----  
Nr. 0012: Programme und Tricks fuer Genie I und Genie II  
Clemens Becher, Frank Seiger --- ?  
Viele Programme, Tips und Tricks fuer den Genie.

### Computertest



Eine große Elektronikfirma hat einen Computer mit einer bis dahin unvorstellbaren Speicherfähigkeit entwickelt. Der Interessent Mr. Miller steht den Beteuerungen der Fachleute skeptisch gegenüber; er verlangt einen Test.  
Der Computer wird mit sämtlichen persönlichen Daten Mr. Millers gefüttert.  
Mr. Miller stellt die Frage: »Wo befindet sich mein Bruder zur Zeit?« Der Programmierer gibt ein, und in Bruchteilen von Sekunden kommt die Antwort: »IHR BRUDER BEFINDET SICH AUF EINER GESCHÄFTSREISE IN CHIKAGO!« »Stimmt!« sagt Mr. Miller verblüfft.  
»Wo ist mein Vater?« fragt Mr. Miller mit hintergründigem Lächeln. Der Programmierer gibt ein, es dauert etwas länger, dann kommt die Antwort: »IHR VATER SITZT AM EAST RIVER UND ANGELT.«  
»Denkste!« triumphiert Mr. Miller, »mein Vater ist schon seit zwei Jahren tot!«  
Ratlosigkeit und Enttäuschung bei den Fachleuten. Die Frage wird nochmals als Kontrollfrage eingegeben. Die Antwort: »TOT IST DER GATTE IHRER MUTTER. IHR VATER SITZT AM EAST RIVER UND ANGELT!«

Da hätte ich doch beinahe die Bücher-Verleihliste vom Rainer Jablotschkin übersehen:

- Anwenderprogramme für TRS-80 und Video Genie (M. Stübs)
- Programmieren mit TRS-80 (M. Stübs)
- 57 praktische BASIC-Programme (C. Lorenz)
- Programmieren in Maschinensprache mit Z-80 (C. Lorenz)
- BASIC, Grundlagen und Beispiele, Anwendung in Mikro- und Minicomputern (Günter Abeldt)
- 77 BASIC-Programme (Lon Poole, Mary Borchers)
- BASIC Computer Spiele Band 2 (David A. Ahl)
- Programmieren mit BASIC (Byron S. Gottfried)
- CHIP Spezial 2/81 Programme II
- PCC Games, People's Computer Company
- Computergrafik, Fred Wagenknecht
- Programmieren in BASIC und Maschinencode mit dem ZX-81
- The ZX-81 Companion (Bob Marnder)
- Programmieren von Taschenrechnern 2; Lehr- und Übungsbuch für den TI-57 (Hans Heinrich Gloistehn)



Die im 4. Info angesprochene Diskette erhaltet Ihr nicht. Die Programmbibliothek muß ja erst sortiert werden und die Datenverwaltung der Clubmitglieder läuft nicht so recht. Außerdem habe ich nur von ca. 12 Leuten eine Diskette bekommen. Auf dem Clubtreffen erhalten alle Mitglieder Ihre Disketten zurück (und können sich dort auch die Clubadressen auf Diskette kopieren). Die Mitglieder, die zum Treffen nicht erscheinen, erhalten Ihre Diskette dann mit dem 7. Info. Alles klar?



Persönliche Vorstellung von Matthias Hallup:

Ich heie Matthias Hallup und bin 20 Jahre alt. Seit Oktober 1984 studiere ich Informatik an der Universitt Dortmund (und 'schlage mich dort mit Computer-Theorie herum').

Meine ersten Kontakte zum Programmieren konnte ich vor ca. 5 Jahren knpfen (programmierbarer Taschenrechner). Da ich von da ab der Computer-Faszination erlegen war, legte ich mir bald ein Video-Genie Mod. I zu. Nach und nach kamen dann Drucker und Disk-Laufwerke hinzu. Auch baute ich meinen Speicher aus, und legte mir vor kurzem eine HRG-1B (hochauflsende Grafik) zu.

Computer und Programmieren begeistern mich noch immer, doch bin ich nicht davon abhngig (kein Sklave des Computers). Meiner Meinung nach sollte der Computer dem Menschen dienen, nicht umgekehrt. Ich verspreche mir vom CLUB 80 einen regen Erfahrungsaustausch, auch hoffe ich auf Tips von erfahrenen TRS-80- und VG-Anwendern. Ferner mchte ich Leute kennenlernen, die dasselbe Computer-Hobby wie ich haben.

BERICHT BER DIE COMPUTERSZENE IN DER DDR (Matthias Hallup)

Ich fand die Berichte ber die Computer-Szene in Norwegen und UDSSR sehr interessant, denn man sollte fter ber die eigenen Grenzen hinaus schauen. Ich selber fahre fter in die DDR (Verwandte in Halle/Saale), wo ich auch in Ost-Berlin einen computerbegeisterten Freund habe.

Meiner Meinung nach ist die Computerszene in der DDR gegenber der unsrigen um ca. 10 Jahre zurck. Der Wissensdurst in Sachen Computer ist jedoch umso grer. Computer amerikanischer oder japanischer Produktion sind zwar sehr begehrt, doch verstndlicherweise nicht zu bekommen. Zwar gibt es z.B. den HP-Computer in Ost-Berlin, doch sind diese einigen ausgewhlten Firmen vorbehalten (und dort nur bestimmten Mitarbeitern). Computer werden - soweit berhaupt vorhanden - zur Automation der Produktion eingesetzt. Es geht nmlich vorrangig darum, die Produktion und deren Qualitt zu steigern. Darum investiert der Staat auch viel in die Roboter-Forschung und -Produktion. Die Firma Robotron sucht mit Ihren Produkten Anschlu an den Weltmarkt (was bei dem japanischen Vorsprung kaum gelingen drfte).

Die Hobby-Computerszene der DDR ist wie erwhnt recht klein. Zu kaufen gibt es programmierbare Taschenrechner Made in UDSSR, die aber durchaus z.B. dem TI-57 vergleichbar sind. Taschen- oder Personalcomputer sind kaum zu kaufen (und wren auch fr den Normalbenutzer nicht zu bezahlen - schon ein programmierbarer Taschenrechner kostet drben ca. 800-1000 M, ein fr Schler und Studenten nicht aufzubringender Betrag). Trotzdem gibt es drben ein paar Exemplare japanischer Taschencomputer (z.B. Sharp PC-1251), die von Dienst-Reisen (z.B. Ungarn) 'eingeschmuggelt' werden. Software, Tips und Hardware stehen natrlich hoch im Kurs, und jedes Computer-Buch ist hei begehrt.

Aber auch hier macht Not erfinderisch: Interfaces u.a. werden selber gebastelt, obwohl man sich die Elektronik-Teile dafr von Freunden im Westen besorgen lassen mu. Auch gibt es in der DDR keine privaten Computer-Clubs wie hier, Informationen werden mndlich weitergegeben und sind nur einem kleinen Kreis vorbehalten. Jugendliche in der DDR sehen mit Wehmut auf die Computer-Szene im Westen, und ein TRS-80- oder VG-Computer zu besitzen wre eine absolute Sensation fr Sie. Hier sieht man doch wieder, welche Computer-Vorteile wir besitzen, obwohl wir uns dessen oft gar nicht bewut sind. Wenn man mal wieder neidisch auf die Computer-Szene in den USA sieht, sollte man sich dies zum Trost vor Augen halten.

Ich heie Arnulf Sopp und bin 36 Jahre alt. Zwar gehe ich zur Schule, stehe aber auf der feindlichen Seite des Katheders. Nachdem ich jahrelang als kultivierter Mensch Computer verachtet hatte, wobei jedoch meine Taschenrechner immer leistungsfhiger wurden, machte ich irgendwann Ngel mit Kpfen und kaufte mir ein gebrauchtes Video-Genie. Das war der Anfang vom Ende. Inzwischen bin ich vollkommen computerkrank und verbringe nahezu meine ganze Freizeit an der Denkprothese. Meine jetzige Anlage sieht so aus: Genie I (83er Modell) mit diversen Um- und Einbauten, Laufwerke 2mal 80/DS/DD und 1mal 40/SS/DD, Gemini-10X, EG 64 MBA (wohl die beste Banking-Logik auf dem Markt zum Preis der schlechtesten: DM 195,-), HRG 1b. Mein bevorzugtes Betriebssystem ist tatschlich "mein" Betriebssystem: Aus G-DOS 2.1b machte ich mit etlichen Zaps und neuen SYS-Files H-DOS, das neben vielen anderen Features das Ansteuern des MBA und der HRG sehr bequem macht. H-DOS kommt von "The HACKTORY", meinem Spitznamen im Computer-Schriftverkehr. Das klingt wie eine Firma und ist auch ungefhr so gemeint. Wenn jemand aus dem Club in der Z80-Maschinensprache fit ist und Lust hat, bei solchen Entwicklungen mitzumischen, wrde ich mich ber Post oder einen Anruf freuen.

F  
R  
A  
G  
E  
N  
  
P  
R  
O  
B  
L  
E  
M  
E

- Hat Jemand das FORTRAN-Reference-Manual zum FORTRAN-Compiler von Microsoft ?
- Hat Jemand ein vollstndiges ALCOR-PASCAL Manual ?
- Hat Jemand fuer das Modell I einen Zeichengenerator mit Umlauten ?  
Wenn Ja, ist es ein EPROM ?  
Kann es kopiert werden ?  
Wo auer bei L.Roeckrath kann man noch einen beziehen ?  
Ist evtl. der vom Modell III oder Komtek verwendbar ?  
Wer hat eine Schaltung des Videoteils von Modell III ?
- Hat Jemand sein Modell I auf hoehere Taktfrequenz umgestellt?  
Wenn Ja, welche Erfahrungen hat er gemacht ?

Josef Konrad

- Wer hat die CT Nr. 5/84 ?  
Bitte sofort mit dem Hartmut in Verbindung treten!
- Wer hat Dotwriter-Zeichenstze ? Diese werden vom Jens 'gesammelt' und passen fr die HRG und auch zum Briefschreiben.
- Wer kann dem Patrick ein Interfacekabel (40-derig) besorgen?

Bitte helft dem Hans Knig weiter! Er hat folgende Probleme:

Mein Problem ist, System Kassetten auf Disk zu bernehmen. Ich habe schon viel versucht, aber ich schaffe es nicht.

Genauso habe ich Schwierigkeiten mit /SRC Files, also Quellcodes zum CMD File zu machen. Ich kenne zwar den Ablauf (mit EDTASM) und im Directory steht der File auch, aber dann klappt nichts mehr, nur die Tren, die ich heftig schliee. Auch hier htte ich gerne einen Tip.

## Ein Auszug eines Briefes vom Matthias Lallup

14.) Vorschlag: Was hältst Du von der Einrichtung einer Diskussions-Seite, z.B. zum Thema "Computer und Kinder"? Ich selber möchte meine Meinung zu diesem Thema kurz schildern: In letzter Zeit ist viel von sog. Computer-Kids die Rede gewesen. Diese Kinder bzw. Jugendliche haben zwar viel Computer- und Programmierwissen, doch wie steht es mit Ihrem Allgemeinwissen? Für mich ist es erschreckend, wenn Kinder im Vorschul-Alter auf den Umgang mit Computern getrimmt werden. Sie beherrschen zwar perfekt das Programmieren, doch können Sie sich auch in normalen Deutsch verständlich machen? Letztlich ist eine Generation, die nur mit dem Computer aufgewachsen ist, leicht manipulierbar. Ein autoritärer Staat könnte diese Menschen über Computer-Informationen in die Unfreiheit lenken, Kritikfähigkeit und zwischenmenschliche Kommunikation wären ausgeschaltet, Verständigung nur noch über Bildschirm möglich. Ich stelle diese Schreckensvision nicht deshalb dar, weil ich computerfeindlich bin. Im Gegenteil: Ich bin für technischen Fortschritt und Computereinsatz. Doch sollte dies in Maßen geschehen. Computerbegeisterung ist nicht schädlich, doch zu viel davon kann schwere Folgen haben. Seid Ihr von Freunden nicht auch schon verständnislos angeschaut worden, wenn Ihr Erklärungen in Computer-Chinesisch abgegeben habt? Es besteht also die Gefahr, daß der einseitig Computerbegeisterte den Kontakt zur Umwelt verliert. Dies sollte und darf man Kindern nicht zumuten. Man darf Kinder nicht zu früh das abstrakte Denken per Computer beibringen. Es ist unbedingt nötig, Jugendliche mit Computern vertraut zu machen und Technikfeindlichkeit abzubauen, doch darf dies nicht zu einer kritiklosen Computerliebe führen. Man sollte Kinder im kritischen Umgang mit Computern schulen, und Sie nicht zu Computer-Fachidioten machen. Nur wer selber zu denken gelernt hat, wird nicht Sklave des Computers werden. Computer-Begeisterung ist schön und nützlich, doch blinde Begeisterung hat noch nie genutzt, sondern immer Schaden hervorgerufen (Genügend Beispiele dafür findet man in der Geschichte). Übrigens <sup>hat</sup> ~~man~~ blinde Computer-Feindlichkeit genauso verheerende Folgen. Nur der kritische, aufgeschlossene und informierte Mensch wird die Probleme der Zukunft meistern. Was denkt Ihr dazu? Welche Meinung habt Ihr zu "Computer und Kinder", "Computer und Gesellschaft" oder "Roboter und künftige Arbeitswelt"? Gerade Computer-Begeisterte wie wir haben eine besondere Verantwortung gegenüber der Gesellschaft und sollten <sup>unseren</sup> ~~ihren~~ Beitrag zur Lösung der Zukunfts-Probleme leisten!

15.) Werbung: Zum Schluß möchte ich noch ein wenig (einseitige?) Werbung betreiben. Ob Du Sie abdruckst, entscheide bitte selbst.

a.) Für alle diejenigen, die noch Software suchen, kann ich den Software-Händler (Privat-Anbieter) Hanke nur empfehlen. Er hat in seiner Programmbibliothek ca. 1300 Programme für TRS-80/VG (Sprachen, Utilities,

Anwender, Spiele etc.), und es dürfte für jeden etwas dabei sein. Auch werden Kopien von vielen Software-Anleitungen zum Kauf angeboten (gegen Erstattung der Kopier- und Portokosten). Wenn Ihr Hanke DM5.- als Schein zusendet, erhaltet Ihr seine umfangreiche Software-Liste. Ich teile Euch dieses nicht mit, weil ich prozentual am Verkauf beteiligt wäre (schön wärs!), sondern weil ich mit Hanke gute Erfahrungen gemacht habe (Wer schlechte hat, sollte dieses bitte mitteilen!). Ich finde es wichtig, eine günstige Quelle für Programme zu kennen, und möchte Euch daher die Adresse mitteilen: H.R. Hanke; Wiener Str. 127; A-2620 Neunkirchen. Für einige wird diese Adresse ein alter Hut sein, doch vielleicht konnte ich einigen einen Tip geben.

b.) Im vorigen Info wurde die hochauflösende Grafik HRG-1B lobend (zu recht!) erwähnt. Leider fehlte eine Angabe von Software-Bezugsquellen für diese Grafik. Zwar wurde das Treiber-Programm von RB-Elektronik und der sog. Super-Treiber erwähnt, doch möchte ich dazu folgendes kritisch anmerken: Das mitgelieferte Treiberprogramm enthält nur einen sehr dürftigen Befehlssatz, und die Beschreibung des Super-Treibers ist als katastrophal und unverständlich zu bezeichnen (Mir wurden zu diesem Programm nur drei Seiten mitgeliefert, ohne Beispiele und Erklärungen). Meiner Meinung nach ist der Super-Treiber sein Geld keineswegs wert. Wesentlich besser sind folgende zwei Programme für die HRG-1B (leider nur für Disk zu haben):

a.) GRAPE für TRS-80/VG mit HRG-1B und Drucker. Dieses Programm würde ich als sehr gut bezeichnen. Der Autor Winter hat die Möglichkeiten der Grafik voll ausgenutzt (z.B. 3D-Zeichnen), und nimmt Anpassungen an Computer-Typ und Drucker vor. Nähere Informationen und Preis Auskunft bei M. Winter; Im Steinergarten 23; 7000 Stuttgart 80.

b.) HRG-PACK von B. Wedell. Programm ist sehr empfehlenswert, zwar nicht ganz so gut wie GRAPE, doch weitaus besser als Supertreiber. Info und Preis bei B. Wedell; Postfach 911265; 3000 Hannover 91 (Anpassung für Computer- und Druckertypen möglich!).

16.) Vorschlag: Einrichtung einer Soft- und Hardware-Test-Seite für TRS-80/VG. Hier könnten Leser Ihre persönlichen Erfahrungen mit Programmen beschreiben, Vor- und Nachteile darlegen. Auch könnten Bau-sätze etc. beschrieben und beurteilt werden. Natürlich hängt so etwas stark vom Engagement und vor allem der Zeit der Leser ab (Test-Berichte erfordern Mühe und Aufwand!).

Okay! Was meint Ihr dazu?

8. Jan. 1985

Dieter Böcker  
Lehmweg 4, 24 (04451) 7630  
2930 Varel 1

Varel, 30.12.84

Sehr geehrter Herr Wagner!

Als erstes möchte ich mich noch nachträglich für die schnelle Beantwortung meines Schreibens bedanken.

Nun kurz zu meinen Problemen:

- 1) Mit einem Kollegen zusammen mache ich Basic-kurse an einer Volkshochschule. Die Hardware stellen wir privat zur Verfügung. Wir besitzen mehrere Video genie I, von denen 2 mit Floppy unter TRS DOS u. G DOS arbeiten. Wir wären daran interessiert, noch zwei Laufwerke incl. Controller günstig zu erwerben. Vielleicht gibt es Club-Mitglieder, die uns verkaufen wollen oder die eine Adresse kennen, wo man preiswert eins erwerben kann.
- 2) Wir sind hauptamtlich Professoren an der Fachhochschule Wilhelmshaven. Da dort an Mikrocomputern überwiegend mit C/PT gearbeitet wird, sind wir daran interessiert, auch unsere privaten Videosmie auf C/PT umzu-rüsten. Deswegen hatte ich mich ja an Sie gewandt. Für Tips u. Adressen wo

wir die erforderliche Hard- u. Software günstig bekommen können, wären wir sehr dankbar.

Weiter: Im Rahmen eines Kooperationsvertrages arbeiten wir (C/PT W'haven) mit einer ungarischen Hochschule für Elektrotechnik in Budapest zusammen. Dort wird in der Mikrocomputer-Ausbildung der Video genie I benutzt. Er wird in Ungarn in Lizenz nachgebaut und ist der einzige Mikrocomputer an den die Ungarn ohne Devisenschwierigkeiten herankommen. Für eine Zusammenarbeit auf diesem Gebiet müssen wir uns also anpassen und auch den Video genie benutzen. Wir möchten also für die Fachhochschule zwei bis drei Video genie I anschaffen. Da er bei uns nicht mehr gebaut wird, haben wir da einige Schwierigkeiten, da wir aus haushaltsrechtlichen Gründen keine gebrauchten Rechner aus Privathand kaufen dürfen. Händleradressen, wo man noch Video genie I kaufen kann (auch gebraucht), wären für uns also sehr interessant.

Abgesehen von diesen aktuellen Problemen sind wir natürlich an Software u. Erfahrungsaustausch immer interessiert. Mein Aufnahme-Formular liegt bei.

Mit freundl. Grüßen!  
Dieter Böcker

Magende noch in das Info aufrechnen konnte ich dieses Schreiben unseres neuen Mitglieds -  
Über dem Dieter helfen kann, wendet sich direkt an ihn.

COBOL eine Einfuehrung

Teil: 01

Was ist COBOL ?

Diese Frage mag sich der geneigte Leser stellen - nun, um es kurz zu machen: COBOL ist eine Programmiersprache die in den 60 ziger Jahren entwickelt wurde und fuer kommerzielle Loesungen eingesetzt wird.

Die Programmiersprache COBOL ist eine bei Home- u. Personalcomputern unuebliche Sprache - das mag vielleicht daran liegen, dass sich gerade diese Gruppe weniger mit kommerziellen Loesungeen beschaeftigt als mehr mit Spielen, kleine Anwenderprogramme und eventuell noch Steuerungen fuer Radios, Luftungen usw.

Warum also COBOL in CLUB 80 ?

Ganz einfach, in meiner Eigenschaft als Systemberater und Anwendungsprogrammierer in einer mittleren Firma mit Sitz in Recklinghausen moechte ich gerne etwas von meinem Wissen hinsichtlich der COBOL-Programmierung im allgemeinen und speziell fuer Anwendungen auf dem TRS 80, Video-Genie, oder Komtek abgeben.

Wenn man die einschlaegige Fachliteratur durchblaetert so findet man fuer unsere Rechner BASIC (Compiler), PASCAL, FORTRAN und eventuell noch FORTH - aber COBOL ?

Nun, vielleicht liegt es auch daran, dass COBOL eine sehr stark strukturierte Sprache (aber mit ganz fantastischen Moeglichkeiten) ist, mit der sich ein Programmierer, der mal eben ein Programm auf die Schnelle entwickelt, einfach nicht anfreunden kann.

Auch ich muss an dieser Stelle sagen, dass ich mich am Anfang recht schwer, nicht mit der Sprache, aber mit der Organisation und Struktur getan habe.

Bevor wir ins Eingemachte gehen, soll hier ein kleiner Vergleich zwischen einer kleinen Anwendung in Basic und Cobol gezeigt werden:

Anhand eines kleinen Programmbeispiels soll ein Vergleich zwischen Basic und Cobol gezogen werden - uebrigens der einzige Vergleich, denn Cobol kann eigentlich mit keiner gaengigen Sprache fuer mathematisch-naturwissenschaftliche Zwecke geeignete Sprache verglichen werden - Cobol ist von seiner Anwendung hauptsaechlich fuer kaufmaennische Aufgaben entwickelte Sprache (was aber nicht heissen soll, dass Cobol nicht nutzbringend fuer CLUB 80 Mitglieder sein kann).

Das Programm soll einen einfachen Taschenrechner mit den vier Grundrechenarten simulieren - jeder wird bereits schon jetzt ein Konzept im Kopf haben wie er das Problem in Basic angeht:

Es sieht bestimmt so aehnlich aus wie ich es nun darstellen will:

```
- Erst einmal CLS
- so, jetzt PRINT "Taschenrechner (4 Funktionen)"
- und dann noch "=====
- eventuell noch ein PRINT
- PRINT "Geben Sie die Rechenart an (+,-,*,./)
- INPUT "Funktion .... ", FU (Dollar)
```

Spaetestens an dieser Stelle folgt ein RUN, um einmal anzuschauen, wie das "Programm" funktioniert.

Und gerade das geht nicht in Cobol!

Das gesamte Problem muss im Rahmen eines Programmes codiert, compiliert (und gelinkt) werden und kann erst dann in Verbindung mit der RUN-TIME "zum Laufen gebracht" werden.

An dieser Stelle moechte ich nicht ein BASIC Programm zur Loesung des o.g. Problems anführen, ein jeder wird in Gedanken dieses Programm schon fertig haben, so dass ich gleich auf die Cobol-Loesung eingehen kann:

IDENTIFICATIN DIVISION.

PROGRAM-ID.	RECHNER.
AUTHOR.	H TRAPP.
INSTALLATION.	CLUB 80.
DATE-WRITTEN.	30-10-84.
DATE-COMPILED.	TODAY.
SECURITY.	KEINE.
REMARKS.	SIMULATION EINES TASCHENRECHNERS.

\*

ENVIRONMENT DIVISION.

WORKING-STORAGE SECTION.

77 WERT-1	PIC 9999.
77 WERT-2	PIC 9999.
77 ERGEBNIS	PIC 99999.
77 WAHL	PIC A.

\*

PROCEDURE DEVISION.

ANFANG.

```
DISPLAY "RECHENPROGRAMM", LINE 1, POSITION 1, ERASE.
ACCEPT WERT-1, LINE 3, POSITION 1.
DISPLAY "+" LINE 3, POSITION 6.
ACCEPT WERT-2, LINE 3, POSITION 8.
DISPLAY "=", LINE 3, POSITION 13.
ADD WERT-1, WERT-2 GIVING ERGEBNIS.
DISPLAY ERGEBNIS, LINE 3, POSITION 15.
DISPLAY "WEITERE RECHNUNGEN (J) E=ENDE", LINE 5, POSITION 1.
```

NOCHMAL.

```
ACCEPT WAHL, LINE 5, POSITION 32.
```

```
IF WAHL = "J" GO TO ANFANG.
```

```
IF WAHL = "E" GO TO ENDE.
```

```
GO TO NOCHMAL.
```

```
ENDE.
```

Wenn man sich aufmerksam das Programm anschaut, so kann man auch als "Noch-nicht-Fachmann" erkennen, das es sich bei dem Beispiel nur um ein Additionsprogramm handelt. - Und dafuer so ein Aufwand ??

Ja gut, man haette sich am Anfang vier oder fuenf Zeilen sparen koennen, aber alles andere ist ein unbedingtes "MUSS".

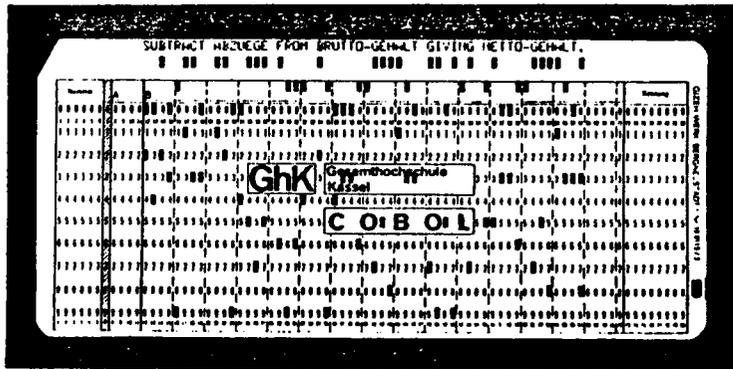
Also ist das Programmieren in Cobol ein recht grosser Schreibaufwand - mit dem Vorteil allerdings, dass, wie auch der "Noch-nicht-Fachmann" bestaetigen wird, das Programm sehr gut lesbar und anschaulich (Dokumentation) ist.

