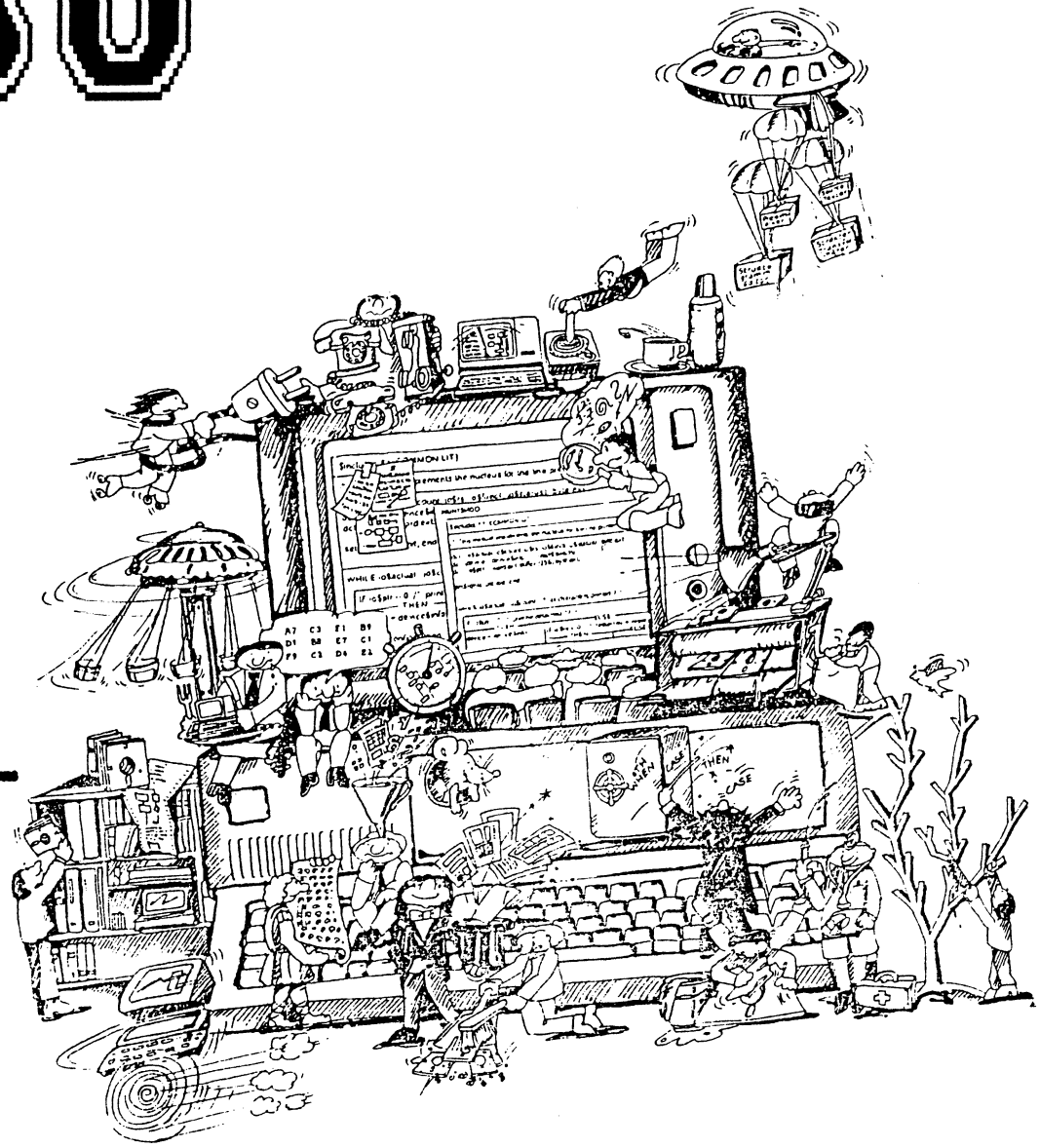
CLUB 80

Clubinfo der TANDY - GENIE - und KOMTEK - ANWENDER



23. AUSGABE

KONTAKTADRESSE : CLUB 80 / HARTMUT OBERMANN / SCHWALBACHER STRABE 6 / 6209 HEIDENROD 1
POSTFACH 27 / TEL.: 06124 / 3913

- I N H A L T S V E R Z E I C H N I S -

Neues vom Vorstand

Das erste Mitglied

Falsch getippt, nicht das erste Neumitglied im Jahr 1988, sondern das erste Mitglied des CLUB 80 überhaupt, welches keinen TRS 80 oder auch nur einen kompatiblen sein eigen nennt! Matthias Homann heißt der Knabe, wohnhaft in Seevetal (bei Gerald Schröder gleich um die Ecke) und stolzer Besitzer eines modifizierten NDR-Klein-Computers mit 64180-CPU, GDP 644-HRG und ... und ... und ... Aber das soll ja weder eine komplette Vorstellung von Matthias, noch eine Werbeaktion für NDR-Klein-Computer werden. Vielmehr wollte ich euer Augenmerk auf den Umstand richten, daß man nicht unbedingt Besitzer eines TRS 80 oder kompatiblen sein muß, um Mitglied im CLUB 80 zu werden. Ich hoffe für Matthias (und auch für eventuell noch folgende Nicht-TRS 80-Besitzer) daß unser Info und die sonstigen Leistungen des CLUB 80 (Bücherei, Programmsammlung usw.) auch für ihn so interessant sind, daß er seinen Entschluß, bei uns einzusteigen nicht bereut und noch lange Zeit zu uns gehört!

Stringy-Floppy

Ebenfalls ein Unikum in unseren Reihen stellt Werner Hentz dar. Er ist zwar Besitzer eines TRS 80 Model 1, arbeitet aber, meinen Informationen nach als einziger im CLUB 80, nur mit Stringy-Floppys. Wer noch solche Dinger irgendwo rumliegen hat oder sogar damit arbeitet, kann sich ja mal mit Werner in Verbindung setzen!

MS-DOS

Die vor einiger Zeit recht heftige Diskussion über eine MS-DOS-Ecke im Info ist zwar etwas abgeflaut, zu einem Ende ist sie aber immer noch nicht gekommen. Nachdem ich jetzt selbst auch einen PC-kompatiblen mein Eigen nenne, würde ich gerne wissen, welche Mitglieder des Clubs ebenfalls zweigleisig fahren. Wer also neben dem TRS 80 auch noch einen anderen Computer (ob PC, Amiga, Atari oder sonst etwas) beackert, möge sich bei mir melden. Ein Anruf oder eine Postkarte genügen, wer mich gleich mit einem längeren Brief beehren möchte, kann das auch tun. Eine Antwort bekommt ihr auf alle Fälle!

Clubtreffen 88

Nach mehreren erfolglosen Versuchen habe ich (stimmt nicht, Jutta hat) endlich ein geeignetes Hotel für unser jährliches Club-Treffen gefunden. Es handelt sich um das Hotel Idsteiner Hof in Idstein. Ursprünglich wollten wir ja in den Raum Darmstadt/Bergstraße, hatten aber bei unserer Suche dort keinen Erfolg. Auf Idstein fiel die Wahl, weil es direkt an einer Nord-Süd-Autobahnverbindung (A 3/E 5), etwa 30 km nördlich von Wiesbaden und ca. 50 km nordwestlich von Frankfurt liegt. Damit ist wieder gewährleistet, daß die Anfahrstrecke weder für die Nordlichter (ich halte mich zu dieser Zeit übrigens in Rendsburg/Schleswig auf) noch für die Südländer zu lang ist.

Idstein ist ein sehr schönes kleines Städtchen, dessen Altstadtkern gerade neu renoviert wurde. Auch die nähere Umgebung (Wiesbaden, Limburg, Diez, Feldberg usw.) bieten einiges, so daß es auch für eventuell mit anreisende "bessere Hälften" nicht langweilig werden dürfte. Das Hotel liegt direkt an der Straße, die von der Autobahn in die Stadt führt und ist damit leicht zu finden. Die Zimmer sind zwar teilweise recht spartanisch, jedoch sehr ordendlich und sauber eingerichtet. Die Preise für Doppelzimmer liegen zwischen 20,-- und 30,-- DM pro Person incl. Frühstück. Die Tagungsräume, die sehr gemütlich ausgestattet und recht groß sind, werden uns wieder kostenfrei zur Verfügung gestellt.

Wo viel Licht ist, findet sich leider immer wieder auch der eine oder andere Schatten. Beim Idsteiner Hof ist es die leider recht beschränkte Unterbringungskapazität von 18 Betten, so daß eventuell einige Mitglieder andersweitig untergebracht werden müssen. Aber auch das ist kein Beinbruch, da die sehr hilfsbereite und freundliche Wirtin versprochen hat, bei der Suche nach Quartieren in Hotels in der Nähe tatkräftige Unterstützung zu leisten.

Da ich nicht weiß, wie lange es bei diesem Info vom Redaktionstermin bis zur Auslieferung dauert, werde ich die Einladung gesondert verschicken. Eigentlich müßte sie euch schon erreicht haben!? Wenn dies nicht der Fall ist (grande Malheur) und ihr am 12. bis 15. Mai am Clubtreffen teilnehmen wollt, ruft mich schleunigst an, damit ich euch noch eine Einladung zukommen lassen kann!

Eine Tagesordnung für das Clubtreffen besteht, wie jedes Jahr, noch nicht. Trotzdem bin ich sicher, daß auch dieses Treffen wieder ein voller Erfolg wird. Der Rahmen wird ungefähr wie folgt aussehen:

Do. 12.05 - Anreise der Computersüchtigen, die das seltene Ereignis voll auskosten wollen. Zeit zum Meinungs-, Erfahrungs- und Programmaustauch

Fr. 13.05 - Anreise für die nicht Abergläubischen, Gelegenheitshacker und diejenigen, die es nicht geschafft haben, für diesen Tag Urlaub (beim Chef oder der Frau) zu bekommen. Nach dem Mittagessen eventuell erste Vorträge (wenn sich jemand findet, der etwas wissenswertes und interessantes vorzutragen hat) und noch mehr Zeit zum Hacken

Sa. 14.05 - Vor- und Nachmittags weitere Vorträge und viel Zeit zum Meinungs- und Erfahrungsaustausch. Nach dem Fünfuhrtee soll es dann etwas offizieller werden, sprich die Jahreshauptversammlung steigt.

So. 15.05 - Da auch das schönste Clubtreffen mal ein Ende haben muß, wird dies wohl der Tag des Abschiednehmens werden. Die einen werden früher, die anderen später das Weite suchen und wie in jedem Jahr bleibt es jedem selbst überlassen, wann er den Tagungort verläßt.

Damit dieser Rahmen sich mit Leben füllt, sind natürlich alle Teilnehmer aufgerufen, sich möglichst aktiv am Treffen zu beteiligen. Gefragt sind vor allem Vorträge über die verschiedensten Aspekte der Hard- und Software (z.B. wird sich Rüdiger Sörensen über Turbo-Pascal auslassen). Aber auch der Gedankenaustausch über die Zukunft des CLUB 80 sollte nicht zu kurz kommen. Themen wie "MS-DOS-Ecke ja oder nein", "Verbesserung des Infos durch ansprechendere Gestaltung" oder "öffnung des CLUB 80 für nicht TRS 80-kompatible Computer" stehen bei der Jahreshauptversammlung mit Sicherheit auf der Tagesordnung. Auf alle Fälle gilt das Motto: je vielfälltiger das Angebot, desto Interessanter das Treffen!

Damit will ich für diesmal Schluß machen. Ich freue mich jetzt schon, euch möglichst vollzählig in Idstein begrüßen zu dürfen und wünsche euch in diesem Sinne alles Gute und viel Spaß am Computerhobby, euer

Hartmut Obermann

03

Hotel-Restaurant-Bierstube
IDSTEINER HOF
Inh. Maria Keil
Wiesbadener Straße 43
6270 IDSTEIN/TAUNUS
Telefon 06126/3227

Wir empfehlen uns für Tagungen, Geschäfts- und Familienfeiern

Der Idsteiner Hof ist sehr leicht zu finden:

- Autobahn A3 Abfahrt Idstein abfahren
- Richtung Innenstadt fahren
- direkt an der Straße, die in die Stadt führt, kommt links eine Texaco-Tankstelle
- direkt nach der Tankstelle liegt der Idsteiner Hof
- Parkplätze gibt es auf der Rückseite des Gebäudes!

Gute Fahrt und sicheres Ankommen wünscht

Hartmut Obermann

G e b u r t s t a g s k i n d e r

Hallo Freunde,

heute ist es wieder soweit. Es weilen wieder viele Geburtstags"kinder" unter uns.
Im letzten Info kamen zwar einige Geburtstagswünsche reichlich spät, aber ihr könnt versichert sein, sie waren genauso herzlich, wie es diese sind.

Da sich bis jetzt noch kein im letzten Infobeitrag vergessenes Mitglied bei mir gemeldet hat, nehme ich an, daß sich niemand übergangen fühlte. Trotzdem möchte ich all diejenigen, von denen ich kein Geburtstagsdatum habe, bitten, mir dieses mitzuteilen.

So, nun endlich geht es los.

Im Monat **Februar** hatten Geburtstag:
am 2. : Spieß Peter
am 4. : Emmrich Helmut (der unseren Club leider vor kurzem verlassen hat)
am 9. : Schmitz Paul-Jürgen und Rensch Richard
am 17. : Schröder Gerald
am 26. : Wucherer Jürgen

Im Monat **März** gibt es nur einen (reichlich schwacher Monat):
am 30. : Sörensen Rüdiger

Im Monat **April** sind es wieder einige mehr:
am 7. : Trapp Harald und Schmid Alexander
am 26. : Loose Gerhard
am 28. : Piller Walter
am 30. : Böckling Ulrich

Weiterhin gratulieren wir zwei frischgebackenen Vätern und den dazugehörigen Müttern und wünschen beiden wenig durchwachte Nächte (wegen des Babys). Klaus Hermann (Monat Dezember '87) und Manfred Held (Januar '88) haben trotz der vielen Computerei noch Zeit für andere Dinge gefunden. Sie wurden beide Väter einer Tochter. Juhuu!!!

So, das war's mal wieder. Wie ihr seht, gab und gibt es wieder viel zu feiern.

Im Namen des Vorstands wünsche ich den Geburtstags"kindern" und allen anderen, die etwas zu feiern haben, viel Glück und hoffe, daß wir uns zum Clubtreffen in Idstein wiedersehen (12.05.-15.05.).

Eure Clubmaus

Jutta

Hallo Clubfreunde,

als neues Mitglied im Club möchte ich mich kurz vorstellen.

Ich heiße Werner Hentz, geb. 3.11.42, wohnhaft in Maintal - einer Stadt zwischen Frankfurt und Hanau. Beruflich bin ich in Sachen EDV schon seit gut 20 Jahren tätig und hoffe von Euch noch einiges dazulernen zu können. Es war reiner Zufall, daß über eine CHIP-Anzeige mich ein Frankfurter Arzt auf den CLUB 80 aufmerksam gemacht hatte.

Meine TRS-80/Modell I Anlage besitze ich seit Sommer 1979, die nach und nach in Stufen erweitert bzw. ausgebaut wurde. Als absolute Besonderheit werden meine Programme über STRINGY FLOPPY - einer Alternative zwischen Kassette und Diskette - geladen und gespeichert. Das Modell I arbeitet mit 48 kRAM, erhöhter Taktfrequenz und Kleinschreibung (incl. Zeichengenerator mit Unterlängen). Seit Herbst letzten Jahres kam der Thermo-Transferdrucker TPX-80 von C. ITOH dazu, der über das TANDY-Bufferkabel 26-1411 am Bus angeschlossen ist.

Von meinem Clubbeitritt im November letzten Jahres erhoffe ich mir in erster Linie gute Kontakte zu den übriggebliebenen Z80-Freaks.

Auf gute Zusammenarbeit
Werner Hentz

Mit freundlichen Grüßen Werner Hentz

Liebe Clubfreunde,

da ich nicht sehr stark in deutscher Sprache bin, möchte ich mich hier nur kurz vorstellen.

Eigentlich bin ich ein Franzose aus Bordeaux (Gironde) und 1935 geboren.

Vor ungefähr 6 Jahren habe ich mit einem Model 1 mit Kassetten begonnen. Dann habe ich mich an den TRS 80 Club von Yverdon angeschlossen. Um mich weiterzubilden, habe ich viele Bücher gekauft, um Software in Basic besser zu verstehen. Dafür habe ich in Hardware keine Erfahrung.

Um CP/M kennenzulernen, habe ich mir einen Commodore 128 und ein Genie III gekauft, sodaß ich die zahlreichen Unterlagen über Z80-Assembler und Pascal anwenden kann.

Soweit meine Vorstellung.

Auf Gute Zusammenarbeit mit euch freue ich mich

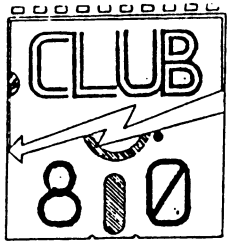
Michel

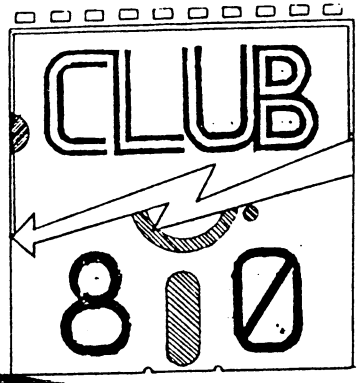
07

Thema „Club-Emblem".
Zum Emblem selbst ist nicht viel zu sagen. Es soll ganz einfach 3 Punkte darstellen
1. den Clubnamen
2. Hardwarebezug
3. Softwarebezug
Das ganze sollte auch noch einigermaßen nett aussehen und auch noch ganz klein erkennbar bleiben. Was daraus kam siehst Du hier.

Bis dann

Matthias





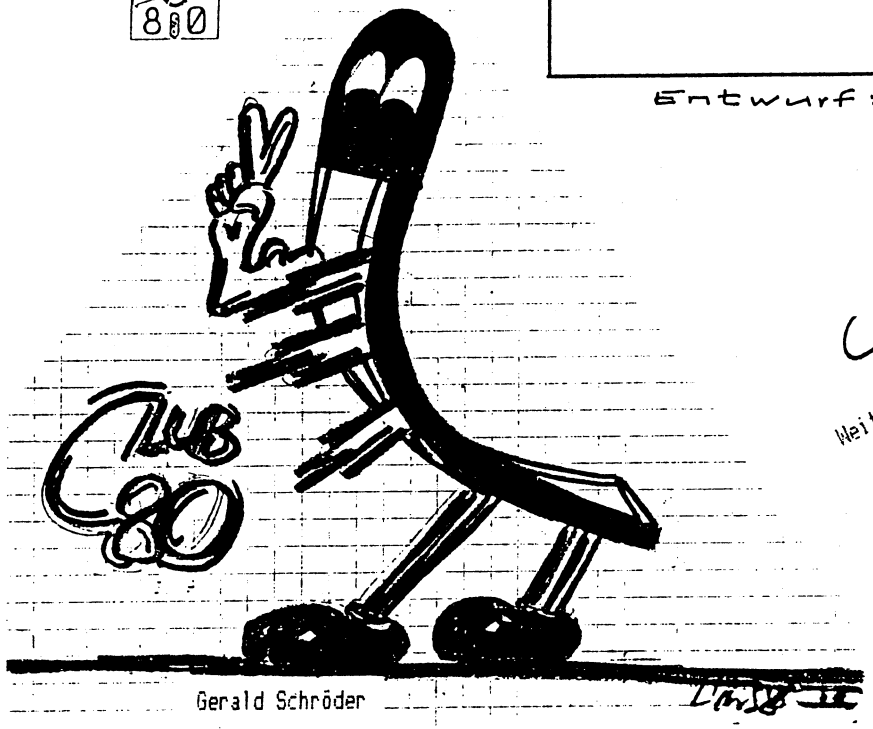
In eigener Sache:

Wir haben keine Druckerei für das Clubinfo mehr!!

Peter Spieß hört aus beruflichen Gründen auf. Da er sich nun nur noch mit IBM-PC's beschäftigt, ist er gleichzeitig auch aus unserem Club ausgetreten.

Aus diesem Grunde suchen wir einen neuen Vervielfältiger für unsere kommenden Club-Info's. Vielleicht hat einer von Euch eine gute Adresse bei der der Club 80 weiterhin kostengünstig das Club-Info herstellen lassen kann?

Meldungen dazu bitte an die Club-80-Redaktion.



Gerald Schröder

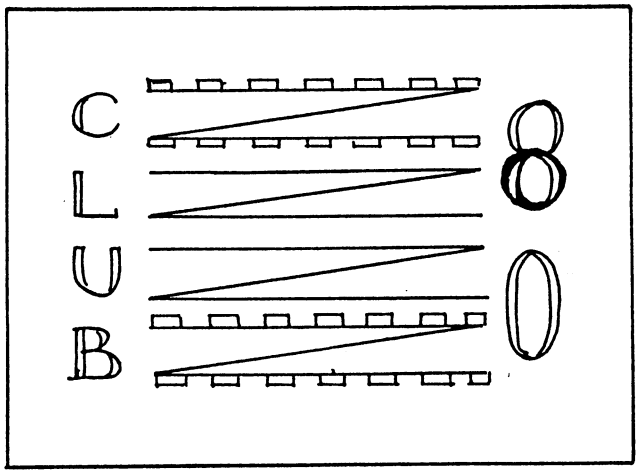
Betr.: Club 80 - Emblem [umseitig]

08

1) Die beiden Außenleisten links + rechts sind klar.

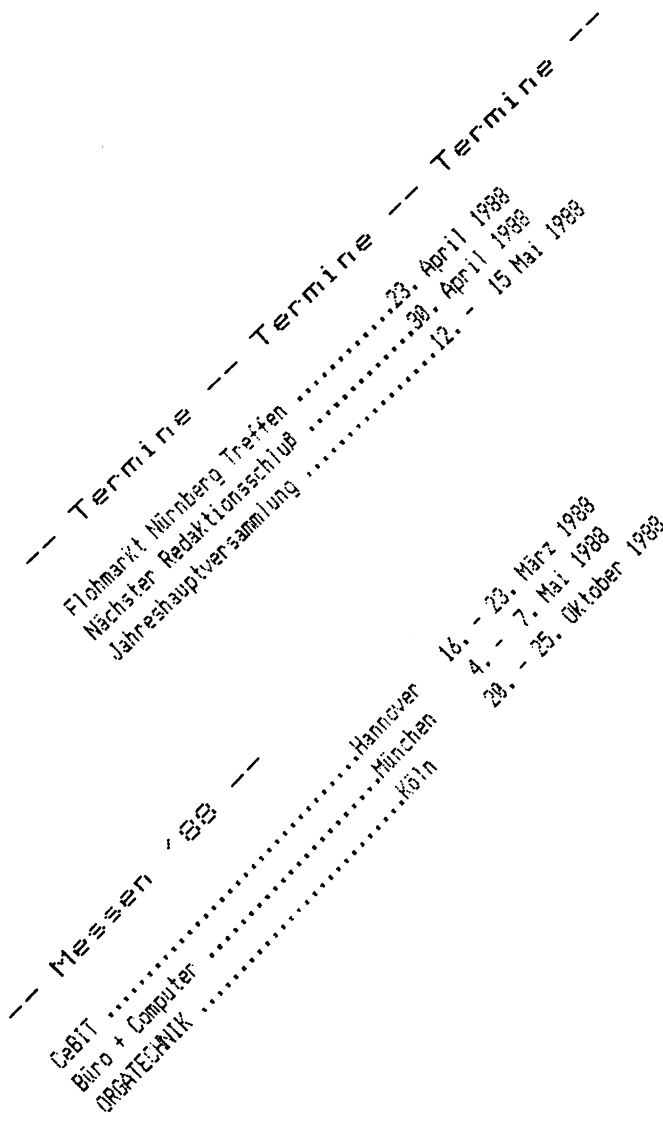
2) Der Mittelteil symbolisiert:
   a) das Z von Z 80
   b) je einen Chip oben und unten
   c) die „endlose" Zickzack-Faltung des Druckpapierstapels

Kajot, Weinheim 18.1.88



Entwurf: Kajot

Weitere Vorschläge zum Club-Emblem



Frank Sandlos

Kleines Textprogramm

Für TRS-80

Hier ein kleines Text-Verarbeitungsprogramm für das TRS-80-Modell 1. Mit wenig Aufwand kann man damit ganz schön Texte erstellen. Vielleicht auch als Beispiel geeignet, wie Besitzer anderer Computermodelle sich so ein Text-System anlegen könnten. Umlaute und Groß-Kleinschreibung sind möglich.

Das Programmlisting (*Bild 1*) enthält in den Zeilen 51000...51020 eine Basic-Routine, die ein Maschinenprogramm in den Speicher ab 32683 (dez.) schreibt (Speicher: 16 KByte), das Großbuchstaben mit einem Grafik-Block zur Unterscheidung von den Kleinbuchstaben versieht. Ferner sind die Kleinbuchstaben direkt wie bei einer normalen Schreibmaschine erreichbar, Großbuchstaben nur mit „Shift". Besitzer eines TRS-80 mit 16-KByte-Speicher und eingebauter Kleinschrift (und möglicherweise Umlauten) lassen die Zeilen 5,51000...51020 weg und benutzen Zeile 104 aus dem Listing in Bild 4. Ist ein Umlautsatz ebenfalls schon eingebaut, so kann auch die Sondertastenbelegung in Zeile 132 wegfallen. Man sollte dann auch die in den Zeilen 355...357 unterstrichenen Druckercodes experimentell überprüfen, die eventuell modifiziert werden müssen. Bei Benutzung der Kleinschreibroutine muß „MEM SIZE?" mit 32683 beantwortet werden. Selbstverständlich kann dieses Programm auch mit einem anderen Druckermodell als dem MX-80 von Epson benutzt werden, es müssen dann die Druckercodes in den Zeilen 132,355...357 geändert werden. Ist bei der Eingabe des Programmes das Programmzeilenende erreicht, d. h. der Computer nimmt keine Zeichen mehr für die betreffende Zeile an, obwohl im Listing noch Zeichen stehen, so gehe man in den Edit-Modus. Nur im Edit-Modus des TRS-80 läßt sich die im Handbuch angegebene Zeilenlänge von max. 255 Zeichen erreichen!

Dieses Programm addiert bei der Druckerausgabe zum ASCII-Wert des betreffenden Zeichens 32 hinzu, wenn er im Bereich von 64...90 liegt. Liegt dieser Wert zwischen 96 und 122, so wird 32 subtrahiert. Deshalb müssen die Druckeranweisungen in diesem Bereich angepaßt werden.

Die Bedienung des Programms

Nach Eingabe und Start des Programms wählt man die Hauptzeilenlänge, entweder 122 oder 72 Zeichen/Zeile. Diese Zeilenlänge wird dann beim Programmablauf und bei der Druckerausgabe beibehalten. Bei 122 Zeichen/Zeile wird der Drucker auf die Schriftart „SI" programmiert, ansonsten wird die Normalgröße beibehalten. Der Text wird mit einem genügend breiten Abstand vom linken Papierrand ausgedruckt (zum Abheften). Um eine optimale Ausnutzung des DIN-A4-Blattes zu erreichen, muß die obere Kante des jeweiligen Blattes mit dem anliegenden Rand der klappbaren Skaleneinteilung des MX-80 bündig abschließen. Nach der Wahl der Hauptzeilenlänge wählt man entweder die Schriftart der nächsten einzugebenden Zeile oder man unterbricht den Eingabemodus durch Drücken der Taste „Pfeil unten". Folgende Eingaben zur Schriftauswahl sind möglich:

‚ENTER'	– Normalschrift (N)
‚E'	– „Emphasized"-Schrift (E)
‚D'	– „Double Printed"-Schrift (D)
‚S'	– „Shift Out"-Schrift (S), dabei autom. Begrenzung der Zeilenlänge auf 36 Zeichen.

Nun erscheint eine Skala, unter der ein Leuchtbalken die Länge der Textzeile markiert. Dahinter befindet sich die Zeilennummer sowie nach einem Grafik-Block eine Abkürzung der Schriftart. Jetzt kann man den Text eingeben, mit Benutzung der in *Bild 2* beschriebenen Sondertasten. Will man die Eingabe beenden und die Textzeile speichern, einfach „Enter" drücken. Drückt man „Enter" ohne Zeicheneingabe, so wird die Zeile beim Ausdrucken des Textes ignoriert, es entsteht keine Leerzeile. Leerzeilen werden erst ausgedruckt, wenn man vor „Enter" mindestens ein Leerzeichen eingibt. Drückt man, anstatt die Schriftart der Folgezeile auszuwählen, die Taste „Pfeil unten", so erscheint eine Auswahltafel auf dem Monitor. Diese erklärt sich weitgehend von selbst, einige Punkte sollen jedoch näher erläutert werden.

Sichten der Textzeilen, „Pfeil unten"

Nach Drücken dieser Taste fragt das Programm nach der Anfangszeilennummer, ab der der Text gesichtet werden soll. „A" + „Enter" bewirkt automatischen Durchlauf des ganzen Textes. „Enter" bewirkt Anzeige der ersten Zeile. Danach manuelles Weiterschalten durch „Pfeil unten".

„xx" + „Enter", die Zeile Nr. xx wird angezeigt, manuelles Weiterschalten durch „Pfeil unten".

Zum automatischen Sichten: Nach Beendigung dieses Vorgangs entweder „Enter" zum Verlassen dieses Programmteiles oder „A" zur erneuten Durchsicht drücken.

Zum manuellen Sichten: „Enter" bewirkt Verlassen dieses Programmteiles, „K" bewirkt anschließende Korrektur der zuletzt angezeigten Zeile.

Einfügen einer Zeile, „Pfeil rechts"

Nach Tastendruck der „Pfeil rechts"-Taste muß die Zeilennummer der neu einzufügenden Zeile eingegeben werden. Diese muß genau in der Mitte zwischen den Nummern zweier schon eingegebener Zeilen liegen, also z. B. 1.5 oder 2.5, wenn die Zeilen 1 und 2 bzw. 2 und 3 schon eingegeben sind (möglich ist auch eine Überschreibung einer schon eingegebenen Zeile). Danach entweder „N", „E", „S" oder „D" zur Schrifttypauswahl drücken (Beantwortung der Frage „Typ?"). Nach Eingabe der Zeile „Enter" drücken, wie im Eingabemodus. Bitte beachten: Die vorgesehene Zeilenzahl pro DIN-A4-Seite beträgt 67 Zeilen. Deshalb nicht mehr als zwei Zeilen zusätzlich einfügen, wenn die volle Zeilenzahl in Anspruch genommen wird.

Korrektur einer Zeile, „K"

Hier gibt man entweder die Zeilennummer an oder drückt „Enter", wodurch die erste Zeile des Textes korrigiert werden kann. Auf dem Bildschirm wird nicht nur die Korrekturzeile, sondern zur besseren Übersicht auch die vorher-

gehende und nachfolgende Zeile angezeigt. Die besonderen Tastenfunktionen bei der Korrektur einer Zeile zeigt *Bild 3*.

Aufbau des Programms

Zeilen:

10...90	Titel, Eingabe der Hauptzeilenlänge
99	Beginn der Hauptschleife
103	Eingabe der Schriftart
131...143	Eingabemodus
150...155	Auswahltafel
160...169	Sichten der Textzeilen
175	Einfügen einer Textzeile
190...230	Korrektur
300...305	Speichern des Textes auf Kassette
310...315	Lesen des Textes von Kassette
350...359	Text Ausdruck
ab 51000	Basic-Loader

TIP: Zur besseren Orientierung kann man die Umlaute mit einem Prägegerät in Spezialfolie stanzen und vor die entsprechenden Tastenkappen kleben.
Bild 1 zeigt, wie gesagt, das komplette Programm für den TRS-80, Modell 1, 16 KByte, ohne Modifikationen. In *Bild 4* ist das Programm in einer Version für den TRS-80 mit 48-KByte-Speicher sowie Kleinschriftmodifikation und eventuell deutschem Zeichensatz zu sehen (verwendet wird der Kleinschrifttreiber aus mc 12/1982). In dieser Version sind einige Bedienungstasten geändert worden, näheres klärt sich beim Programmlauf von selbst. Besonders viel Ärger bereitete der Aufbau von Zeichenkettendateien in der Form PRINT#-1, A$,B$,... Die so aufgebaute Datei wird vom TRS-80 nur unvollständig eingelesen. Alle Versuche, diesen Fehler durch geschickte Programmierung zu umge-

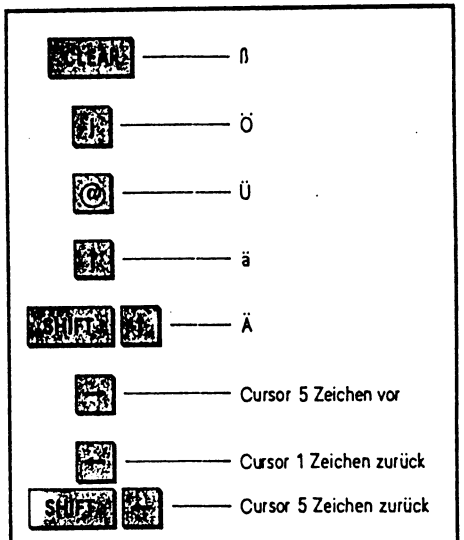


Bild 2. Die Sondertasten des Eingabemodus

TRS-80: Sortieren

In mc 1982, Heft 2, wurde eine Verbesserung der Stringsortierung für Verfahren wie z. B. Quicksort vorgestellt. Die Methode der Sortierung der Ordnungsnummer anstelle der eigentlichen Strings verbraucht aber zusätzlichen Speicherplatz, der dann an Aufnahmekapazität verlorengeht.
Jedem String eine Ordnungsnummer zuzuweisen (ein Integer-Array belegt 2n+m Speicherplätze, wobei n die Anzahl der Strings bzw. Ordnungsnummern ist und m die Anzahl der Bytes der internen Arraybeschreibung), verbraucht also bei großen Text-Arrays zusätzlich 2×n Bytes. Nun ist es aber beim TRS-80 oder beim Video-Genie so, daß sich Vertauschung von String-Inhalten ohne eigentliche Basic-Zuweisung ausführen läßt.
Wie schon in mc geschildert, wird bei einer Zuweisung an eine String-Variable freier Speicherplatz beschrieben und der alte Platz im Textspeicher einfach vergessen. Das kann dann bis zur völligen Belegung des Textspeichers und dem damit verbundenen Aufruf der Garbage-Kollektion führen.
Ist eine Textvariable eingerichtet, das ist bei Komponenten eines Arrays immer der Fall, so besteht diese Variable zunächst nur aus einem Deskriptor, der die Länge des Strings und die Startadresse enthält. Wird nun der Variablen ein (Text-)Wert zugewiesen, so wird dieser Wert im freien Textspeicher abgelegt und der Anfangszeiger des Deskriptors auf die entsprechende Stelle gesetzt. Des weiteren wird der Längenzähler auf den neuen Wert (Länge des zugewiesenen Wertes) gesetzt. Bei einer erneuten Zuweisung wird der alte Deskriptor auf einen neueingerichteten String im Textspeicher gesetzt usw. Die Belegung freien Speicherplatzes läßt sich vermeiden, wenn man nicht einfach eine Zuweisung wie etwa

```
C$=A$(1):A$(1)=A$(2):A$(2)=C$
```

vornimmt, sondern gleich die entsprechenden Deskriptoren verändert. Folgende Fragen tauchen dabei auf:

1. Wo findet man den Deskriptor?

2. An welcher Stelle steht der Längenzähler oder der Zeiger?

Beim TRS-80 liefert die Funktion VARPTR (‹Variable›), „Variablen-Pointer", die Stelle, an der im Speicher der Wert einer Variablen zu finden ist. Bei Textvariablen wird die Adresse des Deskriptors geliefert. Sei A$ eine Textvariable, so liefert

```
P=VARPTR(A$)
```

die Adresse des Deskriptors vom momentanen Stand von A$. Die Speicherstelle (P) enthält nun die Länge des Strings, (P+1) (LSB), (P+2) (MSB) enthalten den Zeiger auf den Textspeicher. Diese drei Parameter braucht man nur gegen andere auszutauschen, um der Textvariable A$ einen neuen Wert zuzuweisen, ohne dabei neuen Textspeicherplatz zu belegen. Folgendes Programm bietet sich daher zum „Swapping" (Austausch) von Texten an:

```
100 DIM A$(200)
...
200 A$="Text1"
210 A$="Text2"
...
1000 P=VARPTR(A$(1):Q
     =VARPTR(A$(2))
1010 P1=PEEK(P):P2=PEEK(P+1):
     P3=PEEK(P+2)
1020 Q1=PEEK(Q):Q2=PEEK(Q+1):
     Q3=PEEK(Q+2)
1030 POKE Q,P1:POKE Q+1,P2:POKE
     Q+2,P3
1040 POKE P,Q1:POKE P+1,Q2:POKE
     P+2,Q3
...
```

Nach Durchlauf der Zeilen 1000 bis 1040 sind die Inhalte der Variablen A$(1) und A$(2) ausgetauscht.
Bei anderen Systemen erfolgt die Verwaltung von Strings ähnlich.
Auf eine weitere Möglichkeit der Ausnutzung der Deskriptoren sei an dieser Stelle noch hingewiesen.
Will man etwa direkt auf den Video-Speicher des TRS-80 zugreifen, so kann dies „peek by peek" (Byte für Byte) oder abschnittweise erfolgen. Die erstere Methode, bei der jedes Byte einzeln „gepeekt" wird und in die entsprechenden Buchstaben umgewandelt werden muß, verbraucht viel Zeit und auch wieder viel Textspeicherplatz, da jeder Buchstabe im Textspeicher (unnützerweise) abgelegt wird.
Folgendes Programmstück bildet den Inhalt des Video-RAM auf einem Text-Array Z$(16) ab, wobei nach dieser Abbildung eine Zuweisung an andere Textvariable vorzunehmen ist, da beim Schreiben auf den Bildschirm unmittelbar in die Variableninhalte geschrieben wird.
Man kann natürlich auch zu Anfang des Programms diese Variablen entsprechend präparieren, und dann deren Inhalte abfragen.

```
100 DIM Z$(16)
...
1000 P=15360:FOR N=0 TO 15
1010 Q=VARPTR(Z$(N+1)):
     P1=N*64+P
1020 Q1=INT(P1/256):Q2=P1-256*Q1
1030 POKE Q,64:POKE Q+1,Q2:POKE
     Q+2,Q1
1040 NEXT N
```

Alle Variablen sollten bereits eingerichtet sein, um eine Verschiebung ihrer Adressen beim Einrichten neuer Variablen zu vermeiden.

Nach Abarbeitung des Programmstücks 1000...1040 zeigen die Pointer der Variablen Z$(1)...Z$(16) genau auf das Video-RAM. Man braucht also nicht mehr byteweise abzufragen, sondern nur noch zeilenweise.

Heinrich Seebauer

```
5 GOSUB51020
10 CLS:CLEAR9400:DIMA$(133),Z(40):DEFINTA-Y:PRINT$86,"T E X T S Y S T E M":PRINT
STRING$(64,42):FORZ=0TO200:NEXT:G$=STRING$(4,176):H$=CHR$(143):I$=CHR$(140):K$=C
HR$(191):M$=CHR$(131)
20 N$="----"+H$+"----1----"+H$+"----2----"+H$+"----3----"+H$+"----4----"+H$+"---
-5----"+H$+"----6---4":P$="--- SCHRIFTSTAERKE(N;D;E;S) ------ ' "+CHR$(92)+" 'SC
HREIBMODUS BEENDEN ---"
25 ST$=">>>>>>>> START <<<<<<<<":KA$="$$ KASSETTENRECORDER BEREIT "
30 PRINT$211,G$" "G$" "G$" "G$" "CHR$(176)" "G$" "G$CHR$(176)$275,H$I$I$CHR$(188)" "K
$"    "K$I$I$CHR$(183)" "K$" "K$I$I$H$"    "K$$339,M$M$M$M$" "M$M$M$M$" "M$"  "M$
" "M$" "M$"      "M$
60 FORZ=0TO650:NEXT:PRINT$143,STRING$(34,176)$207,K$$240,K$$271,K$$304,K$$335,K$
$368,K$$399,STRING$(34,131)
70 FORZ=0TO500:NEXT:PRINT$194,"EIN"$260,"PROGRAMM"$331,"VON"$309,"F R A N K"$371
,"S A N D L O S":FORZ=0TO400:NEXT:PRINT:PRINT"HAUPTZEILENLAENGE:":PRINT:PRINT"
   72 ZEICHEN/ZEILE ----------- N(ORMAL)"
80 PRINT"    122 ZEICHEN/ZEILE ----------- E(NG)":PRINT:PRINT:PRINT"          BIT
TE ANFANGSBUCHSTABEN DRUECKEN !"
90 W$=INKEY$:IFW$="",90ELSEIFW$="N",ZH=72ELSEIFW$="E",ZH=122ELSE90
99 FORZ=0TO132STEP2
100 O$=STR$(-(Z/2+1)):O$=STRING$(4-LEN(O$),45)+O$+"-":P=832:ZL=ZH:CLS:PRINT$768,
P$;:E$=""
103 R$=INKEY$:IFR$="",103ELSEIFASC(R$)=13,R$="N":GOTO104ELSEIFR$="D",104ELSEIFR$
="E",104ELSEIFR$="S",ZL=36ELSEIFASC(R$)=10,150ELSE103
104 CLS:ZZ=0:FORZ1=Z-1TOZ-24STEP-1:IFZ1<0,110ELSEZ2=LEN(A$(Z1)):IFZ2>0,Z3=INT(Z2
/64)+1:ZZ=ZZ+Z3+1:IFZZ>12,110ELSEPRINT$(768-ZZ*64),A$(Z1);:NEXTZ1ELSENEXTZ1
110 PRINT$(P-64),N$E$STRING$(ZL-LEN(E$),143)O$R$;
131 A$=INKEY$:IFA$="",131ELSEA=ASC(A$):IFA=91,E$=E$+CHR$(123):GOTO133ELSEIF64<AA
NDA<123,E$=E$+A$:PRINT$P,E$;:IFLEN(E$)<ZL,131ELSEE=LEN(E$):GOTO140ELSEE=LEN(E$):
IFA=8ANDE>0,E$=LEFT$(E$,E-1):GOTO133ELSEIFA=24ANDE>4,E$=LEFT$(E$,E-5):GOTO133
132 IFA=32,E$=E$+" ":GOTO133ELSEIFA=31,E$=E$+CHR$(126):GOTO133ELSEIFA=59,E$=E$+C
HR$(92):GOTO133ELSEIFA<64ANDA>32,E$=E$+A$:GOTO133ELSEIFA=27,E$=E$+CHR$(91)ELSEIF
A=64,E$=E$+CHR$(93)ELSEIFA=9ANDE+5<ZL,E$=E$+"     "ELSEIFA=13ANDV=0,143ELSE140
133 E=LEN(E$):PRINT$P,E$STRING$(ZL-E,143)O$R$;:IFE<ZL,131
140 IF(V=0ANDE>=ZL)OR(V=2ANDE=>ZL),141ELSEIFV=1ORV=2,A$(ZA)=E$+O$+R$:GOTO205ELSE
131
141 W$=INKEY$:IFW$="",141ELSEW=ASC(W$):IFW=8,E$=LEFT$(E$,E-1):GOTO133ELSEIFW=24,
E$=LEFT$(E$,E-5):GOTO133ELSEIFW=13ANDV=0,143ELSEIFW=13ANDV=2,A$(ZA)=E$+O$+R$:GOT
O205ELSE141
143 A$(Z)=E$+O$+R$:NEXTZ:GOTO150
150 PRINTCHR$(15):CLS:PRINT$64,"KORREKTUR:          '"CHR$(92)" XX'  - SICHTEN DER
 ZEILEN AB XX"$146,"'"CHR$(94)" XX'  - EINFUEGEN EINER ZEILE"$210,"'K XX'  - KOR
REKTUR ZEILE XX"$384,"TEXT AUSDRUCKEN:              'A'"$576,"AUF CASSETTE SPEICHER
N:    'C'"
151 PRINT$768,"VON CASSETTE LESEN:          'L'"$913,"'ENTER'  - SCHREIBMODUS";$98
0,"E      - PROGRAMMENDE";
155 W$=INKEY$:IFW$="",155ELSEW=ASC(W$):IFW=10,160ELSEIFW=9,175ELSEIFW=13,CLS:GOT
O100ELSEIFW$="K",190ELSEIFW$="A",350ELSEIFW$="C",300ELSEIFW$="L",310ELSEIFW$="E"
,400ELSE155
160 ZA$="0":PRINT$114,"NR.";:INPUTZA$:IFZA$="A",168ELSEZA=VAL(ZA$)$2-2:IFZA>Z,16
0ELSEIFZA=-2,ZA=0:CLSELSECLS
```

Bild 1. Das Programmlisting für die 16-KByte-Version mit Kassettenspeicherung (MEM SIZE? 32683)

```
161 PRINTA$(ZA)
165 W$=INKEY$:IFW$="",165ELSEIFW$="K",191ELSEW=ASC(W$):IFW=13,150ELSEIFW=10,ZA=Z
A+1:IFZA=Z,ZA=0:GOTO161ELSEIFA$(ZA)="",165ELSEPRINTA$(ZA):GOTO165ELSE165
168 CLS:FORZV=0TOZ:IFA$(ZV)="",NEXTZVELSEPRINTA$(ZV):NEXTZV
169 W$=INKEY$:IFW$="",169ELSEIFW$="A",168ELSEIFASC(W$)=13,150ELSE169
175 PRINT$178,"NR.";:INPUTZA:PRINT$178,CHR$(201);:PRINT$178,"TYP";:INPUTR$:U$="E
 I N F U E G E N":Q=1:K$=STR$(-ZA):IFLEN(K$)>4,K$=STR$(ZA)ELSEK$=STRING$(4-LEN(K
$),45)+K$
176 K$=K$+"-"+R$:ZA=ZA$2-2:Z(T)=ZA:T=T+1:GOTO200
190 ZA=1:PRINT$236,"NR.";:INPUTZA:ZA=ZA$2-2
191 U$="K O R R E K T U R"
200 WW=0:IFZA=0,203ELSEIFA$(ZA-1)="",ZE=ZA-2ELSEZE=ZA-1
203 IFA$(ZA+1)="",ZF=ZA+2ELSEZF=ZA+1
205 V=0:CLS:PRINT$19,U$"  NR."ZA/2+1$64,A$(ZE);$256,A$(ZA);$448,N$LEFT$(K$,WW)CH
R$(95);$768,A$(ZF);:IFQ=1,230
210 W$=INKEY$:IFW$="",210ELSEW=ASC(W$):K$=A$(ZA):K=LEN(K$):IFW=13,150ELSEIFW=32A
NDK>WW+1,WW=WW+1ELSEIFW=8ANDWW>0,WW=WW-1ELSEIFW=9ANDK>WW+5,WW=WW+5ELSEIFW=27,A$(
ZA)=LEFT$(K$,WW)+RIGHT$(K$,K-WW-1):GOTO205
211 IFW=91,220ELSEIFW=31,230ELSEIFW=25,K=K+1:GOTO220ELSEPRINT$512,LEFT$(K$,WW)CH
R$(95)STRING$(K-WW,32);:GOTO210
220 ZL=WW+1:E$=LEFT$(K$,WW):O$=RIGHT$(K$,K-WW-1):R$="":P=512:V=1:GOTO131
230 ZL=ZH:E$=LEFT$(K$,WW):O$=RIGHT$(K$,6):R$="":P=512:V=2:Q=0:IFRIGHT$(K$,1)="S"
,ZL=36:GOTO110ELSE110
300 CLS:PRINT$279,"S P E I C H E R N":PRINT$390,KA$;:INPUTS:PRINT$788,ST$:PRINT#
-1,ZH,Z,T:TN=0:FORGV=0TOZ-2STEP2:GW=GV
301 W=LEN(A$(GW)):B$="A"+A$(GW):IFW=0,305
303 FORGG=1TOW+1:X=ASC(MID$(B$,GG,1)):Y=LEN(B$):T$=STR$(GG):IF(X>32ANDX<48)OR(X>
57ANDX<65),B$=LEFT$(B$,GG-1)+" "+RIGHT$(B$,Y-GG)+STRING$(4-LEN(T$),32)+T$+STR$(X
):NEXTGGELSENEXTGG
305 B=LEN(B$)-W-1:C=W:PRINT#-1,TN,B,C,B$:NEXTGV:IFT=0ORGV=T,150ELSEFORGV=0TOT-1:
TN=Z(GV):GW=TN:GOTO301
310 CLS:PRINT$282,"L E S E N":PRINT$390,KA$;:INPUTS:PRINT$788,ST$:INPUT#-1,ZH,Z,
T:FORGV=0TOZ-2STEP2:GW=GV:INPUT#-1,TN,B,C,B$
313 IFB=0,A$(GW)="":NEXTGV:GOTO150ELSEX$=RIGHT$(B$,B):FORGG=1TOBSTEP7:Q=VAL(MID$
(X$,GG,5)):IFNOTQ=0,B$=LEFT$(B$,Q-1)+CHR$(VAL(MID$(X$,GG+4,3)))+MID$(B$,Q+1,C+1-
Q):NEXTGG
315 A$(GW)=RIGHT$(B$,C):NEXTGV:IFT=0ORGV=T,150ELSEFORGV=0TOT-1:INPUT#-1,TN,B,C,B
$:GW=TN:GOTO313
350 CLS:PRINT$472,"DRUCKER ON-LINE ";:INPUTS:PRINT$788,ST$:IFZH=122,N=10:S=5:LPR
INTCHR$(15)CHR$(24)ELSEN=5:S=5:LPRINTCHR$(24)
355 C=20:FORGX=0TOZ-1:R$=RIGHT$(A$(GX),1):A=LEN(A$(GX)):IFA=0ORA=6,NEXTGXELSEA$=
LEFT$(A$(GX),A-6):IFR$="N",LPRINTTAB(N)CHR$(27)CHR$(C)A$:C=20:NEXTGXELSEIFR$="S"
,LPRINTTAB(S)CHR$(27)CHR$(C)CHR$(14)CHR$(27)CHR$(101)A$:C=102:NEXTGXELSE357
356 GOTO358
357 IFR$="E",LPRINTTAB(N)CHR$(27)CHR$(C)CHR$(27)CHR$(101)A$:C=102:NEXTGXELSEIFR$
="D",LPRINTTAB(N)CHR$(27)CHR$(C)CHR$(27)CHR$(103)A$:C=104:NEXTGX
358 LPRINTCHR$(27)CHR$(C);:IFZH=122,LPRINTCHR$(18);
359 GOTO150
400 CLS:PRINT" S - LOESCHEN/NEUSTART":PRINT" E - ENDE"
401 W$=INKEY$:IFW$="",401ELSEIFW$="S",RUN10ELSEIFW$="E",END
51000 DATA 42,30,64,34,213,127,33,198,127,34,30,64,42,38,64,34,242,127,33,233,12
7,34,38,64,195,25,26,245,121,205,244,127,56,5,62,130,205,51,0,241,197,205,0,0,19
3,79,254,8,192,42,32,64,43,126,254,130,192,62,8,195,51,0,245,121,246,32,205,244
51010 DATA 127,241,195,0,0,254,123,63,216,254,97,216,121,238,32,79,201
51020 IFPEEK(16526)=171ANDPEEK(16527)=127,RETURNELSEA=32683:FORX=ATO32767:READB:
POKEX,B:NEXT:POKE16526,171:POKE16527,127:A=USR(0)
```