Ein Cobol-Programm dokumentiert sich also selbst, REMARKS sind fast ueberfluessig (erfreulich fuer genervte Basic-Programmierer).

Noch einige Worte zum Programmaufbau:  
Das Programm, (es ist eigentlich der SOURCE-CODE) wuerde beim Compiler-Lauf mit zahlreichen Fehlern aus der Memory "rausgeworfen" werden.

Neben der normalen Syntax ist auch die Position der einzelnen Befehlswoerter von ausschlaggebender Rolle.

Dazu ein kleiner Exkurs in die Geschichte der Lochkarte.  
(Ich gehe an dieser Stelle nur auf die Dinge ein, die fuer Cobol-Programmierung von Bedeutung sind, Behandlung von Ziffer- und Zonentell, sowie einige spezielle Eigenschaften sind hier uninteressant).



Um die Positionierung der einzelnen Befehlswoerter zu verstehen, muss man von einer Lochkarte wissen, dass ihr Informationsgehalt maximal 80 Zeichen betragen kann:

Spalte 01-06	Zeilennummer
Spalte 07	Kennzeichen
Spalte 08-11	Zone A
Spalte 12-71	Zone B
Spalte 72-80	Bemerkungen

Fangen wir von hinten an:  
Spalte 72-80 ist fuer Bemerkungen zustaendig! In der Praxis heisst das, dass der Compiler die Zeichen in der Spalte 72-80 nicht beachtet - somit sind sie auch fuer uns nicht von Interesse.

Zone «B» ist fuer die verschiedenen Befehle zustaendig - man kann auch sagen, das eigentliche Programm steht zwischen Spalte 12 und 71.

Die Zone «A» nimmt bei Cobol eine Sonderstellung ein: Sowohl einzelne Befehle aus den ersten drei DIVISIONS, die DIVISIONS selbst, PARAGRAPEN und SECTION's beginnen (!) in Zone «A». (Es sind nicht allzu viele Befehle, die in Zone «A» beginnen (muessen) - man muss sich die Befehle mit diesem Merkmal einpraegen sonst sucht man oft vergeblich stundenlang nach einem Fehler in der Syntax der ueberhaupt nicht vorhanden ist.

Spalte 7 allein hat nur zwei Aufgaben: Kennzeichen '\*' fuer REMARKS und Kennzeichen '-' fuer Verbindungen zwischen den einzelnen Befehlszeilen.

Steht in Spalte 7 ein '\*' so wird die gesamte Zeile vom Compiler nicht beachtet, (wie REM in Basic).  
Steht in der 7. Spalte ein '-', so bedeutet das fuer den Compiler, dass diese Zeile an die vorhergehende anzuhangen ist. (kommt nur bei Ausgaben auf den Bildschirm oder den Drucker vor - und auch nur dann, wenn die Ausgabezeile + Befehlswort (DISPLAY oder WRITE) laenger als Zone «B» ist.

Spalten 1 bis 6 wiederum werden vom Compiler nicht beachtet, der EDITOR setzt hier seine Zeilennummerierung ein (die bei Cobol Programmen nicht noetig sind).

Mit diesen Ausfuehrungen im Hinterkopf und den Blick auf unser Beispielprogramm wird der aufmerksame Leser an dieser Stelle sofort den Einwand bringen, dass das kurze "Source-Listing" nicht in der fuer Cobol spezifischen Form geschrieben ist.

Ganz richtig, wenn dieser Source-Code kompiliert werden wuerde,

waeere das totale Chaos perfekt - es sollte damit auch nur ein kurzer Ueberblick ueber die Lesbarkeit eines Cobol-Source-Codes gegeben werden.

So, genug der Vorrede, wenden wir uns der ersten von vier DIVISION's zu aus der ein Cobol-Programm besteht.

#### IDENTIFICATION DIVISION.

PROGRAM-ID.	Programm-Name
AUTHOR.	Kommentar
INSTALLATION.	Kommentar
DATE-WRITTEN.	Kommentar
SECURITY.	Kommentar

Die Identification Division besteht aus sechs Anweisungen, zwei davon sind fuer einen korrekten Compiler-Lauf unbedingt erforderlich, die letzten vier Anweisungen koennen entfallen.

Im Hinblick auf Allgemeinguetigkeit sollte man sich jedoch angewoehnen, auch die anderen vier Anweisungen (AUTHOR, INSTALLATION, DATE-WRITTEN und SECURITY) anzugeben, ein so grosser Schreibaufwandd ist es nicht und man hat alle benoetigten Angaben ueber den Urheber, Datum und Installation an der Stelle wo man sie haben sollte und nicht irgendwo zwischen ein paar REMARKS.

Man beginnt also beim Schreiben eines Programms bei der IDENTIFICATION DIVISION. (in Zone A, Spalte 8)

gefolgt von:

PROGRAM-ID.	Programm-Name	(Spalte 8)
AUTHOR.	Kommentar	(Spalte 8)
usw.		

Was bedeutet nun IDENTIFICATION DIVISION ?

Die IDENTIFICATION DIVISION leierte das Programm ein und stellt die erste von vier DIVISIONS dar - Kennzeichen fuer den Compiler, dass hier der Source Code beginnt.

PROGRAMM-ID.  
bedeutet eigentlich PROGRAM IDENTIFICATION, und fordert einen Programm-NAMen. Dieser Programmname kann 32 Stellen lang sein, relevant sind allerdings nur die Stellen 1 bis 6. Mit den ersten sechs Zeichen weist der Compiler bei Programmfehlern in den Source Codes hin oder aber bei der Run-Time auf Fehler waehrend des Programmlaufs (wichtig, wenn man mit CALLS arbeitet, zur Kennzeichnung, in welchem Programm denn nun der Fehler aufgetreten ist).

AUTHOR.  
verlangt den Namen des Erstellers - wichtig: Abkuerzungen mit Punkten sind nicht erlaubt.  
Falsch : H.TRAPP  
Richtig: H-TRAPP  
Man sollte sich von vornherein bei Cobol angewoehnen nur mit Bindestrichen und nicht mit Punkten zu arbeiten. (Diese haben, wie wir gleich sehen werden,eine andere Bedeutung).

INSTALLATION.

Adresse des Erstellers oder freier Kommentar.

DATE-WRITTEN.  
Datum, aber bitte in der Form 05-06-1984 (fuer den 5. Juni)

SECURITY.  
hier gibt man im allgemeinen: KEINE an.

Alle Anweisungen haben es gemeinsam, dass sie in Spalte 8, Zone A anfangen.  
Die einzelnen Statements (PROGRAM-ID,AUTHOR usw.) werden mit einem Punkt '.' zur Ende-Kennzeichnung abgeschlossen.  
Der Programm-Name und die Kommentare koennen direkt hinter dem Punkt des Statements geschrieben werden - hier benutzt man allerdings zur besseren Uebersichtlichkeit die Moeglichkeiten des EDITOR's (mit dem der Source Code erstellt wird) mit dem man z.B. den uebernaechsten TAB anspringt und alle Eingaben in der gleichen Spalte beginnen laesst.  
Auch hier muss zur Ende-Kennzeichnung ein Punkt gesetzt werden.

Beispiel eines COBOL-Source Codes:

```
1----67A---B----- usw. ... --72
      IDENTIFICATION DIVISION.
      PROGRAM-ID.          DEMO-PROGRAMM.
      AUTHOR.              H-TRAPP.
      INSTALLATION.        KRANICHSTR 46 4270 DORSTEN.
      DATE-WRITTEN.        07-11-1984.
      SECURITY.             KEINE.
      *
      *Hier kann ein Kommentar stehen.
      *
```

Sie erinnern sich ??  
Der Stern '\*' in Spalte 7 laesst jegliche Kommentare zu. Kommentare und DISPLAY (auf den Bildschirm) lassen Gross- u. Kleinbuchstaben zu.  
Statements und Anweisungen muessen in Grossbuchstaben angeschrieben werden.

Mit dem Stern '\*' in Spalte 7 kann man den Source-Code noch uebersichtlich gestalten oder, wie in dem ersten Beispiel, zusaetzhliche Dokumentation erstellen werden.

z.B. \*DATE-COMPILED. 10-11-1984.

Die DATE-COMPILED Anweisung ist ueberigens auf Grossrechenanlagen Pflicht.

Das waer's fuer Teil 1. - Im naechsten Teil gehen wir auf die zweite und dritte DIVISION ein - soviel sei schon hier verraten:

ENVIRONMENT DIVISION. (Maschinen- u. Datei (Drucker) Definition).

DATA DIVISION. (Erklaerung der Daten und Arbeitsfelder)

## Wer hat Angst vor A S S E M B L E R ?

BASIC ist eine sehr bequeme Sprache, bei der sich mit einem Wort ein ganzer Rattenschwanz von Maschinenspracheprogrammen aufrufen läßt. Ich habe keine Ahnung, um wieviel ich mich nach oben oder unten verschätze, aber die simple Anweisung PRINT dürfte gut und gerne 1 kB des Interpreters benötigen. Hier liegt auch der Grund, warum BASIC so langsam ist.

Beim Erstellen eines Programms ist es genau umgekehrt. Das liegt daran, daß wir beim Programmieren einer Maschinenspracheroutine in Bahnen denken, die uns sehr ungewohnt sind. Da heißt es meistens nicht, wie in BASIC, "Tu was!", sondern überwiegend "Tu was wohin!". Das bedeutet folgendes: Da unser Z80 nicht rechnen kann, muß er nach einem möglichst sinnvollen Algorithmus so lange Daten schaufeln, bis an der vorher bestimmten Stelle der richtige Wert steht.

Hier ein konkretes Beispiel. An der Bildschirmstelle 0, also an der Speicherstelle 3C00h (15360d) soll der Großbuchstabe A angezeigt werden. Das ist ASCII 41h bzw. CHR\$(65). In BASIC ist das fix getan mit dem Befehl

```
PRINT$0,"A"
```

Der Interpreter erkennt den PRINT-Befehl und sucht nun zunächst nach weiteren Operanden wie TAB, USING, ,, ;, ( und eben auch \$ (at, Klammeraffe). Schon schneller geht es, wenn wir den Interpreter nicht wie oben beschrieben auffordern, etwas zu tun, sonder gleich etwas irgendwohin zu tun:

```
POKE15360,65
```

Jetzt passiert zwar noch immer so allerhand, aber ohne Umschweife wird nun der Code 41h (65d, "A") nach 3C00h geladen. Auch hier habe ich nicht im ROM nachgezählt, aber es dürften vielleicht 200 Bytes sein, die zum Syntaxcheck und zum eigentlichen Ladevorgang durchlaufen werden.

Am schnellsten geht es in Maschinensprache. Das hat man auch schon bei Microsoft gemerkt, denn der Interpreter ist ein gewaltiges Maschinenprogramm. Wie er die Bildschirmcke lädt, weiß ich nicht. Vielleicht so:

```
LD      A,41H
LD      (3C00H),A
```

Der erste Befehl setzt sich aus zwei Bytes zusammen, der untere aus dreien. Wir haben also nun mit 5 Bytes in einer unvorstellbar kurzen Zeit das A angezeigt.

Und das war ganz simpel. Zunächst wurde ein Register (soviel wie Speicher) der CPU mit dem Code geladen (LD = load). Es ist der Akkumulator oder Akku. Es gibt noch eine ganze Reihe weiterer Register, auf die ich jetzt aber nicht eingehen will. Anschließend wurde dieser Code an die Speicherstelle 3C00h weitergegeben. Den Befehl

```
LD      (3C00H),41H
```

kennt die CPU leider nicht, sonst wäre es noch simpler und schneller gegangen.

Es hat übrigens keinen Sinn, diese Assembler-Befehle ohne einen Assembler einzugeben. Da der BASIC-Interpreter sie nicht kennt, kommt nur ein ?SN-Error dabei heraus.

Diesen Info-Beitrag schreibe ich nicht, um meine neue Schreibmaschine auszuprobieren. Er hat vielmehr eine Vorgeschichte und einen ganz bestimmten Grund.

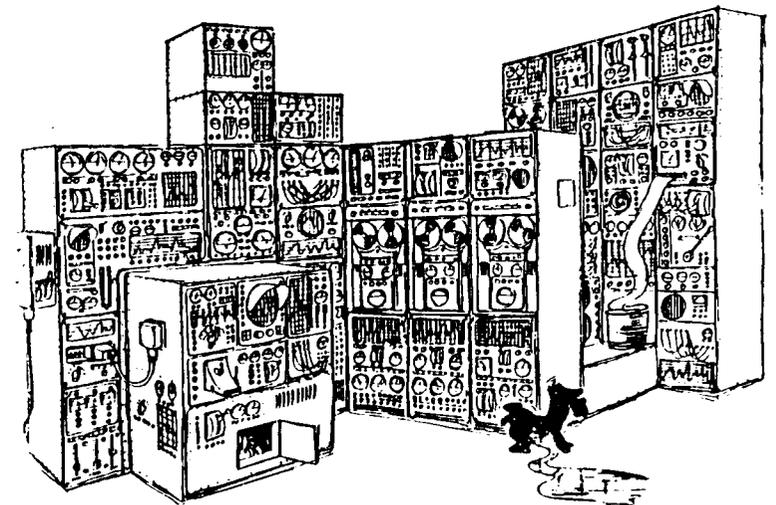
Die Vorgeschichte: Als Noch-nicht-Mitglied des Club 80 wollte ich mich informieren, was bei euch so läuft und bestellte bei Günther die bisherigen Infos. Bei der Lektüre fiel mir auf, daß kein einziges selbstgeschriebenes Assemblerprogramm drinsteht. Ich schlug vor, mit einem zunächst einfachen Artikel zu diesem Thema die Lücke bewußt zu machen und dann eure Reaktionen abzuwarten. Wenn es sich zeigt, daß im Club Interesse an der Maschinensprache besteht, will ich gerne noch mehr (aber nicht mehr so einfache Dinge) bringen.

Mein pädagogischer Ehrgeiz ist allerdings gering - zumindest außerhalb meiner beruflichen Tätigkeit. Günther schlug nämlich vor, einen Assemblerkurs ins Info zu bringen. Dergleichen gibt's im Buchhandel von Berufeneren als mir. Besonders empfehlenswert ist z. B. die Serie "View from the Top of the Stack" in The First Book of 80-US.

Der Grund: Vielmehr ist es mit diesem Artikel meine Absicht, euch für die Maschinensprache zu interessieren, indem ich euch zeige, daß das keine Zauberei ist. Wie jede Sprache hat Assembler natürlich auch ihren Wortschatz, ihre Rechtschreibung, ihre Syntax, genau wie BASIC. Sie sind ebenso einfach erlernbar wie BASIC. Und irgendwann hat man sich auch an die etwas umständlich anmutende Denkakrobatik gewöhnt, die erforderlich ist, um z. B. 7 und 5 zu multiplizieren, denn die CPU kann, wie gesagt, nicht (jedenfalls fast nicht) rechnen.

Wenn ihr jetzt Blut geleckt haben solltet, laßt es Günther wissen. In der Folge schreibe ich dann (oder eben nicht, wenn es keinen interessiert) weitere Beiträge, in denen ich Maschinenspracheprogramme vorstelle und so kommentiere, daß auch der Laie versteht, welchen Weg sie gehen. Die Programme restlos zu durchschauen, erfordert allerdings, daß ihr per Buch oder wie auch immer diese Sprache erlernt. Und genau das will ich mit diesem Beitrag bezwecken.

Arnulf Sopp



In der Programmbibliothek existieren zwei Versionen eines Programms mit dem der BASIC-Befehl RESTORE erweitert wird. RESTOREN/BAS und RESTOREN/CMD ermoeglichen es, den DATA-Zeiger auf eine bestimmte Zeile innerhalb eines DATA-Feldes zurueckzustellen (z.B. RESTORE 140). Dies kann bei bestimmten Anwendungen sehr nuetzlich sein. Auf alle Faelle ist es unpraktisch, dass beide Programme extra geladen und durch das Setzen von HIMEM geschuetzt werden muessen.

Eine elegantere Loesung des Problems sehe ich darin, dass relativ kurze Maschinenprogramm ins BASIC einzubinden. Das hat zur Folge, dass der RESTORE n - Befehl ein fester Bestandteil des Diskbasic wird.

Das Maschinenprogramm RESTOREN/CMD disassembliert:

```

21E17F LD HL,7FE1H
220440 LD (4004H),HL
C3CC06 JP 06CCH
E3CC06 EX (SP),HL
7C LD A,H
FE1D CP 1DH
E3 EX (SP),HL
C2781D JP NZ,1D78H
D7 RST 1DH
FE90 CP 90H
C2791D JP NZ,1D79H
CD5B1E CALL 1E5BH
E5 PUSH HL
CD2C1B CALL 1B2CH
0B DEC BC
ED43FF40 LD (40FFH),BC
E1 POP HL
C1 POP BC
E1D JP 1D1EH

```

Spalte 1 Spalte 2 Spalte 3

Das Programm liest im Bereich 7FD8H bis 7FFDH mit dem Entry-Point 7FD8H.

Das Programm ist in zwei wesentliche Teile unterteilt. Block 1 (Zeile 1-3) schreibt die Adresse der eigentlichen Befehlsweiterung (Block 2, Zeile 4-Ende) in die Speicherstelle 4004H. In 4003H steht ein C3H, was einen Sprung zur nachfolgend angegebenen Adresse (jetzt 7FE1) zur Folge hat. Zeile 3 des ersten Blocks bewirkt einen BASIC-Warmstart.

Wird jetzt in einem BASIC-Programm der Befehl RESTORE benutzt, wird statt der "alten" RESTORE-Routine (Addr 1D91H), die neue RESTORE n-Routine (Addr 7FE1H) angesprungen.

Der Block 2 soll uns hier funktionsmaessig nicht interessieren. Um den neuen Befehl ins BASIC einzubinden, muss man folgendermassen vorgehen:

1. Suche einen freien Bereich im BASIC/CMD (mit Superzap).
2. Suche die Ladeadresse des freien Bereichs. Dazu muss man vom freien Bereich aus rueckwaerts das naechste 01-Byte finden. Diesem 01-Byte folgt ein 1-Byte-Wert, der angibt, wieviel Bytes als naechstes geladen werden sollen (00=256 Bytes). Die naechsten beiden Bytes enthalten die Ladeadresse des zu ladenden Blocks im Format LSB/MSB. Beispiel: 1. 2. 3. 4.  
01 00 C9 65

1. Beginn des Ladeblocks
2. Anzahl der zu ladenden Bytes (hier 256)
3. und 4. Ladeadresse des Blocks LSB/MSB (hier also 65C9H)
3. Nun gehe ins BASIC und lasse einisse Programme laufen. Ueberpruefe, ob der gefundene, ungenutzte Bereich auch tatsaechlich frei ist und nicht vom BASIC aus ueberschrieben wird (z.B. Buffer usw.). Hier hilft der im NEWDOS eingebaute Debugger und ein ROM-Listing.

Bis hier ist alles nur Vorseplaenkel. Jetzt gehts zur Sache!!!

4. Bringe den Block 2 des CMD-Programms RESTOREN als Hex-Dump im freien Bereich von BASIC/CMD unter. Dazu werden einfach ab der Zeile 4 des Disassembler-Listings die Hexzahlen (Spalte 1) mit Superzap (MOD-Funktion) in den entsprechenden Sector der BASIC/CMD-File uebertragen. ACHTUNG: Nur die Zeilen 4-Ende !!!
5. Stelle jetzt die Einsprungsadresse des neuen Befehls fest. Zaehle dazu von dem ersten Byte der Block-Ladeadresse bis zum Beginn des neu eingegebenen Maschinenprogramms. Addiere die gefundene Zahl zur Block-Ladeadresse. Das Ergebnis ist die gesuchte Einsprungsadresse.
6. Es werden noch sechs Bytes fuer den ersten Block des neunten RESTOREN-Programms gebraucht. Diese zwacken wir einfach von den Bytes des geaenderten Blocks ab. Wir zaehlen also vom ersten Byte der Blockladeadresse, um die Anzahl der zu ladenden Bytes nach vorne. Steht z.B. 00H im Block-Groessen-Byte zaehlen wir 256 Bytes. Ist dies geschehen, sehen wir von dort 6 Bytes zurueck.
7. Hier fuegen wir nun folgende Bytefolgen ein:  
01 = neue Blocklademarke  
04 = es sollen vier Bytes geladen werden

0440 = Blockladeadresse (=4004)

xxxx = in Punkt 5 ermittelte Einsprungsadresse des BASIC-Patches.

Diese Bytefolge ersetzt den Block 1 des Maschinenprogramms.

8. Wir muessen jetzt noch den alten Block-Groessenwert um 6 Bytes vermindern (wir haben ja einen neuen Block im Bereich des Alten eroeffnet). Dazu ziehen wir vom alten Block-Groessenwert die 6 Bytes ab, die wir fuer unsere benutzt haben. War der Wert 00H, muss er jetzt durch FAH ersetzt werden.

9. Hoffen wir, dass alles geklappt hat!!!

Ich habe den RESTORE-Befehl in das SUPERDOS- und das NEWDOS-BASIC eingearbeitet und bin sicher, dass es sich auch in andere BASIC-Versionen (z.B.: TRSDOS1.3, 2.3, VTOS-BASIC) einbauen laesst. Selbstverstaendlich koennen auch andere Patches eingebaut werden.

Sollte es jemand zu umstaendlich sein, dieser Anleitung zu folgen, kann er die geaenderten BASIC/CMD-Files fuer SUPERDOS-BASIC und NEWDOS-BASIC bei mir anfordern. Schickt mir dazu eine Systemdiskette (SingleSided/SingleDensity) des gewuenschten Diskoperativsystems und einen frankierten Rueckumschlag. Leider habe ich keine Zeit auch noch andere Diskbasicversionen umzuarbeiten. Deshalb bitte nur SUPER- und NEW-DOS-Disketten einschicken!!!

Aber selbst probieren sollte es meiner Meinung nach jeder einmal. Man bekommt dadurch ungeahnte Sicherheit im Umgang mit Superzap und einen guten Einblick in den Aufbau von System- und CMD-Files.

Quellenhinweise: MC Nr.4 - April 1984, Seite 64

Changing the BASIC-Language, 80 micro 6/83

Und weils so schoen war, gleich noch ein Patch!!!

Wenn man vom DOS aus DIR oder vom BASIC aus CMD"DIR aufruft wird als erstes der Bildschirm geloescht bevor das Inhaltsverzeichnis ausgegeben wird. Dies stoert bei manchen Anwendungen. Man kann dieses Manko aber durch folgendes Patch in der fuer den DIR-Befehl zustaeendigen Systemfile beseitigen. Modifikation des SYS8/SYS mit Superzap (Befehlsfolge):

DFS (ENTER), SYS8/SYS (ENTER), 4 (relativ Sector in File) (ENTER), MOD SE, 031D (ENTER), Y, (ENTER), EXIT. Mit dieser Befehlsfolge wird das relative Byte 5EH im relativen Sector 4 des Systemfiles 8 von 031CH in 031DH geaendert. Das Loeschen des Bildschirms auf Aufruf des Disk-inhaltsverzeichnisses unterbleibt nun.

Quelle: Clubinfo Nr. 6 des Computerclubs Bremerhafen.

### Locked Out Tracks

Verdammt noch mal, jetzt habe ich schon zum dritten mal versucht die Diskette zu formatieren und immer wieder wird der Track 36 als nicht formatierbar ausgewiesen oder beim Verifizieren als fehlerhaft bezeichnet!!!

Wer kennt dieses Problem nicht?!? Ich bin fast sicher, dass jeder von uns schon einmal eine solche Diskette in seinem Bestand hatte. Soll man es riskieren, die nur mit Problemen formatierte Floppy zu benutzen und eventuell ein Programm oder eine Datei zu verlieren oder soll man die Disk gleich zum Muell werfen? Beides ist nicht noetig, es gibt einen tragbaren Kompromiss!

Wer das weniger Gute aber um so aeltere TRSDOS 2.3 fuer das Model 1 noch kennt, erinnert sich sicherlich an die Frage "Lockout any Track" in der Formatierungsroutine. Das heute meist verwendete NEWDOS und seine Abkoemmlinge kennt diese Moeglichkeit den Zugriff auf verschiedene Tracks der Diskette zu verhindern nicht mehr. Man kann aber mit Superzap (wie koennte es auch anders sein) eventuell unsichere Tracks unbenutzbar machen.

Der GAT-Sector (GranulAllocationTable) des Direktorys enthaelt die Information ueber die benutzbaren und die benutzten Granules einer Diskette. Die Bytes 00H-27H enthalten die Information ueber die Ausnutzung der einzelnen Tracks. Die Bytes 60H-87H enthalten die sogenannte Lockout-Table. Ein Track ist dann unbenutzbar, wenn seine beiden repraesentativen Bytes im Gatsector den Wert FFH enthalten. An einem Beispiel erlaeutert klingt die Sache schon verstaendlicher. Nehmen wir also an, der Track 36 (relativ) einer Diskette laesst sich nicht formatieren. Der Rest der Disk funktioniert einwandfrei. Die Formatierung wird mit der Proceed-Funktion der Formatierungsroutiene bis zum Ende gefuehrt. Danach schaut man sich mit Superzap den GAT-Sector (Sector 0) des Direktorys an. Man muss, um den Track 36 unbenutzbar zu machen, mit der MOD-Funktion die Bytes 23H und 83H mit dem Wert FFH ueberschreiben. Das Disk Operating System wird diesen Track nie zur Datenspeicherung hernehmen und man kann ohne Angst vor einem Datenverlust die Diskette benutzen!