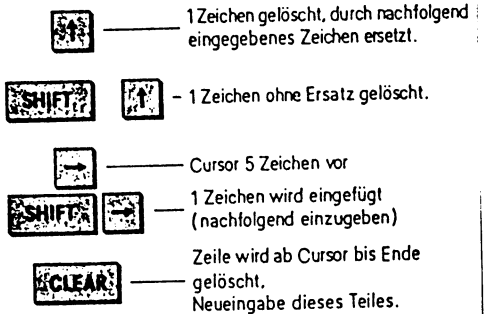


Bild 3. Die Sondertasten des Korrekturmodus

```
5 GOSUB51000
10 CLS:CLEAR15000:DIMA$(133),Z(40):DEFINTA-Y:PRINT$86,"T E X T S Y S T E M":PRIN
TSTRING$(64,45):FORZ=0TO200:NEXT:G$=STRING$(4,176):H$=CHR$(143):I$=CHR$(140):K$=
CHR$(191):M$=CHR$(131)
20 N$="-----"+H$+"-----1-----"+H$+"-----2-----"+H$+"-----3-----"+H$+"-"+CHR$(127)+"--4-
---"+H$+"-----5-----"+H$+"-----6---4":P$="--- Schriftstärke(n;d;e;s) ------- ' & 'Sc
hreibmodus beenden ---"
25 ST$=">>>>>>>> START <<<<<<<<":KA$="** Kassettenrekorder bereit "
30 PRINT$211,G$" "G$" "G$" "CHR$(176)" "G$" "G$CHR$(176)$275,H$I$I$CHR$(188)" "K
$"    "K$I$I$CHR$(183)". "K$" "K$I$I$H$"    "K$$339,M$M$M$M$" "M$M$M$M$" "M$"  "M$
" "M$" "M$"      "M$
60 FORZ=0TO650:NEXT:PRINT$143,STRING$(34,176)$207,K$$240,K$$271,K$$304,K$$335,K$
$368,K$$399,STRING$(34,131)
70 FORZ=0TO500:NEXT:PRINT$194,"Ein"$260,"Programm"$331,"von"$309,"F R A N K"$371
,"S A N D L O S":FORZ=0TO400:NEXT:PRINT:PRINT"Hauptzeilenlänge:":PRINT:PRINT"
  72 Zeichen/Zeile ---------- n(ormal)"
80 PRINT"    122 Zeichen/Zeile ---------- e(ng)":PRINT:PRINT:PRINT"          Bit
te Anfangsbuchstaben drücken !"
90 W$=INKEY$:IFW$="",90ELSEIFW$="n",ZH=72ELSEIFW$="e",ZH=122ELSE90
99 FORZ=0TO132STEP2
100 O$=STR$(-(Z/2+1));O$=STRING$(4-LEN(O$),45)+O$+"-":P=832:ZL=ZH:CLS:PRINT$768,
P$;:E$=""
103 R$=INKEY$:IFR$="",103ELSEIFASC(R$)=13,R$="n":GOTO104ELSEIFR$="d",104ELSEIFR$
="e",104ELSEIFR$="s",ZL=36ELSEIFR$="$",150ELSE103
104 CLS:ZZ=0:FORZ1=Z-1TOZ-24STEP-1:IFZ1<0,110ELSEZ2=LEN(A$(Z1)):IFZ2>0,Z3=INT(Z2
/64)+1:ZZ=ZZ+Z3:IFZZ>12,110ELSEPRINT$(768-ZZ$64),A$(Z1);:NEXTZ1ELSENEXTZ1
110 PRINT$(P-64),N$E$STRING$(ZL-LEN(E$),143)O$R$;
131 A$=INKEY$:IFA$="",131ELSEA=ASC(A$):IF64<AANDA<127,E$=E$+A$:PRINT$P,E$;:IFLEN
(E$)<ZL,131ELSEE=LEN(E$):GOTO140ELSEE=LEN(E$):IFA=8ANDE>0,E$=LEFT$(E$,E-1):GOTO1
33ELSEIFA=24ANDE>4,E$=LEFT$(E$,E-5):GOTO133
132 IFA=32,E$=E$+" ":GOTO133ELSEIFA<64ANDA>32,E$=E$+A$:GOTO133ELSEIFA=9ANDE+5<ZL
,E$=E$+"     "ELSEIFA=13ANDV=0,143ELSE140
133 E=LEN(E$):PRINT$P,E$STRING$(ZL-E,143)O$R$;:IFE<ZL,131
140 IF(V=0ANDE>=ZL)OR(V=2ANDE=>ZL),141ELSEIFV=1ORV=2,A$(ZA)=E$+O$+R$:GOTO205ELSE
131
141 W$=INKEY$:IFW$="",141ELSEW=ASC(W$):IFW=8,E$=LEFT$(E$,E-1):GOTO133ELSEIFW=24,
E$=LEFT$(E$,E-5):GOTO133ELSEIFW=13ANDV=0,143ELSEIFW=13ANDV=2,A$(ZA)=E$+O$+R$:GOT
0205ELSE141
143 A$(Z)=E$+O$+R$:NEXTZ:GOTO150
```

```
150 PRINTCHR$(15):CLS:PRINT$64,"Korrektur:          '# XX'  - Sichten der Zeilen a
b XX"$146,"'"CHR$(94)" XX'  - Einfügen einer Zeile"$210,"'k XX'  - Korrektur Zei
le XX"$384,"Text ausdrucken:          'a'"$576,"Auf Cassette speichern:    'c'"
151 PRINT$768,"Von Cassette lesen:          '1'"$913,"'ENTER'  - Schreibmodus";$98
0,"e     - Programmende";
155 W$=INKEY$:IFW$="",155ELSEW=ASC(W$):IFW=35,160ELSEIFW=123,175ELSEIFW=13,CLS:G
OTO100ELSEIFW$="k",190ELSEIFW$="a",350ELSEIFW$="c",300ELSEIFW$="1",310ELSEIFW$="
e",400ELSE155
160 ZA$="0":PRINT$114,"Nr.";:INPUTZA$:IFZA$="a",168ELSEZA=VAL(ZA$)$2-2:IFZA>Z,16
0ELSEIFZA=-2,ZA=0:CLSELSECLS
161 PRINTA$(ZA)
165 W$=INKEY$:IFW$="",165ELSEIFW$="K",191ELSEW=ASC(W$):IFW=13,150ELSEIFW=10,ZA=Z
A+1:IFZA=Z,ZA=0:GOTO161ELSEIFA$(ZA)="",165ELSEPRINTA$(ZA):GOTO165ELSE165
168 CLS:FORZV=0TOZ:IFA$(ZV)="",NEXTZVELSEPRINTA$(ZV):NEXTZV
169 W$=INKEY$:IFW$="",169ELSEIFW$="a",168ELSEIFASC(W$)=13,150ELSE169
175 PRINT$178,"Nr.";:INPUTZA:PRINT$178,CHR$(201);:PRINT$178,"Typ";:INPUTR$:U$="E
 I N F U E G E N":Q=1:K$=STR$(-ZA):IFLEN(K$)>4,K$=STR$(ZA)ELSEK$=STRING$(4-LEN(K
$),45)+K$
176 K$=K$+"-"+R$:ZA=ZA$2-2:Z(T)=ZA:T=T+1:GOTO200
190 ZA=1:PRINT$236,"Nr.";:INPUTZA:ZA=ZA$2-2
191 U$="K O R R E K T U R"
200 WW=0:IFZA=0,203ELSEIFA$(ZA-1)="",ZE=ZA-2ELSEZE=ZA-1
203 IFA$(ZA+1)="",ZF=ZA+2ELSEZF=ZA+1
205 V=0:CLS:PRINT$19,U$"  Nr."ZA/2+1$64,A$(ZE);$256,A$(ZA);$448,N$LEFT$(K$,WW)CH
R$(95);$768,A$(ZF);:IFQ=1,230
210 W$=INKEY$:IFW$="",210ELSEW=ASC(W$):K$=A$(ZA):K=LEN(K$):IFW=13,150ELSEIFW=32A
NDK>WW+1,WW=WW+1ELSEIFW=8ANDWW>0,WW=WW-1ELSEIFW=9ANDK>WW+5,WW=WW+5ELSEIFW=27,A$(
ZA)=LEFT$(K$,WW)+RIGHT$(K$,K-WW-1):GOTO205
212 IFW=123,220ELSEIFW=31,230ELSEIFW=25,K=K+1:GOTO220ELSEPRINT$512,LEFT$(K$,WW)C
HR$(95)STRING$(K-WW,32);:GOTO210
220 ZL=WW+1:E$=LEFT$(K$,WW):O$=RIGHT$(K$,K-WW-1):R$="":P=512:V=1:GOTO131
230 ZL=ZH:E$=LEFT$(K$,WW):O$=RIGHT$(K$,6):R$="":P=512:V=2:Q=0:IFRIGHT$(K$,1)="s"
,ZL=36:GOTO110ELSE110
300 CLS:PRINT$279,"S P E I C H E R N":PRINT$390,KA$;:INPUTS:PRINT$788,ST$:PRINT#
-1,ZH,Z,T:TN=0:FORGV=0TOZ-2STEP2:GW=GV
301 W=LEN(A$(GW)):B$="A"+A$(GW):IFW=0,305
303 FORGG=1TOW+1:X=ASC(MID$(B$,GG,1)):Y=LEN(B$):T$=STR$(GG):IF(X>32ANDX<48)OR(X>
57ANDX<65),B$=LEFT$(B$,GG-1)+" "+RIGHT$(B$,Y-GG)+STRING$(4-LEN(T$),32)+T$+STR$(X
):NEXTGGELSENEXTGG
305 B=LEN(B$)-W-1:C=W:PRINT#-1,TN,B,C,B$:NEXTGV:IFT=0ORGV=T,150ELSEFORGV=0TOT-1:
TN=Z(GV):GW=TN:GOTO301
310 CLS:PRINT$282,"L E S E N":PRINT$390,KA$;:INPUTS:PRINT$788,ST$:INPUT#-1,ZH,Z,
T:FORGV=0TOZ-2STEP2:GW=GV:INPUT#-1,TN,B,C,B$
313 IFB=0,A$(GW)="":NEXTGV:GOTO150ELSEX$=RIGHT$(B$,B):FORGG=1TOBSTEP7:Q=VAL(MID$
(X$,GG,5)):IFNOTQ=0,B$=LEFT$(B$,Q-1)+CHR$(VAL(MID$(X$,GG+4,3)))+MID$(B$,Q+1,C+1-
Q):NEXTGG
315 A$(GW)=RIGHT$(B$,C):NEXTGV:IFT=0ORGV=T,150ELSEFORGV=0TOT-1:INPUT#-1,TN,B,C,B
$:GW=TN:GOTO313
350 CLS:PRINT$472,"Drucker ON-LINE ";:INPUTS:PRINT$788,ST$:IFZH=122,N=10:S=5:LPR
INTCHR$(15)CHR$(24)ELSEN=5:S=5:LPRINTCHR$(24)
355 C=20:FORGX=0TOZ-1:R$=RIGHT$(A$(GX),1):A=LEN(A$(GX)):IFA=0ORA=6,NEXTGXELSEA$=
LEFT$(A$(GX),A-6):IFR$="n",LPRINTTAB(N)CHR$(27)CHR$(C)A$:C=20:NEXTGXELSEIFR$="s"
,LPRINTTAB(S)CHR$(27)CHR$(C)CHR$(14)CHR$(27)CHR$(69)A$:C=70:NEXTGXELSE357
356 GOTO358
357 IFR$="e",LPRINTTAB(N)CHR$(27)CHR$(C)CHR$(27)CHR$(69)A$:C=70:NEXTGXELSEIFR$="
d",LPRINTTAB(N)CHR$(27)CHR$(C)CHR$(27)CHR$(71)A$:C=72:NEXTGX
358 LPRINTCHR$(27)CHR$(C);:IFZH=122,LPRINTCHR$(18);
359 GOTO150
400 CLS:PRINT" s - Löschen/Neustart":PRINT" e - Ende"
401 W$=INKEY$:IFW$="",401ELSEIFW$="s",RUN10ELSEIFW$="e",END
51000 DATA33,226,255,34,30,64,33,132,255,34,22,64,205,227,3,197,213,87,254,32,32
,17,58,128,56,183,40,11,58,209,255,238,32,50,209,255,175,24,47,122,254,31,40,248
,58,64,56,254,2,32,16,122,33,210,255,1,8,0,237,177,32,5,1,7,0,9,86,122,230,223
51002 DATA254,65,56,11,254,94,48,7,122,33,209,255,174,24,1,122,209,193,201,0,65,
79,85,97,111,117,83,67,91,92,93,123,124,125,126,31,221,110,3,221,102,4,218,154,4
,221,126,5,183,40,1,119,121,254,32,218,6,5,254,128,210,166,4,195,125,4
51004 IFPEEK(16526)=119ANDPEEK(16527)=255RETURNELSEFORX=-136TO-1:READB:POKEX,B:N
EXTX:POKE16526,119:POKE16527,255:A=USR(0)
```

Bild 4. Hier die 48-KByte-Version für TRS-80 mit Kleinschrift und deutschem Zeichensatz (MEM SIZE? 65399)

TRS-80 erstellt Hardcopy

Eine Hardcopy vom Bildschirm erstellt die kleine Maschinenroutine von *Bild 1*.

```
1000 FOR J=15360 TO 16320 STEP 64:
     FOR I=J TO J+63: IF PEEK(I)>122
     THEN LPRINT CHR$(42);:IF PEEK(I)
     =<122 THEN LPRINT CHR$(PEEK(I));
1010 NEXT I:LPRINT" ":NEXT J
```

Bild 1. Eine Bildschirm-Hardcopy erstellt dieses TRS-80-Programm

Dazu ist der im Level-II und Level-III-Basic nicht benutzte Befehl „NAME" im Direktmodus oder von einem Programm aus aufzurufen. Dadurch bleibt der Befehl USR(0) für andere Zwecke frei. Je nach Speicherumfang muß die Startadresse geändert werden. Zu ändern ist eventuell auch die Druckertreiberadresse (037E8H).

Eine Hardcopy in Basic ermöglicht das Programm von *Bild 2*. Dabei kann in begrenztem Umfang auf nicht grafikfähigen Druckern Grafik ausgegeben werden. Alle Zeichen, deren Code größer als 122 ist, werden nämlich als Sternchen - CHR$(42) - gedruckt. Natürlich kann man an seiner Stelle jedes andere Zeichen verwenden.

Wolfgang Bethmann

```
                    0000          ORG   0FFC0H
                    0001 ;HARDCOPY FUER TRS-80/VG/KOMTEK
                    0002 ;W.BETHMANN 11.1985
FFC0 21 C0 FF       0003          LD    HL,0FFC0H   ;Startadresse
FFC3 22 8E 40       0004          LD    (408EH),HL  ;Systemstartadresse
FFC6 22 B1 40       0005          LD    (40B1H),HL  ;MEM-SIZE setzen
FFC9 22 8F 41       0006          LD    (418FH),HL  ;Zeiger"Name"verbiegen
FFCC CD 94 03       0007          CALL  394H        ;UP: CR auf Drucker
FFCF 16 00          0008          LD    D,0         ;D auf 0 setzen
FFD1 DD 21 00 3C    0009          LD    IX,3C00H    ;1.Bildschirmplatz
FFD5 06 3F          0010 PRINT    LD    B,3FH       ;noch 63 Zeichen/Zeile
FFD7 DD 4E 00       0011 LOOP     LD    C,(IX)      ;Bildschirm nach C laden
FFDA DD E5          0012          PUSH  IX          ;Speicherinhalte retten
FFDC C5             0013          PUSH  BC          ;
FFDD CD 5F 30       0014          CALL  305FH       ;Druckertreiberadr.holen
FFE0 C1             0015          POP   BC          ;zurück vom Stack
FFE1 DD E1          0016          POP   IX          ;
FFE3 DD 23          0017          INC   IX          ;nächster Bildschirmplatz
FFE5 10 F0          0018          DJNZ  LOOP        ;Schleife noch 63 mal
                    0019 ;1.SCHLEIFE
FFE7 DD E5          0020          PUSH  IX          ;Speicherinhalte retten
FFE9 C5             0021          PUSH  BC          ;
FFEA D5             0022          PUSH  DE          ;
FFEB CD 94 03       0023          CALL  394H        ;UP: CR auf Drucker
FFEE D1             0024          POP   DE          ;zurück vom Stack
FFEF C1             0025          POP   BC          ;
FFF0 DD E1          0026          POP   IX          ;
FFF2 DD 23          0027          INC   IX          ;nächster Bildschirmplatz
FFF4 14             0028          INC   D           ;Zähler für CR
FFF5 3E 0F          0029          LD    A,0FH       ;noch 15 Zeilen
FFF7 BA             0030          CP    D           ;fertig ?
FFF8 20 DB          0031          JR    NZ,PRINT    ;sonst Schleife 15 mal
FFFA C3 CC 06       0032          JP    6CCH        ;zurück ins Basic
                    0033          END
```

Bild 2. Sogar begrenzte Grafikmöglichkeiten bietet dieses Basic-Programm

Zeichengenerator für Nadeldrucker

Selbst die einfachsten Nadeldrucker können Punktgrafiken drucken. Damit setzt nur die eigene Phantasie dem Zeichenvorrat des Druckers Grenzen. Braucht man ein mathematisches Sonderzeichen oder ein sonstiges Symbol, beginnt die Gestaltungsarbeit mit Papier und Bleistift. Vor jedem Versuch müssen die Punkte dann noch in ein Bitmuster umgesetzt werden. Diese Arbeit kann das abgedruckte Basic-Programm *(Bild 1)* erleichtern. Mit den Tasten des Zahlenfeldes steuert man die Punkte an, die gesetzt oder gelöscht werden sollen. Leider sind die Grafikpunkte des TRS-80 nicht quadratisch – die auf dem Bildschirm gemalte Figur wirkt also leicht verzerrt. Ein Tastendruck genügt, um das Resultat aufs Papier zu bringen. Im übrigen erklärt sich das Programm selber. Es wurde auf einem TRS-80, Model I geschrieben, dürfte aber nach Anpassung der Grafikbefehle (set (x, y)/reset (x,y)/point (x, y): Koordinate(x, y) setzen/löschen/prüfen) auch auf den meisten anderen Rechnern laufen. Bei einigen Druckern (Star DP-510 u. ä.) können nur sieben Bit direkt angesteuert werden. Der oberste Punkt muß dann über eine zusätzliche Abfrage (IF P(I)>127 THEN ...) angesteuert werden.

Von Drucker zu Drucker verschieden sind auch die Befehle zum Aufruf der Grafik. Sie sind in der DATA-Zeile abgelegt.

Als Beispiele für nützliche Sonderzeichen mögen Summenzeichen, Alpha, das Integralzeichen und das Zeichen für Unendlich genügen *(Bild 2)*.

Henning Schulz-Rinne

```
vo CLEAR 100:CLS:DEFINT X,Y,I,J,M,P,C:DEFSTR E
100 MX=10:DIM P(MX),C(3):REM MX = Zahl der Spalten
110 FOR I=0 TO 3:READ C(I):NEXT
120 PRINT "Bedienung über numerisches Tastenfeld:"
130 PRINT "+-----------+-----------+-----------+"
140 PRINT "I    (7)    I   auf-    I    (9)    I"
150 PRINT "I           I   wärts   I           I"
160 PRINT "+-----------+-----------+-----------+"
170 PRINT "I  links    I  Punkt    I  rechts   I"
180 PRINT "I           I  löschen  I           I"
190 PRINT "+-----------+-----------+-----------+"
200 PRINT "I  Spalte   I   ab-     I    (3)    I"
210 PRINT "I  löschen  I   wärts   I           I"
220 PRINT "+-----------+-----------+-----------+"
230 PRINT "I  aus-     I  Punkt    I (ENTER)   I"
240 PRINT "I  drucken  I  setzen   I           I"
250 PRINT "+-----------+-----------+-----------+"
260 PRINT "Alles löschen: '/'-Taste: Ende: '-'"
270 PRINT "Zum Anfangen Taste drücken":
280 E=INKEY$:IF E="" THEN 280
290 CLS
300 FOR X=0 TO MX:SET (X,8):NEXT
310 FOR Y=0 TO 8:SET (MX+1,Y):NEXT
320 X=0:Y=0
330 SET(X,9):SET(MX+2,Y)
340 E=INKEY$:IF E="" THEN 340
350 IF (ASC(E)<44) OR (ASC(E)>56) THEN E="":GOTO 340
360 RESET(X,9):RESET(MX+2,Y)
370 ON ASC(E)-44 GOTO 640,380,520,530,510,400,330,420,440,470,330,490
380 IF POINT(X,Y)=0 THEN P(X)=P(X)+2^(7-Y):SET(X,Y)
390 GOTO 470
400 IF Y<7 THEN Y=Y+1 ELSE Y=0
410 GOTO 330
420 IF X>0 THEN X=X-1 ELSE X=MX
430 GOTO 330
440 IF X>0 THEN X=X-1 ELSE X=MX
450 IF POINT(X,Y) THEN P(X)=P(X)-2^(7-Y):RESET(X,Y)
460 GOTO 330
470 IF X<MX THEN X=X+1 ELSE X=0
480 GOTO 330
490 IF Y>0 THEN Y=Y-1 ELSE Y=7
500 GOTO 330
510 P(X)=0:FOR I=0 TO 7:RESET (X,I):NEXT:GOTO 330
520 FOR I=0 TO MX:P(I)=0:NEXT:CLS:GOTO 300
530 PRINT @512,;:FOR I=0 TO MX:LPRINT PX(I);:NEXT
540 FOR I=0 TO 3
550 IF PEEK(14312)=63 THEN POKE 14312,C(I) ELSE 550
560 NEXT I
570 FOR I=0 TO MX
580 IF PEEK(14312)=63 THEN POKE 14312,P(I) ELSE 580
590 NEXT I
600 LPRINT:LPRINT:GOTO 330
610 REM Es folgen die zur Initialisierung des Druckers nötigen
620 REM Daten (MX=80, Graphik doppelter Dichte, 11 pixels)
630 DATA 27,76,11,128
640 END
```

Bild 1. Das Programm hilft Sonderzeichen zu entwerfen

```
  16  40  68  68  40  16  40  68  68  40  16 ∞
 195 129 165 129 153 129 129 129 129 129  0 Σ
   0   0   2   1   1 126 128 128  64   0   0 ∫
  28  34  34  34  20   8  20  34   0   0   0 α
```

Bild 2. Eingaben und Ergebnisse

Wer suchet, der soll finden

(angeblich)

Zunächst einmal:
All denen, die dieses Bibelzitat lesen, wünsche ich noch ein

Glückliches Neues Jahr 1988,

(welches, wie ich hoffe, inzwischen noch nicht zu alt geworden ist).
Denjenigen, die dies hier nicht lesen, brauche ich das nicht mehr zu wünschen - denn sie *haben* bereits Glück!
Nur für die Unglücklichen, die bei Empfang fremder Disketten sich mit den sogenannten

PDRIVEs

herumquälen, dürfte das folgende vielleicht interessant sein (vorausgesetzt, sie besitzen noch so einen antiquierten Computer wie ich, der mit NEWDOS oder G- resp. H-DOS arbeitet!)

"Als ich noch - der 'Waldbauernbub' war..." hätte ich fast mit *Peter Rosegger* (kein Clubmitglied!) gesagt; also:
"Als ich noch ein Diskothekar war" (in Lauffener Originalton: ein Diskettothekar), habe ich mich oft genug damit herumgequält, eine Diskette, deren PDRIVEs nicht angegeben waren, zum Laufen (diesmal mit nur einem 'f') zu bringen.
(1.Anm.: Dies war aber nicht der Grund für die Aufgabe dieser Aufgabe; ich habe dabei manches gelernt, u.a. eben das, worüber ich hier zu berichten mich anschicke. * Ende der 1.Anm.)
Heute plagt mich das Problem (das übrigens auch von "ID/CMD" keineswegs immer gelöst wird!) nur noch selten. Zum Beispiel, wenn ich versuche, eine Diskette zu lesen, auf der überhaupt nichts drauf ist... O Gott, ist das ein Krampf! Wie erkennt man die Leere einer Diskette?? Ich bedanke mich für die Lehre, die mir hoffentlich jemand darüber erteilt! Es geht kaum etwas über eine solche Leerelehre!

Ein lieber Clubkamerad schickte mir 50 Disketten, die ich mir ansehen durfte, bevor er sie seiner neuen WAA übergab. Ich entschloß mich prompt zu einer Sitzblockade und saß etwa 3 Tage (ohne weggetragen zu werden! Nicht einmal von meiner Frau...)
Als sich die Leseproblematik bei diesem Unternehmen wieder einmal häufte, erinnerte ich mich des hervorragenden Buches von *GROSSER (im RÖCKRATH-Verlag)*, und studierte dort die Seite 4-3 (was soviel wie Seite 3 von Kapitel 4 heißen soll, nicht etwa Seite 4 bis 3! Anscheinend hatte *GROSSER* Nachträge vorgesehen (sind sie erschienen?). Dort ist nachzulesen, wo auf einer ("normalen") Diskette diese verdammten "PDRIVE-Parameter" stehen.
(2. Anm.: Ich halte diese Bezeichnung für unsinnig. Was soll "PDRIVE" anderes sein als die Abkürzung für "Parameters of Drive" ? Also ist 'PDRIVE-Parameter' ein schwarzer Rappe. Ich übersetze das grundsätzlich mit "Drive-Parameter". * Ende der 2.Anm.)

Mit diesem Erwerb unumstößlicher Wahrheiten (danke, großer GROSSER!) gelang es mir dann doch, sämtliche 50 Disks zu lesen - allerdings mit Ausnahme jener, auf denen vermutlich gar nichts stand. SUPERZAP zeigte dennoch einen Sektorinhalt an, den *GROSSER* offensichtlich nicht erfaßt hat...

Da mir z.Z. kein besserer INFO-Beitrag einfällt, möchte ich jenen, die ebenso hilflos vor ihrer Sphinx sitzen, wie einst ich, das Orakel verraten (ohne einen Anspruch auf DELPHI zu erheben) (wo Fräulein SPHINX auch gar nicht sitzt):

Die Drive-Parameter einer Diskette sind im 3.Sektor, d.i. der relative Sektor 2 (DRS 2) gespeichert, und zwar in jeder 16-Byte-Zeile eine Parameter-Gruppe, jeweils für die Laufwerke 0 bis 3 sowie die Pseudo-Laufwerke 4 bis 9.
Dieser Sektor läßt sich "mit Hülfe von SUPERZAP" (jawohl, mir wird ganz feyerlich zumute) bei einer normalen Diskette stets lesen.
Ich gebe hier immer vor, daß die Diskette "normal" sein muß. Es gibt andere, z.B. GENIETEXT, wo das nicht geht. (*ZENDER* hat sie weitgehend kopiergeschützt durch Umfunktionierung wichtiger Strukturen.)
Was die einzelnen Bytes besagen, interessiert in diesem Zusammenhang nur insoweit, als es Auskunft über die gesuchten verfl-, verzwickten Parameter gibt. Und das habe ich in der nachstehenden Tabelle in der Reihenfolge zusammengestellt, die für den forschen Forscher relevant ist.

Auf daß ein jeder schneller finde,
Denn Zeitvergeudung ist 'ne Sünde!

Ka-Jot

* Was TSCRIPS nicht kann - - - *

(Ein Vexierrätsel ?)

Wenn dieses INFO erscheint, steht Ostern vor der Haustür.
Das ist die Zeit, in der das Suchen allgemeine Fröhlichkeit verbreitet; nicht nur unter Kindern.
Drum laßt uns den Kindern gleich tun! (Denn "...der soll finden...")
Es geht um Zeichengestaltung. Aber diesmal heißt der Hase, der die Eier legt, nicht *TSCRIPS*, auch nicht *NEWSCRIPT* oder anderswie.

TSCRIPS kann viel. Die Tastenfolge "@P" erlaubt ein sofortiges Umschalten in den Zeichenmodus, in dem man Zeichen in einer 8x12-Matrix entwerfen oder vorhandene sehr schnell ändern kann - sogar während man einen Text schreibt - so daß auch für den in Arbeit befindlichen Text in aller Kürze beliebige Fantasiegebilde zur Verfügung stehen, die man, wenn sie nicht mehr benötigt werden, auch sofort wieder löschen kann.
Ich halte dies für einen besonderen Vorzug von *TSCRIPS*, weil ich für Mathe-Texte, manchmal aber auch für Briefe, viele Sonderzeichen (manchmal auch "Spaßvögel") brauche, die es im integrierten Zeichenvorrat natürlich nicht gibt.

(Anmerkung: *GENIETEXT* besitzt diesen Modus auch; der ist aber sehr viel umständlicher anzuwenden und zerstört meistens den laufenden Text. Andere Textsysteme mit dieser "facility" - deutsch: "Fazilität" - sind mir nicht bekannt.)

Identifizierung der Drive-Parameter

Die PD's *TI / TD / TC / SPT / TSR / GPL / DDSL / DDGA*

(bzw. für GENIE: *TI / TD / SP / SEK / SWZ / EIB / SBIV / AEIV*)

stehen in DRS 2 in der der jeweiligen rel. Laufwerk-Nr. entsprechenden Zeile; für das Systemlaufwerk 0 in Zeile 0 usw.
Es gilt:

TI ==> Byte 2 in binärer Darstellung interpretieren:

```
Bit 7 gesetzt entspricht: TI = H
 "  6    "         "      TI = K
 "  5    "         "      TI = M

                 { 001 entspricht: TI = A
                 { 011      "      TI = B
Bittriade 4-3-2 = { 100      "      TI = C
                 { 010      "      TI = D
                 { 101      "      TI = E
                 { 000      "       ERROR
Bit  1 + 0 =       Track Stepping Rate
```

Byte E: Wenn Bit 3 gesetzt, dann TI = L

TD ==> Byte F = 00 01 02 03 04 05 06 07
TD = A B C D E F G H

TC ==> Byte 3 gibt die Anzahl Tracks (Spuren) an

SPT ==> Byte 4 gibt die Anzahl Sektoren Pro Track an

TSR ==> Byte C gibt die Track Stepping Rate (Spurwechselgeschw.) an

GPL ==> Byte 5 gibt die Anzahl Granules Pro Lump an

DDSL ==> Byte 0 gibt den "Lump" (Block) an, in dem das Directory beginnt

DDGA ==> Byte 9 gibt die Anzahl der dem Directory zugeteilten Granules an

Disk-Typ: Byte 7 in binärer Darstellung interpretieren:

wenn	Bit:	7	6	4	2	1	0	gesetzt,
dann Disktyp	:	8"	DS	1)	2)	3)		DD
sonst	:	5"	SS	1)	2)	3)		SD

1) wenn gesetzt, dann TI=I, Sektorzählung ab 1, sonst ab 0
2) " " , " TI=L, " " 1, " " 0
3) " " , " TI=K, Spurzählung " 1, " " 0

Byte 1 gibt die Anzahl der Lumps auf der Diskette an.

Kajott

In *TSCRIPS* kann man die selbst gebauten Zeichen auch schnell nach links, rechts, oben, unten verschieben. Rotieren, Dehnen und Spiegeln geht nicht. Wollte man Zeichen z.B. um einen Winkel drehen oder vergrößern/verkleinern, so muß man sie mit *DOTWRITER* entwerfen und mit *"MANIPULATE"* - einem Zusatzprogramm - manipulieren. *DOTWRITER* und *DOTPRINT* erfordern jedoch viel Zeitaufwand. Will man mehr produzieren, als die genannte Fertig-Software hervorbringt (pro-duziert = "vor-führt"), so besinne man sich auf das eigentliche Werkzeug, daß - o Gott, KaJott, jetzt erliegst Du auch schon dem Trend; aber ich lasse es trotz *TSCRIPS*'s wundervoller Editierkunst mal so stehen - also noch einmal: man besinne sich auf das Werkzeug, das solche Kunstwerke ermöglicht: den Heireß-fähigen Matrix-Nadeldrucker (so man hat!) Auch der Meißel des Bildhauers gehorcht nicht nur dem schöpferischen Künstler, sondern jedem Lausbuben, der ihn schwingt und den geduldigen Stein damit verunstaltet...

Verunstalten wir!
Bedienen wir uns also der Bitmuster-Druckroutine selbst, statt sie einem geklauften Textprogramm preiszugeben!
Ich habe den EPSON RX80. Wer einen anderen Drucker hat (natürlich nur einen HRG-fähigen, s.o.), kennt sicher sein entsprechendes Kommando und ändert mein nachstehendes Programm entsprechend (Zeile 360)
Die Bilder werden aus aufeinander folgenden Punktspalten (beim RX80 zu je 9 senkrecht übereinander liegenden Punkten) zusammengesetzt. Jeder Punkt repräsentiert ein Bit; die Bitzählung beginnt unten. Der unterste Punkt ist vom USER (Unsereiner Spricht Englisch, Right?) nicht ansprechbar; der nächste entspricht Bit 0, der oberste Bit 7. Jede Punktspalte wird als Byte codifiziert, daß das sich aus der Addition der gesetzten Bits ergibt. Ein "!" ist also eine Spalte, in der - von ganz unten gezählt - der dritte Punkt (Bit 1) sowie die Punkte 6 bis 9 (Bits 4 bis 7) gesetzt sind. Das ergibt das "Bitmuster" 1111 0010; und dies ist gleich F2Hex = 242dez. So erhält jede Spalte ihren Code, und da wir die Spaltencodes für unsere Bilder selbst konstruieren, können wir ihre Abfolge auch selbst manipulieren. Denn jede Spalte wird in unserem Programm durch eine hierdurch definierte Variable repräsentiert. Es empfiehlt sich, hierfür ein- oder noch besser zweidimensionale Feldvariable zu verwenden, denn diese lassen sich am elegantesten "manipulieren".

Mit nachstehendem Programm wird das Vorgehen verdeutlicht, und zwar an einem Beispiel, das euch etwas er-GOETZ-liches vorführt, wenn ihr es ausdruckt. Wer es nicht gleich erkennt, gehe ins Badezimmer. (Es braucht übrigens weder eine Badewanne noch eine Dusche darin zu sein. Aber die Hardware, die das Vexierbild von Sais entschleiert, ist bestimmt drin... *)

Anmerkung: Da der EPSON die "Characters" 0, 10, 11 und 12 vom TRS80 nicht wunschgemäß akzeptiert, wurde ein Drucker-Hilfstreiber eingebaut. (*ARNULF* hat einen besseren. Aber dieser tut's auch.)

HEFT 23 Februar 1988

Ich bin sicher, ihr spiegelt euch in euren Texten wieder!

* * * Ka - Jot * * *

*) Für das "i" genügte *TSCRIPS*! (p.H.: ï)

```
10 '                Ein "erGOETZlicher" Ausspruch
20 '                (nicht von KaJot Muehlenbein,
30 '      aber widergespiegelt im Februar 1988 in Weinheim)
40 '
50 '   Bitmuster-Codes (Punktspalten)
60 DATA 0, 255, 136, 136, 136, 136, 136, 255, 0     ' A
70 DATA 0, 255, 145, 145, 145, 145, 145, 110, 0     ' B
80 DATA 0, 255, 129, 129, 129, 129, 66, 60, 0       ' D
90 DATA 0, 255, 145, 145, 145, 145, 129, 129, 0     ' E
100 DATA 0, 0, 0, 129, 255, 129, 0, 0, 0            ' I
110 DATA 0, 255, 16, 40, 68, 130, 129, 1, 0         ' K
120 DATA 0, 255, 1, 1, 1, 1, 1, 1, 0                ' L
130 DATA 0, 255, 64, 48, 8, 4, 2, 255, 0            ' N
140 DATA 0, 255, 132, 132, 132, 132, 72, 48, 0      ' P
150 DATA 0, 255, 136, 136, 140, 138, 81, 33, 0      ' R
160 DATA 0, 98, 145, 145, 145, 137, 137, 70, 0      ' S
170 DATA 0, 128, 128, 128, 255, 128, 128, 128, 0    ' T
180 DATA 0, 252, 2, 1, 1, 1, 2, 252, 0              ' U
190 DATA 0, 192, 48, 12, 3, 12, 48, 192, 0          ' V
200 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0                  ' Zwischenraum
210 '       T E X T :
220 DATA1,7,7,4,8,15,5,11,12,15,3,1,11,15
230 DATA3,4,8,6,4,8,15,4,10,7,1,13,2,12,15
240 DATA14,5,4,7,4,8,15,2,7,4,5,2,12,15
250 DATA4,11,15,4,10,11,9,1,10,12
260 GOSUB410                                        ' zum Drucker-Hilfstreiber
270 CLS:CLEAR400:DEFSTRA-Z:DIMX(15),XX(15),C%(53)
280                                 ' Konstruktion der Buchstaben
290 FORJ%=1TO15:FORI%=1TO9
300 READC%:X(J%)=X(J%)+CHR$(C%)
310 NEXTI%
320                                 ' Spiegelung
330 FORK%=9TO1STEP-1
340 XX(J%)=XX(J%)+MID$(X(J%),K%,1)
350 NEXTK%,J%
360 LPRINTCHR$(27);"K";CHR$(221);CHR$(1);' Bitmuster-Modus ein
365 '
370 ' Text aus DATA-Zeilen 220-250 einlesen und drucken :
375 '
380 FORI%=1TO53:READC%(I%):NEXT
390 FORI%=53TO1STEP-1:LPRINTXX(C%(I%));:NEXT
400 END
405 '          Drucker-Hilfstreiber fuer EPSON :
410 B$="21E837CB7E20FC211100397E32E837C9"
420 A=16571:FORP=1TO32STEP2
430 B=ASC(MID$(B$,P,1))-48:IFB>9THENB=B-7
440 T=ASC(MID$(B$,P+1,1))-48:IFT>9THENT=T-7
450 POKEA,B*16+T:A=A+1::NEXTP
460 POKE16422,187:POKE16423,64
470 RETURN
```


Die *"80 MICRO"* hat uns Tandianern ja nicht mehr viel zu bieten. Deshalb greift man gern auf ältere Ausgaben zurück und schaut nach, was dort noch auszuschöpfen ist.
Da ist z.B. in der April-Nummer von 1983 auf Seite 210 ein Artikel von *Arne Rohde*, welcher beschreibt, wie man innerhalb von VISICALC bequem sowohl Eingabekorrekturen vornehmen als auch Drucker-Steuersequenzen eingeben kann.
Während ersteres eine nicht allzu wichtige facility ist, über deren Wert man streiten kann - denn oft ist es nicht weniger bequem, die Eingabe in ein Feld schlicht zu wiederholen, als eine Korrektur an der fehlerhaften Eingabe mit dieser trickreichen Methode durchzuführen - bietet die zweite die häufig erwünschte Möglichkeit, den Ausdruck eines VISICALC-Rechenblattes individuell zu formatieren, z.B. durch Unterstreichen, Kursivschrift, Fettschrift etc. Besonders häufig benutze ich diese "facility", um den Drucker von VISICALC aus auf ELITE-Schrift oder Schmalschrift einzustellen, wenn es sich wegen unvorhergesehen größerer Spaltenanzahl, die mit PICA nicht in die Zeile paßt, als notwendig herausstellt oder wenn ich vor dem Laden von VISICALC vergessen hatte, den Drucker auf irgendeinen erwünschten besonderen Modus einzustellen.
Auch Grafik-Zeichen können mit dieser Utility ausgegeben werden (falls der Drucker dazu fähig ist).
Bei der Eingabe von Drucker-Steuersequenzen bedient das Programm sich eines Trickes: Da Zeichen wie *ESC, @, FF (Form Feed), SI (Schmalschrift ein), DC2 (Schmalschrift aus), SO (Breitschrift ein), DC4 (Breitschrift aus), BEL* usw. nicht von der Tastatur aus direkt eingegeben werden können, wurde eine Umleitung programmiert: Auf ← (Linkspfeil) wird das eingegebene Zeichen in der ASCII-Tabelle um zwei Spalten nach links verschoben, auf → (Rechtspfeil) um acht Spalten nach rechts.

Ein Beispiel für den EPSON RX 80 und Verwandte:
Wenn einem erst unmittelbar vor dem Ausdruck des Rechenblattes einfällt, daß der Druck *fett* sein soll, kann man die entsprechende Steuersequenz wie folgt an den Drucker ausgeben:
Nachdem die bekannte Zeichenfolge / P P eingegeben wurde, erscheint in der Menue-Zeile:
LOWER RIGHT, "SETUP, &, -, +
Statt nun die "untere rechte Ecke" einzugeben, drückt man zunächst die Anführungsstriche ("). Dadurch gelangt man in den Modus der Druckersteuerung. Die Fettdruck-Einstellung lautet *ESC CHR$(69)* bzw. *CHR$(27);"E"*. Gibt man nun den Linkspfeil ← ein, so wird der ASCII-Wert des nächsten Zeichens um 32 vermindert. Um das Zeichen ESC = CHR$(27) zu erhalten, lautet also die Eingabe <← ;>. Denn der ASCII-Wert des Semikolons ist 59. Da 59-32=27, empfängt der Drucker durch diese Eingabe das richtige Signal, das ihn "hellhörig" macht, nämlich 'ESC' (ESCAPE = 'Fluchtsequenz'). Gibt man anschließend 'E' ein, so reagiert er dann entsprechend durch Fettdruck, sobald der Druckbefehl durch 'ENTER' und Eingabe der "unteren rechten Ecke" abgeschlossen ist. Stattdessen kann man aber beliebig oft wieder das Anführungszeichen eingeben, um weitere Steuersequenzen anzuschließen. Will man z.B. die drei Effekte :

ELITE (ESC M), Kursivschrift (ESC 4) und Unterstreichung, so lautet die Tastenfolge:

/ P P " ← ; M ENTER " ← ; 4 ENTER " ← ; - 1 ENTER

Danach weiter wie gehabt: "Untere rechte Ecke" usw. * Die Aufhebung dieser Befehle erfolgt entsprechend.
Mit dem Rechtspfeil (→) kann man zu anderen Zeichen gelangen, die der Drucker zwar bereithält, die von der Tastatur aus aber sonst nicht erreichbar sind. Er bewirkt die Addition von 128 zum ASCII-Wert des eingegebenen Zeichens.
(Ein sehr ähnliches "Umlenkungsverfahren" verwendet *ZENDER* in seinem *GENIETEXT*.)

Man lasse sich nicht beirren (be-irren; nicht beir-ren), wenn auf Drücken des Linkspfeils ein 'H' und statt des Rechtspfeils ein 'I' erscheint. Diese Buchstaben werden durch die anschließende Eingabe weggeschoben. Was sie bedeuten, habe ich nicht herausgefunden. Wahrscheinlich dienen sie nur als Bestätigung, daß man den richtigen Pfeil gedrückt hat. Oder als Gelächter über den User, wenn er zuviele ←→ ohne weitere Eingaben hintereinander drückt. Denn dann antwortet sein Dialogpartner eben 'HIHI'!

Eine Kurzanleitung findet sich unten (oder nebenan).
Ich hätte auch das Source-Listing assembliert ausdrucken können. Doch da das sowie niemand abtippt*) und ich das Programm gern jedem Einsender einer Diskette mit Rückporto und Parameterangabe (möglich sind alle) zur Verfügung stelle, unterläßt dies:

Euer *K*a-J*o*t*

*) 3 (drei) Seiten (Anlage!)

Kurzanleitung für VCMOD/CMD

ACHTUNG: VCMOD/CMD läuft nur, wenn auf VC/CMD zugegriffen werden kann!

1) EDIT-Mode

; ↓	Replace-Mode einschalten. - Dann: Insert-Mode und zurück (durch Vorsetzen von " kann aus einer numerischen Eingabe ein Label gemacht werden)
CLEAR	Zeichen löschen (durch Löschen von " kann aus einem Label ein numerischer Eintrag gemacht werden)
BREAK	EDIT-Mode ohne Änderung verlassen
ENTER	Rückkehr in den normalen Eingabe-Modus. (Danach ENTER oder Pfeiltaste drücken)

2) Druckersteuerung

Tastenfolge: / P P " ←					
;	ESC (Prefix)				
/	Schmalschrift	ein	2	aus	
.	Breitschrift	ein	4	aus	
; - 1	Unterstreichg.	ein	- 0	aus	
; 4	Kursivschrift	ein	5	aus	
; E	Fettdruck	ein	F	aus	
; M	ELITE	ein	P	aus	
; @	Drucker RESET				

u.s.w.

Anm.1: ← schiebt das eingegebene Zeichen in der ASCII-Tabelle zwei Spalten nach links;
→ schiebt das eingegebene Zeichen 8 Spalten nach rechts.

Anm.2: Mehrere Steuerbefehle können, durch ENTER getrennt, hintereinander eingegeben werden.
Sie werden durch Anführungszeichen <"> eingeleitet.

Kleine Hexen bevorzugt

Nur noch unverbesserliche Nostalgiker im Club benutzen in selbstquälerischer Mühwaltung DISASSEM, um für EDTASM eine Source zu erzeugen. Nun ja, ein Indianer kennt keinen Schmerz. Allgemein hat aber sich die Einsicht verbreitet, daß man das mit DSMBLR für ZEUS tun sollte.

DSMBLR erzeugt 16 Bit-Labels, die mit M beginnen, gefolgt vom Zahlenwert in Hex. Bei 8 Bit-Werten wird entweder die ASCII-Entsprechung zwischen Hochkommata gesetzt oder die Hexzahl mit einem H dahinter. Schönere Labels und Zahlenoperanden kann man sich nicht wünschen.

Der Haken: DSMBLR kennt nur Großbuchstaben. So kann eine Zeile z. B. so aussehen:

```
MFF00   LD      A,0A4H
```

Natürlich macht es wenig Spaß, mit ZEUS die entsprechenden Labels mit dauerhaft niedergehaltener Shift-Taste für die Majuskeln einzutippen. Deshalb ist die erste Aktion nach einer Disassembly immer, mit der C-Funktion von ZEUS Kleinbuchstaben zu erzeugen:

```
cA,a - cB,b - cC,c - ... - cH,h - cM,m
```

Wozu ist ein Computer da, wenn nicht zum Automatisieren solch stumpfsinniger Vorgänge?

Zuvor hatte ich in DSMBLR nach den Stellen gespürt, wo die Großbuchstaben erzeugt werden. M und H konnten bereits dort in m und h verwandelt werden (s. u.). Aber die Hex-Ziffern werden leider auf die gleiche undurchsichtige Weise erzeugt wie durch das DOS in der Routine an 4068h. Hier konnte ich nichts manipulieren, weil ich den äußerst krausen (aber kurzen und schnellen) Algorithmus nicht durchschaue.

Das hier vorzustellende Programm erledigt das. Es geht ziemlich simpel; da wird bei den Hexziffern einfach nur das Bit 5 gesetzt. Daher verweise ich zur nicht allzu komplizierten Programmlogik nur auf die reichlichen Kommentare. Etwas schwieriger war, das Programm herausfinden zu lassen, was eine Hex-Ziffer ist und was z. B. ein Zeichen aus einem Mnemonic.

Eine DSMBLR-Source hat als mögliche Zeichen oberhalb ASCII 30h nur die Ziffern (bis F), H, M und die Großbuchstaben, aus denen sich die Mnemonics zusammensetzen. Daher konnten die nunmehr kleinen Buchstaben m und h für die Suche nach einem Label oder einer Hex-Zahl herhalten. Eine Verwechslung mit dem M aus dem Befehl JP M,nn oder dem H von PUSH qq war nun nicht mehr möglich.

Um dies zu erreichen, müssen in DSMBLR/CMD folgende Änderungen durchgeführt werden: Im filerelativen Sektor 7 wird das sektorrelative Byte 01h von 4Dh (M) in 6Dh (m) geändert. Im Sektor 6 wird das Byte A4h und im Sektor 8 das Byte 6Ah von 48h (H) auf 68h (h) gesetzt. Möglicherweise sind durch fleißiges Patchen nicht bei allen in Umlauf befindlichen Versionen die Diskadressen dieselben. Dennoch lassen sich die zu ändernden Stellen leicht finden; es sind die einzigen, die LD A,'M' (3E-4Dh) und LD A,'H' (3E-48h) lauten. Daher sind sie leicht zu finden, z. B. mit FED.

Um die Konversion von Groß- in Kleinbuchstaben abzufahren, wird das Programm (bei mir heißt es CONVERT/CMD) mit seinem Namen, gefolgt vom Namen des zu modifizierenden Source-Files aufgerufen:

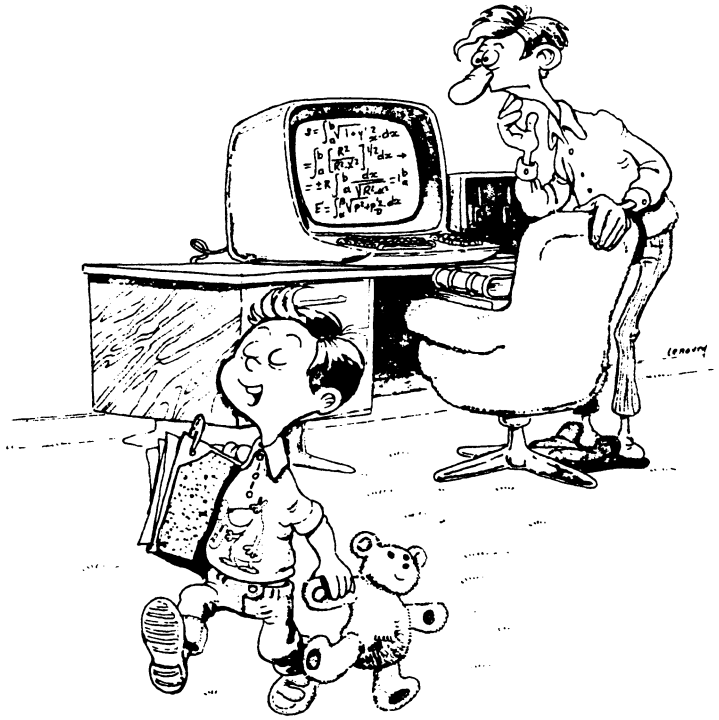
```
CONVERT,SOURCE<CR>
```

Wenn keine Typbezeichnung folgt, wird automatisch /SRC angenommen. Das ist die Extension, die DSMBLR generiert, und die ZEUS als gegeben unterstellt, wenn nichts anderes eingegeben wurde.