\*\* Hartmut Obermann \*\*

Sector-Dump von BASIC/CMD aus NEWDOS80 vor den Aenderungen

```

DRV 00 D977 BEC2 C957 7022 B140 11CE FF19 22A0
1 10 4021 5C67 CD67 4421 FEFF 22A2 4021 CA64
1H 20 22A7 4021 6943 CBBE 3A6C 43CB 773E C928
30 067C 3221 643E C332 1243 21BB 6711 5241
DRS 40 0193 00F3 EDB0 3E03 3289 5F21 BE66 3134
304 50 65FB B728 1108 ED4B C564 3600 0B23 78B1
130H60 20F8 083D 20EF 3600 2322 A440 112C 0119
70 EB2A B140 DFDA C957 CD4D 1B21 0000 7EFE
80 2A20 1701 0000 2AA4 4071 2370 2100 00CD
90 5A1B 21E9 5422 5E65 1804 FE0D 2806 2189
A0 6522 0552 C319 1AE1 2133 0022 0552 C521
B0 AC65 CD67 44ED 5B5E 652A A740 1AFE 0D77
FRS C0 1323 20F8 AFF5 2BC3 7903 1D1B 1F03 0100
14 D0 C965 0000 0000 2A2A 237E FE3D C9C5 CDE1
EH E0 6179 C1C9 ED5B C564 C9DD CB04 FEC9 C24A
F0 1E3A 0251 B7FD 2100 4228 04FD 2117 42E5

```

```

DRV 00 DD6E 05DD 6606 FD5E 05FD 5606 DF20 04DD
1 10 CB04 FE11 1700 FD19 FD7E 00B9 38E8 E1C9
1H 20 DDCB 0476 C032 024F C9DD 5E13 DDCB 0476
30 C9F1 09C9 DDCB 0476 C019 10FD C9DD 740A
DRS 40 09EB DDCB 0476 C078 B128 042B 1BED B8AF
305 50 C900 0000 0000 0000 0000 0000 0000
131H60 0000 0000 0000 0000 0000 0000 0000 0000
70 0000 0000 0000 0000 0000 0000 0000 0000
80 0000 0000 0000 0000 0000 0000 0000 0000
90 0000 0000 0000 0000 0000 0000 0000 0000
A0 0000 0000 0000 0000 0000 0000 0000 0000
B0 0000 0000 0000 0000 0000 0000 0000 0000
FRS C0 0000 0000 0000 0000 0000 0000 0000 0000
15 D0 012A C766 0000 0000 0000 0000 0000 0000
FH E0 0000 0000 0000 0000 0000 0000 0000 0000
F0 0000 0000 0000 0000 0000 0000 0100 BC66

```

Sector-Dump von BASIC/CMD aus NEWDOS80 nach den Aenderungen

```

DRV 00 D977 BEC2 C957 7022 B140 11CE FF19 22A0
1 10 4021 5C67 CD67 4421 FEFF 22A2 4021 CA64
1H 20 22A7 4021 6943 CBBE 3A6C 43CB 773E C928
30 067C 3221 643E C332 1243 21BB 6711 5241
DRS 40 0193 00F3 EDB0 3E03 3289 5F21 BE66 3134
314 50 65FB B728 1108 ED4B C564 3600 0B23 78B1
13AH60 20F8 083D 20EF 3600 2322 A440 112C 0119
70 EB2A B140 DFDA C957 CD4D 1B21 0000 7EFE
80 2A20 1701 0000 2AA4 4071 2370 2100 00CD
90 5A1B 21E9 5422 5E65 1804 FE0D 2806 2189
A0 6522 0552 C319 1AE1 2133 0022 0552 C521
B0 AC65 CD67 44ED 5B5E 652A A740 1AFE 0D77
FRS C0 1323 20F8 AFF5 2BC3 7903 1D1B 1F03 0100
14 D0 C965 0000 0000 2A2A 237E FE3D C9C5 CDE1
EH E0 6179 C1C9 ED5B C564 C9DD CB04 FEC9 C24A
F0 1E3A 0251 B7FD 2100 4228 04FD 2117 42E5

```

```

DRV 00 DD6E 05DD 6606 FD5E 05FD 5606 DF20 04DD
1 10 CB04 FE11 1700 FD19 FD7E 00B9 38E8 E1C9
1H 20 DDCB 0476 C032 024F C9DD 5E13 DDCB 0476
30 C9F1 09C9 DDCB 0476 C019 10FD C9DD 740A
DRS 40 09EB DDCB 0476 C078 B128 042B 1BED B8AF
315 50 C900 E37C FE1D E3C2 781D D7FE 90C2 791D
13BH60 CD5B 1EE5 CD2C 1B0B ED43 FF40 E1C1 C31E
70 1D00 0000 0000 0000 0000 0000 0000 0000
80 0000 0000 0000 0000 0000 0000 0000 0000
90 0000 0000 0000 0000 0000 0000 0000 0000
A0 0000 0000 0000 0000 0000 0000 0000 0000
B0 0000 0000 0000 0000 0000 0000 0000 0000
FRS C0 0000 0000 0000 0000 0000 0104 0440 4966
15 D0 012A C766 0000 0000 0000 0000 0000 0000
FH E0 0000 0000 0000 0000 0000 0000 0000 0000
F0 0000 0000 0000 0000 0000 0000 0100 BC66

```

Geaenderte Bytes unterstrichen !!!

### Schnelle Hardcopy fuer TRS-80

Unter diesem Titel erschien das folgende, sehr kurze BASIC-Programm nach einer Frage in der Computer Personlich 24/84. Der Verfasser ist P. Wollschlaeger. Das Programm ermoglicht den Ausdruck des Bildschirms innerhalb kurzester Zeit. Mich hat erstaunt, dass auch die Grafik bei dieser Art der Hardcopy sehr gut wekommt. Die Zeilen 10-1070 stellen das in der CP abgedruckte Programm dar. Die Zeile 1 zeigt das Programm nach der Behandlung mit PACKER.

```

1 I=I:V=V:V$="" :V=VARPTR(V$):POKEV,64:FORI=15360TO16320
STEP64:POKEV+1,IAND255:POKEV+2,INT(I/255):LPRINTV$:NEXT
10 'TRS 80, MODEL I/III, VG : SCREENDUMP
20 GOSUB 1000
30 END
1000 I=I:V=V:V$=""
1010 V=VARPTR(V$)
1020 POKEV,64
1030 FORI=15360TO16320STEP64
1040 POKEV+1,IAND255
1050 POKEV+2,INT(I/255)
1060 LPRINTV$
1070 NEXT
    
```

### Zeilennummer 00000

Die unten abgedruckten sieben BASIC-Zeilen entstammen einer 80-US, die ich mir beim Josef Konrad ausgeliehen habe. Es ermoglicht, die Zeilennummern eines BASIC-Programms alle zu 0 zu machen. Solange ein Programm keine Spruenge (GOSUB, GOTO, RESUME, ON ERROR GOTO usw.) enthaelt, hat das keinen Einfluss auf den Programmablauf. Treten Spruenge in einem Programm auf, muss die aufrufende und die auferufene Programmzeile erhalten bleiben, die Zeilen dazwischen koennen zu 0 gemacht werden. Das Programm wird mit RUN 65300 auferufen. Es zeigt die Jeweilige Zeilennummer an und fragt, ob diese gerettet werden muss. Man muss sich also, bevor man ein Programm "behandelt", darueber im klaren sein, welche Zeilen stehen bleiben muessen. Vielleicht faellt einem ja ein Trick ein, damit das Programm die Entscheidung darueber, ob eine Programmzeile stehen bleiben muss oder nicht, selbst faellt!?! Die Sache funktioniert uebrigens auch, wenn man ein BASIC-Programm als ASCII-File abspeichert und die Aenderung der Zeilennummern mit einem Textverarbeitungsprogramm vornimmt.

```

65300 P=PEEK(&H40A4)+PEEK(&H40A5)*256
65305 IFP)32768THENP=P-65536
65310 P1=PEEK(P)+PEEK(P+1)*256
65320 IF PEEK(P+2)+PEEK(P+3)*256=65300THENEND
65330 PRINT"SAVE LINE # ";PEEK(P+2)+PEEK(P+3)*256;
65340 INPUTZ$;IFZ$="Y"THENZ$="";P=P1:GOTO65305
65350 POKEP+2,0:POKEP+3,0:P=P1:GOTO65305
    
```

### In eisener Sache

Ganz kurz moechte ich hier in eisener Sache ein paar Worte an euch richten. Ich bringe nun zum dritten Mal einen Beitrag in einem Clubinfo und frage mich oftmals, wie diese Beitrage bei euch ankommen. Ob das, was ich bringe, neu und interessant fuer euch ist oder ob man es als "olie Kammeln" handelt. Ob meine Formulierungen bei manchen, vielleicht etwas schwierigeren Problemen verstaendlich sind oder ob ich im 7. Programmierer-himmel schweben und keiner mich versteht. Ich moechte jetzt nicht noch eine Fragebogenaktion starten und damit mit Kanonen auf Spatzen schiessen. Aber ich moechte meine Bereitschaft kundtun jegliche Kritik und jeden konstruktiven Vorschlag der an mich herangetragen wird zu beachten. Ich freue mich ueber jede Anregung und verspreche jede an mich gestellte Frage zu beantworten (soweit ich eine Antwort weis!!!).

Damit Schluss fuer heute. Da ich das Auslieferungsdatum des Infos noch nicht kenne, wuensche ich euch an dieser Stelle ein froehliches Weihnachtsfest und ein glueckliches neues Jahr!!!

\*\* Hartmut Obermann \*\*

Den Computer besser nutzen:

## Bildschirm Ausdruck ohne Probleme beim TRS/80 + Video Genie

Der TRS-80 enthaelt in seinen ROM's leider kein Programm zum Ausdrucken des Bildschirms. Ein solches Programm ist jedoch recht zweckmaeBig, deswegen gibt es schon viele Programme, z. B. JKL beim NEW-DOS/80, mit denen dieser Fehler wieder gutgemacht wird. Moechte man jedoch z. B. mitten in einem Spiel ein Schachfeld oder die Highscores fuer immer festhalten, versagen auch diese Programme. Das liegt daran, daB sie als eine Interruptroutine arbeiten, welche von den meisten Maschinen-spracheprogrammen abgeschaltet oder geaendert wird. Das hier vorgestellte Programm kann jedoch immer den Bildschirminhalt ausdrucken, egal, welches Programm gerade laeuft. Es ist fuer den EPSON MX-80 mit Grafrax geschrieben, kann jedoch leicht an andere Drucker mit TRS-80-Blockgrafik angepaBt werden.

The image contains three main sections:

- Top:** A listing of assembly code with German comments. Comments include: "10110 101-80/Grafrax wird auf TRS-80 Grafik eingestellt", "4200 3E18 LD A,18H", "4202 CD3842 CALL PRINT", "4203 3E40 LD A,40H", "4207 CD3842 CALL PRINT", "4208 3E18 LD A,18H", "420C CD3842 CALL PRINT", "420F 3E3A LD A,3AH", "4211 CD3842 CALL PRINT", "4214 2103C 00210 (Bildschirminhalt aus Drucker)", "4217 0A40 00220 MEM.IN LD B,40H", "4219 7E 00150 NEXT LD B,A,163", "421A FE80 00240 CP J O80H", "421C 3802 00250 JP C,O81", "421E C8F7 00260 RES A", "4220 FE30 00270 OR1 CP 20H", "4222 3002 00280 JR NC,DK2", "4224 C8F7 00290 RET A,A", "4226 CD3842 00300 DK2 CALL PRINT", "4229 23 00310 INC HL", "422A 10E0 00320 DJNZ NEXT", "422C 3E00 00330 LD A,00H", "422E CD3842 00340 CALL PRINT", "4231 3E40 00350 LD A,40H", "4234 20E1 00360 CP H", "4234 20E1 00370 JK NI,MEM.IN", "4236 18FE 00380 READY JR READY", "4238 32E837 00390 PRINT LD (37E8H),A", "423B 30E37 00400 TEST LD A,(37E8H)", "423E EAF0 00410 AND 0F0H", "4240 FE30 00420 CP 30H", "4242 20F7 00430 JR NI,TEST", "4244 C9 00440 RET A", "4200 00450 END 4200H", "00400 TOTAL LENGTH", "30161 TEST AREA BYTES LF1"
- Middle:** A chessboard displayed on a grid. The board shows pieces in their starting positions. The grid has coordinates like '300' and '301'.
- Bottom:** A continuation of the assembly code listing, including the same comments as the top section.

Oben: Adressen des Programms

Mitte: Beispiel

Unten: Listing des Programms

Es wird direkt auf die Diskette geschrieben, ist aber zum besseren Verstaendnis auch als Source-Code abgedruckt.

Zuerst muB eine Diskette formatiert werden. Wichtig ist, daB Spur 0 in Single Density formatiert wird. Dann wird Sektor 0 mit Superzap oder aehnlichem Utility-Programm wie folgt geaendert:

Das Laufwerk, auf dem geaendert wird, muB auf Single Density eingestellt sein.

Zum Gebrauch wird diese Diskette in Laufwerk 0 eingelegt, waehrend ein Maschinenspracheprogramm laeuft. Soll zu einem bestimmten Zeitpunkt der Bildschirminhalt ausgedruckt werden, wird einfach auf den RESET-Knopf gedrueckt.

Zum Programm: bei einem RESET wird beim TRS-80 bei angeschlossener Floppy Diskette in den Sektor 0 der in Laufwerk 0 liegenden Diskette in den Speicherbereich 4200H bis 42FFH geschrieben und anschliessend nach 4200H gesprungen. Auf unserer Diskette steht in Sektor 0 das Programm zum Ausdrucken des Bildschirms, welches sofort startet, ohne daB der Bildschirm zuvor geaendert wird. Bedingt durch den TRS-80-ROM kann nicht in das Maschinenspracheprogramm zurueckgesprungen werden, da der Stackpointer bei einem RESET auf 407DH geaendert wird. Nach dem Bildschirm Ausdruck wartet der TRS-80 in einer Endlosschleife auf erneutes Druecken der RESET-Taste. Die geaenderte Diskette kann uebrigens weiterhin als Datendiskette benutzt werden!

Christof Ueberschuur

## Willkommen in der ADVENTURE-ECKE!

Heute stellt Euch der Gerald Schröder das Adventure GOLDEN BATON vor. Ich verabschiede mich bis zum nächsten Club-Info, Euer

### Adventure - Alex

-----

Bahn frei

foooooooooooo

Golden Baton Gerald !!!

Die Situation und der Sinn des Spiels lassen sich aus dem Anfangstext ersehen.

Die Schauplätze

Jeder Platz erhält eine Nummer. Die möglichen Ausgänge werden den Nummern der Lokalitäten zugeordnet, zu denen sie führen.

1. Dense Forest: N=3, S=2
2. Tangle of briars: N=1
3. At a small stream: N=7, S=1, W=4
4. Old Tree: N=5, E=3
5. Hut: S=4, Tür=6
6. Inside hut: W=5, D=24
7. Road 1: S=3, N=8
8. Road 2: S=7, N=92
9. Road 3: S=8, N=10
10. Pond: S=9, N=11
11. Moat: S=10, (12)
12. Portcullis: (13), (12)
13. Battlements: D=14
14. Courtyard: U=13, N=15, E=16, W=17
15. Huge oaken door: S=14, N=23
16. Stables: W=14
17. Outbuilding: E=14, D=18
18. Smelly chamber: E=19, U=17
19. Torture chamber: W=18, N=22, S=20
20. Sorcerer's laboratory: N=19, E=21
21. Bare room: W=20
22. Servants quarters: S=19
23. Cold room: S=15
24. Small cave: U=6, N=28, S=25
25. Larger cave: N=24, S=26
26. Empty room: N=25, S=27
27. Storage chamber: N=26
28. Shore: S=24 (29)
29. Lake.

Im folgenden werden jeweils die Nummer des Schauplatzes und die zu erfolgenden Aktionen aufgeführt. Dabei ist nicht berücksichtigt, wieviel getragen werden kann.

1. get cloak, wear cloak, search leaves, get sword
2. cut briars, get rope
4. throw rope, climb rope, get ring, wear ring, d, get rope
7. kill wolf
8. get staff
11. throw matches, swim
12. get matches, throw rope, climb rope
16. get helmet, wear helmet, read runes
17. get lamp, search straw
18. open door
19. light lamp, get hammer
20. wave staff, say akurz, get quartz
21. wave quartz, search lizard, get knife
22. get mirror
15. rub ring, get key, unlock door, hold mirror, open door.
- N
23. get parchment, read parchment
16. get horn
5. open door
6. open barrel, get salt
25. throw salt, get slugs
26. smash padlock
27. get raft
28. drop slugs, drop raft, sail raft
29. blow horn, throw knife, get baton

# ADVENTURE-ECKE

Frage:

"Wie macht ein Schotte  
Gruppensex?"

"Mit seiner Frau vor dem  
Spiegel."

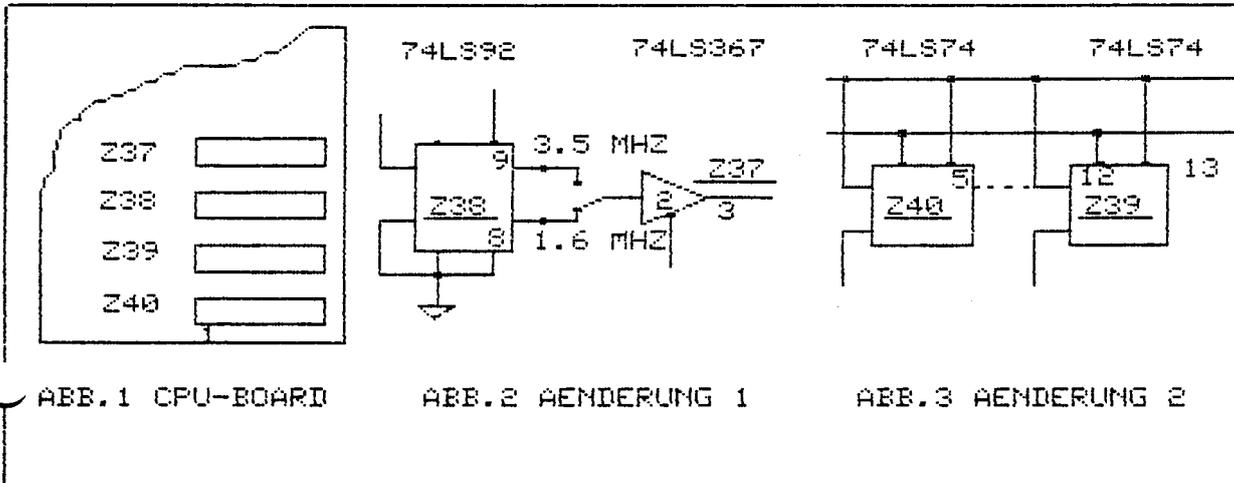


"Hat die Mc. Gregory  
sich zu Hause  
vorgesetzt?"  
"Ja, einen Schaden!"

## Verdoppelung der Taktfrequenz für Genie I + II

Für verhältnismäßig wenig Geld läßt sich die Taktfrequenz des Videogenies verdoppeln. Notwendig sind nur eine neue CPU Z80A und ein Umschalter (1 X um).

Voraussetzung für diese Hardware-Änderung ist allerdings, daß die RAM IC's keine größere Zugriffszeit als 300 ns ( bzw. 200 ns für RAM IC's im Expander) haben. Mit den neuen 64 K Chips dürfte es keine Probleme geben.



Im Einzelnen sind folgende Handgriffe zu verrichten:

1. Austausch des Z80 Prozessors gegen einen Z80A Prozessor.
2. Einbau des Umschalters ins Tastaturgehäuse.
3. Unterbrechung der Verbindung zwischen Pin 8 von Z38 und Pin 2 von Z37 (auf der Unterseite der CPU-Platine). Pin 2 von Z37 wird über den Umschalter (Abb.2) an Pin 8 und 9 von Z38 angeschlossen.
4. Unterbrechung der Verbindung zwischen Pin 5 von Z40 und Pin 12 von Z39 (auf der Oberseite der CPU-Platine, in der Nähe von Pin 5).
5. Pin 12 und 13 von Z39 werden durch eine Drahtbrücke verbunden (Abb.3).

Abb.1 zeigt die Lage von Z37 - Z40 auf der CPU-Platine (Aufsicht). Pin 1 befindet sich jeweils unten links am IC.

Am besten testet man den Computer erst einmal unter der alten Taktfrequenz (z.B. mit einem Diagnoseprogramm). Ist dieser Test erfolgreich verlaufen, wiederholt man die gleiche Prozedur mit der doppelte Taktfrequenz. Erforderlich hierzu ist eine Anpassung des Betriebssystems an die doppelte Taktrate. Unter NEWDOS80 z.B. ändert man den Systemparameter BJ von 1 auf 2.

in Nachteil gegenüber den käuflichen 'SPEED UPS' läßt sich bei dieser Billiglösung nicht vermeiden, ein Umschalten der Taktrate bei eingeschaltetem Gerät führt meist zum "Hängen" des Computers. Dann hilft nur noch Ausschalten (incl. Programmverlust).

Zum Schluß noch ein Hinweis für eiserne Sparer. Beim Verfasser läuft ein Genie I auch mit alter CPU unter doppelter Taktfrequenz einwandfrei.

G.Dreyer

### Neues aus der Gerüchte-Küche:

Wundert Euch nicht, wenn Ihr nächstens in Euer Tandy Computer-Center reinschaut und dann plötzlich 'apricot'-Computer seht. Angeblich werden die Tandy-Computer-Center von Tandy abgezwickelt und heißen dann 'TA-Computerworld' und vertreiben außer Tandy dann die aus England stammenden Computer der Firma 'apricot'. Dabei soll es sich um leistungsstarke und sehr preiswerte MS-DOS-Computer handeln.

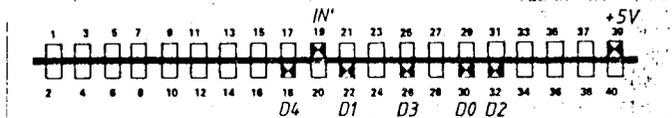
# JOYSTICK für den TRS-80 Modell I

## Anschluß des Joystick :

Der Joystick benötigt folgende TRS-80 Anschlüsse :

- a) IN' (Pin 19 des TRS-80-Bus) == gemeinsamer Anschluß des Joystick auf Peripherie-Lese-Taktsignal.
- b) D0 (Pin 30 des TRS-80-Bus) == hoch-, vor Anschluß des Joystick auf Datenleitung 0.
- c) D1 (Pin 22 des TRS-80-Bus) == runter-, zurück Anschluß des Joystick auf Datenleitung 1.
- d) D2 (Pin 32 des TRS-80-Bus) == links Anschluß des Joystick auf Datenleitung 2.
- e) D3 (Pin 23 des TRS-80-Bus) == rechts Anschluß des Joystick auf Datenleitung 3.
- f) D4 (Pin 18 des TRS-80-Bus) == Feuer- Anschluß des Joystick auf Datenleitung 4.
- g) + (Pin 39 des TRS-80-Bus) == Dauerfeuer- Anschluß des Joystick auf Plus 5V. (gegebenenfalls)

Die entsprechenden Joystickanschlüsse sind mit dem Ohmmeter und Betätigung des Joystick leicht auszumessen.



Numerierung der Bus-Anschlußleitungen (von der Rückseite des Computergehäuses aus gesehen).

Es gibt nun zwei Möglichkeiten, einen Joystick an den TRS-80 anzuschließen.

### 1. Ohne große Bastelei (zum Probieren)

Man besorge sich einen 40-poligen Busstecker für die TRS-80-Platine bzw. für das EXPANSION. Sowie fünf Dioden, die zur Entkopplung im Joystick in die Leitungen b - f eingelötet werden. Die Dioden müssen mit der Kathode (schwarzer Ring / - Pol) zur Joystick-Platine eingelötet werden. Danach sind die Drähte a - f bzw. a - g - je nach Joystick - auf den entsprechenden Kontakt des Bussteckers (wie oben angegeben) zu verpressen.

### 2. Wer seinen TRS-80-Bus bzw. EXPANSIONS-Bus benötigt

In diesem Fall braucht man einen 7-poligen Diodenstecker und eine passende Diodenbuchse anstatt des TRS-80 Bussteckers. Anschließend wird der TRS-80 oder das EXPANSION vorsichtig geöffnet, und auf der Hauptplatine werden vor dem 40-poligen Bus sieben Drähte auf die oben angegebenen Punkte aufgelötet. Es ist zu beachten, daß die Folienverbindung zwischen Haupt- und Tastaturplatine nicht beschädigt wird! Nun sucht man sich am Computergehäuse einen günstigen Platz zum Einbau der Diodenbuchse. Nachdem die Buchse eingebaut ist und an die Anschlüsse derselben die fünf Dioden (Kathode zur Buchse) angelötet sind, können die sieben Drähte von der Hauptplatine aufgelötet werden. Es ist darauf zu achten, daß die Belegung des Diodensteckers mit der der Diodenbuchse identisch ist. Die Verbindungen sind zu überprüfen. Bei dieser Einbauvariante kann der Joystick so belassen werden wie er ist.

Es besteht auch die Möglichkeit, den Original-Joystickstecker zu belassen und sich eine entsprechende Buchse dafür zu kaufen und einzubauen.

Der Einbau des Joysticks ist recht einfach. Ich möchte aber darauf hinweisen, daß ich für eventuelle Schäden, die am Gerät entstehen sollten, nicht hafte.

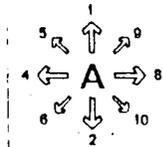
Für Verbesserungsvorschläge bin ich allerdings dankbar.

## Funktionstest des Joystick :

Der Joystick ist durch die Benutzung des Port 0 unabhängig von der Tastatur. Der Joystick muß demzufolge auch anders angesprochen werden. Der Befehl dazu ist INP (0).

Nun ein kurzes Programm in BASIC zum Testen :

```
5 CLS
10 A=255-INP(0)
20 Print A;A
30 GOTO 10
```



Nach Starten des Programms müssen auf dem Monitor je nach Betätigung des Joysticks die Zahlen - wie auf der Abbildung rechts- angezeigt werden.

Ist der Joystick in Ruhelage, erscheint eine 0.

Bei Feuerknopfdruck sollte eine 16 erkennbar sein.

Erscheinen die Zahlen entsprechend vorheriger Erklärung ist der Joystick betriebsbereit. Andernfalls sollten die Anschlüsse noch einmal auf ihre Richtigkeit überprüft werden.

## Basicprogramme mit Joysticksteuerung :

Um nun die Joystickfunktionen in ein Basicprogramm einzubinden, ist folgendes zu tun.

Tastaturbetrieb	Joystickbetrieb
1000 A%=INKEY\$: IFA\$=""THEN1000	1000 A=255-INP(0): IFA=0THEN1000
1010 IFA\$="I"THEN GOSUB AAAA	1010 IFA=1THEN GOSUB AAAA
1020 IFA\$="J"THEN GOSUB AABA	1020 IFA=2THEN GOSUB AABA
1030 IFA\$="L"THEN GOSUB AACA	1030 IFA=4THEN GOSUB AACA
1040 IFA\$="M"THEN GOSUB AADA	1040 IFA=8THEN GOSUB AADA
1050 IFA\$="N"THEN GOSUB AAFA	1050 IFA=16THEN GOSUB AAFA

Damit die Joystickabfrage schneller wird, sollte die entsprechende Variable als DEFINT-Variable ausgegeben werden.

## Maschinenprogramme mit Joysticksteuerung :

Die schnellsten Spiele sind natürlich Programme in Maschinensprache. Hier passiert im Prinzip das gleiche wie bei den Basicprogrammen. Der Computer muß nur so programmiert werden, daß er den Port 0 liest, anstatt die Tastaturabfrageroutine zu nutzen.

Die meisten Programme der Firmen BIG FIVE SOFTWARE, ADVENTURE INTERNATIONAL, CORNSOFT GROUP und ALPHA PRODUCTS sind Joystickkompatibel.

Ich wünsche viel Spaß beim Spiel.

Peter Wolf

# EPROM- Programmier- gerät

Mehr und mehr Anwendungen von Computern benötigen programmierte EPROMs, beispielsweise die zahlreichen Einplatinencomputer. Ein EPROM-Programmiergerät soll zum einen möglichst alle gängigen EPROM-Typen programmieren können, zum anderen an möglichst viele Computer anschließbar sein. Da in Zukunft weitere EPROMs auf den Markt kommen dürften, muß der Anwender die Änderungen der Software selbst durchführen können.

Die vorliegende Schaltung ist in der Lage, die EPROM-Typen 2716, 2732, 2732A und 2764 zu programmieren. Ziel war es, mit möglichst wenig zusätzlicher Hardware dieses Problem zu lösen. Zum Rechner hin ist eine Parallelschnittstelle erforderlich, in der vorliegenden Form sind 18 Portleitungen notwendig. Die Software ist in Basic geschrieben und zeigt den grundsätzlichen Umgang mit dem Programmiergerät. Der einzelne Anwender kann diese Software noch leicht erweitern.

## Nicht viel Hardware

Die Schaltung wird im vorliegenden Fall an einem 8085-System betrieben und ist dort an einem PIO-Baustein 8255 angeschlossen. Dieser wird von Basic aus über die Befehle IN und OUT angesprochen, bei anderen Computern ist dies analog mit PEEK und POKE möglich. Die Gesamtschaltung des Programmiergerätes (Bild 1) ist nicht sehr aufwendig (Bild 2). Der PIO-Baustein 8255 besitzt drei Ports von je 8 Bit. Port A wird zur Übertragung der Daten von und zum Rechner benutzt. Port B steuert mit den Bits 0 und 1 die beiden Binärzähler IC1 und IC2. Diese erzeugen die Adressen für die EPROMs, da sonst eine erheblich größere Zahl von Portleitungen erforderlich wäre. Port C ist für die Steuersignale zuständig. Hier werden die Programmierspannung, das Chip-Enable-Signal sowie der

Programmierimpuls den jeweiligen Pins zugeführt. Das NAND-Gatter IC 3 verhindert, daß bei unsachgemäßer Programmierung (alle Leitungen des Ports C auf High) die 25-V-Versorgung kurzgeschlossen wird. Die unterschiedlichen Spannungen zum Programmieren bzw. zum Lesen der EPROMs werden mit den Z-Dioden D3 und D4 erzeugt. D3 ist so zu wählen, daß sich eine Programmierspannung von 21 V einstellt. Da Z-Dioden Toleranzen haben, kann es sein, daß man sich mit Kombinationen von Dioden behelfen muß. Das gilt natürlich auch für D4, die mit D3 zusammen eine Spannung von < 5 V (High-Pegel) ergeben muß. In gewissen Grenzen kann man auch mit der 25-V-Versorgung nachhelfen, denn diese Spannung kann ja zwischen 24 und 26 V liegen.



Bild 1. Der Prototyp des EPROM-Programmiergerätes

Die gesamte Schaltung ist auf einer einseitig kaschierten Leiterplatte im Format 100 x 160 mm untergebracht (Bild 3). Die Platine wird nach Bild 4 bestückt. Da es sich um eine einseitige Leiterplatte handelt, lassen sich Brücken nicht ganz vermeiden, aber doppelseitige Leiterplatten kann nicht jeder so ohne weiteres herstellen und sind zudem erheblich teurer. Haben Rechner und Programmiergerät getrennte Stromversorgungen, sollten zwei Widerstände von etwa 10 kΩ in die beiden Leitungen zum Adreßzähler geschaltet werden. Ansonsten werden die Schutzdioden des CMOS-Schaltkreises überlastet, wenn das EPROM-Programmiergerät nicht eingeschaltet ist.

## Unterschiedliche EPROMs

Eine Aufstellung, wie die einzelnen EPROMs programmiert oder auch gelesen werden, zeigt Bild 5. In Zweifelsfällen ist es immer sinnvoll, ein Datenblatt des Herstellers zu Rate zu ziehen. Allgemein gilt aber, daß die EPROMs 2716 und 2732 mit 25 V programmiert werden, die Typen 2732A und 2764 hingegen mit 21 V. Die Länge des Programmierimpulses soll 50 ms ± 5 ms betragen. Durch das EPROM 2764 bedingt besitzt das Programmiergerät einen 28poligen Sockel. Alle übrigen Typen hingegen haben nur 24 Anschlüsse. Deshalb ist beim Einsetzen der EPROMs auf deren richtige Position zu achten. Wer nur 24polige EPROMs programmieren will, kann von vornherein einen 24poligen Sockel montieren.

## Die Programmier-Software

Um das Programm (Bild 6) auf andere Rechner übertragen zu können, sind die spezifischen Adressen des PIO-Bausteins 8255 zu erklären. Es bedeuten im einzelnen:

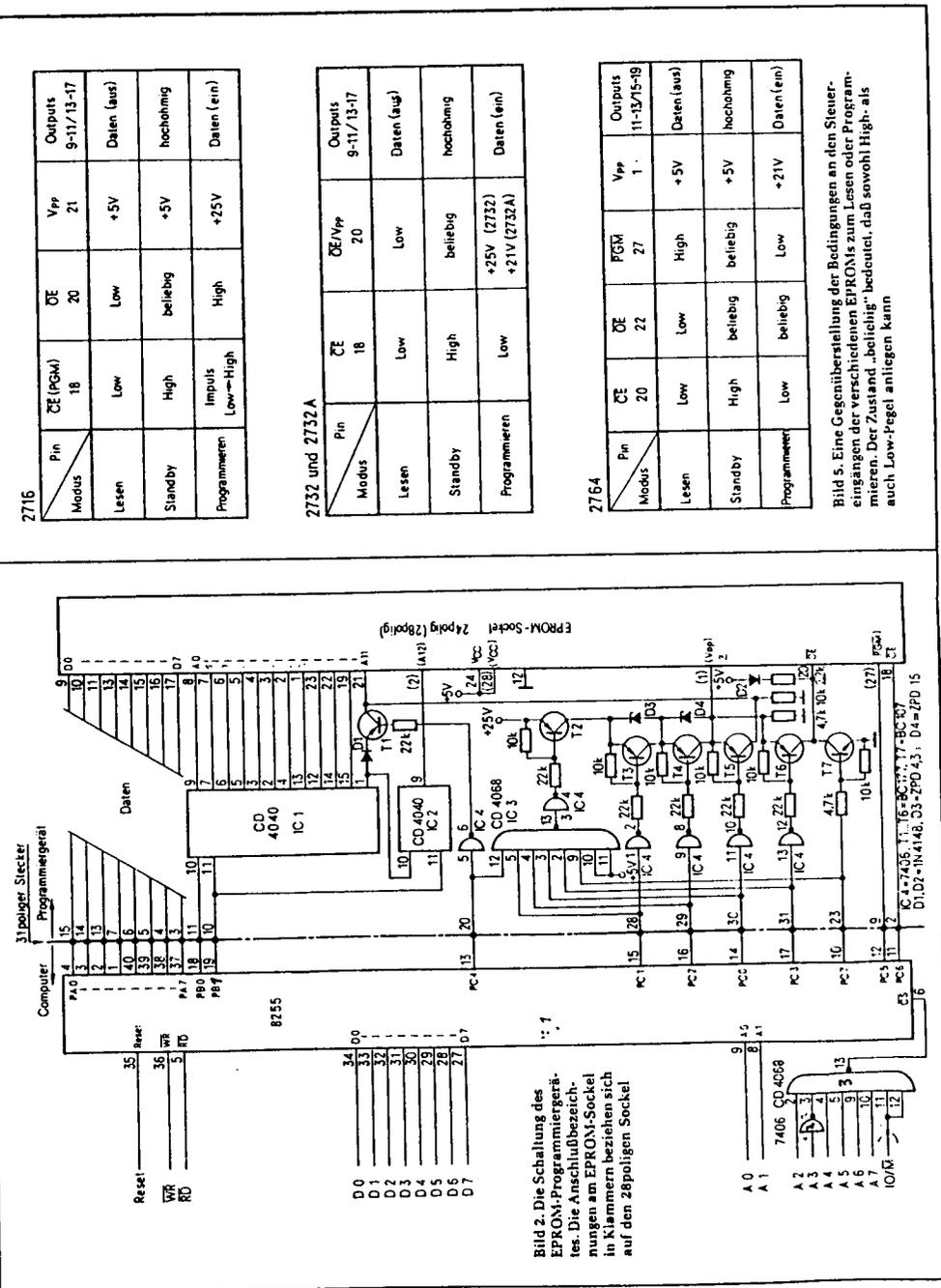


Bild 2. Die Schaltung des EPROM-Programmiergerätes. Die Anschlußbezeichnungen am EPROM-Sockel in Klammern beziehen sich auf den 28poligen Sockel

Pin	CE (PGM)	OE	Vpp	Outputs
Modus	18	20	21	9-11/13-17
Lesen	Low	Low	+5V	Daten (aus)
Standby	High	beliebig	+5V	hochohmig
Programmieren	Impuls Low-High	High	+25V	Daten (ein)

Pin	CE	OE/Vpp	Outputs
Modus	18	20	9-11/13-17
Lesen	Low	Low	Daten (aus)
Standby	High	beliebig	hochohmig
Programmieren	Low	+25V (2732) +21V (2732A)	Daten (ein)

Pin	CE	OE	Vpp	Outputs
Modus	20	22	1	11-13/15-19
Lesen	Low	Low	+5V	Daten (aus)
Standby	High	beliebig	+5V	hochohmig
Programmieren	Low	beliebig	+21V	Daten (ein)

Bild 5. Eine Gegenüberstellung der Bedingungen an den Steuereingängen der verschiedenen EPROMs zum Lesen oder Programmieren. Der Zustand „beliebig“ bedeutet, daß sowohl High- als auch Low-Pegel anliegen kann

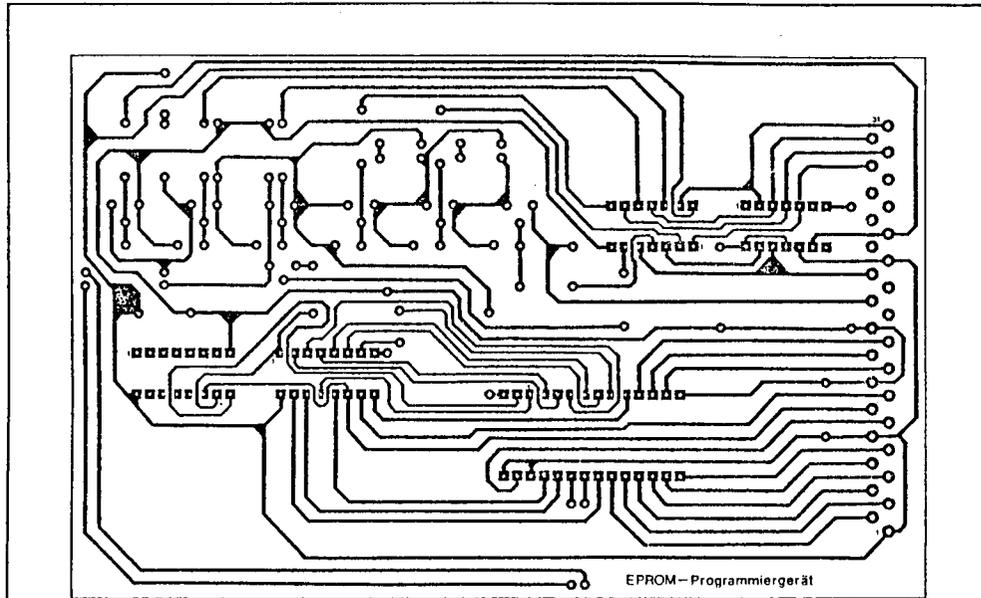


Bild 3. Die Schaltung des Programmiergerätes ist auf einer Europakarte untergebracht

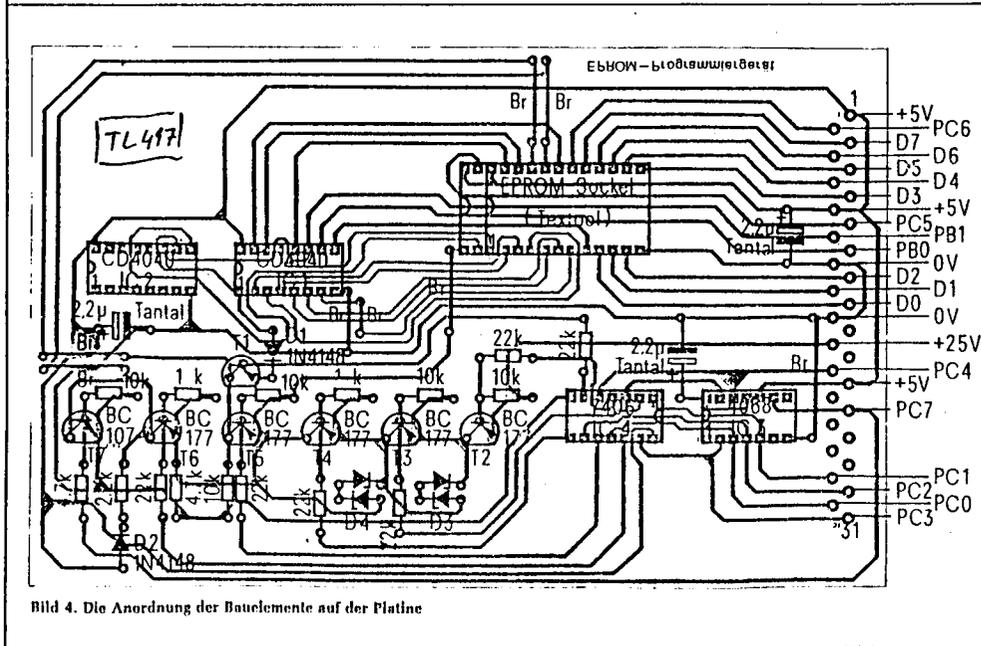
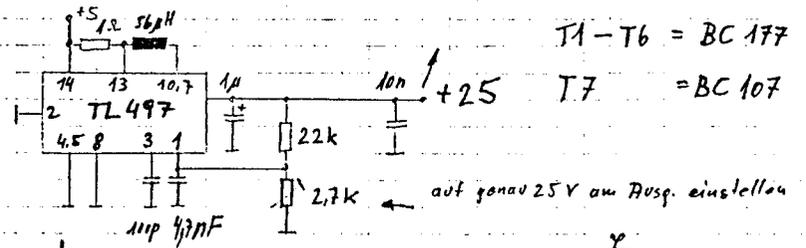
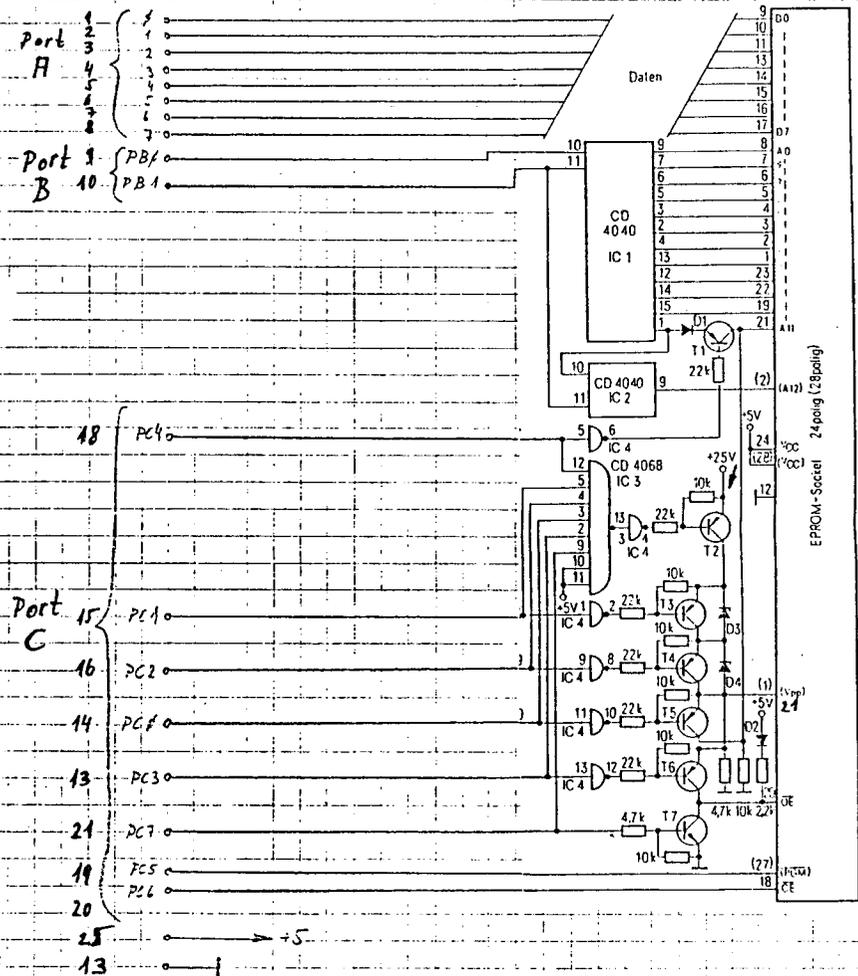


Bild 4. Die Anordnung der Bauelemente auf der Platine

EPROM - HARDWARE FÜR KOMTEK, TRS, VG

Interface OE2ZL



Die Schaltung mit dem TL497 erzeugt aus der normalen +5V die Programierspannung 25V. Die Bauteile dazu sind frei auf der Platine verdrahtet.

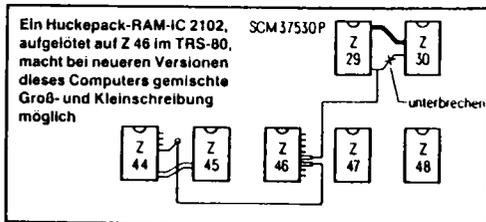
T1 - T6 = BC 177  
T7 = BC 107

auf genau 25V am Ausg. einstellen

gehört an den Schluss  
dieses Beitrages. **Kleinbuchstaben für den TRS-80**

Da ich auch einen TRS-80 besitze, habe ich den Artikel in mc 1982, Heft 12, auf Seite 61, mit großem Interesse gelesen. Bei neueren Versionen des TRS-80 mit dem Zeichengenerator SCM 37530 genügt ein einzelnes Video-RAM, um den Rechner von Groß- auf Kleinschreibung umzustellen. Zur Umschaltung kann man den Kleinbuchstaben-Treiber aus mc 1982, Heft 6, verwenden. Die Schaltung ist im Bild wiedergegeben. Das neue 2102-IC wird huckepack auf Z 46 gelötet, wobei die Pins 11 und 12 rechtwinklig abgeknickt werden.

Friedhelm Lütkecke, Münster



mc 2/1983

## Kleinbuchstaben-Treiber für den TRS-80

Wer in letzter Zeit einen TRS-80 Modell I gekauft hat, der hat wahrscheinlich, ohne es zu wissen, Kleinbuchstaben mit erworben. Die neuen, die in Japan unter Lizenz hergestellt werden

(Merkmal: am Tastaturgehäuse unten steht „MADE IN JAPAN“), werden serienmäßig mit dem Zeichen-ROM ausgestattet, das sonst inklusive Treiberprogramm in Tandy-Läden nur für rund 150 DM erhältlich ist. Nur: mit dem normalen Zeichentestprogramm des Handbuchs mit PRINT CHR\$ bleiben diese Zeichen verborgen. Erst mit POKE kommen sie zum Vorschein, und zwar mit folgendem kleinen Testprogramm: 10 CLS:Y=15360:FOR X=0 TO 255 20 POKE Y,X:POKE Y+1,32:Y=Y+2 30 NEXT:PRINT@512,;

Nach RUN erscheinen die Großbuchstaben in der ersten Reihe, die normalerweise angesprochen werden. In der vierten Reihe, ab dezimal 97, kommen die Kleinbuchstaben. Es galt also, diese Kleinbuchstaben verfügbar zu machen, und zwar so, daß Großbuchstaben wie bei der Schreibmaschine mit der SHIFT-Taste zu erreichen sind; aber auch der alte Modus mit Großbuchstaben sollte nicht dabei ganz verlorengehen. Das Programm KBUCH macht's möglich. Dieser Treiber für Kleinbuchstaben auf dem TRS-80 Modell I zeichnet sich durch einfaches Umschalten aus. Es wird das Basic-Kommando NAME, das weder mit ROM- oder mit Disk-Basic benutzt wird, verwendet, um von der TANDY-Tastatur (Großbuchstaben) zur geänderten und zurück zu gehen. Angenommen, man hat nur 16 KByte RAM; der Treiber liegt bei hexadezimal 7FA2 bis 7FFF. Bei der Frage MEM SIZE? muß also 32673 (oder weniger) eingegeben werden. Das Programm wird von Kassette unter dem Namen KBUCH eingelesen und bei 7F92 (32658) gestartet. Hierdurch zeigt NAME auf den Treiber. Anschließend kann mit dem Aufruf NAME beliebig hin- und hergeschaltet werden.

Wirkungsweise: wenn der Treiber eingeschaltet ist, werden alle alphabetischen Zeichen von der Tastatur „umgedreht“. Im Bildschirmtreiber wird die Ausgabe von Kleinbuchstaben ermöglicht. Eine Kassettenaufzeichnung kann mit dem DEBUG-Monitor folgendermaßen erstellt werden: Zunächst wird DEBUG eingelesen, dann tippt man mit dem M-Kommando das Programm im Bild ein: M ADDRESS = 7F92 usw. Mit dem W-Kommando schreibt man nun auf Band: S=7F92 E=7FFF T=7F92 N=KBUCH Bei Systemen mit 32/48 KByte müssen die Bytes an den Adressen 7F99, 7FB0 und 7FB3 angepaßt werden; ansonsten ist das Programm frei verschiebbar.

Mary Jo Kostya

7F90	00	00	F5	E5	21	8F	41	11
7F9B	A2	7F	73	23	72	E1	F1	C3
7FA0	19	1A	F5	E5	D5	C5	21	16
7FAB	40	7E	FE	E3	20	13	11	C9
7FB0	7F	01	E2	7F	73	23	72	2E
7FBB	1E	71	23	70	C1	D1	E1	F1
7FC0	C9	11	E3	03	01	5B	04	18
7FCB	FE	CD	E3	03	FE	41	DB	FE
7FDD	3B	C8	FE	60	C8	FE	7B	D0
7FDB	CB	6F	20	03	CB	EF	C9	CB
7FE0	AF	C9	DD	6E	03	DD	66	04
7FEB	DA	9A	04	DD	7E	05	87	2B
7FF0	01	77	79	FE	20	DA	06	05
7FFB	FE	80	D2	A6	04	C3	7D	04

Das Treiberprogramm als Hex-Dump

Luidger Röckrath

## Kleinbuchstaben für den TRS-80

Kleinbuchstaben sind für Textverarbeitung auf Computern notwendig, leider jedoch bei der Konzeption vieler handelsüblicher Kleincomputer (wie z. B. beim TRS-80) schlichtweg vergessen worden. Oder kam den Schöpfern dieser Kleincomputer Textverarbeitung als mögliche Anwendung überhaupt nicht in den Sinn? Besonders sträflich erscheint dieses Versäumnis beim TRS-80, da durch hinzufügen von nur zwei preisgünstigen Bausteinen (Gesamtkosten ca. 5 DM) Kleinbuchstaben implementiert werden können.

Textverarbeitung ist beim Autor eine der häufigsten und nützlichsten Anwendungen des TRS-80. Mit einem TRS-80, zwei Disk-Laufwerken, einer Kugelpf- oder Typenradmaschine und entsprechender Software läßt sich für ein paar Tausender ein System zusammenstellen, mit dem komfortabel und schnell Briefe, Manuskripte und auch größere Arbeiten eingegeben, korrigiert, dauerhaft gesichert und in Korrespondenzqualität ausgegeben werden können. Kleinbuchstaben bilden die geringste Voraussetzung für halbwegs vernünftige Textverarbeitung. Wie die weiteren Voraussetzungen auch erfüllt werden können, dazu auch mehr im letzten Abschnitt.