Arnulf Sopp


```
                        00001 ;       Programm zur Konversion von Groß- in Kleinbuchstaben
                        00002 ;       in Sources, die von DSMBLR erzeugt wurden
                        00003 ;       (notwendige Änderung in DSMBLR: M -) m und H -) h)
                        00004 ;       Arnulf Sopp 1988
                        00005
5200                    00006         ORG     5200h
5200 CDD54C             00007 start   CALL    4cd5h           ;HL auf Dateinamen im Befehlsstring
5203 2844               00008         JR      Z,synterr       ;falls keine Datei angegeben
5205 EB                 00009         EX      DE,HL           ;DE (- Dateiname
5206 215C52             00010         LD      HL,extensn      ;Default-Extension /SRC
5209 CD7344             00011         CALL    4473h           ;anhängen, falls keine Ext. angegeben
520C EB                 00012         EX      DE,HL           ;HL (- Dateiname
520D 118044             00013         LD      DE,4480h        ;Adresse des DOS-FCB
5210 CD1C44             00014         CALL    441ch           ;Namen in den FCB übertragen
5213 2036               00015         JR      NZ,errexit      ;falls Fehler (z. B. kein gültiger Name)
5215 210053             00016         LD      HL,5300h        ;Sektorpuffer
5218 CD2444             00017         CALL    4424h           ;File eröffnen
521B 202E               00018         JR      NZ,errexit      ;falls Fehler aufgetreten
521D EB                 00019         EX      DE,HL           ;HL (- FCB-Adresse
521E 23                 00020         INC     HL              ;FCB +1. Bit 6: EOF-Wert nicht nach jedem
521F CBF6               00021         SET     6,(HL)          ;Schreibzugriff auf den NEXT-Wert setzen!
5221 2B                 00022         DEC     HL              ;zurück auf FCB +0
5222 EB                 00023         EX      DE,HL           ;Register zurücktauschen
5223 CD3644             00024 loop0   CALL    4436h           ;einen Sektor einlesen
5226 2805               00025         JR      Z,seekm         ;falls kein Fehler aufgetreten ist
5228 FE1C               00026         CP      1ch             ;Fehlercode "Ende der Datei angetroffen"?
522A C8                 00027         RET     Z               ;fertig, falls ja
522B 181E               00028         JR      errexit         ;sonst anderer Fehler: anzeigen und raus
522D E5                 00029 seekm   PUSH    HL              ;Pufferadresse retten
522E 010001             00030         LD      BC,0100h        ;Zähler für CPIR
5231 C5                 00031         PUSH    BC              ;brauchen wir noch für CPDR
5232 3E6D               00032 loop1   LD      A,'m'           ;Suchbyte für Labels
5234 EDB1               00033         CPIR                    ;'m' finden
5236 201F               00034         JR      NZ,seekh        ;falls Suche erfolglos beendet
5238 0604               00035         LD      B,4             ;4 Stellen des Labels
523A 7E                 00036 loop2   LD      A,(HL)          ;Zeichen laden
523B FE41               00037         CP      'A'             ;Hex-Ziffer?
523D 3802               00038         JR      C,ciphern       ;falls Dez-Ziffer
523F CBEE               00039         SET     5,(HL)          ;Klein- aus Großbuchstaben machen
5241 23                 00040 ciphern INC     HL              ;auf nächste Ziffer stellen
5242 0D                 00041         DEC     C               ;Bytezähler für CPIR nachstellen
5243 280E               00042         JR      Z,seekh         ;dort weiter, falls Sektor überschritten
5245 10F3               00043         DJNZ    loop2           ;bis gegf. 4 Zeichen umgewandelt
5247 18E9               00044         JR      loop1           ;das nächste Label mit 'm' suchen
5249 3E34               00045 synterr LD      A,34h           ;Fehlercode "Syntax o. Trennzeichen ..."
524B F680               00046 errexit OR      80h             ;Bit 7 setzen für Rücksprung aus 4409
524D C30944             00047         JP      4409h           ;Fehler anz., evtl. zum Aufrufer zurück
5250 53                 00048 extensn DM      'SRC'           ;Default-Extension für Source-Files
5253 2B                 00049 seekh   DEC     HL              ;Zeiger auf Ende des Sektorpuffers 53FF
5254 C1                 00050         POP     BC              ;Zähler 0100 für CPDR
5255 3E68               00051 loop3   LD      A,'h'           ;Kenner von Hex-Zahlen
5257 EDB9               00052         CPDR                    ;nach 'h' suchen
5259 2019               00053         JR      NZ,wrsect       ;Skt. zurückschr., falls Suche erfolglos
525B 0604               00054         LD      B,4             ;max. 4 Stellen der Zahl
525D 7E                 00055 loop4   LD      A,(HL)          ;Zeichen laden
525E FE30               00056         CP      '0'             ;eine Ziffer?
5260 3004               00057         JR      NC,hexciph      ;weiter, falls nicht TAB, CR oder Komma
5262 0600               00058         LD      B,00h           ;für restlichen CPDR-Zähler 00xx
5264 18EF               00059         JR      loop3           ;nächstes 'h' suchen
5266 FE41               00060 hexciph CP      'A'             ;Hex-Ziffer?
5268 3802               00061         JR      C,cipherh       ;falls Dez-Ziffer
526A CBEE               00062         SET     5,(HL)          ;Klein- aus Großbuchstaben machen
526C 2B                 00063 cipherh DEC     HL              ;eine Stelle zurück auf nächste Ziffer
526D 0D                 00064         DEC     C               ;Sektorpuffer schon unterschritten?
```

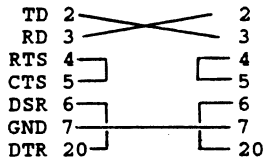
```
526E  2804    00065          JR     Z,wrsect       ;falls ja
5270  10EB    00066          DJNZ   loop4          ;bis gegf. 4 Zeichen umgewandelt
5272  18E1    00067          JR     loop3          ;nächstes 'h' suchen
5274  E1      00068 wrsect   POP    HL             ;Pufferadresse
5275  CD4544  00069          CALL   4445h          ;FCB auf alten Sektor zurückpositionieren
5278  20D1    00070          JR     NZ,errexit     ;falls Fehler aufgetreten
527A  CD3944  00071          CALL   4439h          ;Sektor zurückschreiben
527D  20CC    00072          JR     NZ,errexit     ;falls Fehler
527F  18A2    00073          JR     loop0          ;sonst nächsten Sektor bearbeiten
5200          00074          END    start          ;dort Einsprung

00000 Fehler

cipherh 526C   cipherm 5241   errexit 524B   extensn 5250   hexciph 5266   loop0  5223
loop1   5232   loop2   523A   loop3   5255   loop4   525D   seekh   5253   seekm  522D
start   5200   synterr 5249   wrsect  5274
```



Tut mir leid, Papi – Dein Computer ist zu primitiv für meine Hausaufgaben.

Datenübertragung / Model 4P als Terminal

Durch die bereits serienmäßig in das Model 4P (Model 4 nachrüstbar) eingebaute RS 232 Schnittstelle und dem unter TRSDOS 6.x mitgelieferten Kommunikationsprogramm COMM ist das Model 4 gut gerüstet für Daten(fern)übertragung.
Datenfernübertragung mittels Akustikkoppler war mir bereits bekannt. Durch einen mir kürzlich zugelegten frei steckbaren RS 232-Stecker war es mir nun aber möglich, einen Null-Modem-Stecker herzustellen und so die Möglichkeiten einer Direktverbindung zwischen zwei Computern zu testen. Der Null-Modem-Stecker läßt sich auch mit wenig Aufwand selbst herstellen. Allgemein üblich ist nachstehende Verbindung:



Verbindet man nun die beiden Geräte, so lassen sich nicht nur Daten übertragen, was bei zwei verschiedenen Systemen oft die einzige Möglichkeit darstellt, einen Datenaustausch vorzunehmen. Eine weitere Möglichkeit ergibt sich auch durch die Benutzung des einen Geräts als Hauptcomputer und des anderen als Ein- oder Ausgabeterminal. Auf diese Art und Weise müßte man ein für uns nicht kompatibles System (z.B. MS-DOS) mit einem Model 4 bedienen können. Oder man setzt das Model 4 einfach für einen nicht vorhanden Monitor/Tastatur eines Fremdsystems ein.

Nachfolgend nun einige Verbindungsmöglichkeiten, die ich mal versucht habe. Als zweites Gerät hatte ich ebenfalls ein Model 4, was natürlich eine einfache Installierung zuließ. Große Schwierigkeiten dürften sich jedoch bei anderen Systemen ebenfalls nicht ergeben.

Zunächst muß bei beiden Geräten der Treiber für die RS 232 Schnittstelle installiert werden. Dies geschieht durch den Befehl

```
SET *CL TO COM/DVR
```

Anschließend werden die Parameter für die Datenübertragung eingegeben (Baud, Word, Parity). Gute Erfolge hatte ich mit folgender Einstellung:

```
SETCOM (BAUD=9600,WORD=8,PARITY=NO)
```

Bei einem Versuch, die Übertragung mit 19200 Baud vorzunehmen, stieg das System aus.
Nun noch kurz der Unterschied zwischen den zu benutzenden Befehlen LINK und ROUTE. LINK koppelt ein Gerät an das andere an, während ROUTE ein Gerät auf ein Ersatzgerät umleitet.
Der Monitor wird als "DO", die Tastatur als "KI" und die RS 232 Schnittstelle als "CL" bezeichnet.
Durch COMM *CL wird das Kommunikationsprogramm von TRSDOS aus aufgerufen.
Bei den nachstehenden Beispielen ist davon auszugehen, daß es sich bei Gerät 2 um das Model 4/4P handelt. Gerät 1 stellt das evtl. Fremdsystem dar.

Bsp. 1:

```
Gerät 1:        LINK *DO *CL
Gerät 2:        COMM *CL
```

Mit diesem Aufbau wird Gerät 2 im Prinzip nur als Zweit- oder Ersatzmonitor benutzt. Das Gerät 1 dient als Hauptcomputer, während Gerät 2 nur die Daten von Gerät 1 empfängt, jedoch keine Daten an Gerät 1 senden kann.

Bsp. 2

```
Gerät 1:        LINK *KI *CL
Gerät 2:        COMM *CL
```

Gerät 1 ist wieder der Hauptcomputer. Dieses Mal ist anstatt des Monitors jedoch die Tastatur angekoppelt. Die Befehle für Gerät 1 können nun noch zusätzlich/ersatzweise über Gerät 2 eingegeben werden. Sollen die eingegebenen Befehle auch auf Gerät 2 angezeigt werden, sind innerhalb COMM noch die Anweisungen DUPLX ON und ECHO ON einzugeben. Befehle an Gerät 1 können von beiden Geräten eingegeben werden.

Bsp. 3

```
Gerät 1:        LINK *KI *CL
                LINK *DO *CL
Gerät 2:        COMM *CL
```

Wie Bsp. 2, nur daß jetzt das Ergebnis des an Gerät 1 gesandten Befehls auch auf Gerät 2 angezeigt wird.

Bsp. 4

```
Gerät 1:        LINK *KI *CL
                ROUTE *DO *CL
Gerät 2:        COMM *CL (evtl. HNDSHK ON)
```

Hier wird die Bildschirmausgabe auf Gerät 1 unterdrückt und direkt an Gerät 2 geleitet. Gerät 1 wird nun direkt von Gerät 2 aus bedient. Dieser Aufbau läßt ein Arbeiten mit Gerät 1 zu, wenn dort z.B. Tastatur und Monitor nicht vorhanden sind. Voraussetzung wäre dann allerdings, daß beim Booten von Gerät 1 die o.a. Einstellung automatisch vorgenommen wird.

Bei der Bildschirmausgabe auf Gerät 2 ist zu beachten, daß die Daten seriell, also Zeile für Zeile, ausgegeben werden. Ursprünglich direkt adressierte Bildschirmpositionen werden also auf Gerät 2 direkt hintereinander ausgegeben.

Weiterhin ist noch zu beachten, daß die Datenausgabe von Gerät 1 über das Betriebssystem erfolgen muß. Teilweise werden die auszugebenden Daten unter Umgehung des Betriebssystem durch direkte Adressierung ausgegeben bzw. eingegeben. Öfters findet man dies bei MS-DOS unter 8088-Proz., um so das System etwas schneller zu machen. Auch bei der Grafikausgabe kann es zu Problemen führen.
Hauptsächlich einsetzen lassen sich diese Verbingungen, um Betriebssysteme zu bedienen. Für die Bedienung von Anwenderprogrammen ist diese Verfahrensweise nur bedingt geeignet.

Es wäre interessant zu wissen, bei welchen Systemen sich das Model 4 als Terminal einsetzen läßt. An einem Erfahrungsaustausch wäre ich interessiert.

(Den oben erwähnten RS-232-Stecker gibt es für DM 18,25 DM bei Westfalia Technica, 5800 Hagen. Die Signale sind durch Brücken beliebig zu vertauschen).

Klaus Hermann

Druckausgabe für COBOL-Programme

Der Schreibaufwand für COBOL-Druckausgaben ohne Report-Generator ist auf Grund der fehlenden expliziten Adressierung und Längenangaben sehr groß. Jede Druckzeile muss in der Form

```
01  Druckzeile1.
    02  FILLER                 PIC  X.
    02  LAUFENDE-NR            PIC  X(5).
    02  FILLER                 PIC  X.
    02  NAME                   PIC  X(20).
    02  FILLER                 PIC  X.
    02  VORNAME                PIC  X(20).
    02  FILLER                 PIC  X.
        .
        .
```

definiert werden. Müssen viele unterschiedliche Druckzeilen programmiert werden, so ist der Schreibaufwand nicht unerheblich.

Auf folgende Weise läßt sich hier Abhilfe schaffen. Man schreibt nachstehendes COPY-Element ab und speichert sich dieses unter einem beliebigen Namen (z.B. DRUCK).

```
    01  DRUBER                 PIC  X(80).
    01  D001 REDEFINES DRUBER  PIC  X(01) JUSTIFIED RIGHT.
    01  D002 REDEFINES DRUBER  PIC  X(02) JUSTIFIED RIGHT.
    01  D003 REDEFINES DRUBER  PIC  X(03) JUSTIFIED RIGHT.
        .
        .
    01  D080 REDEFINES DRUBER  PIC  X(80) JUSTIFIED RIGHT.
```

Dieses COPY-Element kann nun mit der COBOL-Anweisung 'COPY DRUCK' für alle zukünftigen Programme benutzt werden. Die langweilige Definition des oben stehenden Ausgabebereichs ist somit nur eine einmalige Sache.

Das Füllen des Druckbereichs erfolgt nun von RECHTS nach LINKS. Vor Bearbeitung des Ausgabebereichs wird dieser gelöscht. Wenn als ein auszugebender Wert auf die Schreibposition 60-63 plaziert werden soll, so erfolgt die übertragung mit (es muß der Name der höchsten Druckstelle angegeben werden)

z.B.

```
        MOVE SPACE TO DRUBER.
        MOVE "TEXT" TO D063.
        MOVE "UEBERSCHRIFT" TO D037.
        WRITE ZEILE FROM DRUBER AFTER 1.
```

Viel Spaß

Werner Förster

Vergleichstest

Vorgeschichte

Wie ihr im letzten Info lesen konntet, habe ich ein Model 100 in Besitz, auf dem viele meiner Briefe und Texte (so z.B. dieser) entstehen. Zum Erstellen der Texte kann man den 100'ter gut verwenden, zum Drucken derselben taugt das Computerchen allerdings nicht. Daher schiebe ich die Geistesergüsse nach dem tippen über die RS232-Schnittstelle auf das Model 4p. Hier werden sie mittels TSCRIPS formatiert und gedruckt.

Störend wirkt sich dabei aus, daß die Umlaute des Model 100 (selbstverständlich) an einer anderen Stelle im Zeichensatz liegen, als die des 4p. Dieses Problemchen ist aber mit einem kleinen Konvertierungsprogramm sehr leicht zu lösen. Dieses Programm entstand zunächst in BASIC und glänzte vor allem durch seine Lahmheit. Aus diesem Grunde schrieb ich, in einer ruhigen Minute, ein Programm in Pascal.
Unter NewDOS laufen auf dem TRS 80 mehrere Pascal-Dialekte. Den ersten Versuch startete ich mit PASCAL 80. Nachdem das relativ gut klappte, kam ich auf die wahnwitzige Idee, das Programm auch in PASCAL 4.5 und Alcor-PASCAL zu realisieren und das ganze als eine Art Vergleichstest zu gestalten.

Als nach mehreren Stunden (teilweise hatte ich keine Dokumentation zu den PASCAL-Compilern) alle drei Programme liefen, wollte ich das Problem auch noch in RPNL umschreiben. Dazu hatte ich zwar eine wirklich gute Dokumentation, war aber trotz mehrstündiger verzweifelter Versuche nicht in der Lage, ein funktionsfähiges Programm zu erstellen. Als ich dann auf dem Nordlichtertreffen Kurt Müller traf, der ja RPNL unter NewDOS nutzbar machte, verdonnerte ich ihn sofort dazu, ein entsprechenden Programm in "seiner" Sprache zu erstellen. Last but not least wurde Gerald Schröder damit beauftragt, das Umlauteproblem in Assembler zu lösen.

Im Anschluß könnt ihr also zunächst einmal den Vergleichstest zwischen den drei PASCAL-Compilern und im Anschluß daran die Erläuterung der Problemlösung in RPNL und Assembler lesen. Ich wünsche euch viel Spaß bei der Lektüre und hoffe, ihr werdet dadurch dazu angeregt, einmal ein eigenes Problem in PASCAL, RPNL oder Assembler statt in BASIC zu lösen.

Pascal 80

Pascal 80 erinnerte mich bei der ersten Benutzung an Turbo-Pascal, weil es, genau wie dieser wohl bekanntest Pascal-Compiler unter CP/M, Editor, Compiler und Runtimelibary in einem Programm vereint. Nach dem Start erscheint ein Menü mit den Optionen:

E - Editor
Q - Quit to DOS
K - Kill (Clear) Editor
C - Compile Program in Editor
R - Run Program in Editor
S - Save Program in Editor
L - Load Program to Editor
A - Append Text to Editor
W - Write Object Code to Disk
X - Execute Program from Disk.

Der Editor ist zwar bei weitem nicht so komfortabel wie der, den man von Turbo gewohnt ist, aber im wesentlichen läßt sich gut damit arbeiten. Das Compilieren geht recht flott vonstatten. Dabei wird der Quelltext auf dem Bildschirm gelistet und, falls aufgetreten, ein Fehler angezeigt. Wenn man nach einem Fehler wieder in den Editor geht, befindet sich der Cursor in der Zeile, in der der Fehler aufgetreten ist!

Wurde ein Quelltext fehlerfrei übersetzt, kann man das entstandene Programm mit 'R' direkt vom Menü aus starten. Will man es als /CMD-File abspeichern, tritt eine deutliche Schwäche des Sprachsystems zutage. Um das zu tun, muß man nämlich folgende Schritte durchlaufen:
Quelltext compilieren und Objectcode abspeichern
PASCAL verlassen
AUTHCODE/CMD starten
PASCAL starten
AUTHOR/SRC laden, compilieren und starten
mit der Option 'X' wird dann das gewünschte Programm als /CMD-File auf Diskette geschrieben.
Diese Procedur ist recht umständlich und ziemlich zeitaufwendig. Die Programme AUTHCODE/CMD und AUTHOR/SRC dienen hier dazu, den Objectcode, zusammen mit der Runtimelibary (getrennt von Editor und Compiler) abzuspeichern. Einen richtigen Linker, wie er bei anderen PASCAL-Systemen üblich ist, gibt es hier nicht. Das erklärt auch die ungewöhnliche Größe des entstehenden Programms, in dem immer das gesamte Laufzeitsystem vorhanden ist, unabhängig davon, was gebraucht wird und was nicht!

Auffällig an Pascal 80 ist die Möglichkeit der Verwendung von INCLUDE- und CHAIN-Files, wobei auch diese beiden Funktionen an Turbo-Pascal erinnern. Mit INCLUDE kann man Quelltexte, die in einer Diskettendatei stehen, whärend des Compilierens in den im Editor befindlichen Sourcecode einbinden. CHAIN ähnelt den Overlays von T.-P. und erlaubt es, separat compilierte Objectdateien aus einem laufenden Programm heraus aufzurufen.

Abschließend wäre zu sagen, daß man Pascal 80 sein Alter (1981) nicht ansieht und nur manchmal, vor allem bei Diskoperationen, daran erinnert wird, daß es urspünglich für TRSDOS geschrieben war (aber unter NewDOS sowohl auf dem Model 1 als auch auf dem 3/4/4p lauffähig ist). Das mit 60 Seiten nicht allzu umfangreiche Handbuch erklärt die Möglichkeiten, Erweiterungen (gegenüber Standard-Pascal) und auch die Einschränkungen des Systems sehr gut.

Pascal 4.5

Pascal 4.5 besteht, im Gegensatz zu Pascal 80 oder Turbo-Pascal, aus drei Komponenten:
Editor -> PASEDIT/CMD
Compiler -> PASCT/CMD
Laufzeitsystem -> PASCAL/CMD.
Das mag den BASIC-Freak zunächst einmal abschrecken, ist aber bei näherem hinsehen (und einigen Testläufen) gar nicht so schlimm.

PASEDIT ist ein sehr komfortabler Editor, der mich allerdings beim ersten Diskfehler (Schreibversuch auf schreibgeschützte Diskette) fürchterlich erschreckt hat. Nach Auftreten eines DOS-Fehlers landet man nämlich grundsätzlich im Betriebssystem. Welch ein Glück, daß dabei der soeben erstellte Quelltext nicht zerstört wird, sondern bei erneutem Aufruf von PASEDIT restauriert werden kann. Statt des systemeigenen Editors kann man übrigens jederzeit auch ein Textverarbeitungssystem benutzen, solange es in der Lage ist, ASCII-Files auf Diskette abzulegen. Im Prinzip ist also von SCRIPSIT bis LESCRIPT jeder Editor verwendbar!

Nachdem man den Quelltext erstellt hat, steht man vor dem Problem, den Compiler aufrufen zu müssen. Manchem, der sich noch nie mit einem Compiler herumgeschlagen hat, wird diese Formulierung seltsam vorkommen, aber der Compileraufruf ist tatsächlich ein nicht zu unterschätzendes Problem.

Schließlich muß man dem Compiler (ebenso wie manchen Linkern) zusätzlich zum Namen des zu übersetzenden Programms auch noch allerhand weitere Informationen geben (wer sich einmal mit BASCOM herumgeschlagen hat, wir ein Lied davon singen können). PASCT geht mit dem Programmierer da sehr viel sanfter um. Er fragt nicht nur nach dem Namen für Ein- und Ausgabedatei sondern auch nach den gewünschten Optionen, und davon gibt es immerhin sieben.

Weniger gut finde ich die Art, wie der Compiler Fehler anzeigt. Statt Klartextfehlermeldungen, wie es sich in der heutigen Zeit wohl gehört, bekommt man einen Fehlercode (z.B. Error 104) an den Kopf geworfen und muß sich aus einer Liste den Fehlergrund heraussuchen. Und dann beginnt das Spielchen, das einem die Freude an Pascal (und auch an anderen Compilersprachen) vermiesen kann:

1. Quelltext erstellt und abgespeichert
2. Compiler aufrufen
3. Fehlercode notieren (dabei vergißt man mit an Sicherheit grenzender Wahrscheinlichkeit, sich die Zeile zu notieren, in der der Fehler aufgetreten ist!)
4. Editor aufrufen und Fehler beheben

und auf ein neues!

Auch Pascal 4.5 hat keinen Linker, vielmehr kann man mit dem Programm PASCAL/CMD die als Objectcode abgespeicherten Programme starten. Um eigenständige /CMD-Files zu erzeugen, muß man den compilierten Objectcode mittels des DOS-Befehls APPEND an die Datei PASCAL/OBJ anhängen. Zuvor sollte man sich natürlich von PASCAL/OBJ ein Backup angefertigt haben!!! Obwohl sich Pascal 4.5 in diesem Punkt nur in der Verfahrensweise, nicht aber im Prinzip von Pascal 80 unterscheidet, entstehen erstaunlich kleine /CMD-Programme. Weiterhin hilft einem das Programm PASJCL/CMD, welches auch im Quellcode vorliegt, beim Erstellen von lauffähigen Programmen, indem es nach verschiedenen Abfragen (Name des zu übersetzenden Quelltextes, Compileroptionen etc.) eine /JCL-Datei anlegt und mit ihr sowohl den Compiler startet als auch das "Binden" zum fertigen Programm erledigt!

Die Möglichkeiten von INCLUDE- und CHAIN-Dateien fehlen bei Pascal 4.5, dafür glänzt es aber durch eine sehr gute Zusammenarbeit mit NewDOS und ein paar sehr brauchbare Sonderbefehle. So kann man mit MOVEDN und MOVEUP Speicherbereiche verschieben (ähnlich dem Z80-Befehlen LDDR und LDIR), PEEKen und POKEn und, was sich bei meinem Problem als besonders nützlich erwiesen hat, mit TITLE(file) Dateinamen direkt von der Tastatur einlesen!

Die Anleitung zu Pascal 4.5 ist noch spärlicher ausgefallen, als die Pascal 80-Anleitung, durch die klarere Gliederung findet man sich aber sehr schnell zurecht. Etwas schwach finde ich die Erklärung der Befehle, die nicht zum Pascal-Standard gehören (z.B. TITLE usw.). Hier hilft nur das Studium der mitgelieferten Quelltexte (PASJCL/PAS) und viel probieren!

Alcor-Pascal

Eigentlich dürfte ich mich über dieses Sprachsystem gar nicht äußern, da ich zu meinen Versuchen damit keinerlei Anleitung vorliegen hatte. Daraus resultierten natürlich viele Fehlschläge und eine gewisse Antipathie gegen diesen Compiler. Trotzdem möchte ich wenigstens einige Worte darüber verlieren, jedoch an dieser Stelle schon bemerken, daß meine Aussagen rein subjektiv sind!

Im Gegensatz zu den Editoren von Pascal 80 und 4.5 kann man mit ED/CMD, dem Alcor-Editor, ohne Anleitung überhaupt nichts anfangen. Selbst heftigem und langwierigen Herumprobieren wiedersetzt er sich erfolgreich! Einziger Vorteil ist der Umstand, daß der Compiler Quellcodes im ASCII-Format verarbeitet und damit praktisch wieder jeder Texteditor zum Erstellen der Pascalprogramme nutzbar ist. Ich benutze auch hier PASEDIT/CMD, der mir unter Pascal 4.5 ans Herz gewachsen ist!

Der Compiler fragt nach dem Start nach Ein- und Ausgabe-, sowie nach der Listdatei, in die er das Compilerprotokoll ablegen soll. Auch hier werden leider nur Fehlernummern angegeben, den Klartext muß man sich aus einer Fehlerliste holen. Und natürlich kommt dann auch hier die, beim Pascal 4.5 schon beschriebene, umständliche und langwierige Entwanzung des Quelltextes. Über die übersetzungsgeschwindigkeit an sich kann man sich nicht beschweren.

Nach dem Compilieren kann man sein Programm, inzwischen liegt es als /OBJ-File vor, mit Hilfe des Laufzeitsystems starten (RUN progname/obj). Um ein eigenständiges Programm zu erhalten muß man den Linker zu Hilfe rufen, der die benötigten Routinen aus den verschiedenen Dateien zusammensucht und bindet. Natürlich muß auch hier die komplette Runtimelibary mitgeschleppt werden und die /CMD-Files sind entsprechend groß, dafür kann man aber (so man kann!) von anderen Compilern (z.B. Alcor-C) erzeugte /REL-Dateien in die Pascalprogramme einbinden.