### Die Video-RAM-Schaltung des TRS-80

Bild 1 zeigt die Video-RAM-Schaltung des TRS-80. Das Video-RAM besteht aus 1024 Speicherplätzen, entsprechend der Anzahl der auf dem Bildschirm darstellbaren Zeichen. Aber leider bietet jeder Speicherplatz nicht Platz für die Speicherung eines Bytes, wie der Leser vielleicht annimmt, sondern nur für 7 Bit, da ja auf dem Bildschirm auch nur 128 verschiedene Zeichen dargestellt werden können. Der Speicherbaustein für Bit 6 wurde einfach weggelassen, und durch eine kleine Logikschaltung (Z30 in Bild 1) werden beim Auslesen des Video-RAM in Abhängigkeit von Bit 6 und 7 auf der Datenleitung von Bit 6 Logikzustände erzeugt, so daß man beim Auslesen des Video-RAM-Inhaltes die den auf dem Bildschirm dargestellten Zeichen entsprechenden ASCII-Codes

erhält. Hierin (und nicht im Zeichengenerator!) liegt der Grund der Unmöglichkeit Kleinbuchstaben auf dem Bildschirm darzustellen. Tabelle 1 zeigt, wie sich diese Logikschaltung auf das Verhalten des Video-RAM auswirkt. Aus ihr kann man leicht entnehmen, daß alle Zeichen jeweils durch zwei verschiedene Bitmuster auf den Bildschirm gebracht werden können. Beim Auslesen erscheinen allerdings immer die ASCII-Codes der dargestellten Zeichen – eine Tatsache, die auch in der ROM-Software des TRS-80 ihren Niederschlag findet, und uns weiter unten noch beschäftigen wird. (Neueste Modelle aus japanischer Produktion, die mit Kleinschreibung ausgerüstet sind, verhalten sich in dieser Beziehung anders.)

### Die modifizierte Video-RAM-Schaltung

Der einfachste – und folglich auch meist beschrittene – Weg zur Kleinschreibung besteht in der schlichten Ergänzung des fehlenden RAM-Bausteins für Bit 6.

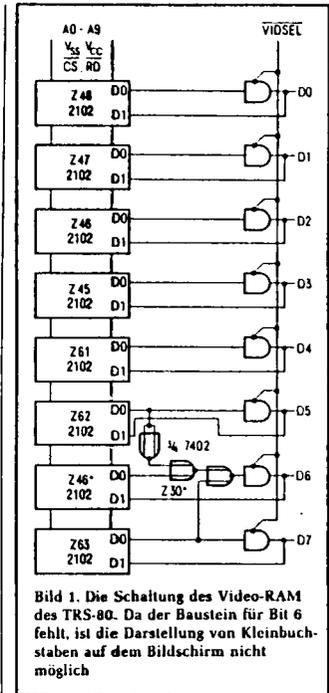


Bild 1. Die Schaltung des Video-RAM des TRS-80. Da der Baustein für Bit 6 fehlt, ist die Darstellung von Kleinbuchstaben auf dem Bildschirm nicht möglich

Aber dieser Weg führt wegen gewisser Eigenschaften des Displaytreibers im ROM, dessen Anfang in Bild 2 als Auszug aus [1] aufgelistet ist, zu auf den ersten Blick recht eigenwilligen Ergebnissen. Es drängt sich fast der Verdacht auf, daß bei der Entwicklung des TRS-80 (hier Displaytreiber) Wert darauf gelegt wurde, durch „geschickte“ Programmierung Erweiterungen (hier Kleinschreibung) möglichst schwer zu machen. Wie am Schluß des letzten Abschnittes schon erläutert, können alle Zeichen durch zwei verschiedene Codes erzeugt

schreiben	im Speicher	auf Bildschirm	lesen
765	765	← Bit-Nr. →	765
000	0X0	Großbuchstaben	010
001	0X1	Ziffern, Sonderz.	001
010	0X0	Großbuchstaben	010
011	0X1	Kleinbuchstaben	011
100	1X0	Grafik	100
101	1X1	Grafik	101
110	1X0	Grafik	100
111	1X1	Grafik	101

werden. Aus unbegreiflichen Gründen (welche unter dem Blickwinkel obiger Argumentation wiederum auch recht begreiflich sind) hat man bei Radio Shack für die Buchstaben den Code gewählt, der nicht dem Standard-ASCII-Code entspricht, sondern demselben mit Bit 6 = 0. Den Erfolg dieser Strategie hat man nach dem Umbau auf der Hand bzw. auf dem Bildschirm: Statt dem gewohnten „MEMORY SIZE?“ erscheinen beim Einschalten allerlei merkwürdige Zeichen, und selbst derjenige, der glaubte seinen TRS-80 zu kennen, ist erstaunt ob der ungeahnten Möglichkeiten grafischer Darstellung, die der TRS-80 noch in sich birgt.

Durch die Hinzufügung des zusätzlichen RAM-Bausteins werden beim Auslesen die Codes für Buchstaben nicht mehr in den ASCII-Code umgewandelt, sondern in der Form, wie sie vom Displaytreiber eingeschrieben wurden, an den Zeichengenerator weitergegeben. Und der erzeugt dann diese interessanten Grafiken. Was tun? Als Lösungsmöglichkeiten bieten sich hard- und softwaremäßige Umschaltung an, die den zusätzlichen RAM-Baustein erst zuschalten, wenn ein neuer Displaytreiber geladen ist. Oder noch umständlicher: Blüdes Laden des neuen Displaytreibers, der die rechten „ASCII-Verhältnisse“ wieder herstellt.

Die Lösung, die in diesem Artikel vorgestellt werden soll, geht radikalere Wege. Auf die Hieroglyphen, die gerade noch auf dem Bildschirm erschienen, müssen wir dabei allerdings verzichten. Dafür erhält man aber normales Verhalten beim Einschalten mit dem Displaytreiber im ROM und Kleinschreibung nach Laden eines neuen Treibers.

Und wie funktioniert diese Lösung? Durch eine einfache Logikschaltung (Bild 3) werden beim Auslesen des Video-RAM alle Inhalte im Bereich von 00-1FH nach 40-5FH transferiert. Das heißt, es können weiterhin Buchstaben im „Tandy-Code“ eingeschrieben wer-

```

0458 DD 4E 03 LD L, (IX+03) ;CURSORADRESSE LADEN
0459 DD 46 04 LD H, (IX+04) ;NACH HL
045E 38 3A JR E, 049A ;CY? -> SPRUNG
0460 DD 7E 05 LD A, (IX+05) ;CURSOR ON?
0463 B7 OR A
0464 28 01 JR Z, 0467 ;NEIN
0466 77 LD (HL), A ;JA, ALTES ZEICHEN AUF CURSORPOSITION
0467 79 LD A, C ;ZEICHEN NACH A
0468 FE 20 CP 20 ;CONTROL CODE?
046A DA 04 05 JF C, 0506 ;JA
046D FE 80 CP 80 ;GRAPHIC ODER SPACE COMPRESSION?
046F 30 35 JF NC, 04A6 ;JA
0471 FE 40 CP 40 ;BUCHSTABE?
0473 38 08 JR C, 047D ;NEIN
0475 D4 40 SUB 40 ;NACH 0-3F TRANSFORMIEREN
0477 FE 20 CP 20 ;LOWER CASE?
0479 38 02 JR C, 047D ;NEIN
047B D4 20 SUB 20 ;IN UPPER CASE UMWANDELN
047D CD 41 05 CALL 0541 ;IN VIDEORAM UND SCROLL, WENN NOETIG
    
```

Bild 2. Der Displaytreiber des TRS-80, dessen Anfang hier aufgelistet ist, hat die Eigenschaft, für Buchstaben nicht ASCII ins Video-RAM einzuschreiben, was den Umbau auf Kleinbuchstaben unnötig erschwert

den, welche dann doch als Buchstaben sichtbar werden. Tabelle 2 zeigt das Verhalten der so modifizierten Schaltung.

**Der Einbau**

Nun endlich zur alles entscheidenden Operation, dem Eingriff ins Herz des Computers. Händler reagieren auf solche Eingriffe zwar meist mit Verlust der Garantie, aber das soll uns nicht stören, insbesondere wenn wir den Rechner schon länger als 6 Monate unser eigen nennen können. Außerdem kann mit folgender Anleitung sowie ein wenig Kenntnis im Umgang mit Schraubenzieher und Lötcolben kaum etwas schiefgehen. Also frisch ans Werk!

**Öffnen des Gehäuses**

Alle hierzu erforderlichen Schritte sind im folgenden zusammengestellt:  
a) Alle Kabelverbindungen von der Tastatur lösen und die Tastatur auf den Arbeitstisch legen.

b) Rechner umdrehen und mit einem Schraubenzieher die 6 Schrauben entfernen. Man bemerke hierbei die verschiedenen Längen der Schrauben und deren Korrelation zur Gehäusedicke, und be-

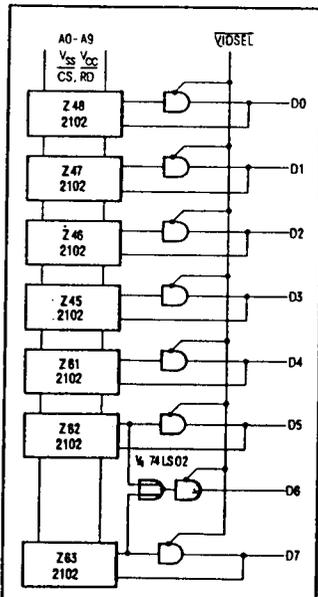


Bild 3. So muß die Videoschaltung des TRS-80 modifiziert werden, um die Darstellung von Kleinbuchstaben auf dem Bildschirm zu ermöglichen

**Tabelle 2: Modifizierte Zeichencodierung**

	schreiben	im Speicher	auf Bildschirm	lesen
	765	765	- Bit-Nr. -	765
	000	000	Großbuchstaben	010
	001	001	Ziffern, Sonderz.	001
	010	010	Großbuchstaben	010
	011	011	Ziffern, Sonderz.	001
	100	100	Grafik	100
	101	101	Grafik	101
	110	110	Grafik	100
	111	111	Grafik	101

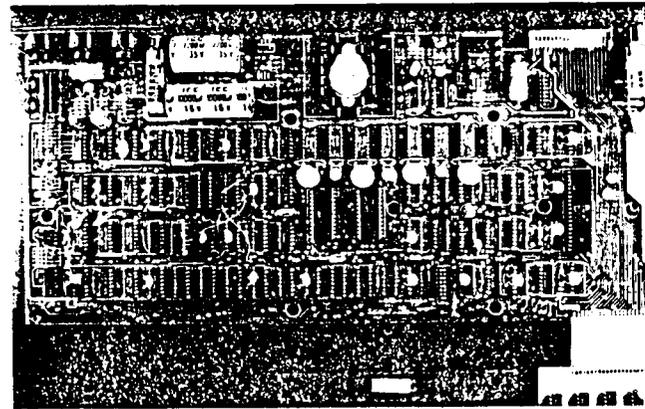


Bild 4. Die Hauptplatte des TRS-80. Die modifizierte Videoschaltung ist an den zahlreichen Drahtverbindungen zu erkennen

wahre die so gewonnene Kenntnis bis zum Zusammenbau.  
c) Nun wird die Tastatur vorsichtig umgedreht. (Man beachte, daß nach Entnahme der Schrauben nicht mehr viel Zusammenhalt vorhanden ist.)  
d) Deckel abnehmen. Bei älteren Modellen ist hierbei zu beachten, daß die Betriebsanzeige-LED nicht immer auf der Tastaturplatte montiert ist, sondern di-

rekt im Deckel und mit zwei Litzen mit der Tastaturplatte verbunden ist. Falls dies der Fall ist, löse man die LED vorsichtig aus dem Deckel.  
e) Tastaturplatte nach vorne wegklappen. Bei diesem und den folgenden Schritten ist darauf zu achten, daß das Verbindungskabel zwischen Haupt- und Tastaturplatte nicht belastet wird. (Durch Mißachtung dieser Vorsichts-

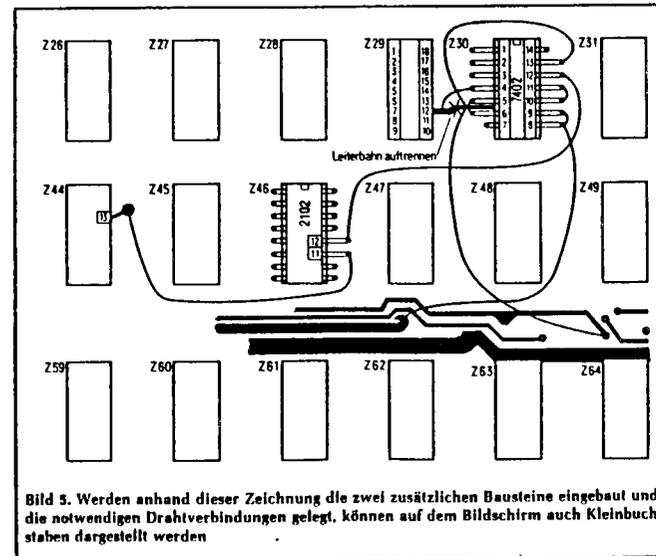


Bild 5. Werden anhand dieser Zeichnung die zwei zusätzlichen Bausteine eingebaut und die notwendigen Drahtverbindungen gelegt, können auf dem Bildschirm auch Kleinbuchstaben dargestellt werden

maßnahme kam der Verfasser zu dem zweifelhaften Vergnügen, dieses Verbindungskabel wegen Kabelbruchs austauschen zu dürfen.)

f) Die 6 Abstandhalter entfernen und den Boden des Gehäuses nach vorsichtigem Anheben der Hauptplatte entfernen.

g) Haupt- und Tastaturplatte umdrehen (Vorsichtsmaßnahme aus e) beachten), so daß Bestückungsseite oben zu liegen kommt.

**Der Umbau des TRS-80**

Nach der nun hoffentlich erfolgreich vollendeten Demontage folgt der eigentliche Eingriff. Dazu legt man die beiden Platinen mit dem benötigten Werkzeug zurecht. Auf der in Bild 4 dargestellten Hauptplatte ist der entscheidende Bereich eingerahmt. Man orte diesen Bereich auf der Platine und nehme dann die Skizze Bild 5 zur Hand. Anhand dieser Skizze, dem Foto in Bild 6, welches den gleichen Ausschnitt zeigt, und folgender Anleitung, ist der Umbau schnell und problemlos vollzogen. Man suche die Video-RAM-Bausteine Z45-48 und Z61-63. Zumindest einer von ihnen ist auf einen Sockel montiert. Wenn nicht (was bei sehr frühen Modellen vorkommen soll), nehme man entweder eine Entlötpumpe zur Hand und entferne damit Z46 aus der Platine oder vergesse den ganzen Umbau und baue den TRS-80 wieder zusammen. Diesen auf einen Sockel montierten Baustein (im folgenden sei angenommen, es handle sich um Z46) hebe man mit einem kleinen Schraubenzieher vorsichtig in leitenden Kunststoff. Unter diesen Vorichtsmaßnahmen sind kaum Probleme beim Umgang mit diesem MOS-IC zu befürchten. Zuvor hat man sich einen Baustein gleichen Typs besorgt (2102), von dem die Anschlüsse 11 und 12 vorsichtig waagrecht abgebogen werden.

Der so vorbereitete Baustein wird nun Huckepack auf den herausgenommenen Z46 gesteckt, wobei darauf zu achten ist das alle Pins übereinstimmen (Pin-1-Markierung!). Nun werden alle Anschlüsse außer Pin 11 und 12 miteinander verlötet. Wenn man nun den Baustein in den Sockel von Z46 zurücksteckt, ist der schwierigste Teil der Operation schon vollbracht. Nun orte man die Leiterbahn zwischen Z29 und Z30 und trenne sie mit einem scharfen Messer an der in Bild 5 bezeichneten Stelle auf. Jetzt nehme man den zweiten Baustein (74LS02) und biege alle Pins bis

auf die Pins 7 und 14 waagrecht ab. Die Pins 7 und 14 werden mit gleichen Pins des Z30 verlötet, wodurch Z30\* zugleich seine Stromversorgung erhält und mechanisch fixiert wird. Mit sieben kurzen Litzen werden schließlich noch die in Bild 5 eingezeichneten Verbindungen gelegt. Um unerwünschte Zusatzverbindungen zu vermeiden, wurden für alle Signale leicht zugängliche Lötunkte gewählt.

Test der Modifikation

Ist der Aufbau soweit gediehen und noch einmal überprüft, kann ein erster Test gewagt werden. Bildschirm und Stromversorgung werden dazu an die Hauptplatine angeschlossen und der Rechner eingeschaltet. Wenn jetzt die typische „MEMORY-SIZE“-Frage erscheint, kann man ziemlich sicher sein, daß man nicht allzuviel falsch gemacht hat. Mit dem Programm aus Bild 7 kann man den Erfolg des Umbaus nun testen. Beim erfolgreichen Umbau sollte dasselbe erscheinen wie in Bild 7 dargestellt. Falls dieses Erfolgserlebnis ausbleiben sollte, bleibt nichts anderes übrig als den gesamten Umbau noch einmal gründlich auf Fehler zu untersuchen. Durch Wiederherstellen der durch Durchtrennen der Leiterbahn zerstörten Verbindung (vorher Litze zu Pin 4 von Z30\* lösen!) kann der ursprüngliche Zustand zu Testzwecken wiederhergestellt werden.

Zusammenbau

Zu diesem Vorgang muß ich nicht viele Worte verlieren. Man lese die Anleitung und Bildfolge zu Schritt 1 rückwärts und gelangt hoffentlich zum gewünschten Ergebnis. Bleiben schließlich noch Abstandhalter, Schrauben oder sonstige Bauteile übrig, dann muß wohl irgend etwas schiefgegangen sein.

Die Software

Ganz ohne Software geht es leider nicht. Wer Diskettenbetriebssysteme mit Kleinbuchstaben-Treibern besitzt (z. B.: LCDVR im NEWDOS80 V.1) oder sogar solche, die durch Setzen einer System-Option auf Kleinbuchstaben umgeschaltet werden können, ist natürlich aus dem Schneider (mit System :0,BF=Y,BG=Y im NEWDOS80 V.2; in beiden NEWDOS80-Versionen empfiehlt es sich, zusätzlich nach Installation des Kleinschreibungsombaus die Optionen AI und AS folgendermaßen zu setzen: SYSTEM :0,AI=Y,AS=N um dadurch

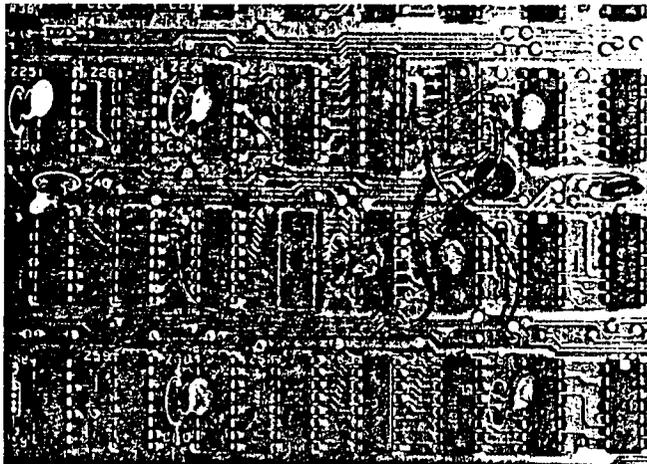


Bild 6. Dieser Ausschnitt zeigt die Videoschaltung auf der Hauptplatine des TRS-80, die für Kleinschreibung umgerüstet ist. Der zusätzliche RAM-Baustein Z46 wird Huckepack auf Z46 montiert. Dieser Aufbau empfiehlt sich, da beide Bausteine bis auf 2 Pins identisch verdrahtet sind (Alle Fotos: Aegidius Röckrath)

die Umwandlung von Klein- in Großbuchstaben bei Ein- und Ausgabe zu verhindern).

Benutzer weniger komfortabler Betriebssysteme können mit den im folgenden beschriebenen Treibern den gleichen Effekt und bei Verwendung des AUTO-Kommandos den gleichen Komfort erreichen. Letzteres wird Kassettbenutzern allerdings verwehrt bleiben. Sie müssen weiterhin von Hand den Treiber laden,

um in den Genuß der Kleinschreibung zu gelangen.

Bild 8 zeigt den Treiber, für dessen bereitwillige Zur-Verfügung-Stellung ich mich an dieser Stelle bei Michael Büning bedanken möchte. Das Treiberprogramm enthält einen neuen Tastatur- und Displaytreiber. Der Displaytreiber verhindert nur die Umwandlung von Klein- in Großbuchstaben und deren Umwandlung in den „Tandy-Code“. Der neue Tastaturtreiber ermöglicht die Eingabe von Kleinbuchstaben im normalen



Bild 7. Mit diesem einzelnen Testprogramm kann der Erfolg des Umbaus überprüft werden. Wenn alles in Ordnung ist, erscheint auf den obersten vier Zeilen des Bildschirms das gleiche wie abgebildet

```

00100 ;***** LOWER-CASE-UMLAUTE-DRIVER FUER MODIFIZIERTEN TRS-80 *****
00110 ;
FF75 21E2FF 00120 ORG OFF75H
FF7B 221E40 00130 LCLUM1 LD HL,NEWVID
FF7B 2184FF 00140 LJ (401EH),HL ;NEUE VIDEO-DRIVER-ADRESSE
FF7E 221640 00150 LD HL,NEWKEY
FFB1 C3E700 00160 LD (4016H),HL ;NEUE KEYBOARD-DRIVER-ADRESSE
00170 JP 00E7H ;SPRUNG NACH BASIC-READY
00180 ;
00190 ;***** NEW KEYBOARD-DRIVER-SUBROUTINE *****
00200 ;
FFB4 CDE303 00210 NEWKEY CALL 03E3H ;NORMALER KEYBOARD-DRIVER
FFB7 C5 00220 PUSH BC
FFB8 D5 00230 PUSH DE
FFB9 57 00240 LD D,A
FFBA FE20 00250 CP ZOH
FFBC 2011 00260 JR NZ,NOSPAC
FFBE 3A803B 00270 LD A,(3B80H) ;SHIFT GEDRUECKT?
FF91 87 00280 OR A
FF92 2B08 00290 JR Z,NOSPAC
FF94 3A01FF 00300 LD A,(FLAG)
FF97 EE20 00310 XOR ZOH ;SHIFTLOCK INVERTIEREN
FF99 32D1FF 00320 LD (FLAG),A ;FLAG STATUS UMGKEHRT
FF9C AF 00330 IGNORE XOR A ;AUSSPRUNG MIT A=0
FF9D 1B2F 00340 JR RUECK
FF9F 7A 00350 NOSPAC LD A,D
FFA0 2E1F 00360 CP 1FH ;CLEAR IGNORIEREN
FFA2 2BFB 00370 JR Z,IGNORE
FFA4 3A403B 00380 LD A,(3B40H)
FFA7 FE02 00390 CP Z ;CLEAR ZUSAETZLICH GEDRUECKT?
FFA9 2010 00400 JR NZ,NORMAL
FFAB 7A 00410 LD A,D
FFAC 21D2FF 00420 LD HL,ALT
FFAF 010B00 00430 LD BC,NEU-ALT ;BC=LAENGE DER TABELLE
FFB2 EDB1 00440 CPIR
FFB4 2005 00450 JR NZ,NORMAL
FFB6 010700 00460 LD BC,NEU-ALT-1
FFB9 09 00470 ADD HL,BC
FFBA 56 00480 LD D,(HL) ;UMCODIERTES ZEICHEN IN D LADEN
FFBB 7A 00490 NORMAL LD A,D
FFBC E6DF 00500 AND 0DFH
FFBE FE41 00510 CP 41H
FFC0 3B08 00520 JR C,SONDER ;AUSSPRUNG FALLS NICHT ALPHA ZEICHEN
FFC2 FE5E 00530 CP 5EH
FFC4 3007 00540 JR NC,SONDER ;AUSSPRUNG FALLS NICHT ALPHA ZEICHEN
FFC6 7A 00550 LD A,D
FFC7 21D1FF 00560 LD HL,FLAG
FFCA AE 00570 XOR (HL) ;GEMAESS SHIFTLOCK GROSS ODER KLEIN GENERI
EN
FFCB 1B01 00580 JR RUECK
FFCD 7A 00590 SONDER LD A,D
FFCE D1 00600 RUECK POP DE
FFCF C1 00610 POP BC
FFD0 C9 00620 RET
FFD1 00 00630 FLAG DEFB 00
FFD2 41 00640 ALT DEFB 41H ;A
FFD3 4F 00650 DEFB 4FH ;D
FFD4 35 00660 DEFB 53H ;U
FFD5 61 00670 DEFB 61H ;KLEIN A
FFD6 6F 00680 DEFB 6FH ;KLEIN O
FFD7 75 00690 DEFB 75H ;KLEIN U
FFD8 53 00700 DEFB 53H ;S
FFD9 43 00710 DEFB 43H ;C
FFDA 5B 00720 NEU DEFB 5BH ;AE
FFDB 3C 00730 DEFB 3CH ;OE
FFDC 5D 00740 DEFB 5DH ;UE
FFDD 7B 00750 DEFB 7BH ;KLEIN AE
FFDE 7C 00760 DEFB 7CH ;KLEIN OE
FFDF 7D 00770 DEFB 7DH ;KLEIN UE
FFE0 7E 00780 DEFB 7EH ;SZ
FFE1 1F 00790 DEFB 1FH ;CLEAR
00800 ;
00810 ;***** NEW VIDEO-DRIVER-SUBROUTINE *****
00820 ;
    
```

Bild 8. Leider kann die Kleinschreibung nur mit einem kleinen Treiberprogramm genutzt werden, da sie von der ROM-Software nicht unterstützt wird. Die Kassettversion für 48-KByte-Systeme ist hier im Sourcelisting abgedruckt

# TRS 80 und Video Linie — INLINE, PRINT und LPRINT

Manchmal ist es erforderlich, Satzzeichen in einen String zu bringen, wozu der Befehl **INLINE** benützt wird, welcher in Level II nicht enthalten ist. Mit drei Basic-Zeilen (120 Byte) kann In-line durch ein Unterprogramm ersetzt werden. Es können 255 Zeichen im String übernommen werden, wobei aber auch **LEN** und **VARPTR** funktioniert.