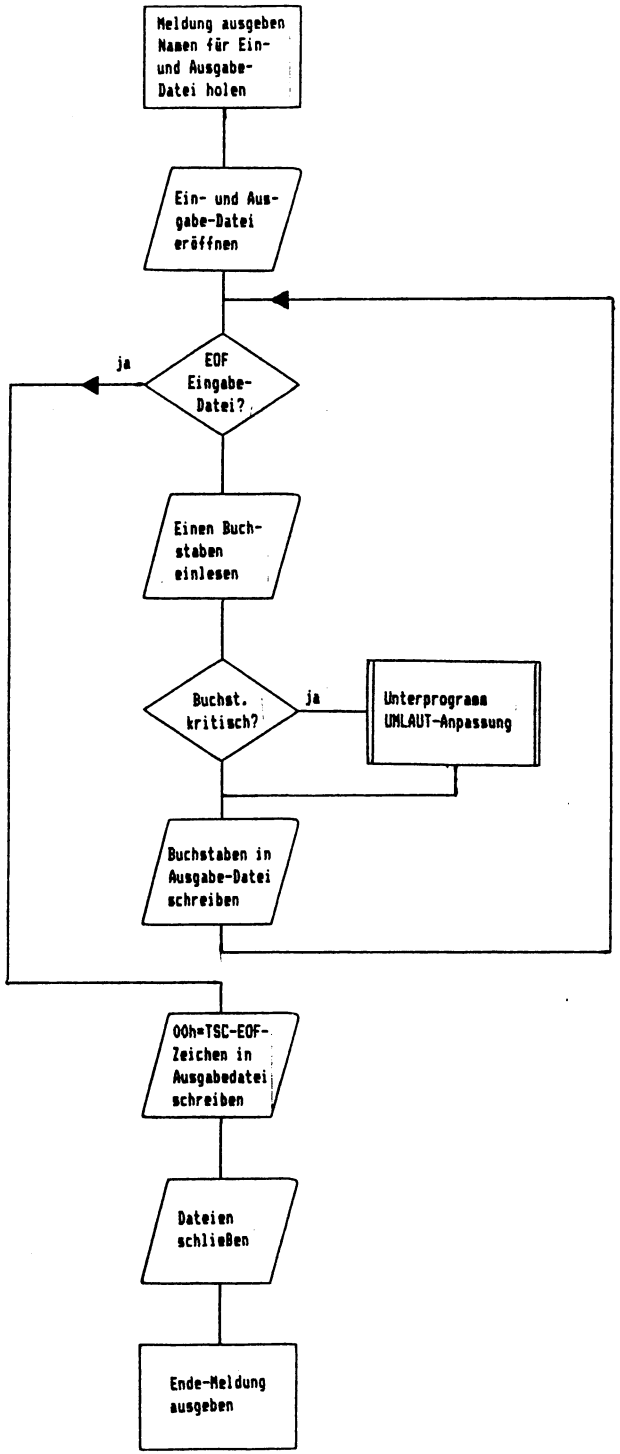
Etwas was mich an Alcor-Pascal sehr stört ist die Tatsache, daß jedes damit erstellte Programm nach seinem Start erst einmal fragt, welches die normalen Wege für INPUT und OUTPUT sein sollen. Sicher ist es manchmal sehr angenehm, an dieser Stelle z.B. den Printer als Ausgabemedium angeben zu können, trotzdem nervt nach mehreren Programmstarts die ständige Fragerei beträchtlich.

Leider kann ich hier nichts über das Handbuch zu diesem System aussagen, denn es wäre sicher ein Pluspunkt für Alcor! Auch scheint der Sprachumfang recht beachtlich zu sein, was ich aus den mir vorliegenden Quelltexten ersehen kann. Was mir weniger gefällt ist das Filehandling, das mir etwas umständlich erscheint. Zu bemerken wäre noch, daß es Alcor-Pascal sowohl für das Model 1 als auch für die Modelle 3/4/4p gibt. Im Gegensatz zu Pascal 80 und 4.5, von denen es nur eine Version für alle TRS 80-Modelle gibt, ist aber der Compiler für das M.1 nicht auf den anderen Maschinen lauffähig und umgekehrt.

Problem und Programmablaufplan

Schon im Vorwort habe ich kurz das Problem angesprochen, welches zu lösen gilt. Hier nun die genaue Problemdefinition:
aus einer Datei unbestimmter Länge soll der gesamte Inhalt in eine zweite Datei übertragen werden, dabei sollen verschiedene Zeichen durch andere ersetzt werden.

Aus dieser Aufgabenstellung ergibt sich folgender Ablaufplan:
(in den sich übrigens ein Fehler eingeschlichen hat! Die Abfrage "EOF Eingabe-Datei?" muß natürlich am Ende der Schleife "Buchst. einlesen / Buchst. kritisch? / Buchst. ausgeben" stehen. Da ich aber zu faul bin, das Flußdiagramm noch einmal zu zeichnen, lasse ich es bei dieser Anmerkung!)



Programm

Ohne größere Umschweife möchte ich euch jetzt auch gleich die Lösung des Problems in Pascal 4.5 präsentieren.

```
PROGRAM M100TSC (INPUT,OUTPUT,FILEA,FILEB);

VAR C             : CHAR;              (*Variable deklarieren*)
    CIN           : INTEGER;

PROCEDURE UMLAUT(CIN:INTEGER);         (*Unterprogramm UMLAUT*)

BEGIN
  CASE CIN OF                          (*je nach Wert von CIN
    182: CIN:=123;                       eine der folgenden
    183: CIN:=124;                       Aktionen ausführen!*)
    184: CIN:=125;
    185: CIN:=126;
    177: CIN:=91;
    178: CIN:=92;
    179: CIN:=93;
  END;
  C:=CHR(CIN);                         (*Wieder in Buchstaben
END;                                     umwandeln*)

BEGIN
  WRITE(CHR(28),CHR(31));              (*Bildschirm löschen*)
  WRITELN('UMLAUT');                   (*Startmeldung ausg.*)
  WRITELN;
  WRITELN('Model 100 -> TSCrips Umlaute-Umwandler');
  WRITELN('C by H. Obermann');
  WRITELN;
  WRITE('Eingabe-Datei? ');
  TITLE(FILEA);                      (*Eingabe-Datei abfr.*)
  WRITE('Ausgabe-Datei? ');
  TITLE(FILEB);                      (*Ausgabe-Datei abfr.*)
  RESET(FILEA);                      (*Eingabe-Datei  eröffnen*)
  WRITELN;
  REWRITE(FILEB);                    (*Ausgabe-Datei eröffnen
                                       sollte eine Datei
                                       gleichen Namens schon
                                       einmal bestehen, wird
                                       diese zuvor gelöscht!*)
  REPEAT                             (*wiederhole folgendes -*)
      READ(FILEA,C);                 (*Buchstabe einlesen*)
      CIN:=ORD(C);                   (*in Integer umwandeln*)
      IF (CIN > 176) AND (CIN < 186) THEN UMLAUT(CIN);
                                     (*im kritischen Bereich?
                                     Wenn ja, dann Umwandeln!*)
      WRITE(FILEB,C);                (*Buchstabe schreiben*)
  UNTIL EOF(FILEA);                  (*bis EOF erreicht!*)
  WRITE(FILEB,CHR(0));               (*EOF für TSCRIPS*)
  CLOSE(FILEA);                      (*Dateien schließen*)
  CLOSE(FILEB);
  WRITELN('Umlaute-Umwandlung beendet!');
  END.
```

Selbstverständlich weiß ich, daß das Programm noch nicht ganz sauber programmiert ist und daß sich die eingefleischten Pascal-Programmierer die Haare raufen werden! Trotzdem zeigt das Prgrämmchen meiner Meinung nach, daß der Umstieg von BASIC zu Pascal gar nicht so schwer ist! Vor allem die Lesbarkeit des Programms ist um Welten besser als die des, von mir an zunächst erstellten, BASIC-Programms. Ganz zu schweigen, von der erheblichen Geschwindigkeitssteigerung!

Vergleich

An dieser Stelle möchte ich einen kurzen Vergleich zwischen den verschiedenen Programmen machen, die, in verschiedenen Sprachen bzw. Dialekten, zur Lösung des Problems inzwischen erstellt wurden.

Sprache/Dialekt	Größe in Grans Quelltext	CMD-File	Geschwindigkeit in Sekunden
Alcor-Pascal	1	9	11.7 - 13.0
Pascal 80	1	10	11.0 - 15.0
Pascal 4.5	1	3	8.5 - 9.2
RPNL	3	3	10.5
Assembler	3	1	7.5 - 8.5

Die Größe der Dateien habe ich in Grans angegeben, weil ich es (zumindest bei einer solchen Anwendung) für uninteressant halte, ob ein Programm 196 oder 248 Byte lang ist. Beide Programme benötigen auf der Diskette auf alle Fälle 1 Granule!

Die Geschwindigkeit wurde mit einem Text gemessen, den ich gerade auf dem Model 100 erstellt hatte und der durch Zufall vorlag. Der Text war ca. 1.5 kByte lang und die Tests erfolgten unter NewDOS auf dem Model 4p mit einer Taktfrequenz von 4 MHz. Die Zeitendifferenzen bei den einzelnen Programmen ergeben sich daraus, daß einmal die Programmnamen mit und einmal ohne Drivenummer angegeben wurden.

Fazit

Eigentlich sollte sich jeder Leser das Fazit aus den obrigen Angaben selbst herauslesen können. Trotzdem hier noch ein paar zusammenfassende Sätze:

1. Wer Wert auf Geschwindigkeit legt und sich von BASIC (auch da gibt es ja Compiler) trennen möchte, dem stehen vor allem die Sprachen PASCAL, RPNL und Assembler zur Verfügung. Ich weiß natürlich, daß ich mir mit dieser Behauptung alle Anhänger anderer Sprachen wie COBOL, Forth usw. zum Feind mache, bin aber erst dann von dieser Behauptung abzubringen, wenn eine Problemlösung in der entsprechenden Sprache im Info erscheint!

2. Am schnellsten ist sicherlich die Assemblerlösung, aber nur Freaks wie Gerald Schröder und Arnulf Sopp schütteln ein solches Assemblerprogramm (so einfach es auch, gegenüber den Programmen, die wir von den beiden sonst gewohnt sind, auch sein mag) innerhalb von ein paar Minuten aus dem Ärmel.

3. Am wenigsten zu Schreiben hat man in Pascal (die Quelltexte in allen drei Dialekten umfassen nur ein Granual, die RPNL- und die Assemblerlösung bringen je 3 Grans auf die Waage!). Hat man die Qual der Wahl zwischen den Pascal-Compilern, weil man alle drei zur Verfügung hat, würde ich folgende Wahl treffen:
Pascal 80 - für Einsteiger und zum Lernen!
Pascal 4.5 - für normale Anwendungen, für Anfänger geeignet!
Alcor Pascal - für den Fortgeschrittenen (und nur mit Handbuch!)

4. Wer sich gerne einer sehr guten (schließlich vereinigt RPNL die Vorzüge von Forth mit denen von Pascal), leider aber sehr seltenen (besser vielleicht exotischen) Sprache zuwenden möchte, die noch dazu sehr maschinennahes Programmieren erlaubt und außergewöhnlich gut dokumentiert ist (sie liegt sogar im Quelltext vor!), kann sich auf RPNL stürzen!

Egal für welche Sprache und welchen Dialekt ihr euch entscheidet, ich wünsche euch auf jeden Fall viel Spaß und Erfolg beim Programmieren, euer

Hartmut Obermann

RPNL

Nachdem Hartmut sich nun ausgiebig zu den PASCAL-Versionen in seinem Test geäußert hat, folgt nun mein bescheidener Beitrag dazu. Da das zu lösende Problem bereits ausgiebig beleuchtet wurde, kann ich mich auf eine Beschreibung des Programmaufbaus beschränken.

Das Programm basiert auf einer Tabelle, in der die fraglichen ASCII-Codes und ihr Ersatz aufgelistet sind:

```
CALL ASCII.TAB  ;Konvertierungs-Tabelle
   B6 7B        ;Erster Eintrag altes Zeichen,
   B7 7C        ;...zweiter Eintag neues Zeichen
   B8 7D
   B9 7E
   B1 5B
   B2 5C
   B3 5D
   00 00        ;Tabellenende.
CEND


CODE TABELLE    ;( -> adr1) Tabellenzeiger auf den Stack
   21 ASCII.TAB ;LD     HL,ASCII.TAB
   E5           ;PUSH   HL
CEND
```

Da RPNL keine DATA-Anweisung kennt, muß, um derartiges zu erreichen, ein Trick angewandt werden. Die Daten-Tabelle wird einfach mit Hilfe der CALL-Definition erstellt (s. CALL ASCII.TAB). Auf dieser Definitionsstufe ist es dem Compile nämlich egal, was für Daten vom Anwender vorliegen. Es muß nicht notwendigerweise ein Unterprogramm sein ! Wenn wir schon freie Hand haben, dann ist es leider auch 'unser Bier', dafür zu sorgen, daß wir auch an die Daten kommen. Dazu muß eine Ebene höher ein weiteres Wort definiert werden, welches eben diesen Zeiger heranschafft (s. CODE TABELLE). Damit wäre der Zugriff auf das Wesentliche vorerst sichergestellt.

Da die Losung 'Austausch von Unerwünschtem gegen wichtigeres' lautet, ist natürlich noch ein Programm von Nöten, welches solches zu leisten im Stande ist. Als erstes der Gegenstand der Diskusion:

```
PROGRAM ?CONVERT          ;(char1 adr1 -> char2 adr1 flag)
    OVER OVER ?(B) =      ;TOS = Tabelleneintrag ?
    IF
       COUNT ? INC COUNT :=
       DUP INC ?(B)       ;Ersatzzeichen holen und...
       SWAP ROT DROP      ;...altes Zeichen ersetzen
       TRUE               ;Setze Erfolgs-Flag
    ELSE
       FALSE              ;Keine Konvertierung
    ENDIF
END
```

```
PROGRAM TEST.ASCII      ;(char1 -> char2)
    DUP DUP             ;prüfe ob Zeichen in der Tabelle
    177 < SWAP          ;Diese Bereichseingrenzung ist
    185 > OR NOT        ;Problemabhängig !!!
    IF
        TABELLE         ;Tabellen-Zeiger auf den Stack
        REPEAT
           ?CONVERT             ;Zeichen ersetzen ?
           OVER ?(B) 0= OR      ;Tabellenende ?
        UNTIL
           INC INC              ;Nein !, nächster Eintrag
        LOOP
        DROP            ;Nur das Zeichen selbst bleibt...
    ENDIF               ;... auf dem Stack
    CHARACTERS ? INC CHARACTERS :=
END
```

Das Wort ?CONVERT prüft das Zeichen auf dem Stack, ob es in der Tabelle enthalten ist. Wenn ja, wird es gegen das Ersatzzeichen ausgetauscht. Über den Erfolg oder Mißerfolgt gibt ein Flag auskunft, welches mit auf dem Stack übergeben wird. Damit TEST.ASCII nicht an Informationen erstickt, ist es unbedingt notwendig, daß die Menge der Daten nicht zu groß wird. Mehr wie vier Zahlen sind auf dem Stack einfach nicht zu verwalten, ohne daß man in unvermutete Probleme gerät. Deshalb ist es wichtig, das ?CONVERT im Erfolgsfall das alte Zeichen vom Stack entfernt. Wer das nicht einsehen kann, versuche das Problem im Wort TEST.ASCII zu lösen.

Da nur einige wenige Zeichen zu Prüfen sind, wird durch eine Bereichsabfrage eine Vorauswahl durchgeführt. Liegt ein Zeichen im interessierenden Bereich, wird weiter geprüft und die Tabelle durchsucht. Am Ende der Routine wird noch etwas mitgezählt, damit am Schluß die Anzahl der durchgeleierten Zeichen bekannt gegeben werden kann. Das Gleiche passiert in der Routine ?CONVERT, hier wird die Anzahl der Ersetzungen mitgezählt.

Damit ist der Job für ein einzelnes Zeichen bereits erledigt. Fehlt nur noch das Prinzip auf eine beliebige Menge auszudehnen. Im Grunde nichts leichter als das: Man programmiert ein weiteres Wort ! Womit wir beim Hauptprogramm wären. Damit bei der Eingabe der Filenamen kein Murks passieren kann, sorgt das Wort GET.NAMES für saubere Verhältnisse. Hier wird geprüft, ob der Quellfile existiert und das Quelle und Ziel nicht identisch sind. Sind diese Dinge OK, kann die eigentliche Arbeit starten und die Hatz nach den falschen Bytes beginnen.

Ein weiterer Höhepunkt der Byteschaufelei findet sich in der Hauptschleife im Hauptprogramm. Als Ende-Kennung sind drei Zustände vorgesehen:

1. *Ein Fehler wird bei READF oder WRITEF erkannt*
2. *Das Fileende wird erkannt,*
3. *Die NULL für SCRIPSIT File-Ende wird erkannt.*

Damit beides richtig zur Geltung kommt, wird ein eifriges Bäumchen-Wechsle-Dich-Spiel auf dem Stack getrieben. Damit die Sache beim ersten gelesenen Byte nicht außer Tritt gerät muß die Dummy-'0' auf den Stack gelegt werden. Damit verständlich wird,

was sich da so abspielt, einige Momente im Leben eines Datenstacks zum mitschreiben: 0 = Dummy, d = Databyte, f = Flag

	READF	ROT	OR	OVER	0=	OR
	0	d	-	d	d	-
	d	f	d	f1	f1	d
TOS->	f	0	f1	d	f2	f3

Wie jeder unschwer erkennen kann, fehlt eine NULL !!!. Die wird von dem WRITEF im zweiten Teil der REPEAT/UNTIL/LOOP Schleife erzeugt (das ist deren Flag über die saubere Abwicklung des in den Zielfile geschriebenen Bytes). Tritt also beim Schreiben oder beim Lesen ein Fehler auf, so bleibt das nicht unbeachtet. Ebenso wie das SCRIPSIT- oder File-Ende nicht übersehen werden kann.

```
:
:
:
0                       ;Dummy '0'
REPEAT
   FILE1 READF          ;hole ein Byte
   ROT OR               ;(Dummy -'0') OR (flag)
   OVER 0= OR           ;etwa Tscript Fileende ?
UNTIL                   ;nur weiter, kein Fehler
   TEST.ASCII
   FILE2 WRITEF         ;schreibe ein Byte
LOOP
DROP                    ;vernichte Stop-Flag
0 FILE2 WRITEF DROP     ;File-Ende markieren
:
:
:
```

Bei diesem Hürdenlauf der Flags war bis nach Hartmuts Test noch einen Fehler im Programm, den ich nicht bemerkt hatte: Nachdem der File gelesen war, blieb die SCRIPSIT-NULL auf dem Stack. Kein 'Fatal Error', aber unschön. Die hier abgelichtete Version ist jetzt korrekt. An den Testergebnissen ändert sich dadurch nichts ! Da für die Konvertierung eine Tabelle benutzt wird, wäre es natürlich auch denkbar, diese nicht fest im Programm zu verankern, sondern extern zu verwaren und je nach bedarf von Diskette nachzuladen. Dazu müßte das Wort ASCII.TAB das Einlesen der Daten übernehmen und im Wort TABELLE der Zeiger auf die Tabelle im RAM bereitgestellt werden. Im Wort GET.NAMES ist die Abfrage auf den Namen der externen ASCII-Tabelle einzubauen. Im Anschluß folgt nun das vollständige Programm.

```
;**=======================================**
;**        ASCII Filter-Programm          **
;**                                       **
;**        Vers. 1.0       12.87          **
;**=======================================**
;
;
DECLARE
   VARF   FILE1
   VARF   FILE2
   VAR    FILE.NAME1
   VAR    FILE.NAME2
   VAR    COUNT
   VAR    CHARACTERS
DEND


CALL ASCII.TAB          ;Konvertierungs-Tabelle
   B6 7B                ;Erster Eintrag altes Zeichen,
   B7 7C                ;...zweiter Eintag neues Zeichen
   B8 7D
   B9 7E
   B1 5B
   B2 5C
   B3 5D
   00 00                ;Tabellenende. Doppel-Null für den Fall,
CEND                    ;daß jemand unbedingt die Null in eine
                        ;Null konvertieren moechte


CODE TABELLE            ;( -> adr1) Tabellenzeiger auf den Stack
   21 ASCII.TAB         ;LD     HL,ASCII.TAB
   E5                   ;PUSH   HL
CEND


PROGRAM GET.NAMES       ;( -> flag)          ;holt FILENAMEN
   SYSFIELD ?                                ;RAM-Pointer retten
   ;
   ;Hole die Filenamen
   ;
   14 OUTCHAR                                   ;Cursor ON
   WRITE 'File1: ? ' FILE.NAME1 READ(CH)        ;erster FILE
   WRITE 'File2: ? ' FILE.NAME2 READ(CH)        ;zweiter FILE
   15 OUTCHAR                                   ;Cursor OFF
   CR
   ;
   ;Beide Namen müssen ungleich sein !
   ;
   FILE.NAME1 ? FILE.NAME2 ? =(S) NOT
   IF
      FILE.NAME1 ? FILE1 FILESPEC  ;neue Filenamen...
      FILE.NAME2 ? FILE2 FILESPEC  ;...einsetzen
      OR 0=                        ;ERROR-Flags zusammenfassen
   ELSE
      WRITE '*** ERROR: Input-File = Output-File !' CR
      FALSE                        ;Eingabefehler !
   ENDIF
   SWAP SYSFIELD :=                ;Restore RAM-Pointer
```



```
END


PROGRAM ?CONVERT        ;(char1 adr1 -> char2 adr1 flag)
   OVER OVER ?(B) =     ;TOS = Tabelleneintrag ?
   IF
      COUNT ? INC COUNT :=
      DUP INC ?(B)      ;Ersatzzeichen holen und...
      SWAP ROT DROP     ;...altes Zeichen ersetzen
      TRUE              ;Setze Erfolgs-Flag
   ELSE
      FALSE             ;Keine Konvertierung
   ENDIF
END


PROGRAM TEST.ASCII              ;(char1 -> char2)
   DUP DUP                      ;prüfe ob Zeichen in der Tabelle
   177 < SWAP                   ;Diese Bereichseingrenzung ist
   185 > OR NOT                 ;Problemabhängig !!!
   IF
      TABELLE                   ;Tabellen-Zeiger auf den Stack
      REPEAT
         ?CONVERT               ;Zeichen ersetzen ?
         OVER ?(B) 0= OR        ;Tabellenende ?
      UNTIL
         INC  INC               ;Nein !, nächster Eintrag
      LOOP
      DROP                      ;Nur das Zeichen bleibt...
   ENDIF                        ;... auf dem Stack
   CHARACTERS ? INC CHARACTERS :=
END


PROGRAM FILTER                  ;>>>>  Hauptprogramm  <<<<
   CLS
   0 COUNT :=
   0 CHARACTERS :=
   WRITE '**=======================================**' CR
   WRITE '**              ASCII-Filter             **' CR
   WRITE '**=======================================**' CR
   CR
   GET.NAMES                    ;Liefert bei OK TRUE-Flag
   IF                           ;Filespec's OK ?
     FILE1 OPEN(I)
     0=                         ;Quell-File vorhanden ?
     IF
        FILE2 OPEN(O)
        0=
        IF                      ;Abbruch bei FILE-ERROR
           WRITE 'Writing data to file'
           0                    ;Dummy '0'
           REPEAT
              FILE1 READF       ;hole ein Byte
              ROT OR            ;(Dummy -'0') OR (flag)
              OVER 0= OR        ;etwa Tscript Fileende ?
           UNTIL                ;nur weiter, kein Fehler
              TEST.ASCII
              FILE2 WRITEF      ;schreibe ein Byte
```

```
        LOOP                    ;vernichte stop-Flag
        DROP
        0 FILE2 WRITEF DROP     ;File-Ende markieren
        ENDIF
      FILE2 CLOSE
    ENDIF
    FILE1 CLOSE
  ENDIF
  COUNT ? 0 >
  IF
     CR WRITE 'Number of substitutions:  ' COUNT PRINT
     CR WRITE 'Total # of characters   :  ' CHARACTERS PRINT CR
     CR
  ELSE
     CR WRITE 'No substitutions !'
     CR WRITE 'File length is      :' CHARACTERS PRINT
        WRITE ' Byte' CR CR
  ENDIF
END
```

Die Stadt ist nicht groß genug für uns beide!

UMLAUT.M

Hartmuts Artikel über die Umlaut-Konvertierung in Pascal habt Ihr sicherlich schon gelesen (wenn nicht, tut es am besten jetzt!). Hartmut bat mich, ein paar Zeilen zu der Assembler-Lösung zu sagen, die Ihr hinter diesem Artikel abgedruckt findet. Eigentlich ist das Programm fast "selbstdokumentierend", wenn auch in ganz anderem Sinne als die Pascal-Lösung: in Assembler sind die Kommentare wichtiger als die Befehle.

Der Anfang mit den Abfragen der beiden Filenamen dürfte recht klar sein. Ich will nicht näher darauf eingehen. Nur der Ablauf sei kurz dargestellt: Zuerst wird ein Einleitungstext angezeigt, der hier sehr kurz ausfällt. Dann fragt das Programm nach dem Namen des Files, der die fehlerhaften Umlaute enthält. Falls dieser File nicht existiert, kommt eine Fehlermeldung und die Eingabe muß wiederholt werden, bis der Filename korrekt ist oder mit BREAK abgebrochen wird. Der zweite File soll die korrigierte Version des ersten aufnehmen. Es wird nur einmal gefragt; falls ein Fehler auftritt, bricht das Programm ab (mit einer DOS-Fehlermeldung).

Nun kommt der interessante Teil, den ich mit "***..." markiert habe. Ihn will ich näher erläutern: Zuerst wird ein Byte aus dem ersten File gelesen. Dabei muß DE auf den FCB des ersten Files zeigen! Zurückgegeben wird das gelesene Byte in A und ein Fehlerstatus. Bei Z (Zero-Flag gesetzt) war das Lesen erfolgreich, bei NZ (Zero-Flag gelöscht) ist ein Fehler aufgetreten, wobei jetzt die Nummer des Fehlers in A steht; in diesem Fall wird die Schleife abgebrochen, wobei noch nicht klar ist, ob dieses Ende erzwungen (Disk-Fehler) oder gewollt (Ende von File 1 = ich bin fertig) wurde. Falls das Lesen erfolgreich war, stellen die beiden ComPare-Befehle fest, ob es sich um einen Umlaut handelt. Falls nicht, kann das Byte sofort in den zweiten File geschrieben werden. Falls es sich um einen Umlaut handelt, wird 177 von seinem ASCII-Code abgezogen, so daß der erste Umlaut jetzt zur 0 wird, der zweite zur 1 usw. Diese Zahl gibt an, um welchen Umlaut es sich handelt. Nachdem H und L gesetzt wurden, hole ich durch LD A,(HL) den neuen Umlaut aus einer Tabelle. Dabei zeigt HL genau auf den richtigen Umlaut, weil der Anfang der Tabelle immer bei xx00h liegt. Also steht der erste Umlaut (der zu dem mit der Nummer 0 gehörende) in (xx00h), der zweite (der zu Nummer 1 gehört) in (xx01h) usw. Das Gleiche hätte ich erreicht, wenn ich die Nummer auf den Tabellenanfang tab addieren würde, aber das dauert länger (ein paar Micro-, Nano- oder sonstwas-Sekunden). Zuletzt wird der neue Umlaut (bzw. ein normaler Buchstabe, der kein Umlaut ist) in den zweiten File geschrieben. Dabei ist zu beachten, daß DE diesmal auf den zweiten FCB zeigen muß. Falls beim Schreiben ein Fehler auftritt, bricht das Programm ab und zeigt den Fehler an. Ansonsten (im Normalfall) wird die Schleife hier nie verlassen.

Bevor das Programm endet, werden alle beiden Dateien geschlossen (was bei der Lese-Datei evtl. nicht nötig ist, aber es sieht schöner aus). Je nachdem, ob ein "ordentlicher" Abbruch mit dem "richtigen" Fehler, nämlich "Dateiende erreicht", erfolgt ist oder nicht, wird eine "Alles roger"- oder eine Fehlermeldung (vom DOS) angezeigt.

Gerald Schröder

```
                        00001 ;       UMLAUT.M
                        00002 ;       letzte Änderung: 8.2.88 von Gerald Schröder
                        00003
                        00004 ;       konvertiert File mit Umlauten
                        00005
                        00006 ;       DOS- und ROM-Routinen:
                        00007
0013                    00008 readbyt EQU     0013h           ;ein Byte lesen
001B                    00009 writbyt EQU     001bh           ;ein Byte schreiben
05D9                    00010 inbuf   EQU     05d9h           ;B Zeichen eingeben
402D                    00011 dos     EQU     402dh           ;Abgang ins DOS
4409                    00012 derror  EQU     4409h           ;DOS-Error ausgeben
441C                    00013 exfil   EQU     441ch           ;Filenamen in FCB
4420                    00014 init    EQU     4420h           ;File neu öffnen
4424                    00015 open    EQU     4424h           ;exist. File öffnen
4428                    00016 close   EQU     4428h           ;File schließen
4467                    00017 scrmes  EQU     4467h           ;Meldung ausgeben
                        00018
                        00019 ;       Konstante:
                        00020
001C                    00021 okerr   EQU     1ch             ;"Dateiende gefunden"
                        00022
                        00023
5200                    00024         ORG     5200h
                        00025
5200  210953            00026 start   LD      HL,anfang       ;Anfangsmeldung
5203  CD6744            00027         CALL    scrmes          ;anzeigen
                        00028
5206  213A53            00029 nochmal LD      HL,frage1       ;Abfrage 1. Filename
5209  CD6744            00030         CALL    scrmes          ;ausgeben
520C  218F53            00031         LD      HL,buf1         ;Buffer für Eingabe
520F  060F              00032         LD      B,15            ;Länge 15
5211  CDD905            00033         CALL    inbuf           ;B Zeichen nach HL
5214  DA2D40            00034         JP      C,dos           ;ab, falls BREAK-Taste
5217  11AD53            00035         LD      DE,fcb1         ;Zeiger auf FCB-Buffer
521A  CD1C44            00036         CALL    exfil           ;Name übertragen in FCB
521D  21ED53            00037         LD      HL,sekbuf1      ;Sektor-Buffer
5220  0600              00038         LD      B,0             ;Sektor-Länge 256
5222  CD2444            00039         CALL    open            ;File öffnen
5225  2808              00040         JR      Z,zweiter       ;kein Fehler, weiter
5227  214A53            00041         LD      HL,fehler       ;Zeiger auf Fehlermeldung
522A  CD6744            00042         CALL    scrmes          ;ausgeben
522D  18D7              00043         JR      nochmal         ;nochmal eingeben
                        00044
522F  216953            00045 zweiter LD      HL,frage2       ;ebenso für 2. File
5232  CD6744            00046         CALL    scrmes
5235  219E53            00047         LD      HL,buf2
5238  060F              00048         LD      B,15
523A  CDD905            00049         CALL    inbuf
523D  DA2D40            00050         JP      C,dos
5240  11CD53            00051         LD      DE,fcb2
5243  CD1C44            00052         CALL    exfil
5246  21ED54            00053         LD      HL,sekbuf2
5249  0600              00054         LD      B,0
524B  CD2044            00055         CALL    init            ;diesen neu schreiben
524E  2034              00056         JR      NZ,error        ;falls Disk-Fehler
                        00057
                        00058 ; ********************************************************
5250  11AD53            00059 loop    LD      DE,fcb1         ;Zeiger auf 1. FCB
5253  CD1300            00060         CALL    readbyt         ;Byte aus File 1 lesen
5256  2016              00061         JR      NZ,ende         ;Ende oder Fehler
5258  FEB1              00062         CP      177             ;Buchstabe < 177?
525A  380A              00063         JR      C,ok            ;ja, ok, kein Umlaut
525C  FEBA              00064         CP      186             ;Buchstabe > 185?
525E  3006              00065         JR      NC,ok           ;ja, ok, kein Umlaut
5260  D6B1              00066         SUB     177             ;177=>0 usw.
5262  2653              00067         LD      H,tab/100h      ;MSB der Umw.tabelle
5264  6F                00068         LD      L,A             ;Code als Tab-Offset
5265  7E                00069         LD      A,(HL)          ;geändertes Zeichen holen
5266  11CD53            00070 ok      LD      DE,fcb2         ;Zeiger auf 2. FCB
5269  CD1B00            00071         CALL    writbyt         ;Byte in File 2 schreiben
526C  28E2              00072         JR      Z,loop          ;kein Fehler: nächstes Z.
                        00073 ;********************************************************
                        00074
                        00075 ;       Status NZ erreicht: irgendein Diskfehler trat auf
                        00076 ;       Bei "Dateiende erreicht" ist das OK, ansonsten
                        00077 ;       hat was nicht geklappt!
                        00078
526E  F5                00079 ende    PUSH    AF              ;Fehlercode retten
526F  11AD53            00080         LD      DE,fcb1         ;File 1
5272  CD2844            00081         CALL    close           ;schließen
5275  11CD53            00082         LD      DE,fcb2         ;ebenso File 2
5278  CD2844            00083         CALL    close
527B  F1                00084         POP     AF              ;Fehlercode zurück
527C  FE1C              00085         CP      okerr           ;Dateiende gefunden?
527E  217C53            00086         LD      HL,endmel       ;Zeiger auf OK-Meldung
5281  CA6744            00087         JP      Z,scrmes        ;falls ja: ausgeben und
                        00088                                 ;Abgang ins DOS
                        00089
5284  F6C0              00090 error   OR      0c0h            ;Bit 7 für Fehler-Ausgabe
5286  C30944            00091         JP      derror          ;setzen und Ausgabe u. ab
                        00092
                        00093 ;-------------------------------------------------------
                        00094
                        00095 ;       Tabelle zur Umwandlung der kritischen Zeichen
                        00096
0077                    00097         DS      $+100h&0ff00h-$ ;Anfang auf "gerader"
                        00098                                 ;Adresse (LSB=00)
5300  5B                00099 tab     DB      91,92,93,180,181,123,124,125,126
                        00100
                        00101
                        00102 ;       Texte
                        00103
5309  1C                00104 anfang  DM      1ch,1fh,'UMLAUT.M',0ah,0ah,'Alles klar?'
5320  0A                00105         DM      0ah,'Sonst ist es auch egal.',0ah,0dh
533A  0A                00106 frage1  DM      0ah,'Welcher File? ',03
534A  0A                00107 fehler  DM      0ah,'So nicht! Gleich noch einmal!',0dh
5369  0A                00108 frage2  DM      0ah,'In welchen File? ',03
537C  0A                00109 endmel  DM      0ah,'Alles roger, bye!',0dh
                        00110
                        00111
                        00112
                        00113 ;       Buffer
                        00114
000F                    00115 buf1    DS      15              ;Eingabe Filename 1
000F                    00116 buf2    DS      15              ;ebenso Filename 2
0020                    00117 fcb1    DS      32              ;FCB File 1
0020                    00118 fcb2    DS      32              ;ebenso File 2
0100                    00119 sekbuf1 DS      256             ;Sektorbuffer File 1
0100                    00120 sekbuf2 DS      256             ;ebenso File 2
                        00121
5200                    00122         END     start

00000 Fehler
```

CP/M intern - kurz und kompakt

0. Vorwort

Nachdem ich angefangen hatte, mich mit CP/M zu beschäftigen, wurmte mich gleich eine Sache ungemein: wie beim alten Newdos brauchte ich aus den dicken Anleitungsbüchern wieder nur ein paar Seiten, die wiederum nicht vollständig oder zu ausführlich waren. Also wollte ich mir eine kurze Zusammenfassung der Sachen schreiben, die ich oft brauchen würde. Daher auch der Titel "Kurz und kompakt". Leider mußte ich feststellen, daß die Kürze das Verständnis erschwerte und es wieder nötig machte, die dicken Wälzer heranzuziehen. Also habe ich einen Kompromiß beschlossen: die Kapitel sind kurz gehalten und mit massenhaft Abkürzungen versehen. Im Anhang dagegen werden einige Begriffe und Themen nochmal ausführlicher erläutert, damit auch ein Neuling damit besser zurechtkommt. Klassischerweise müßte die Aufteilung anders sein, aber das war nicht der Sinn der Sache.

Die Informationen habe ich mir aus dem "Handbuch zum CP/M" von Digital Research und dem CP/M-Buch von Plate zusammengesucht.

Leider bin ich mir bei vielen Sachen nicht sicher, ob sie stimmen, und wäre sehr froh, wenn das jemand korrigieren würde und Erweiterungen vorschlüge. Vor allem fehlen mir Infos über Terminal-Steuercodes.

Vielen Dank an Hartmut Obermann für das Korrekturlesen und viele Informationen und an Helmut Bernhardt für die CP/M-Fähigkeit meines Rechners.

Gerald Schröder

1. CCP

Der CCP wird direkt unter das BDOS in die TPA geladen. Er kann mit einem laufenden Programm überschrieben werden. Allerdings muß das Programm dann mit einem Warm Boot (JP 0000) beendet werden. Ansonsten kann man den CCP wieder direkt anspringen (wenn die Adresse bekannt ist). Er erwartet das Default-Laufwerk in Register C.

1.1 Wie sieht ein Filename aus?

a) bfile = bestimmter File, z.B. TURBO.COM
b) ufile = unbestimmter File, '?'=beliebiger Buchstabe, '*'=Rest beliebig, z.B. TURBO.??? oder TURBO.*

1.2 CCP-Kommandos

'ERA ufile'	: löscht Files
'DIR'	: Inhaltsverzeichnis akt. Laufwerk anzeigen
'DIR lw'	: Inhaltsverzeichnis eines bestimmten Laufwerks anzeigen
'DIR ufile'	: nur bestimmte Files anzeigen
'REN bfile=bfile'	: File umbennen (1. Namen in 2. umändern)
'SAVE n bfile'	: n Sektoren (n dez.) je 100h ab 100h sichern
'TYPE bfile'	: File anzeigen
'USER n'	: auf anderen Userbereich (für alle Laufwerke gültig) umschalten, n<16

1.3 CTRL-Codes (wichtige)

CTRL-C: Warm-Boot (nur am Zeilenanfang)
CTRL-P: Drucker-Listing an/aus
CTRL-R: Zeile nochmal anzeigen
CTRL-S: List-Pause, bis irgendeine Taste
CTRL-Z: Ende einer Aktion, auch: EOF (1ah)

2. Programmierung

2.1 Zero-Page

0000: Ansprung Warm-Boot
0001/2: Zeiger auf Warm-Boot-Routine (=BIOS+3)

0003: I/O-Byte:

Bits:	7/6	5/4	3/2	1/0
	LIST	PUNCH	READER	CONSOLE

Zuordnung:

	LIST (LST:)	PUNCH (PUN:)	READER (RDR:)	CONSOLE (CON:)
00	TTY:	TTY:	TTY:	TTY:
01	CRT:	PTP:	PTR:	CRT:
10	LPT:	UP1:	UR1:	BAT:
11	UL1:	UP2:	UR2:	UC1:

0004: Bits 7-4: USER-Nr.; Bits 3-0: Default-Lw.
0005: Ansprung BDOS (-Funktionen)
0006/0007: Zeiger auf BDOS (bis hierhin kann jedes Programm den Speicher benutzen, wenn es mit JP 0000 endet)
005ch-007fh: FCB für Benutzer
0080h-00ffh: Sektor-Buffer für Benutzer

2.2 Parameter-Übergabe

Übergabe von Parametern beim Aufruf von COM-Programmen
am Beispiel von: 'turbo a:abc.pas xyz.com'

1. FCB: (005ch) ff.: 01,'ABC PAS',0,0,0,0
2. FCB: (006ch) ff.: 00,'XYZ COM',0,0,0,0 (muß gerettet werden!)

Rest der Zeile: (0080h) ff.: 18,' A:ABC.PAS XYZ.COM'
(vom CCP in Großbuchstaben umgewandelt, (0080h)=Länge)

das Anwender-Programm wird vom CCP geCALLt; es steht ein Stack mit 7 Ebenen zur Vefügung; also lieber eigenen Stack einrichten!

2.3 Der FCB

Format des FCB:

	DR	F1	F2	..	F8	T1	T2	T3	EX	S1	S2	RC	D0	..	DN	CR	R0	R1	R2
+	00	01	01		08	09	10	11	12	13	14	15	16		31	32	33	34	35

Erklärung FCB-Bytes (bzw. -Felder):

DR: Lw. für Zugriff: 00=Default-Lw.
01=Lw. A
..
16=Lw. P
F1..F8: Filename in Großbuchstaben
T1..T3: Extension in Großbuchstaben
Bit 7 von T1 gesetzt: File kann nur gelesen werden (R/O)
Bit 7 von T2 gesetzt: SYS-File (bei DIR nicht angezeigt)
EX: akt. Extent-Nr., vom Benutzer auf 0 zu setzen, bei File E/A 0-31
S1: reserviert
S2: reserviert, bei OPEN/SEARCH/MAKE auf 0 setzen (oder gesetzt???)
RC: Record-Nr. im Extent , 0-127
D0..DN: reserviert
CR: akt. Record-Nr. bei seq. File, vom Benutzer auf 0 zu setzen
R0..R2: Record-Nr. bei Random-File, LSB=R0, MSB=R1, Overflow=R2

2.4 BDOS-Funktionen

BDOS-Funktionen:
(die Funktionsnr. ist jeweils nach C zu laden; es werden keine Register gerettet; s. auch Anhang A)

00: sysres
System-Reset, wie Warm-Boot (zurück zum CCP, Lw. auf A)
01: bconin OUT: A=Tasten-Code
02: bconout IN: E=auszugebendes Zeichen
03: breader OUT: A=gelesenes Zeichen
auf READER-Eingabe warten
04: bpunch IN: E=auszugebendes Zeichen
auf PUNCH ausgeben
05: blist IN: E=auszugebendes Zeichen
06: condir IN: E=0ffh: CONSOLE-Eingabe (Tastatur)
<0ffh: CONSOLE-Ausgabe (Bildschirm)
OUT: (nur bei E=0ffh) A=00: keine Taste
>00: Tastencode
07: getio OUT: A=I/O-Byte
08: setio IN: E=I/O-Byte
09: prstr IN: DE=Zeiger auf String, der mit '$' endet
String-Ausgabe auf den Bildschirm
10: getstr IN: DE=Zeiger auf Buffer
(DE)=max. einzugebende Zeichen (1-255)
OUT: (buffer+1): Anzahl eingebene Zeichen
(buffer+2) ff.: eingebene Zeichen
11: bconst OUT: A=0ffh: Taste gedrückt
=00h: keine Taste gedrückt
12: getver OUT: H=00: CP/M
=01: MP/M
L=Versions-Nr. (z.B. 20h=2.0, 22h=2.2)
13: resdsk
Disk zurücksetzen (bei Diskettenwechsel), DMA auf 0080h, Laufwerk A anwählen
14: bseldsk IN: E=Lw. (00h=A)
das angewählte Lw. wird Default-Lw.
15: open IN: DE=Zeiger auf FCB
OUT: A=0-3: alles OK
0ffh: Fehler
im Filenamen auch '?' erlaubt
16: close IN: DE=Zeiger auf FCB
OUT: A=Fehlerstatus (s. 15)
17: search1 IN: DE=Zeiger auf FCB
OUT: A=Fehlerstatus (s. 15)
DMA mit entsprechenden DIR-Sektor geladen, File-Eintrag bei DMA + A*32
wenn DR='?' wird das Default-Lw. auf allen USER-Ebenen durchsucht
18: searchn IN: DE=Zeiger auf FCB
OUT: A=Fehlerstatus (s. 15)
nächsten passenden Eintrag suchen, ansonsten siehe 17
19: delete IN: DE=Zeiger auf FCB
OUT: A=Fehlerstatus (0ffh: nicht gefunden)
Filename darf '?' enthalten, aber DR nicht, alle passenden Files werden gelöscht
20: readseq IN: DE=Zeiger auf FCB
OUT: A=00: OK
>00: Fehler
128-Byte-Sektor wird gelesen, CR und evtl. EX +1
21: writseq IN: DE=Zeiger auf FCB
OUT: A=Fehlerstatus (s. 20)
22: make IN: DE=Zeiger auf FCB
OUT: A=0ffh: Disk ist voll
=0/1/2/3: OK
neuen File öffnen
23: rename IN: DE=Zeiger auf FCB 1 und 2
OUT: A=0ffh: File nicht gefunden
<0ffh: OK
erste 16 Bytes vom FCB: vorhandener Filename
zweite 16 Bytes vom FCB: neuer Filename, DR muß 00 sein
24: getlog OUT: HL=Login-Vektor
welche Laufwerke wurden bisher angesprochen? (Bit 0 von L = Lw. A)
25: getdsk OUT: A=akt. Default-Lw. (00h=Lw. A)
26: setdma IN: DE=DMA-Adresse
27: getalc OUT: HL=Zeiger auf ALLOCation-Vektor
28: wpdsk
akt. Lw. schreibschützen (R/O)
29: getro OUT: HL=R/O-Vektor
welche Laufwerke sind schreibgeschützt? (Bit 0 von L = Lw. A)

```
30: setatt    IN:  DE=Zeiger auf FCB
              OUT: A=Fehler-Status (s. 15)
              R/O bzw. SYS eines Files neu setzen im Directory
31: getdpb    OUT: HL=Zeiger auf DPB
32: user      IN:  E=0ffh: hole akt. USER-Nr.
                   =0-15: USER neu setzen
              OUT: (nur bei E=0ffh): A=akt. USER-Nr.
33: readrnd   IN:  DE=Zeiger auf FCB
              OUT: A=00: OK
                     01: nicht geschriebene Daten gelesen
                     03: Extent kann nicht geschlossen werden
                     04: Versuch, ungeschriebenen Extent zu finden
                     06: hinter Ende der Diskette (R2>0)
              bei Fehler 3: nochmal Extent 0 lesen
              Record-Nr. wird nicht inkrementiert
34: writrnd   IN:  DE=Zeiger auf FCB
              OUT: Fehler-Status, s. 33
                   A=5: neuer Extent kann nicht geöffnet werden
                        (Inhaltsverzeichnis voll)
              Record-Nr. wird nicht inkrementiert
35: getsiz    IN:  DE=Zeiger auf FCB
              OUT: R0/R1/R2: File-Größe
              R0/R1/R2 werden auf letzte Record-Nr.+1 gesetzt
36: setrnd    IN:  DE=Zeiger auf FCB
              OUT: R0/R1/R2
              R0/R1/R2 bei einem File setzen, auf den bis jetzt seq.
              zugegriffen wurde; nun Random-Zugriff möglich
37: resdrv    IN:  DE=Lw.-Vektor
              OUT: A=00
              Laufwerke zurücksetzen, wenn Bits im Vektor gesetzt
40: write0    IN:  DE=Zeiger auf FCB
              OUT: Fehlerstatus wie bei 34
              Block wird vor dem Schreiben mit Nullen gefüllt, wenn er
              vorher unbelegt war, ansonsten wie 34
```

2.5 BIOS-Funktionen

so erhält man die Startadresse: (0001/2)=Zeiger auf BIOS+3

folgende Tabelle: jeweils BIOS (Basisadresse) + Offset

```
+00: boot   Kaltstart
+03: wboot  Warmstart
+06: const  Tastatur-Status, OUT: A=0ffh: Taste grdückt
+09: conin  auf Tastendruck warten, OUT: A=Tastencode
+12: conout auf den Bildschirm ausgeben, IN: C=Zeichen
+15: list   auf den Drucker ausgeben, IN: C=Zeichen
+18: punch  auf den PUNCH ausgeben, IN: C=Zeichen
+21: reader auf Eingabe vom READER warten, OUT: A=Zeichen (1ah=EOF)
+24: home   angewähltes Laufwerk auf Track 0
+27: seldsk Laufwerk anwählen, IN: C=Lw. (00h=A); OUT: HL=Adresse
     der Sektor-übersetzungs-Tabelle, HL=0000: Fehler
+30: settrk Track anwählen, IN: BC=Track-Nr.
+33: setsec Sektor anwählen, IN: BC=Sektor-Nr.
+36: setdma DMA-Adresse setzen, IN: BC=DMA-Adresse
+39: read   Sektor lesen, OUT: A=00: OK, A=01: Fehler
+42: write  Sektor schreiben, IN: C=0: normales Schreiben; C=1: DIR-
     Sektor schreiben (sofort ausgeführt); C=2: 1. Sektor eines neuen
     Blocks; OUT: A=00: OK, A=01: Fehler
+45: listst Status des Druckers abfragen, OUT: A=0ffh: READY
+48: sectrn übersetzung der Sektor-Nr., IN: BC=Sektor-Nr., DE=Adresse
     der übersetzungs-Tabelle; OUT: HL=übersetzte Sektor-Nr.
```


3. Disk-Parameter-Tabellen

3.1 DPH (Disk Parameter Header)

Format:	XLT	0000	0000	0000	DIRBUF	DPB	CSV	ALV
Bytes: +	0	2	4	6	8	10	12	14

Bedeutung:
- XLT: Zeiger auf Sektor-Übersetzungs-Tabelle
- 0000: für BDOS reserviert
- DIRBUF: Zeiger auf einen 128-Byte-Buffer für Directory-Zugriffe kann für alle DPHs gleich sein
- DPB: Zeiger auf den DPB für dieses Lw.
- CSV: Zeiger auf einen Prüfsummenblock, mit dem festgestellt werden kann, ob die Diskette gewechselt wurde; muß für jeden DPH ein anderer sein; Länge: s. DPB
- ALV: Zeiger auf einen Block, in dem BDOS die Diskettenbelegung festhält (Allocation-Vektor); muß für jeden DPH ein anderer sein; Länge: (DSM/8)+1 (DSM: s. DPB; evtl. auch: (DSM+1)/8)

3.2 DPB (Disk Parameter Block)

Format:	SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
Bytes: +	0	+2	+3	+4	+5	+7	+9	+10	+11	+13

Achtung: indirekt festgelegt: BLS (Blockgröße in Bytes): 1024 * 2 hoch x, wobei x von 0 bis 4 gehen darf (BLS von 1024 bis 16384)

Bedeutung:
- SPT: Anzahl 128-Byte-Sektoren per Track
- BSH: x+3 (x aus BLS, s. oben), d.h.: wenn BSH=3, ist BLS=1024 usw.
- BLM: 2 hoch BSH - 1, d.h. wenn BSH=3, ist BLM=7 usw.
- EXM: hängt von DSM ab; wenn DSM<256: EXM=2 hoch x - 1, d.h.: x=0: EXM=0 usw.; wenn DSM>255: EXM=2 hoch (x-1) - 1; (x=1: EXM=0)
- DSM: größte Blocknummer (in BLS gemessen); (DSM+1)*BLS ergibt die totale Speicherkapazität der Diskette in Bytes, ohne die System-tracks; also: DSM=(Tracks-OFF)*SPT*128/BLS - 1
- DRM: Anzahl der Directory-Einträge-1 (d.h.: DRM=127: 128 Dir-Eintr.)
- ALO/AL1:

Bytes	ALO								AL1							
Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
belegt:	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15

  es sind (DSM+1)*32/BLS Bits zu belegen, angefangen von AL0, Bit 7, über AL0, Bit 0, und AL1, Bit 7, bis AL1, Bit 0; bei BLS=1024 und DRM=128 sind AL0=0f0h und AL1=00h (4 Bits belegt)
- CKS: Länge des Prüfblockes CSV (s. DPH); CKS=(DRM+1)/4; nur bei Disketten, nicht bei festen Speichern (RAM-Disk oder Festplatte: CKS=0)
- OFF: Anzahl der System-Tracks; werden am Disk-Anfang ignoriert

4. Genie IIs-CP/M: Besonderheiten

Beim Genie IIs-CP/M handelt es sich um eine CP/M-Version, die stark an das Montezuma-CP/M für das Model IV angelehnt wurde. Allerdings wurden nicht alle Features übernommen. Hier nur die für die Programmierung wichtigen Einzelheiten.