**Beschreibung:**  
 Zeile 1: S\$ zurücksetzen.  
 Zeile 2: Cursor ausgeben.  
 Zeile 3: T\$ zurücksetzen, auf Tastendruck warten.

Nach Tippen einer Taste eine Position zurück und Zeichen ausgeben. War Eingabe Linkspfeil, letztes Zeichen aus S\$ entfernen, Cursor ausgeben, erneut auf Tastendruck warten, sonst zurück nach 2  
 Zeile 4: Ist das Zeichen ENTER, zurück in das Programm, sonst letztes Zeichen an S\$ anfügen, Cursor ausgeben und nach Zeile 3.

```
1 S$=""
2 PRINT " ";GOSUB3:IF ASC(T$)=13, RETURN ELSE
S$=S$+T$:GOTO2
3 T$=" ";T$=INKEY$:IFT$="" , 3 ELSE PRINT
CHR$(8);IFASC(T$)=8,S$=LEFT$(S$,LEN(S$)-1):
PRINT " ";GOTO3 ELSE RETURN
```

**Tip für PRINT und LPRINT:**  
 Wenn Sie Text oder Daten auf dem Bildschirm oder Drucker ausgeben müssen, schreiben Sie **POKE P,L:PRINT**.  
**P = 16540** und **L = 0** Ausgabe auf dem Bildschirm.  
 oder **L = 1** Ausgabe auf den Drucker.  
 So ersparen Sie sich, für **PRINT** und **LPRINT** getrennte Zeilen zu schreiben.

(G.Mulzheim)



„Gestehen Sie, daß Sie Ihre Schwiegermutter vom Balkon gestoßen haben, Angeklagter?“ „Ja, Herr Richter, das stimmt, aber ich wollte mich nur überzeugen, ob Drachen wirklich fliegen können.“

	00100 ;***** LOWER-CASE-UMLAUTE-DRIVER FUER MODIFIZIERTEN TRS-80 *****			
	00110 ;			
FF72	00120	DRG	OFF72H	
FF72 21E2FF	00130 LCUML1	LD	HL, MEMVID	
FF75 221E40	00140	LD	(401EH), HL	;NEUE VIDEO-DRIVER-ADRESSE
FF78 2184FF	00150	LD	HL, MEMKEY	
FF7B 221640	00160	LD	(4016H), HL	;NEUE KEYBOARD-DRIVER-ADRESSE
FF7E 2B	00170	DEC	HL	
FF7F 224940	00180	LD	(4049H), HL	;DOS-HIEM UNTER LCUML1
FF82 AF	00190	XOR	A	
FF83 C9	00200	RET		;ZURUECK INS DOS
	00210 ;			

Bild 9. Die Version des Treibers für Diskette erfordert einige kleine Änderungen gegenüber der Kassettenversion (Bild 8). Die abweichenden Quellprogramm-Statements sind hier abgedruckt. Der gesamte Rest ist identisch

Schreibmaschinenmodus und umgekehrt und bietet eine behelfsmäßige Möglichkeit zur Verarbeitung von Umlauten (der verwendete Algorithmus zur Codewandlung, der sich des CP1R-Befehls bedient, ist recht universell zu verwenden und sehr schnell). Die Bedienung des Treibers ist denkbar einfach. Sobald der Treiber aktiv ist, können durch zusätzliches Drücken der Shift-Taste alle Kleinbuchstaben erzeugt werden und werden auch als solche auf dem Bildschirm angezeigt. Mit Shift-Space und Shift-0 kann in den Schreibmaschinen-Modus geschaltet werden bzw. dieser wieder verlassen werden. Schreibmaschinen-Modus bedeutet: Kleinbuchstaben ohne und Großbuchstaben mit Shift, ganz wie man es von einer normalen Schreibmaschine gewohnt ist. Mit Clear und A, O, U können die Umlaute Ä, Ö, Ü erzeugt werden, und mit Clear-S das ß. Man wundere sich nicht über die merkwürdigen Dinge, die dabei auf dem Bildschirm erscheinen (auch da ist Abhilfe möglich, s. u.). Mit Shift kann auch hier wieder zwischen groß und klein unterschieden werden. Da die Clear-Taste fortan diese Umschaltfunktion erfüllt, kann der Bildschirm nur noch durch gleichzeitiges Drücken von Clear und C gelöscht werden. Bild 8 zeigt die Kassettenversion des Treibers in einem Quellprogrammlisting. Mit EDTASM oder einem Monitor

erstellt man daraus ein Objekt-File, welches dann mit dem SYSTEM-Befehl geladen und mit / gestartet werden kann. Dabei ist keine Antwort auf die „MEMORY SIZE?“-Frage beim Einschalten erforderlich, denn das Programm schützt sich selbst vor dem Zugriff durch den Basic-Interpreter. Bild 9 zeigt die ersten Quellprogrammzeilen der Diskettenversion des gleichen Treibers, die sich von denen der Kassettenversion unterscheiden. Unter DOS kann man ihn schnell und einfach mit dem DOS-Debugger in den Speicher schreiben und dann mit dem DUMP-Kommando als CMD-File auf Diskette abspeichern. Der Treiber wird fortan im DOS durch Angabe seines Filenamens geladen und gestartet, wobei er wie die Kassettenversion den Speicherbereich, den er belegt, vor unberechtigten Zugriffen schützt, so daß beim Aufruf des Basic die „MEMORY SIZE?“-Frage guten Gewissens mit Enter beantwortet werden kann.

### Textverarbeitung

Nachdem der Kleinschreibungsumbau mühevoll vollzogen ist, stellt man leider fest, daß der TRS-80 in den USA entwickelt wurde. Für die dortigen Verhältnisse wäre er jetzt für Textverarbeitung vollkommen zufriedenstellend ausgerüstet und nichts stünde mehr im Wege. Aber leider erlaubt die deutsche Sprache

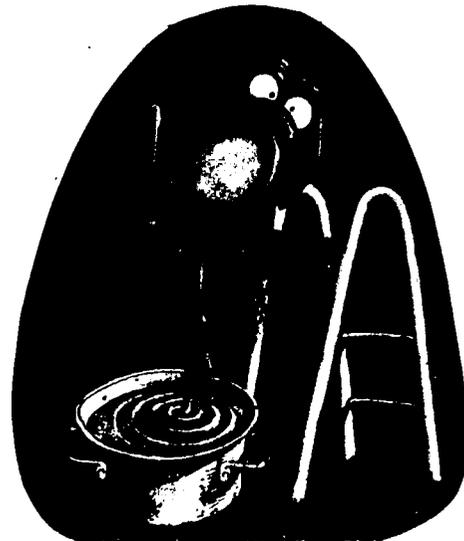
sich den Luxus von Umlauten und des scharfen ß; Buchstaben, die weder als Tasten vorhanden sind noch auf dem Bildschirm dargestellt werden können. Durch Definition einer Umlaut-Umschalttaste (wie in dem oben vorgestellten Treiber) kann die Eingabe der Umlaute ermöglicht werden, wobei die etwas merkwürdige Darstellung auf dem Bildschirm etwas Kompromißbereitschaft erfordert.

Wer diese Kompromißbereitschaft nicht besitzt und auch noch die fehlenden Umlauten bemängelt, der sollte einen weiteren Umbau in Kauf nehmen und einen neuen Zeichengenerator installieren, der all diese Mängel beseitigt und dabei noch bis zu drei verschiedene Zeichensätze bietet. Wenn man einmal soweit ist, ist auch das letzte Hemmnis, die fehlenden Umlauttasten, keine unüberwindliche Barriere mehr. Hier gibt es neuerdings Umbauten, die solche Tasten zusätzlich in den TRS-80 einbauen, wobei allerdings solche Umbauten immer nur so wertvoll wie die mitgelieferte Softwareunterstützung sind, sprich Modifikationen bekannter Textsysteme, wie SCRIPSIT oder PENCIL, die mit solchen Umbauten zusammenarbeiten. Die eleganteste Möglichkeit ist wohl, eine Typenrad-Schreibmaschine sowohl zur Eingabe als auch Ausgabe zu verwenden.

Hier angekommen wird man sich fragen, ob es nicht einfacher gewesen wäre, gleich einen Rechner mit entsprechender Ausrüstung zu erwerben, als nachträglich alle Fehler mühsam zu korrigieren. Für Erstanwender, die einen neuen Computer, der auf ihre Anwendung zugeschnitten ist, erwerben wollen, ist die Frage natürlich eindeutig mit ja zu beantworten. Aber wenn ein Rechner einmal vorhanden ist, ist die nachträgliche Umrüstung erheblich billiger und bietet unschätzbare Einblicke in das Innenleben des Rechners.

### Literatur

[1] Röckroth, Luidger: TRS-80-ROM-Listing, mc 1982, Heft 1, S. 12.



FFE2 D06E03	00830 NEWVID	LD L, (1X+3)	ALT FFD2 00640	00420 00430 00460
FFE5 D06604	00840	LD H, (1X+4)	FLAG FFD1 00630	00300 00320 00560
FFEB D49A04	00850	JP C, 049AH	IGNORE FFC9 00330	00370
FFED D07E03	00860	LD A, (1X+5)	LABEL FFF2 00900	008B0
FFEE B7	00870	OR A	LCUML1 FF75 00130	00960
FFEF 2B01	00880	JR Z, LABEL	NEU FFDA 00720	00430 00460
FFF1 77	00890	LD (HL), A	MEMKEY FFB4 00210	00150
FFF2 79	00900	LABEL LD A, C	MEMVID FFE2 00830	00130
FFF3 FE20	00910	CP 20H	NORMAL FFB8 00490	00400 00450
FFF5 D40603	00920	JP C, 0506H	NOSPAC FFF9 00350	00260 00290
FFF8 FE80	00930	CP 80H	RUECK FFCE 00600	00340 00580
FFFA D2A604	00940	JP NC, 04AAH	SONDER FFCD 00590	00520 00540
FFFD C37D04	00950	JP 047DH		
FF75	00960	END LCUML1		
00000	TOTAL ERRORS			
33210	TEXT AREA BYTES LEFT			

Alle 3 Programme stammen vom Dieter. Mit dem 3. Programm brach er alle Rekorde!

### PRIMZAHLEN 1

```
1 DEFSNGA-Z:DIMP(8500):P(1)=1.1:Z=1:CLS:INPUT"Anfang: ":A:INPU
T"Ende : ":E:IFA=10RA=2THENPRINT"2":A=3
2 IFCINT(A/2)=A/2THENA=A+1
3 FORI=ATOESTEP2:FORJ=1TOZ-1:IFSQR(I)<P(J)THENGOTO4ELSEIFCINT
(I/P(J))=I/P(J)THENNEXTIELSENEXTJ
4 P(Z)=I:Z=Z+1:PRINTI,Z:NEXTI:PRINT"FERTIG !"
```

### PRIMZAHLEN 2

```
1 CLS:'----- PRIMZAHLENBERECHNUNG -----
2 INPUT" Bis zu welcher Grenze sollen die Primzahlen berechnet
werden ":H:Z
3 FORA=3TOH/STEP2:W=INT(SQR(A))
4 FORB=3TOWSTEP2:X=INT(A/B)*B:IFX=ATHEN6
5 NEXTB:PRINTA:
6 NEXTA
```

### PRIMZAHLEN 3

```
1 CLS:'----- PRIMZAHLENBERECHNUNG -----
2 DEFINTA-H
3 INPUT" BIS ZU WELCHER GRENZE SOLLEN DIE PRIMZAHLEN BERECHNET
WERDEN ":H
4 DIMA(Z)
5 FORA=2TOSQR(H):IFAZ(A)=1THEN7
6 PRINTA:FORB=A*ATOHSTEPA:A%(B)=1:NEXT
7 NEXT
8 FORB=ATOH:IFAZ(B)=0PRINTB:
9 NEXT
```

Berechnung nach dem 'Sieb von Eratosthenes'  
Primzahlen bis 10000 in 3-4 Minuten !!!



Unser Sieger

### Ein (steribser BASIC-Befehl: IsA

Wenn man Maschinenprogramme à la Leo Christopherson in Stringvariable packt und die BASIC-Programme mit diesen Variablen anschließend listet, erlebt man so manche Überraschung. Z. B. kann dabei der Befehl IsA auftauchen. Ich wollte untersuchen, was er bewirkt. Um das Ergebnis vorwegzunehmen: Nichts als einen ?SN-Error.

Da gibt Richard Straw in seinem Artikel "How the Level II Interpreter Sees It"<sup>1</sup> eine Tabelle der BASIC-Befehle mit ihren Tokens wieder. So gut der Artikel, so schlecht ist die Tabelle. Das Token FB (251)<sup>2</sup> steht für das Zeichen ', den REM-Ersatz. In der Tabelle steht nichts davon. Diese Lücke setzt sich (von hier ab allerdings zu Recht) fort bis FE (254). Diese Codes entsprechen wirklich keinen BASIC-Befehlen. Aber bei FF (255) schreibt Straw brav ISA hin, weil er das bei seinem Tandy ohne Kleinschrift so gesehen hat. Er ist der Sache wohl nicht nachgegangen, sonst hätte er feststellen müssen, how the level II interpreter nämlich wirklich sees it.

Da die CPU kein Englisch und auch kein BASIC kann, schlägt ihr Dolmetscher, der Interpreter (engl. Dolmetscher) bei jedem BASIC-Token in einer Art Wörterbuch nach, einer Tabelle, die bei 1650 (5712) beginnt. Die einzelnen Befehlsörter, die (fast) in Klarschrift hier stehen, werden durch das gesetzte Bit 7 des jeweils ersten Buchstabens eines jeden Wortes voneinander unterschieden. So heißt z. B. REM hier 02 45 40 (210 69 77) statt 52 45 40 (82 69 77). Der Interpreter findet nach dem LIST-Befehl im Programmtext das Token 93 (147) und setzt von diesem Token zunächst das Bit 7 zurück. Das tut er, indem er 80 (128) subtrahiert. So entsteht aus 93 (147) 13 (19). Jetzt sucht er 19mal nach einem gesetzten Bit 7 in der Tabelle und findet die obigen Codes. Er setzt vom ersten Zeichen das Bit 7 zurück und schreibt dieses und die nachfolgenden Zeichen auf den Bildschirm. Wenn er ein Zeichen findet, bei dem wieder das Bit 7 gesetzt ist, hört er damit auf, denn dort beginnt das nächste Befehlswort. Auf dem Screen stehen jetzt die Buchstaben R-E-M für REM.

Bei dem "Token" (es ist keins) FF (255) sucht er eben 127mal, denn 255-128=127. Er findet bis zum nächsten Bit 7 ab 182A (6186) folgende Bytes: C9 01 73 41 (201 1 115 65). Er setzt sodann das Bit 7 von C9 (201) pflichtgemäß zurück, so daß 49 (73) daraus wird. Kreuzbrav, denn er hat nichts gemerkt, schreibt er nun nach dem oben skizzierten Strickmuster die ASCII-Äquivalente dieser Codes auf den Bildschirm. 01 bedeutet ihm nichts, der Rest heißt dann eben IsA. So kann auch ein Genie irren.

Der casus craxus ist, daß hier die Tabelle längst zuende ist. Das unwiderruflich letzte Token, FB (251) für ' (REM-Ersatz) ist bereits überschritten. Wir befinden uns schon in der anschließenden Tabelle, wo die Adressen der Bearbeitungs-routinen dieser BASIC-Befehle verzeichnet sind.

Dieselbe Erklärung gilt auch für das Phänomen, daß bei derartigen Listings häufig auf 32 Zeichen/Zeile umgeschaltet wird, der Cursor plötzlich in die Home-Position geht usw.. Beim Genie 3 oder beim Genie 1/2 mit H-DOS und EG 64 MBA piept es sogar gelegentlich. Alle Codes ≥FC (252) werden irrtümlich für Tokens gehalten und die in der falschen Tabelle gefundenen "Befehlsörter" (die keine sind) buchstabengetreu angezeigt. Bei Codes ≤19 (31) passiert im Prinzip dasselbe: Wenn z. B. 1C (28) auftritt, geht der Cursor an den Bildschirmanfang.

Dick Straw ist gewiß nicht dumm, aber vielleicht ein bißchen zu fix mit der Feder bzw. der Tastatur bei der Hand.

Und was bringt das alles dem Programmierer? Rein gar nichts. Aber ist es nicht von allgemeinem Vorteil, die Eigenheiten des Interpreters zu kennen, wenn man sich seiner via BASIC bedient?

Arnulf Sopp, Tel. 0451-791926

<sup>1</sup>80-US 5/79, in "The First Book of 80-US", Hofacker-Verlag  
<sup>2</sup>alle Zahlen sedezimal, dahinter in Klammern dezimal

Ein Beitrag von Josef Konrad  
 =====

```

10 REM DATUM-KONVERTIERUNG VERSION 1
20 REM
30 DIM MO$(12)
40 CLS
50 DEFFN MO$(M)=MID$(" JANFEBMAEAPRMAIJUNJULAUAGSEPOKTNVDEZ",M*3,3)
)
60 M=VAL(LEFT$(TIME$,2)):Y$=FN MO$(M)
70 DA$=MID$(TIME$,4,2)
80 YR$=MID$(TIME$,7,2):YR$="19"+YR$
90 DT$=DA$+"."+"Y$+" "+"YR$
100 PRINT@40,DT$
110 REM
120 REM
130 REM
140 REM
150 REM DATUM-KONVERTIERUNG VERSION 2
160 REM
170 CLS
180 X=VAL(LEFT$(TIME$,2))
190 DA$=MID$(TIME$,4,2)
200 YR$=MID$(TIME$,7,2):YR$="19"+YR$
210 FOR NX=1 TO X:READ MO$:NEXT NX
220 DT$=DA$+"."+"MO$+" "+"YR$
230 DATA "JANUAR","FEBRUAR","MAERZ","APRIL","MAI","JUNI","JULI","AUGUST",
"SEPTEMBER","OKTOBER","NOVEMBER","DEZEMBER"
240 PRINT@45,DT$

```

## Am Anfang war das Chaos

Ein Arzt, ein Architekt und ein Datenverarbeiter sitzen gesellig beisammen. Unversehens taucht die Frage auf, wer von ihnen wohl den ältesten Beruf ausübe.

Der Mediziner verkündet siegesicher, diesen Ruhm könne er gewiß für seinen Stand beanspruchen. Denn schließlich heiße es, daß Gott die Eva aus einer Rippe Adams geschaffen habe, und das sei ja wohl eine chirurgische Tat.

Entgegnet lächelnd der Architekt: sein Beruf sei doch um einiges älter. Schon im Ersten Buch Moses stehe schwarz auf weiß, wie Gott die Welt aus dem Chaos schuf – und dabei handele es sich offenkundig um eine architektonische Leistung.

Daraufhin erhebt sich der Datenverarbeiter und fragt trocken: »Und wer schuf das Chaos?«

von Josef Konrad

```

*****
00100 ***** SCHUETZEN VON ASSEMBLER-PROGRAMMEN MODELL I *****
00110 ***** DIESE BEFEHLSFOLGEN RESERVIEREN DEN SPEICHER- *****
00120 ***** BEREICH AB 'START'. *****
00130 ***** SIE WERDEN VOR DAS ZU SCHUETZENDE PROGRAMM *****
00140 ***** GESCHRIBEN. *****
00150 *****
00160 *****
00170 *****
00180 ***** FUER LEVEL II BASIC *****
00190 *****
00200 ***** LD HL,START *****
00210 ***** JP 00E7H *****
00220 *****
00230 ***** FUER DOS *****
00240 *****
00250 ***** LD HL,START *****
00260 ***** DEC HL *****
00270 ***** LD (4049H),HL *****
00280 ***** XOR A *****
00290 ***** RET *****
00300 *****
00310 *****
00320 ***** EQU $$$ *****
00330 ***** START *****
00340 *****
00350 ***** END *****
00000 ***** TOTAL ERRORS *****
32552 ***** TEXT AREA BYTES LEFT *****

0000 210000
0003 03E700

0006 210000
0009 2B
000A 224940
000D AF
000E C9

0000
0000
0000 TOTAL ERRORS
32552 TEXT AREA BYTES LEFT

START 0000 00330 00200 00260

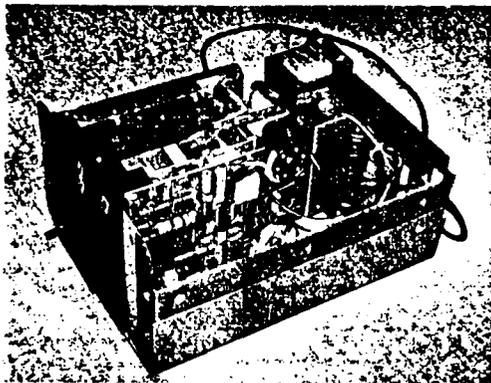
```



# Modernisierung statt Neukauf

Wer schon seit einigen Jahren glücklicher Besitzer eines Computers ist, muß feststellen, daß die Technik in der Zwischenzeit nicht stehengeblieben ist. Er steht dann vor der Frage, ob er sein Gerät gegen ein neues austauschen soll, was meistens auch einen Wechsel von Systemsoftware nach sich zieht. Der Verfasser des folgenden Artikels hat sich für eine Modernisierung seines TRS-80 entschieden.

Bild 3. Marktgängiges Diskettenlaufwerkgehäuse mit erweitertem Netzteil



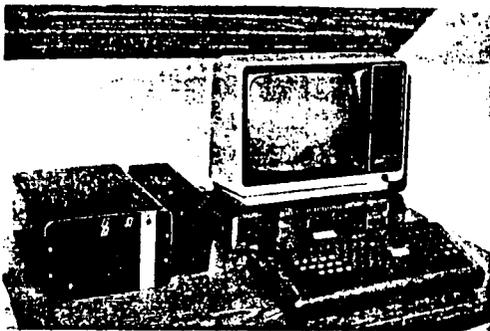
Der TRS-80 Modell 1 des Verfassers stammt aus dem Jahre 1978. Er hat die Seriennummer 16286 und dürfte somit einer der ältesten in der BRD laufenden TRS-80 Modell 1 sein. Seit 1978 hat es bei den Tischcomputern viele technische Fortschritte gegeben. Ende 1982 kam der Verfasser auf die Idee, sich auf dem Markt einen modernen Nachfolger für den TRS-80 auszusuchen. Das Ergebnis der dabei gemachten Marktstudie war unerfreulich für die Computershops und erfreulich für den eigenen Geldbeutel:

1. Die vielgepriesenen Fortschritte

2. Zum beim Verfasser benutzten Disketten-Betriebssystem NEWDOS-80-V2 gibt es keine gleichwertige Alternative. Ein Wechsel zu einem CP/M-fähigen System oder zum DOS 3.3 des Apple II erscheint beim Vergleich der Möglichkeiten der Betriebssysteme als Rückschritt.
3. Obwohl der TRS-80 insgesamt und sein Basic insbesondere recht



▲ Bild 1. Zeichengenerator mit Unterlängen und deutschen Umlauten



◀ Bild 2. Modernisierter TRS-80 Modell-1 mit Zenith-Bildschirm

langsam ist (das heißt in Zeitschriften-Benchmarktests schlecht abschneidet), ist das Basic des TRS-80 insbesondere unter NEWDOS-80-V2 leistungsfähiger als sämtliche dem Verfasser bekannten Interpreter-Basic-Dialekte für Tischcomputer.

Unter dem Eindruck dieser Erkenntnisse entschloß sich der Verfasser, anstelle einer Neuanschaffung eine Modernisierung des vorhandenen TRS-80 durchzuführen. Ziel der Modernisierung sollte sein:

- die Leistungsfähigkeit der TRS-80-Anlage erhöhen,
- die technisch sinnvolle Nutzungsdauer der TRS-80-Anlage so weit zu verlängern, bis die technischen Fortschritte auf dem Gebiet der Mikroelektronik den im TRS-80 Modell 1 verwendeten Mikroprozessor Z80 eindeutig als veraltet erscheinen lassen.

Ein wesentlicher Vorteil der Modernisierung gegenüber einem Neukauf eines anderen Systems darf nicht übersehen werden: Bei jedem Systemwechsel ist ein beachtlicher Aufwand zur Übernahme der Software und zur Neubeschaffung der Systemsoftware zu leisten. Bei einer Modernisierung entfällt dieser Aufwand vollkommen.

## Alles aus dem Versandhandel

Praktisch alles wurde über den Versandhandel beschafft. Aufgrund von Anzeigen in den bekannten Tischcomputer-Zeitschriften

entsprechend angepaßtes Betriebssystem: Am besten ist einmal mehr NEWDOS-80-V2 geeignet. Nach Einbau des Doublers ist die Nutzung der Disketten mit einfacher Aufzeichnungsdichte weiterhin möglich, und der TRS-80 bleibt

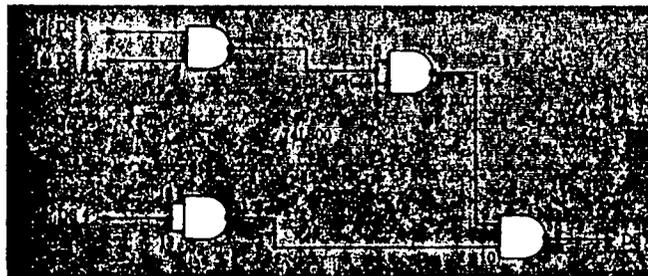


Bild 4. Selektion von Laufwerk 3 für den TRS-80 Modell-1 mit einem 74LS00

ten wurden die Kontakte zu den Händlern aufgenommen. Händler, die Anfragen nicht beantworteten (so etwas gibt es wirklich!), erhielten selbstverständlich keine Bestellung. Bezahlt wurde mit Verrechnungsscheck (der Verfasser hat eine Abneigung gegen Nachnahme) oder — bei Bestellungen in den USA — per Visa-Kreditkarte. Die schnellste Lieferung kam innerhalb von vier Tagen (Computerstudio Braunschweig). Keine innderdeutsche Bestellung hat länger als zwei Wochen gedauert, keine Ware mußte reklamiert werden.

Bei der Lieferantenauswahl war der Preis das wesentliche Auswahlkriterium. Preisunterschiede von 20 Prozent sind bei gleicher Hardware normal, und es lohnt sich deshalb für sachkundige Käufer, die Lieferanten zu wechseln.

## Die ersten Schritte zum »neuen« Gerät

Die beiden ersten Modernisierungsstufen erfolgten bereits in den letzten Jahren:

1. Stufe: Geschwindigkeitsumschalter für den CPU-Takt.
2. Stufe: Neuer Zeichengenerator mit Unterlängen und deutschen Umlauten (Bild 1).

Die beiden für die Erweiterungen erforderlichen zusätzlichen Platinen wurden selbst eingebaut.

3. Stufe: Doppelte Schreibdichte auf Disketten.  
Beim TRS-80 besteht die Möglichkeit, die Diskettenaufzeichnungskapazität durch Einbau eines sogenannten »Doublers« in das Expansion Interface um 80 Prozent zu erhöhen. Die Doubler benötigen ein

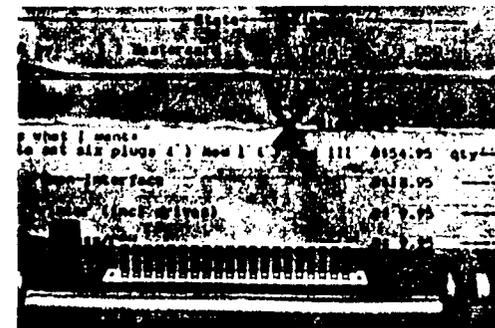


Bild 5. Die aufgelöteten, vergoldeten Kartenstecker stehen etwas vor

voll kompatibel zu seinem Ursprungszustand. Mit einiger Mühe beim Ausschuchen konnte der Verfasser etwa 30 Prozent der vorhandenen, für einfache Schreibdichte vorgesehenen Disketten auch für doppelte Schreibdichte verwenden. Wichtig ist, daß man sich vor dem Kauf des Doublers beim Lieferanten erkundigt, ob die vorhandenen Laufwerke für doppelte Schreibdichte geeignet sind.

Der übliche Preis für einen Doubler beginnt je nach Lieferanten bei etwa 230 Mark. Wenn man ausländischen Computerzeitschriften Glauben schenken darf, bietet Tandy/Radio Shack in den USA Doubler für den TRS-80 Modell-1 an.

## 4. Stufe: Ein neues Gehäuse für zwei Diskettenlaufwerke

Ursache für diese Investition war das bei einem vorhandenen Diskettenlaufwerk mitgelieferte Gehäuse. Es war von billigstmöglicher Machart, und seine Seitenteile bestanden aus Kunststoff (einer der Freunde des Verfassers pflegt in diesem

Zusammenhang nicht ganz wertfrei von Pappmachegehäusen zu reden). Wenn das Laufwerk in einem solchen Gehäuse nahe am Bildschirmmonitor steht, treten beim Betrieb mit doppelter Schreibdichte Schreib-/Lesefehler auf. Dieses Gehäuse wurde gegen ein marktgängiges aus Aluminiumblech für zwei Diskettenlaufwerke mit eingebautem Netzteil ersetzt. Es wurde durch den Verfasser so umgebaut, daß es nun zwei Netzteile für insgesamt vier Diskettenlaufwerke enthält (Bilder 2 und 3).

## 5. Stufe: Das neue Diskettenlaufwerk

Anfang 1983 bewegten sich die Preise für Diskettenlaufwerke rapide nach unten. Es gelang dem Verfasser, ein BASF-Laufwerk preiswert zu erwerben, welches als logisches Laufwerk mit der Adresse 3

in das oben genannte Gehäuse eingebaut wurde (Bild 3).

## Probleme bei der Steckerbeschaffung

Immer dann, wenn man Laufwerke ohne Gehäuse erwirbt, hat man Probleme mit der Kabel- und Steckerbeschaffung. Für den Verfasser war es ein glücklicher Umstand, daß Radio Shack an seinem Wohnort Ende 1982 ein Computer Center eröffnet hat. Ein Telefonanruf genügte, um zu erfahren, daß die benötigten Stecker auf Lager waren. Die Ersatzteilversorgung bei Radio Shack ist — der Verfasser hat es bisher dreimal ausprobiert — ausgezeichnet.

Da das Laufwerk die logische Adresse 3 hat, muß die Selektion des Laufwerkes angepaßt werden. Dazu bietet sich eine Gatterschaltung an, die man in das Laufwerk zusätzlich einbauen muß. Die Selektionstabelle beim TRS-80-Modell-1 sieht wie folgt aus (siehe nächste Seite).



Somit genügt zur Selektion des Laufwerkes 3 die Verwendung der Signale DS2, DS3 und DS4 (Bild 4).

#### 6. Stufe: Der neue Bildschirm

Der beim TRS-80 Modell-1 anno 1978 mitgelieferte Bildschirm war nicht gerade sehr schön — die Helligkeit schwankte, und das Bild

stand auch nicht still, sondern wackelte mit einer Frequenz von 10 Hertz. Dieser Bildschirm wurde gegen einen Zenith-Bildschirm ersetzt (Bild 2). Der Bildschirm findet auf dem Expansion-Interface gut Platz, und er ist zum TRS-80 Modell-II-1 voll kompatibel.

Wie auch bei anderen Bildschirmen, muß man möbelschonende Füße aus Weich-PVC zusätzlich kaufen und selbst unterkleben.

Dieser Bildschirm — er wird von verschiedenen Lieferanten angeboten — war das Gerät mit den größten Preisabweichungen: je nach Händler kostete er zwischen 265 Mark und 395 Mark.

#### 7. Stufe: Goldplug als Stecker — teuer aber gut

Die einzelnen Aggregate des TRS-80 Modell-1 sind durch steckbare Flachkabel miteinander ver-

bunden. Die Kartenstecker sind verzinkt und geben nur für die Dauer der Garantiezeit des TRS-80 Modell-1 ausreichend Kontakt. Später beeinträchtigt der Übergangswiderstand der Steckverbindungen die Zuverlässigkeit. Man erkennt es als Benutzer daran, daß der TRS-80 gelegentlich Speicherinhalte verändert und plötzlich mitten im Betrieb einen Kaltstart durchführt.

Die Abhilfe kommt aus den USA: auflötbare, vergoldete Kartenstecker. Sie haben den eingetrag-

nen Markennamen »Goldplug«. Sie funktionieren ausgezeichnet, haben aber zwei Nachteile:

1. Die Kunststoffabdeckteile passen nicht mehr auf die Stecker, denn die aufgelöteten Goldplugs stehen etwas vor (Bild 5).

2. Ein Steckersatz kostet ein kleines Vermögen, und zwar US\$ 54,95.

Leider gibt es keine Importfirma für diese von vielen TRS-80-Modell-1-Besitzern dringend benötigten Stecker. Der Hersteller und Lieferant ist: E.A.P. Co., P.O.Box 14, Keller, Texas 76248.

#### Was hat der Umbau gekostet, was hat er gebracht?

Die Investition für die Umbauarbeiten außer den vorgezogenen Stufen 1 und 2 lagen um 1080 Mark. Dieser Betrag ist im Vergleich zu einer Neuanschaffung und gegengerechnetem Verkauf der Altanlage sehr gering. Ebenso betrug der für die Realisierung benötigte Zeitaufwand nur einen Bruchteil der für eine Systemumstellung benötigten Zeit.

(Rolf-Fr. Matthaei)

## VORWORT ZUM 2. TEIL BASIC F&B

Viele Zuschriften ermunterten mich, weitere Seiten zu veröffentlichen. Es sind also dieses Mal insgesamt 24 Seiten. Einige Male wird auf den Anhang verwiesen, dieser wird im nächsten Info abgedruckt.

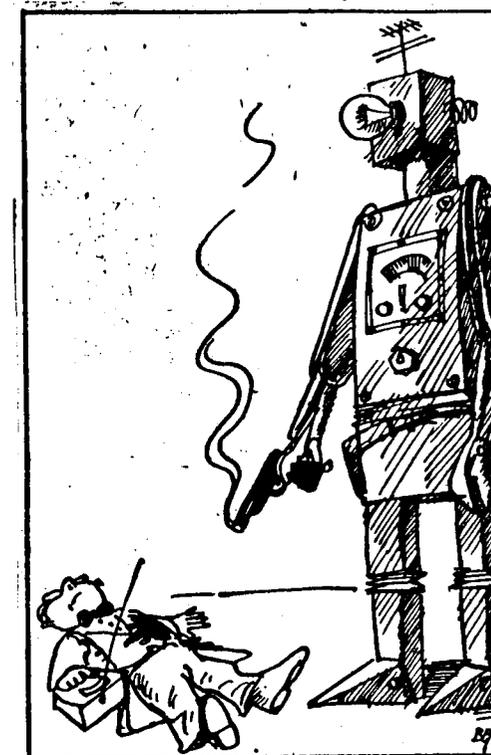
Warum mache ich mir diese Arbeit?

Nun, zum einen sitze ich sehr oft pro Tag ca. 2 Stunden im Zug. Lernen o.ä. kann ich da nicht — erstaunlicherweise aber kann ich dort dieses Buch übersetzen. Das verbinde ich mit dem Englischen, so daß ich da ein klein wenig drin bleibe.

Ein weiterer wichtiger Punkt: Was ich übersetzt habe, habe ich auch (meistens) verstanden. Ich hoffe, daß mich das weiterbringt. Ich glaube auch, daß einigen Mitgliedern durch diese Übersetzung geholfen wird. Nicht alle werden ausreichenden Englisch-Unterricht gehabt haben.

So — einige kleine Fehler bzw. Unstimmigkeiten entdeckte ein Mitglied im Assembler-Teil. Da das Buch sich mit BASIC befaßt, gehe ich an dieser Stelle nicht näher ein (vielleicht zu einem späteren Zeitpunkt). Aber bitte — meldet mir alle Fehler. Insbesondere die Beispiele und Listings sind sehr fehlerträchtig.

Aber nun genug. Ich hoffe, daß der 2. Teil allen etwas bringt!



Falsch programmiert ...

(Zeichnung: Georg Beyer)

DAS LADEN VON DISKETTE UND DAS AUSFÜHREN VON USR-ROUTINEN

Wir wollen annehmen, daß wir die Bildschirm-Füll-Routine als Disk-Datei 'BFUELL' abgespeichert haben. Unser erster Schritt ist das Laden der Routine in den Speicher (RAM). Von 'DOS READY' aus können wir die Routine BFUELL durch LOAD BFUELL laden.

Nun rufen wir BASIC auf. Wir dürfen dabei nicht vergessen, daß wir den Speicher entsprechend unserer Routine begrenzen müssen. Ansonsten kann unser BASIC-Programm die Routine zerstören. Wir schauen uns das Assembler-Listing an, und sehen als Beginn der Routine die Adresse BFF0, das der Dezimalzahl 49136 entspricht. Unsere Antwort auf MEMORY SIZE darf nicht größer als 49136 sein. Im BASIC können wir den Speicher ganz einfach begrenzen, indem wir eingeben: PRINT 65536 + &HBFF0

Wenn wir im BASIC sind, muß unser Programm die Startadresse der USR-Routine festlegen. Der DEFUSR-Befehl im Disk-BASIC erlaubt uns die Definition von maximal 10 Startadressen für 10 USR-Routinen (0 bis 9). Um unsere Routine als USR-Routine 0 zu definieren könnte unser Programm wie folgt aussehen:

```

10 DEFUSR0=&HBFF0
oder,
10 DEFUSR=&HBFF0
oder,
10 DEFUSR0=49136
oder,
10 DEFUSR=49136

```

Falls wir mehr als eine USR-Routine haben, können wir die zweite als DEFUSR1, die dritte als DEFUSR2, etc. definieren. Man sollte sich merken, daß man USR-Routinen in einem Programm so oft undefinieren kann, wie man will. Außerdem bleiben die Adressen von USR-Routinen so lange erhalten, bis sie undefiniert werden oder BASIC neu geladen wird. Man kann andere BASIC-Programme laden und starten so oft man will - die definierten USR-Routinen bleiben erhalten.

Zur Ausführung unserer Bildschirm-Füll-Routine können wir das folgende kleine Programm eingeben:

```

10 DEFUSR0=&HBFF0
20 PRINT@0,"X"
30 J=USR0(0)

```

Nach RUN wird der Bildschirm sofort mit 'X' gefüllt. Man muß nur die Zeile 20 ändern, wenn man den Bildschirm mit anderen Zeichen füllen will. Die Zeile 30 ruft die USR-Routine auf. In diesem Fall ist 'J%' eine Schein-Variable, genauso wie die '0' in den Klammern. In raffinierteren Programmen werden wir die '0' durch eine Integer-Zahl oder einen Ausdruck ersetzen, und teilen so der USR-Routine einen Wert mit, den diese für Berechnungen bzw. die Ausführung benötigt. Wir werden 'J%' oder andere Integer-Variablen benutzen, um von USR-Routinen einen Wert für das BASIC-Programm zu erhalten.

DAS POKEN VON USR-ROUTINEN IN DEN SPEICHER

Jede USR-Routine dieses Kurses wird im POKE-Format vorgestellt. Anders ausgedrückt wird eine Liste von Zahlen angegeben, die man benötigt, wenn man eine USR-Routine in den Speicher POKE'n will. Auf diese Weise benötigt man keinen Editor-Assembler; außerdem muß man die Z-80 Maschinensprache nicht verstehen. Die besprochene Bildschirm-Füll-Routine kann durch das POKE'n der folgenden 12 Zahlen in beliebige 12 benachbarte Adressen des RAM 'geladen' werden:

```
33, 0, 60, 17, 1, 60, 1, 255, 3, 237, 176, 201
```

Probieren Sie diese Schritte um zu sehen, wie es geht:

1. Von DOS READY aus laden Sie BASIC mit einer MEMORY SIZE von 49136.
2. Geben Sie folgendes Programm ein:

```

10 DEFUSR0=&HBFF0
15 DATA 33,0,60,17,1,60,1,255,3,237,176,201
16 FORX=0TO11: READ P: POKE &HBFF0+X,P: NEXT
20 PRINT@0,"X"
30 J=USR0(0)

```

3. Starten Sie das Programm. Der Bildschirm wird sofort mit 1024 'X' gefüllt. Ersetzen Sie nun die Zeile 20 durch:

```
20 PRINT@0,CHR$(191)
```

Starten Sie das Programm wieder, und Sie haben sofort einen weißen Bildschirm.

Die DATA-Ausdrücke in Zeile 15 entsprechen den 12 Bytes aus der USR-Routine. In Zeile 16 werden diese 12 Zahlen in 12 Adressen des geschützten Speichers gepoked, beginnend bei BFF0 (dezimal 49136), so daß die Routine ausgeführt werden kann.

Da die Bildschirm-Füll-Routine veränderlich ist, kann die Adresse &HBFF0 in den Zeilen 10 und 16 durch jede andere Adresse ersetzt werden. Wer einen TRS-80 mit 48K hat, wird z.B. FFF0 anstatt BFF0 verwenden. In Vereinbarung mit MEMORY SIZE können natürlich auch niedrigere Adressen gewählt werden.

Fragen Sie sich, wie man die POKE-Werte erhält? Unser assembliertes Listing zeigt uns die hexadezimalen Werte der USR-Routine. Das Kommando 'LD HL,15360' in Zeile 40 ergab die Maschinensprache-Anweisung 21003C. Diese Anweisung wird in Dezimalzahlen umgewandelt:

```

21 entspricht 33 dezimal.
00 entspricht 0 dezimal.
3C entspricht 60 dezimal.

```

Wir fahren fort mit den Zeilen 50 bis 80 unseres Assembler-Listings und erhalten so der Reihe nach unsere 12 Werte. Noch einfacher erhalten wir diese Werte, wenn wir das assemblierte Programm von Diskette oder Kassette in den Speicher laden. Wir können dann von BASIC aus mittels PEEK die Werte einfach von der ersten bis zur letzten Adresse der Routine ausgeben. Wir verwenden dazu folgende Anweisung:

```
FOR X=&HBFF0 TO &HBFFB: PRINT PEEK(X):: NEXT
```

#### DAS ABSPEICHERN VON USR-ROUTINEN AUF DISKETTE

Jede USR-Routine in diesem Kurs wird im POKE-Format angegeben. Das heißt, daß eine Liste von Zahlen angegeben wird, die mittels POKE in den Speicher gebracht wird, beginnend mit einer Startadresse, die im geschützten Speicher liegt. Wenn die zu einer USR-Routine gehörenden Zahlen einmal in den Speicher gepoked worden sind, kann die Routine unter jedem beliebigen Namen auf der Diskette abgespeichert werden. Nehmen wir an, wir wollen die Bildschirm-Füll-Routine auf Diskette abspeichern:

1. Zuerst gehen wir ins BASIC unter Beachtung von MEMORY SIZE. Sie muß für die USR-Routine ausreichend sein, so daß die USR-Routine auf alle Fälle im geschützten Speicherbereich liegt. In unserem Beispiel legten wir als MEMORY SIZE 49136 fest, so daß wir unsere 12-Byte USR-Routine bei BFF0 beginnen lassen können.

2. Dann schreiben oder laden wir ein Programm, welches uns die benötigten Zahlen in die gewünschten Adressen poked. Hier sind die Programmzeilen für unsere 'BFUELL'-Routine:

```
15 DATA 33,0,60,17,1,60,1,255,3,237,176,201
16 FORX=0T011: READ P: POKE &HBFF0+X,P: NEXT
```

Beachten Sie, daß es sich dabei um die Zeilen 15 und 16 unseres Test-Programms handelt.

3. Als nächstes starten wir das Programm. Es liest die DATA-Ausdrücke und poked die Zahlen in den Speicher.

4. Nun gehen wir in das DOS zurück (CMD"S").

5. Im DOS benutzen wir DUMP. Um die 12 Bytes, die sich noch ab Adresse BFF0 im Speicher befinden, in eine Disk-Datei mit dem Namen 'BFUELL/CIM' zu dumpen, benutzen wir folgende Anweisung:

```
DUMP SFILL (START=X'BFF0',END=X'BFFB')
```

Beachten Sie, daß das DUMP-Kommando an den Datei-Namen automatisch den Zusatz '/CIM' anfügt, wenn nicht eine andere Erweiterung festgelegt wird. Das DOS-Handbuch erklärt diese und andere Details des DUMP-Befehls.

6. Immer wenn Sie in einem Programm die BFUELL-Routine benötigen, geben Sie einfach vor Aufruf des BASIC das Kommando BFUELL ein. Es wird dann die Routine in den Speicher geladen, und zwar genau an die Adresse, an der diese Routine gedumpt worden ist. Wenn Sie ins BASIC gehen, müssen Sie wieder einen Speicherbereich für diese Routine schützen.

Wer will, kann 'BFUELL/CIM' in jeden anderen möglichen Datei-Namen umbenennen. Man verwendet hierzu den RENAME-Befehl. Wenn Sie die Datei zum Beispiel in 'BILDFUEL' umbenennen, so hat diese nicht mehr die Erweiterung '/CIM' und die Anweisung zum Laden vom DOS aus lautet LOAD BILDFUEL.

Wenn Sie ein Model III haben, oder auf Model I NEWDOS benutzen, so können Sie die Routine direkt von BASIC aus laden. Im BASIC unter NEWDOS schreiben wir

```
10 CMD "BFUELL"
oder
10 CMD "LOAD BFUELL"
```

Je nachdem, ob BFUELL den Zusatz '/CIM' hat oder nicht.

Bei einem Model III unter TRSDOS 1.3 lautet die DUMP-Anweisung:

```
DUMP BFUELL (START=BFF0,END=BFFB)
```

Nun können Sie die unter dem Namen BFUELL/CMD abgespeicherte Routine von TRSDOS READY aus einfach durch Eingabe von BFUELL starten. Im BASIC können Sie folgende Anweisung verwenden:

```
10 CMD "L","BFUELL/CMD"
```

SUPER-STRINGS  
=====

#### DAS LADEN VON USR-ROUTINEN IN STRINGS

Wir können jede veränderliche USR-Routine in einen String laden, wenn sie nicht länger als 255 Bytes ist. Diese Technik bietet einige große Vorteile. Zunächst können wir die geforderte Reservierung von Speicherplatz (die 'MEMORY-SIZE'-Frage) vermeiden, wenn wir die USR-Routine in einen String laden. Weiterhin können wir die Routine sehr leicht von einer Speicherstelle zur anderen bewegen, indem wir unseren String mittels VARPTR in einen anderen String poken. Und schließlich können wir die Routine in eine ganz gewöhnliche Disk-Datei abspeichern, welche eine umfassende Bibliothek von Routinen beinhalten kann, und von der man schnell und bequem von BASIC aus die Routinen laden kann.

Unsere Bildschirm-Füll-Routine können wir mit folgender Anweisung in den String S\$ laden:

```
S$=CHR$(33)+CHR$(0)+CHR$(60)+CHR$(17)+CHR$(1)+CHR$(60)+
CHR$(1)+CHR$(255)+CHR$(3)+CHR$(237)+CHR$(176)+CHR$(201)
```

Um nun die Routine auszuführen, können wir die Adresse unserer USR-Routine so definieren, daß sie auf die Daten im String hinweist:

```
DEFUSR0=PEEK(VARPTR(S$)+1)+256*PEEK(VARPTR(S$)+2)
```

Zur Sicherheit sollten wir jedoch vor jedem Aufruf der Routine die Adresse neu definieren, da sich während eines Programmablaufes die Adresse des Strings S\$ mehrmals ändern kann. Es kann sich z.B. die Adresse ändern, wenn neue Variablen zum ersten Mal benutzt werden, oder wenn eine DEF-Anweisung ausgeführt wird.

Hier ist nun eine einfachere Möglichkeit, um längere USR-Routinen in einen String zu laden, besonders wenn die Routine bereits im geschützten Speicher geladen und getestet worden ist:

1. Laden Sie (in einen geschützten Speicherbereich) die Routine von Diskette (Datei, welche mit einem Editor-Assembler erstellt wurde) oder poken Sie die Routine. Wie dies bei der Bildschirm-Füll-Routinen gemacht wird, ist bereits erklärt worden.

2. Definieren Sie mittels DEFUSR den Beginn der USR0-Routine. Bei unserem Beispiel startet die Bildschirm-Füll-Routine bei BFF0:

```
DEFUSR0 = &HBFF0
```

3. Definieren Sie einen String wie folgt: S\$ = ""

4. Poken Sie VARPTR von S\$, so daß die Stringlänge der USR-Routine-Länge entspricht. In unserem Beispiel:

```
POKE VARPTR(S$),12
```

5. Poken Sie den Zeiger der USR-Routine in VARPTR des Strings hinein. Im Anhang 2 steht eine Liste von USR-Routinen-Zeiger-Adressen für die meisten bekannten DOS-Systeme. Für das Model I unter NEWDOS verwenden wir:

```
POKE (VARPTR(S$)+1),PEEK(&H5B14)
POKE (VARPTR(S$)+2),PEEK(&H5B15)
```

Nun enthält der String S\$ die USR-Routine und wir können S\$ in einer Random-Diskdatei abspeichern. So können wir in zukünftigen Programmen die Routinen ganz einfach laden und ausführen, ohne daß wir uns mit Speicherplatz-Reservierung oder DATA-Ausdrücken quälen müssen.

Die Random-Diskdatei, welche wir erstellen, kann so viele USR-Routinen enthalten, wie auf einer Diskette gespeichert werden können. Um die in S\$ abgespeicherte Routine in den Satz 1 einer mit 'USR' benannten Random-Diskdatei abzuspeichern, benutzen wir folgendes Kommando:

```
OPEN R,1,"USR"
FIELD 1, LEN(S$) AS A$
LSET A$ = S$
PUT 1,1
CLOSE
```

Immer wenn wir in einem zukünftigen Programm die Bildschirm-Füll-Routine benötigen, können wir irgendwo im Bereich des Programmmanfanges die folgenden Anweisungen zum Laden der Routine in S\$ programmieren:

```
OPEN R,1,"USR"
FIELD 1,12 AS A$
GET 1,1
S$=A$
CLOSE 1
```

Wenn nötig, können wir die Routine wie folgt aufrufen:

```
POKE &H5B14, PEEK (VARPTR(S$)+1)
POKE &H5B15, PEEK (VARPTR(S$)+2)
J=USR0(0)
```

Die zwei Poke's erfüllen die Funktion des DEFUSR-Befehls; die Adressen werden durch den Befehl VARPTR aus S\$ eingelesen. Die Adressen &H5B14 und &H5B15 im obigen Beispiel müssen ersetzt werden durch die Adressen aus dem Anhang 2, wenn Sie ein anderes DOS verwenden.

Als Alternative können Sie die USR-Routine während der Ausführung im Disk-Buffer belassen. Jeder Disk-Buffer ist ein 256 Byte langer, geschützter Speicher, der durch die Frage 'HOW MANY FILES?' reserviert wird. Die Adressen der Disk-Buffer sind im Anhang 2 wiedergegeben.

Um nun den Buffer 1 zur Ausführung unserer Bildschirmfüll-Routine verwenden zu können, verwenden wir folgende Anweisung zum Laden der Routine:

```
OPEN R,1,"USR"           'Eröffnet die Datei USR
GET 1,1                  'Einlesen des Satzes 1
DEFUSR0 = &H6575         'Adresse des Disk-Buffers defin.
```

Die Routine können wir dann immer wie folgt aufrufen:

```
J = USR(0)
```

Sie werden erkennen, daß diese Technik der 'Superstrings' eine der schnellsten, flexibelsten und wirksamsten Methoden für die Anwendung von USR-Routinen darstellt.

## SUPER-DATENFELDER

=====

DAS LADEN UND AUSFÜHREN VON 'SUPER-DATENFELDERN'

Genauso wie man eine USR-Routine in einen String laden kann, und dann diesen 'String' ausführt, kann man auch eine USR-Routine in ein Integer-Feld laden, und dann dieses 'Super-Datenfeld' ausführen. Bei dieser Technik braucht man keinen Speicherplatz reservieren. Ein Integer-Feld von 15 Elementen reserviert und schützt automatisch 30 Bytes des Speichers. Ein ebenso bedeutender Vorteil dieser Technik ist die bequeme und wirtschaftliche Methode Integer-Werte an USR-Routinen weiterzugeben.

Um zu sehen, wie man mit 'Super-Datenfeldern' arbeitet, geben Sie dieses kurze Programme ein und starten es. Das Programm entspricht wieder unserer Bildschirm-Füll-Routine.

```

5  DEFINT A-Z: J=0
10 US(0)=8448:US(2)=4352:US(4)=256:US(6)=-20243:US(7)=201
20 US(1)=15360:US(3)=15361:US(5)=1023
30 PRINT00,"X"
40 DEFUSR0=VARPTR(US(0))
50 J=USR0(0)
60 GOTO 60

```

Wir laden 7 Integer-Werte in ein Integer-Feld. In Zeile 40 definieren wir die Adresse unserer USR-Routine. In Zeile 50 rufen wir die USR-Routine auf, welche im 'Super-Datenfeld' abgespeichert ist.

Nun betrachten wir uns die Zeile 20. Wir übergeben an die USR-Routine 3 Elemente über die Feldelemente Nr. 1, 3 und 5. Element 1 gibt die Adresse an, welche kopiert werden soll, in diesem Fall 15360 als Adresse der oberen linken Ecke unseres Bildschirms. Das Element 3, das hier um 1 größer ist wie Element 1, gibt an, daß nur 1 Byte zu kopieren ist. Das 5. Element bestimmt, daß das ganze 1023 mal ausgeführt werden soll.

Wir wollen jetzt eine Veränderung versuchen, indem wir andere Parameter verwenden. Das Wort "TEST" soll 63 mal kopiert werden. Es sind lediglich die Zeilen 20 und 30 wie folgt zu ändern:

```

20 US(1)=15360:US(3)=15364:US(5)=63*LEN("TEST")
30 PRINT00,"TEST"

```

Nun starten Sie das Programm und "TEST" wird 63 mal kopiert. Wir änderten also nur die Argumente für unsere USR-Routine. Wie Sie sehen, ist diese Methode der POKE-Methode überlegen.

Seien Sie vorsichtig, wenn Sie mit dieser oder anderen Routinen zu experimentieren anfangen. Nur ein falsches Argument, und der Computer 'taucht unter'. Deshalb befolgen Sie bitte folgende Vorsichtsmaßnahmen, bevor Sie mit Experimenten beginnen:

- Speichern Sie das Programm ab
- Nehmen Sie alle Disketten aus den Laufwerken

Außerdem müssen zuerst die Vorschriften für die Benutzung von 'Super-Datenfeldern' besprochen werden:

1. Das Datenfeld muß ein Integer-Feld sein. In unserem Beispiel benutzen wir einfach die Anweisung 'DEFINT A-Z', um sicherzustellen, daß US% ein Integer-Feld ist.
2. Das Programm darf zwischen dem DEFUSR-Befehl und dem Aufruf der USR-Routine keine neuen Variablen benutzen. Um dieser Regel zu entsprechen, wird die Variable 'J%' in der Zeile 5 unseres Beispiel-Programms vorinitialisiert.

Diese Regel ist notwendig, weil BASIC Integer-Felder immer im Speicher nach oben verschiebt, wenn in einem Programm eine neue Variable benutzt wird. Falls wir die Variable 'J%' in der Zeile 50 zum ersten Mal benutzen, wandert die Adresse unseres Feldes nach oben und unsere DEFUSR-Anweisung in Zeile 40 wird ungültig. Es ist gut, wenn man unmittelbar vor Aufruf einer 'Super-Datenfeld'-Routine die DEFUSR-Anweisung ausführt. Auf diese Weise führt in komplexen Programmen das zufällige Verschieben der USR-Routine durch Verwendung einer neuen Variablen zu keinem Programm-Fehler.

Jede USR-Routine dieses Kurses wird im Format eines 'Super-Datenfeldes' angegeben. Es wird eine Liste der Integer-Zahlen angegeben, die man zum Laden eines Integer-Feldes benötigt, falls man die Methode des 'Super-Datenfeldes' benutzt. Für längere Routinen kann man DATA-Ausdrücke verwenden, um die Integer-Zahlen in das Feld zu bekommen. Die Technik des 'Super-Datenfeldes' ist bestens geeignet für kurze USR-Routinen bis ungefähr 50 Bytes Länge. Sie werden vielleicht bemerkt haben, daß die Technik des Super-Datenfeldes etwas langsam wird, wenn in einem Programm mehrere große Felder während der Programm-Ausführung benutzt werden. Aber für kurze USR-Routinen ist die Technik des Super-Datenfeldes in der Tat 'Super'!

DAS SCHREIBEN VON 'SUPER-DATENFELD'-USR-ROUTINEN

Wie Sie gesehen haben, liefern die Super-Datenfelder einen einfachen Weg zum Laden von Argumenten aus BASIC in eine Maschinensprache-USR-Routine. Wer die Z-80 Assemblersprache kennt, kann hier nachlesen, wie man USR-Routinen als Super-Datenfeld schreibt:

- Schreiben Sie eine Z-80 Routine und assemblieren Sie diese mit einem Editor-Assembler. Es muß eine veränderliche Routine sein.
- Betrachten Sie das assemblierte Listing um festzustellen, wo Sie Argumente benötigen. Wenn nötig, fügen Sie entweder 'NOP'-Befehle ein, oder reorganisieren die Routine so, daß die Argumente bei jedem geraden Byte innerhalb der Routine starten. Falls die Länge der Routine nicht gleichmäßig durch 2 teilbar ist, so fügen Sie ein NOP als letzte Anweisung an, damit die Routine durch 2 teilbar wird. Nun assemblieren Sie die Routine nochmals und kontrollieren anschließend, ob die Routine nun richtig ausgerichtet ist.

Nun betrachten wir das Assembler-Listing, welches zur Erstellung des 'Super-Datenfeldes' für unsere Bildschirm-Füll-Routine benötigt wurde. Von jetzt an wollen wir diese Routine als 'DATA-Treiber'-Routine bezeichnen, da diese Routine bei vielen Anwendungen nützlich ist, wo wir DATA-Blöcke von einer Speicherstelle zu einer anderen verschieben wollen.

BFF0	00100	ORG	0BFF0H	!BEGINN-VERÄND.
BFF0	00	NOP		!NO-OP F. AUSR.
BFF1	210000	LD	HL,0	!LADE "VON"-ADR.
BFF4	00	NOP		!NO-OP F. AUSR.
BFF5	110000	LD	DE,0	!LADE "NACH"-ADR
BFF8	00	NOP		!NO-OP F. AUSR.
BFF9	010000	LD	BC,0	!LADE # V. BYTES
BFFC	EDB0	LDIR		!BC BYTES HL-DE
BFFE	C9	RET		!RUECKSPRUNG
BFFF	00	NOP		!F. GERADE LAENG
0000	00200	END		;
00000	TOTAL	ERRORS		

Im Assembler-Listing der DATA-Treiber-Routine sehen wir, daß in den Zeilen 120, 140 und 160 jeweils Integer-Nullen von 2 Byte Länge in die Register HL, DE und BC geladen werden. Im BASIC werden diese Nullen durch den Inhalt der Elemente 1, 3 und 5 ersetzt. So wie wir Parameter in die erforderlichen Feldelemente innerhalb eines BASIC-Programms laden, laden wir auch unsere Argumente in die Routine.

In den Zeilen 110, 130 und 150 fügten wir NOP's ein, damit sich die Parameter auf gerade Bytes ausrichten. Die Z-80 NOP-Anweisung ist einfach eine 8-Bit Null und bedeutet 'keine Anweisung' (no operation). Der Computer beachtet diese NOP's einfach nicht und fährt mit der nächsten Anweisung fort.

In der Zeile 170 steht die gewaltige Z-80 LDIR-Anweisung. Sie bewegt ein Byte mit der Adresse aus dem BC-Register in die Adresse aus dem DE-Register. Dann werden das BC- und DE-Register erhöht, das BC-Register wird erniedrigt. Das Ganze wiederholt sich, bis BC=0 geworden ist.

In der Zeile 190 steht nochmals ein NOP. Dadurch wird die Länge der Routine gerade. Es ist wichtig, daß Super-Datenfeld-Routinen von gerader Länge sind.

Wenn Sie die Routinen assembliert haben, laden Sie diese in den Speicher und gehen ins BASIC, wobei Sie die geschützte Speichergröße so wählen, daß Ihre Routine nicht überschrieben wird.

Um nun die Integer-Werte zu erhalten, die wir für unser Super-Datenfeld benötigen, schreiben wir folgendes Programm:

```

10 SX = &HBFF0           'Startadresse
20 EX = &HBFFF           'Endadresse
30 FOR X = SX TO EX STEP 2
40 PRINT CVI(CHR$(PEEK(X))+CHR$(PEEK(X+1)));
50 NEXT

```

Natürlich stehen in den Zeilen 10 und 20 die Start- und Endadressen Ihres Programmes. Gewöhnlich wird man in Zeile 40 anstatt PRINT ein LPRINT schreiben, um eine Ausgabe auf Drucker zu erreichen.

#### DAS ABSPEICHERN VON 'SUPER-DATENFELDERN' IN RANDOM-DISK-DATEIEN

Die Technik des Super-Datenfeldes hat einige feine Vorteile beim Laden von USR-Routinen in den Speicher. So benötigt eine Routine im Super-Datenfeld-Format nur die Hälfte der Argumente, welche beim POKE-Format benötigt werden.

Wenn man einmal die benötigten Werte einer Routine in ein Datenfeld gebracht hat, möchte man gewöhnlich die im Datenfeld befindliche USR-Routine in einer Random-Disk-Datei abspeichern. Dadurch verschenkt man in zukünftigen Programmen, in denen man diese Routine einsetzen möchte, keinen Speicher, welcher von DATA-Ausdrücken belegt werden würde. Wie speichert man nun 'Super-Datenfeld'-Routinen mit einer maximalen Länge von 127 Elementen in einer Random-Disk-Datei ab:

- Eröffnen Sie Ihre Random-Disk-Datei.
- Erstellen Sie FIELD mit 255 Bytes für A\$.
- Definieren Sie eine Schein-String-Variable S\$ durch S\$="".
- Poken Sie in VARPTR von S\$ die Länge Ihrer im Datenfeld abgespeicherten Routine. Die Länge entspricht der doppelten Element-Anzahl, da jedes Element 2 Bytes beinhaltet.
- Poken Sie in VARPTR von S\$+1 die Adresse von LSB (Least Significant Byte = erstes bedeutendes Byte) Ihres Datenfeldes. Wenn Ihr Datenfeld mit US\$(0) beginnt, so schreiben Sie:

```
POKE VARPTR(S$)+1, ASC(MKI$(VARPTR(US$(0))))
```

6. Poken Sie in VARPTR von S#+2 die Adresse von MSB (Most Significant Byte = höchstes bedeutendes Byte) Ihres Datenfeldes. Wir schreiben:

```
POKE VARPTR(S#)+2, ASC(RIGHT$(MKI$(VARPTR(US$(0))),1))
```

Nun enthält S# die USR-Routine. Führen Sie LSET A#=S# aus und bringen Sie A# mittels PUT in den gewünschten physikalischen Satz.

Immer dann, wenn Sie diese Routine in einem Programm benutzen wollen, eröffnen Sie die Disk-Datei und lesen mit GET den gewünschten Satz in den Speicher ein. Sie können dann Ihre Routine über den Disk-Buffer ausführen, oder auch die Routine in einen anderen geschützten Speicherbereich bringen. Sie können die Routine aber auch wieder in ein Datenfeld bringen.

Ein Beispiel. Nehmen wir an, Sie haben 58 Werte über DATA-Ausdrücke in ein Super-Datenfeld US# gebracht. Ihre USR-Routine beginnt mit US\$(0). Um nun diese Routine in den physikalischen Satz 2 einer mit "ROUTINEN" bezeichneten Datei abzuspeichern, verwenden Sie folgende Anweisungen:

```
OPEN "R",1,"ROUTINEN": FIELD1, 255 AS A#
S#="": POKE VARPTR(S#),116
POKE VARPTR(S#)+1, ASC(MKI$(VARPTR(US$(0))))
POKE VARPTR(S#)+2, ASC(RIGHT$(MKI$(VARPTR(US$(0))),1))
LSET A#=S#: PUT 1,2: CLOSE
```

Wenn Sie nun diese Routine zu einem späteren Zeitpunkt ohne Verwendung von DATA-Ausdrücken in ein Super-Datenfeld zurückbringen wollen, können Sie folgende Anweisungen benutzen:

```
DIM U$(58)
OPEN "R",1,"ROUTINEN": FIELD1, 116 AS A#
GET 1,2
S#="": POKE VARPTR(S#),116
POKE VARPTR(S#)+1, ASC(MKI$(VARPTR(US$(0))))
POKE VARPTR(S#)+2, ASC(RIGHT$(MKI$(VARPTR(US$(0))),1))
LSET S#=A#
```

Wenn Sie die Routine nicht in Feldelementen benötigen, können Sie natürlich auch die Techniken zum Laden und Ausführen von Super-Strings verwenden.

#### VERWANDLUNG VON USR-ROUTINEN DURCH KONTROLLFELDER

Dies ist eine weitere gewaltige Technik, die Sie in Ihrem DOS-Handbuch vergeblich suchen werden. Wir erstellen einfach ein Integer-Feld, welches die Argumente beinhaltet, die wir an eine USR-Routine weitergeben wollen. Dieses 'Kontrollfeld' kann ebenso Integer-Werte enthalten, welche von der USR-Routine erstellt wurden und an das BASIC übergeben worden sind.

Zum Beispiel benötigt die USR-Routine 'SORT1', welche String-Felder in aufsteigender Reihenfolge sortiert, 2 Werte. Das aufrufende BASIC-Programm muß das zu sortierende String-Feld und die Anzahl der zu sortierenden Elemente angeben. Diese 2 Argumente sind in einem Integer-Feld enthalten. Das Element 0 beinhaltet VARPTR des String-Feldes und Element 1 beinhaltet die höchste zu sortierende Element-Nummer des String-Feldes.

Um die ersten 600 Elemente des Feldes S# zu sortieren, benutzen wir folgende Anweisungen, um die USR-Routine mit dem Feld CX als Kontrollfeld zu benutzen:

```
100 CX(0)=VARPTR(S$(0))
101 CX(1)=599
102 J=USR0(VARPTR(CX(0)))
```

Im Programm müssen wir schon vorher mit dem DEFUSR-Kommando die Adresse unserer SORT1 USR-Routine geladen haben. Ebenso muß bereits die Schein-Integer-Variable J# definiert worden sein, damit Sie beim USR-Aufruf richtig arbeitet. Die Methode des Kontrollfeldes zur Übergabe von Werten kann für jede beliebige USR-Routine angewandt werden, egal ob diese im geschützten Speicher, im Super-String oder im Super-Datenfeld steht.

Kontrollfelder sind besonders nützlich, wenn viele Argumente zwischen der USR-Routine und BASIC ausgetauscht werden müssen. Bei allen USR-Routinen, die Kontrollfelder benutzen, wird angegeben, welche Elemente bzw. Angaben benötigt werden.

Es gibt einiges bei der Benutzung von Kontrollfeldern zu beachten:

1. Ein Kontrollfeld muß ein Integer-Feld sein. Deshalb sollten Sie entweder immer das Prozent-Zeichen '%' benutzen, oder die benutzten Variablen mit DEFINT definieren.
2. Erinnern Sie sich daran, daß sich die Feld-Adressen ändern, wenn Sie während der Ausführung eines BASIC-Programms eine neue Variable definieren. Falls eines der Elemente des Kontroll-Feldes das VARPTR eines anderen Feldes beinhaltet, müssen Sie sicherstellen, daß während des Ladens des Kontroll-Feldes und dem Aufruf der USR-Routine keine neuen Variablen benutzt werden.
3. Sie müssen nicht mit dem Element 0 des Kontrollfeldes starten. Sie können bei anderen Anwendungen andere Elemente benutzen. Zum Beispiel könnten wir die SORT1-Routine wie folgt aufrufen:

```
100 CX(14)=VARPTR(S$(0))
101 CX(15)=599
102 J=USR0(VARPTR(CX(14)))
```

Wenn Sie in eigenen USR-Routinen Kontrollfelder benutzen wollen, so werfen Sie einen Blick auf irgendeine USR-Routine dieses Kurses, welche diese Technik anwendet.

Sie werden feststellen, daß die ersten drei Z-80 Anweisungen der Routine immer wie folgt lauten:

```
CALL 0A7FH
PUSH HL
POP  IX
```

'CALL 0A7FH' lädt den Klammerwert der USR()-Funktion des BASIC-Programmes in das HL-Register. Die Hilfsroutine 0A7F vom ROM führt dies für uns aus. Weil der vom BASIC übergebene Wert der VARPTR-Wert eines Kontrollfeldes ist, bezieht sich HL auf das erste Element des Feldes.

Die PUSH- und POP-Anweisungen kopieren den Inhalt des HL-Registers in das IX-Register. Dann könnten wir z.B. den Inhalt des zweiten Elementes unseres Kontrollfeldes in das DE-Register laden:

```
LD  E,(IX+4)
LD  D,(IX+5)
```

Durch das entgegengesetzte Verfahren können wir Daten an das Kontrollfeld übergeben. Wenn wir den Inhalt von BC in das 3. Feldelement übergeben wollen, so können wir schreiben:

```
LD  (IX+6),C
LD  (IX+7),B
```

Falls wir nur einen Wert an das BASIC-Programm übergeben wollen, so lautet die letzte Anweisung der USR-Routine:

```
JP  0A9AH
```

Dies bewirkt einen Sprung zu einer ROM-Routine, welche den Inhalt von HL dem BASIC übergibt. Falls wir diesen Sprung benutzen um ins BASIC zurückzukehren, und wenn unser Aufruf

```
J=USR0(VARPTR(CZ(0)))
```

gelaufen hat, so erhält die Variable 'J' den letzten Wert von HL, welcher durch die USR0-Routine berechnet wurde. Wenn wir nur eine 'RET'-Anweisung benutzen, um ins BASIC zurückzukehren, so bleibt der Inhalt der Variablen J% von dem USR-Aufruf unberührt.

VERÄNDERLICHER VIELWERT-VERWALTER FÜR USR-AUFRUFE

Für das Programmieren in Assembler wird nun eine Routine angegeben, welche man bei USR-Routinen als eine alternative Methode zum Verwalten von verschiedenen Werten benutzen kann. Der VW-Verwalter (Viel-Wert-Verwalter) ermöglicht es im BASIC, alle Werte, die der USR-Routine in einfacher Genauigkeit übergeben werden sollen, festzulegen.

Zum Beispiel benötigt unsere 'DATA-Treiber'-Routine 3 Werte:

1. von Adresse
2. nach Adresse
3. Anzahl der zu übergebenden Werte

Mit dem VW-Verwalter können wir z.B. 50 Bytes von Adresse 15360 nach 15384 übertragen. Unser USR-Aufruf lautet:

```
J=USR(15360) OR USR(15384) OR USR(50)
```

Der Verwalter unterstützt einen Zähler für die bereits übertragenen Werte. Wenn alle Werte (in diesem Fall 3) anerkannt worden sind, so wird die Kontrolle für die Ausführung der Werte an die USR-Routine übergeben:

```

00000 ;VW-VERWALTER
00001 ;
FF00 00100      ORG 0FF00H      ;BEGINN
FF00 CD7F0A    00110      CALL 0A7FH      ;WERT IN HL STELLEN
FF03 DD2A145B 00120      LD  IX,(05B14H) ;IX = DEFUSR-ADR.
FF07 DD7535    00130      LD  (IX+53),L   ;
FF0A DD7436    00140      LD  (IX+54),H   ;WERT SPEICHERN
FF0D DD3409    00150      INC  (IX+9)     ;
FF10 DD3409    00160      INC  (IX+9)     ;1. ZÄHLER + 2
FF13 DD340C    00170      INC  (IX+12)    ;
FF16 DD340C    00180      INC  (IX+12)    ;2. ZÄHLER + 2
FF19 DD7E09    00190      LD  A,(IX+9)   ;
FF1C 0635      00200      LD  B,53       ;
FF1E 90        00210      SUB  B         ;A=ÜBERTRAGENE WERTE * 2
FF1F DD4634    00220      LD  B,(IX+52)  ;B=RESTLICHE WERTE * 2
FF22 90        00230      SUB  B         ;
FF23 2806      00240      JR   Z,PASS1   ;WENN 0, ALLE WERTE ÜBERT.
FF25 210000    00250      LD  HL,0000H   ;CLEAR FÜR RETURN
FF28 C39A0A    00260      JP  0A9AH      ;RETURN FÜR NÄCHSTEN WERT
FF2B DD360935  00270      LD  (IX+9),53  ;
FF2F DD360C36  00280      LD  (IX+12),54 ;ZÄHLER ZURÜCKSETZEN
FF33 1806      00290      JR   START    ;
FF35 0000      00300      DEFW 0         ;WERT 1 SPEICHERN
FF37 0000      00310      DEFW 0         ;WERT 2 SPEICHERN
FF39 0000      00320      DEFW 0         ;WERT 3 SPEICHERN
FF3B 00        00330      START NOP         ;ROUTINE STARTET HIER
402D          00340      END  402DH     ;
00000 TOTAL ERRORS

```

Wie schreibt man eine Z-80 Hilfsroutine mit dem VW-Verwalter:

1. In Bezug auf die verwendete Nummer der USR-Routine und auf das verwendete DOS suchen Sie sich im Anhang 2 die Zeiger-Adresse der USR-Routine. Ändern Sie Zeile 120 entsprechend. Unser Beispiel zeigt 5B14 als Adresse des Zeigers für USR0 unter NEWDOS 2.1.

2. Zwischen den Zeilen 290 und 330 müssen genau so viele DEFW-Anweisungen stehen, wie Sie Werte von BASIC an die USR-Routine übergeben wollen. Wichtig ist, daß nichts anderes zwischen 'JR START' (Zeile 240) und 'START' (Zeile 330) steht, da der Verwalter die Differenz zwischen diesen beiden Anweisungen benutzt, um festzustellen, wieviele Werte übergeben werden müssen, bevor die Hauptroutine aufgerufen werden kann.
3. Die Logik Ihrer Routine muß vor 'JR START' und nach 'START' stehen. Zugang zu den übertragenen Werten erreichen Sie über das IX-Register:

```
(IX+53) und (IX+54) beinhalten den ersten Wert
(IX+55) und (IX+56) beinhalten den zweiten Wert
(IX+57) und (IX+58) beinhalten den dritten Wert, etc.
```

Um nun z.B. den zweiten Wert in das DE-Register zu laden, müssen Sie schreiben:

```
LD E,(IX+55)
LD D,(IX+56)
```

Der VW-Verwalter ist wahrscheinlich die bequemste Art, um USR-Routinen von BASIC aus aufzurufen. Behalten Sie für dessen Benutzung bitte folgendes im Gedächtnis:

1. Bei jedem Aufruf können maximal 25 Werte übertragen werden.
2. Sie müssen entscheiden, welche USR-Routine benutzt werden soll, weil der Zeiger im Verwalter assembliert ist. (Für mehr Flexibilität können Sie auch in das 6. und 7. Byte poken).
3. Der Verwalter fügt etwa 50 Bytes an Ihre Routine an.
4. Die Logik ändert sich während der Ausführungszeit selbst. Alle Variablen müssen ordnungsgemäß übertragen werden, oder der Verwalter wird sich nicht mehr in seinen Ursprungs-Zustand zurückinitialisieren.
5. Sie können Speicher sparen, falls Ihre USR-Routine nicht veränderlich sein muß. Der Hauptvorteil des VW-Verwalters ist seine Veränderlichkeit, aber auch, daß das Arbeiten mit Werten (Speicherstellen) in der Routine integriert ist.

SUPER - SPEICHERTECHNIKEN

---

'Jedes Programm will sich ausweiten, um den gesamten vorhandenen Speicherplatz zu belegen'

---

Falls Sie schon längere Zeit auf Ihrem Computer programmiert haben, werden Sie diese Aussage bestätigen können. Egal, wieviel Speicher- oder Diskettenkapazität man hat - man findet immer irgendeinen Weg, diesen zu belegen. Dieses Kapitel erklärt nun Techniken, die helfen sollen, das Beste aus den vorhandenen Speicher herauszuholen.

Bestimmt kennen Sie Vorführungen, wo Experten das Publikum mit schneller Speicherung von Namen oder Buchseiten unterhalten. Diese schönen Vorführungen sind im Prinzip nur auf einfache Techniken aufgebaut, die jeder erlernen kann. Dieses Kapitel will Ihnen einige einfache Techniken nahe bringen, mit deren Hilfe Sie von Fall zu Fall Ihrem Computer erstaunliche Speicherfähigkeiten geben können. Sie werden sehen, daß Ihre Programme ganz neue Leistungen schaffen, wenn Sie erstmal wissen, wie man den Speicher kontrolliert, wie man schnell Daten verschiebt und wie man Programm-Teile von Diskette lädt und wieder abspeichert.

WIEVIEL SPEICHER HABEN SIE WIRKLICH ?

Falls Sie einen 48K TRS-80-Computer gekauft haben, haben Sie in Wirklichkeit einen 64K Speicher. Wenn Sie einen 32K TRS-80 haben, sind es 48K. Da ein Teil des Speichers vom ROM belegt wird, kann man diesen als Programmierer nicht verändern, und man wird sich überlegen, wo die Grenze des verwendbaren Speichers liegt:

---

Radio Shack Katalog	Höchste Adr. hexadezimal	höchste Adr. dezimal	höchstes Byte Integer-Format
'16K'	7FFF	32767	32767
'32K'	BFFF	49151	-16385
'48K'	FFFF	65535	-1

---