4.1 DPB-Anhängsel

DPB+15: Zeiger auf DCB für das Laufwerk (WORD)
DPB+17: Hardware-Byte

Der DCB enthält nur drei Byte-Einträge: Select-Bits für das Laufwerk, momentaner Track (0ffh=keiner), momentane Select-Maske (mit Seitenanwahl).

Das Hardware-Byte sorgt für korrektes Lesen von Fremd-Formaten. Die Bedeutung der Bits:

7-5: noch unbelegt
4: Schreibdichte (0=einfache, 1=doppelte)
3: Doppelstep (0=nein, 1=ja, für 40er-Disks in 80er-Laufwerken)
2: Track-Zählung (0=gleiche Track-Nr. auf Vorder- und Rückseite, 1=ungerade Track-Nummern auf der Rückseite)
1/0: phys. Sektorgröße (00=128, 01=256, 10=512, 11=1024 Byte je Sektor)

4.2 Tastatur-Codes

Beim IIs übernimmt die Taste "P1" die Funktion der Control-Taste. Shift P1 stellt Groß-/Kleinschreibung um. Die anderen Funktionstasten liefern folgende Codes:

Taste	entspricht ohne Shift	mit Shift
Hochpfeil	CTRL-K (0bh)	CTRL-^ (ESC, 1bh)
Senkr.pfeil	CTRL-J (LF, 0ah)	CTRL-Z (EOF, 1ah)
ENTER	CTRL-M (CR, 0dh)	CTRL-M (CR, 0dh)
EOF	7fh, DEL	5fh, " "
Linkspfeil	CTRL-H (BS, 08h)	CTRL-X (18h)
Rechtspfeil	CTRL-I (TAB, 09h)	CTRL-Y (TAB, 19h)
CLEAR	CTRL-Q (11h)	CTRL-R (12h)
BREAK	CTRL-C (03h)	CTRL-C (03h)
ß	7eh, "ß"	5eh, "^"
P2	-	-

5. Terminals

CP/M war ursprünglich ein Terminal-Betriebssystem, d.h. es lief auf einem Hauptrechner, an den ein oder mehrere Terminal(s) angeschlossen war(en). Je nach Terminal wurden verschiedene Tastatur-Codes geliefert und (was wichtiger ist) verschiedene Reaktionen auf Bildschirmsteuer-Codes bei der Ausgabe verursacht. Das macht es für Programmierer schwer, Programme zu schreiben, die auf jedem CP/M-Rechner lauffähig sind (z.B. mit welchem Code wird der Bildschirm gelöscht?). Daher bieten die meisten Programme eine Anpassung an verschiedene Terminals, z.B. durch Installations-Programme wie TINST bei Turbo-Pascal. Bei heutigen Rechnern ist die Bildschirmansteuerung und die Tastatur zwar meist direkt integriert, aber es werden immer noch verschiedene Terminals simuliert, um CP/M gerecht zu werden. Deshalb muß auch heute der Programmierer die Terminal-Codes kennen, wenigstens die seines Rechners. Deshalb hier eine Auflistung gängiger Terminal-Codes. Ergänzung und Korrektur dringend erwünscht.

5.1 ADM-3A

Name	Code	Funktion
BELL	7	Ton ausgeben
BS	8	Backspace
TAB	9	zur nächsten Tabulator-Pos.
LF	10	in die nächste Zeile
VT	11	in die vorherige Zeile
CRT	12	ein Zeichen nach rechts
CR	13	an den Zeilenanfang
INVOFF	14	Invers aus
INVON	15	Invers an
EOL	21	lösche bis Zeilenende
INVTOG	22	Invers umschalten
EOS	25	lösche bis Bildschirmende
CLS	26	lösche Bildschirm u. HOME
ESC	27	ESC-Sequenz einleiten (s.u.)
HOME	30	Cursor an Bildschirmanfang

Mit der Escape-Sequenz kann der Cursor beliebig auf dem Bildchirm positioniert werden. Nach dem ESC-Code (27) muß ein "=" (61) zum Bildschirm geschickt werden; dieses Zeichen wird nicht ausgegeben. Dann folgen zwei Angaben: zuerst die Zeile und dann die Spalte, in die der Cursor positioniert werden soll. Dabei bedeutet jeweils ein Blank (32) den Wert 0. Also wird durch die Sequenz "27,61,35,40" der Cursor in der 4. Zeile (Zeile 3 von Null an gezählt) an der 9. Stelle (Spalte 8 von Null an gezählt) ausgegeben.

5.2 VT-52

Alle Steuercodes werden durch ein ESC (27) eingeleitet.

Code	Funktion
'A'	Cursor eine Zeile hoch
'B'	Cursor eine Zeile runter
'C'	Cursor rechts
'D'	Cursor links
'E'	Bildschirm löschen & HOME
'H'	HOME
'I'	Cursor hoch; scrollen, wenn nötig
'J'	Bildschirm ab Cursor löschen
'K'	Zeile ab Cursor löschen
'L'	Zeile einfügen
'M'	Zeile löschen (Rest hochscrollen)
'Y' (y-32) (x-32)	Cursor an Position x/y setzen
'b' Farbe	Schriftfarbe neu wählen
'c' Farbe	Hintergrundfarbe neu wählen
'd'	Bildschirm bis Cursor löschen
'e'	Cursor einschalten
'f'	Cursor ausschalten
'j'	Cursorposition speichern
'k'	Cursor auf gespeicherte Position setzen
'l'	Zeile löschen
'o'	Zeile bis Cursor löschen
'p'	Invers ein
'q'	Invers aus
'v'	überlauf an (Cursor geht über Zeilenende)
'w'	überlauf aus (Cursor bleibt am Zeilenende stehen)

(aus: Data-Becker-Führer für Atari-ST)

5.3 TeleVideo TV950

ASCII	Hex	Funktion
^G	07	Ton ausgeben
^H	08	Backspace
^I	09	TAB
^J	0a	Cursor runter
^K	0b	Cursor hoch
^L	0c	Cursor rechts
^M	0d	CR
^V	16	wie ^J, aber ohne Scroll
^Z	1a	Bildschirm löschen & HOME
^^	1e	HOME
ESC '('	1b 28	normale Leuchtstärke
ESC ')'	1b 29	halbe "
ESC '*'	1b 2a	CLS & HOME (auch '+', ',', ':', ';')
ESC '.' n	1b 2e n	Cursor-Attribute setzen, n= '0' (30h): Cursor aus '1' (31h): Cursor als Block (blinkend) '2' (32h): " (nicht bl.) '3' (33h): Cursor als Unterstrich (blinkend) '4' (34h): " (nicht bl.)
ESC '=' s z	1b 3d s z	Cursor auf Spalte s-20h, Zeile z-20h
ESC '?'	1b 3f	Cursor-Pos. abfragen
ESC 'D' x	1b 44 x	Modus setzen: 'L'=lokal, 'H'=halb-duplex, alles andere: voll-duplex
ESC 'E'	1b 45	Zeile einfügen
ESC 'G' x	1b 47 x	Zeichen-Attribute setzen (x-30h), Bits: 6: Grafik-Modus (Zeichen als Grafik) 5: doppelt hoch 4: doppelt breit 3: unterstrichen 2: invers 1: blinkend 0: unsichtbar
ESC 'I'	1b 49	zurück zum letzten Tabulator
ESC 'Q'	1b 51	Zeichen bei Cursor einfügen
ESC 'R'	1b 52	Zeile löschen
ESC 'T'	1b 54	Zeile ab Cursor löschen
ESC 'U'	1b 55	Control-Modus ein (alle Steuerzeichen als Buchstaben mit halber Leuchtstärke)
ESC 'W'	1b 57	Zeichen löschen
ESC 'Y'	1b 59	bis Bildschirmende löschen
ESC 'b'	1b 62	Bildschirm invertieren
ESC 'd'	1b 64	Bildschirm normal
ESC 't'	1b 74	Zeile ab Cursor löschen
ESC 'y'	1b 79	bis Bildschirmende löschen
ESC 'z' n	1b 7a n	Zeichensatz wählen von '0' (30h): USA über Frankreich, Deutschland, England, Dänemark, Schweden, Italien bis '7' (37h): Spanien
ESC 'ä' ...	1b 7b ...	Schnittstelle initalisieren 1. Parameter: Baudrate ('?'=19200, '0' oder '>'= 9600, '<'=4800, ':'=2400, '8'=1200, '7'=600, '6'=300, '5'=150, '3'=110) 2. Parameter: '0'=ein, '1'= zwei Stopbits 3. Parameter: Parity ('0'=aus, '1'=ungerade, '3'=gerade) 4. Parameter: '0'=8-, '1'=7-Bit-Wortlänge

(aus: c't-Artikel über c't-Text-Terminal)

Anhänge

A) Beispielprogramm für BDOS-Aufrufe

```
wboot   EQU     0000h           ;Warm-Boot-Ansprung
bdos    EQU     0005h           ;BDOS-Aufruf-Ansprung
bconin  EQU     1               ;hole Zeichen
bconout EQU     2               ;gib Zeichen aus
prstr   EQU     9               ;gib String aus
getstr  EQU     10              ;hole String

start   LD      DE,string       ;Zeiger auf String
        LD      C,prstr         ;BDOS-Fkt.-Nr.
        CALL    bdos            ;String ausgeben
        LD      C,bconin        ;BDOS-Fkt.-Nr.
        CALL    bdos            ;Zeichen holen
        LD      E,A             ;Zeichen nach E
        LD      C,bconout       ;BDOS-Fkt.-Nr.
        CALL    bdos            ;Zeichen ausgeben
        LD      DE,buffer       ;Zeiger auf Buffer
        LD      C,getstr        ;BDOS-Fkt.-Nr.
        CALL    bdos
        LD      A,(buffer+1)    ;Länge holen
        OR      A               ;=0?
        JR      Z,ab            ;ja, ab
        LD      L,A             ;Länge nach HL
        LD      H,0
        LD      DE,buffer+2     ;Zeiger setzen
        ADD     HL,DE           ;Zeiger auf Buffer-Ende
        LD      (HL),'$'        ;Ende-Zeichen setzen
        LD      C,prstr         ;BDOS-Fkt.-Nr.
        CALL    bdos            ;Eingabe nochmal anzeigen
ab      JP      wboot           ;Abgang
;oder:  RET                     ;direkt zum CCP

string  DM      0dh,0ah,'Dies ist ein Test','$'
buffer  DB      5               ;Länge des Buffers
        DB      0               ;tatsächlich eingegeben
        DS      6               ;Buffer für Zeichen
```

B) Begriffserklärungen

In der Anleitung habe ich einige Begriffe und Abkürzungen verwendet, die nicht jedermann geläufig sein könnten. Deshalb hier einige Erläuterungen. Außerdem habe ich noch einige Begriffe aufgenommen, die im Zusammenhang mit CP/M (bzw. Computern allgemein) verwendet werden. Viele Informationen stammen aus der englischen Ausgabe des "Handbuch zum CP/M" von Digital Research.

BAT:, Batch Batch nennt man das Hintereinanderablaufen von mehreren Programmen, ohne daß eine Person eingreift, wobei die Aktionen (Ein-/Ausgaben) auf einem Drucker protokolliert werden; BAT: bezeichnet im CP/M einen Zustand, in dem alle Eingaben von Reader kommen und die Ausgaben auf List gehen; ->I/O-Byte

BDOS Basic Disk Operating System; das BDOS ist der Teil des Betriebssystems CP/M, der die Disketten-Ein-/Ausgabe für alle CP/M-Rechner standardisiert; es ist deshalb bei allen CP/M-Rechnern gleich und wird von Programmierern intensiv genutzt; das BDOS benutzt das BIOS; -> Anhang C

BIOS Basic Input/Output System; das BIOS ist der "unterste" Teil von CP/M; es regelt die Ein-/Ausgabe speziell für den Rechner, auf dem ein CP/M läuft; es muß für jeden Rechner neu programmiert werden; auf das BIOS sollte man möglichst nicht direkt zugreifen; -> Anhang C

Block eine Organisationsgröße auf der Diskette; die Größe wird im DPB festgelegt und kann 1K, 2K, 4K, 8K oder 16K betragen

Blocking ein Verfahren, das hilft, auf Disketten schneller zuzugreifen; da CP/M nur 128-Byte-Sektoren kennt, die Disketten-Systeme aber oft größere Sektoren (256, 512, 1024 Byte) unterstützen, werden mehrere logische Sektoren in einen physikalischen Sektor gepackt; dieses Packen wird Blocking genannt; -> Deblocking

CCP Console Command Processor; dieser Teil des Betriebssystems ist eigentlich ein Programm, das nur dazu dient, dem Benutzer eine Benutzeroberfläche zur Verfügung zu stellen, mit der er leicht die Disketten verwalten kann; es interpretiert die Benutzerkommandos (ERA, DIR usw.) und setzt sie in konkrete Aktionen um; der CCP ruft Anwenderprogramme auf und diese können, nachdem sie abgelaufen sind, in den CCP zurückkehren, falls sie ihn nicht überschrieben haben; falls sie ihn überschrieben haben, müssen sie mit einem Warm Boot enden, der den CCP neu lädt; es gibt verschiedene CCP-Versionen, die alle das Merkmal haben, daß sie direkt unter dem BDOS im Speicher stehen und laufen; -> Anhang C

Checksum -> Prüfsumme

CLS Clear Screen; den Bildschirm löschen

Cold Boot Kaltstart heißt, daß das Gerät noch nicht "warm" ist bzw. erst gerade eingeschaltet wurde und jetzt erstmal die grundsätzlichen Initialisierungen durchzuführen sind; normalerweise sollen danach nur noch Warmstarts (Warm Boot) durchgeführt werden; wenn Sie RESET geben, entspricht dies einem Kaltstart

COM Extension (z.B. TURBO.COM) von direkt aufrufbaren Programmen, d.h. man muß in der Benutzeroberfläche (CCP) nur den Namen des Programms eingeben, um es zu starten; COMmand-Programme werden immer nach 100h geladen und dort gestartet

Console, CON: Tastatur und Bildschirm in modernen CP/M-Systemen, Teletype (Fernschreiber) oder sonstwas bei älteren; über die Console werden Eingaben bzw. Befehle geholt und Ausgaben gemacht; über CON: kann die Ein-/Ausgabe auf andere Geräte umgeleitet werden; -> I/O-Byte

CP/M Control Program for Microcomputers; unser Thema

CR Carriage Return; an den Anfang der Zeile gehen; beim Newdos gleichzeitig auch "in die nächste Zeile", was totaler Quatsch ist (aber sehr effizient)

CRT: Cathode Ray Tube; ein Bildschirm, wie ihn heute jeder benutzt

DCB Device Control Block; ein Datenblock, der direkt die Eigenschaften eines physikalischen Geräts beschreibt bzw. beeinflußt; unter dem DPB anzusiedeln und oft auch für Tastatur etc. vorhanden; nur in bestimmten BIOS-Versionen vorkommend

Deblocking das Umgekehrte des Blocking: aus einem großen physikalischen Sektor den vom CP/M angeforderten 128-Byte-Sektor raussuchen; wird im BIOS erledigt

Default bezeichnet einen Wert, der angenommen wird, falls nichts anderes an-/eingegeben wurde; z.B. ist das Default-Laufwerk bei Diskettenzugriffen (wie bei "DIR") immer das eingeloggte Laufwerk, dessen Kennung als Prompt vom CCP ausgegeben wird ("A>")

DMA Direct Memory Access; im CP/M die Adresse, an der ein 128 Byte langer Buffer existiert, in den Daten von der Diskette eingelesen bzw. von dem Daten auf die Diskette geschrieben werden; im anderen Zusammenhang ein Baustein, der LDIR-ähnliche Befehle über den 64K-Speicher hinaus ausführt

DPB Disk Parameter Block; der DPB gibt die Charakteristika eines bestimmten Speichermediums (z.B. Diskettenlaufwerk) wieder; falls mehrere Diskettenlaufwerke gleichen Typs in einem System vorhanden sind, muß für alle zusammen nur ein DPB existieren, auf den dann mehrere DPHs zeigen; da die DPB-Einträge nicht für alle Formate ausreichende Informationen enthalten, werden sie je nach CP/M-BIOS-Version durch zusätzliche Einträge ergänzt, die leider keinerlei Normung unterliegen

DPH Disk Parameter Header; eine Datenstruktur, die über dem DPB angesiedelt ist (d.h. einen Zeiger auf einen DPB enthält); der DPH besteht u.a. aus Zeigern auf verschiedene Buffer, die das BDOS benutzt; für jedes logische Laufwerk im System (A:, B: usw.) muß ein eigener DPH existieren

EOF End Of File; ein reservierter Steuer-Code (1ah), der anzeigt, daß ein ASCII-File hier endet; oft auch die Tastenkombination Ctrl-Z oder Fehlermeldung

EOL End Of Line; Ende der Zeile; wie EOS verwendet

EOS End Of Screen; Ende des Bildschirms; oft im Zusammenhang "to EOS", z.B. bei "löschen bis Bildschirmende"

ESC Escape; besonder Taste bzw. besonderer Steuer-Code (1bh); wird benutzt, um eine Folge von Zeichen anzukündigen, die zur Steuerung eines Gerätes (Bildschirm, Drucker) dienen und nicht ausgegeben werden sollen

Extent Directory-Eintrag aus 32 Byte; Byte 0-15: Angaben zum File (Name, User-Bereich, Extension usw.); Byte 16-31: vom File belegte Blöcke (Nummern je 2 Byte); falls mehr als 8 Blöcke belegt werden, wird ein zweiter Directory-Eintrag für den File angelegt; nach Digital Research "16 K aufeinanderfolgende Byte in einem File"

FCB File Control Block; Daten-Block, über den der Programmierer auf einen File zugreift; im FCB stehen u.a. Name, Extension und Laufwerk, momentane Record-Nr. usw.

Flag Flagge; ein Bit, das etwas anzeigt, z.B. "der eben berechnete Wert war Null"; -> Overflow

HOME nach Hause (telefonieren, s. ET); Cursor in die linke obere Ecke des Bildschirms setzen bzw. Schreib-/Lesekopf des Diskettenlaufwerks über Track 0 positionieren

Interleave -> Skew-Faktor

I/O-Byte Input/Output-Byte; ein Byte in der Zero-Page, mit dem logische Geräte auf verschiedene physikalische umgeleitet werden können, falls dies Ihr CP/M-BIOS unterstützt; z.B. kann das logische Gerät CON: (Console) die Eingaben von unterschiedlichen physikalischen Geräten erhalten; läßt sich am besten per STAT ändern

K Kilobyte; 1024 Byte

Kaltstart -> Cold Boot

LF Linefeed; eine Zeile weiter gehen (Cursor oder Druckkopf); meist nach CR zu finden, um einen Zeilenvorschub zu gewährleisten

LIST, LST: das Gerät LIST bzw. LST: ist eines der logischen Geräte, die CP/M kennt und ist normalerweise ein Drucker

loggen meist: ein-/ausloggen; im CP/M: eine neu eingelegte Diskette zum ersten Mal ansprechen bzw. sie dem System bekanntmachen; bei Großrechnern: die Eingabe von Benutzername und -paßwort, bevor man anfangen darf, zu arbeiten

logisch alles, was nicht physikalisch ist; für das CP/M existieren nur logische Geräte, die je nachdem unterschiedlichen physikalischen Geräten zugeordnet sein können; eine Console-Ausgabe, die normalerweise auf einen Bildschirm gelangen würde, kann so auf einem Drucker erscheinen; außerdem: logische Sektoren sind Sektoren mit 128 Byte, die CP/M als einzige kennt; das Speichermedium "dahinter" (z.B. ein Laufwerk) kann ruhig mit größeren Sektoren arbeiten

LPT: Line Printer; das physikalische Gerät, das meist LST: zugeordnet wird

LSB Lowest Signifikant Byte; die unteren 8 Bit eines 16-Bit-Wortes; manchmal auch: Lowest Signifikant Bit, Bit 0 eines Bytes

MP/M Multi-Programming Monitor Control Program; der große Bruder des CP/M; für den Betrieb mehrerer Terminals an einem Haupt-Rechner

MSB Most Signifikant Byte; obere 8 Bit in einem 16-Bit-Wort; manchmal: Most Signifikant Bit, Bit 7 eines Bytes

Nibble obere oder untere 4 Bit in einem Byte

Offset fester Wert, der zu jeder Variable (z.B. Adresse oder Track-Nr.) addiert wird

Overflow Überlauf; falls eine Rechnung oder ein Wert eine gesetzte Grenze überschreitet; meist durch ein Flag angezeigt

Page Seite; 256 Byte im Speicher, die allein durch Änderung eines Bytes, des LSB, adressiert werden können (00-ffh)

PTP:, PTR: irgendwelche physikalischen Geräte; P=Punch, R=Reader; -> I/O-Byte

Prüfsumme eine Quersumme oder ähnliches von mehreren Bytes; wird beim CP/M vom Directory gebildet, um festzustellen, ob die Diskette gewechselt wurde

Punch, PUN: Lochkartenstanzer; früher gebräuchliches Gerät aus der Steinzeit der Rechenanlagen; im CP/M ein logisches Gerät, an dem z.B. physikalisch eine serielle Schnittstelle (mit Modem oder so) hängen kann; -> I/O-Byte

Random Zufall; Random-Files sind Daten-Files, bei denen man die Datensätze (Records) direkt adressieren (lesen/schreiben) kann; -> sequentiell

Reader, RDR: Lochkartenleser; s. Steinzeit der Rechenanlagen; logisches Gerät des CP/M; zum Lesen einer seriellen Schnittstelle nicht schlecht; -> I/O-Byte

Record Diskettenorganisationseinheit; je nachdem mal (physikalisch) ein Sektor von 128 Byte oder (logisch) ein vom Benutzer definierter Datensatz anderer Länge

R/W, R/O Read/Write, Read/Only; Attribut eines Files oder einer Diskette; je nachdem darf nur gelesen oder auch geschrieben werden; -> SYS

Sector Translation Table Sektor-übersetzungs-Tabelle; oft XLT abgekürzt; eine Tabelle, die logische Sektornummern in phsyikalische übersetzt; wird benutzt, um die Drehung der Disketten auszunutzen beim Lesen von hintereinanderliegenden Sektoren; dabei heißt "logisch" hier manchmal auch, daß es sich schon um physikalische (256, 512 ... Byte-) Sektoren handelt; -> Skew-Faktor

Select hier: Anwahl(-Bits); eine Bit-Maske, in der 1 Bit gesetzt ist, um ein bestimmtes Laufwerk anzuwählen; manchmal wird noch ein zweites Bit gesetzt, um bei diesem Laufwerk die Rückseite anzuwählen; unterste Ebene des Disketten-Zugriffs; allein dem BIOS überlassen

sequentiell hintereinander; Zugriffsform auf Files, bei der nur der jeweils nächste Datensatz (Record) gelesen werden kann; -> Random

Skew-Faktor Konstante, die auf die zuletzt adressierte physikalische Sektor-Nummer addiert wird, um den nächsten logischen Sektor zu erhalten; mit diesem Faktor wird die Sector Translation Table aufgebaut, um den Diskettenzugriff zu beschleunigen

String Ansammlung von Zeichen; beim CP/M komischerweise immer mit einem "$" abgeschlossen (Zusammenhang mit Turing-Maschine?)

SYS Attribut von Files, die unsichtbar und nur für den system-internen Gebrauch bestimmt sind; SYS-Files auf User 0 können nach Digital-Research-Informationen von jeder User-Nummer aus aufgerufen werden; -> R/W, R/O

System-Tracks eine bestimmte Anzahl Tracks (oft 2) am Anfang einer Diskette, auf denen das CP/M (Boot-Sektor, BIOS, BDOS, CCP) steht, das beim Booten von dort geladen wird; diese Tracks werden nicht als Diskettenplatz gerechnet, sondern per Offset übersprungen; dieser Offset kann auch benutzt werden, um aus einer physikalischen Diskette zwei logische zu machen (eine aus den ersten 40 Tracks und eine mit Offset 40 aus den oberen 40 Tracks oder so)

TAB Tabulator; ein Steuercode (09), der Cursor oder Druckkopf veranlaßt, zur nächsten durch 8 teilbaren Position zu springen

TPA Transient Program Area; Bereich von 100h bis zum Anfang des BDOS, in den die Anwenderprogramme geladen und ausgeführt werden; der CCP steht auch in diesem Bereich; -> Anhang C

TTY: Teletype; Fernschreiber; uraltes Gerät, das als Console zur Ein-/Ausgabe benutzt wurde

UC:, UL:, UP:, UR: physikaliche Geräte, die "U"ser-definiert sind; nicht mit "USER-Bereich" zu verwechseln!; C=Console; L=List; P=Punch; R=Reader

User Benutzer; hier: es gibt im Directory verschiedene Bereiche (0-15), die jeweils nur zugänglich sind, wenn vorher der entsprechende Userbereich (z.B. per Kommando "USER 1") angewählt wurde; zur Benutzung bei Mehrplatzsystemen bzw. als Urform des Sub-Directory gedacht

Warm Boot soll heißen: jetzt ist das Gerät schon warm (länger am Laufen) und muß nicht mehr voll initialisiert werden; die meisten Programme enden mit einem Warm Boot, der dann CCP und evtl. auch BDOS lädt und den CCP startet

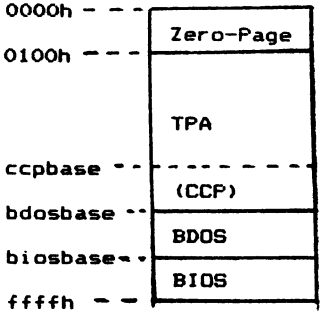
Warmstart -> Warm Boot

Wildcard ein Platzhalter in einem String (z.B. Filenamen) für einen beliebigen Buchstaben oder Teil; im CP/M "?" und "*"

Word, Wort 16-Bit-Datum bzw. "Informationseinheit aus 16 Bit"

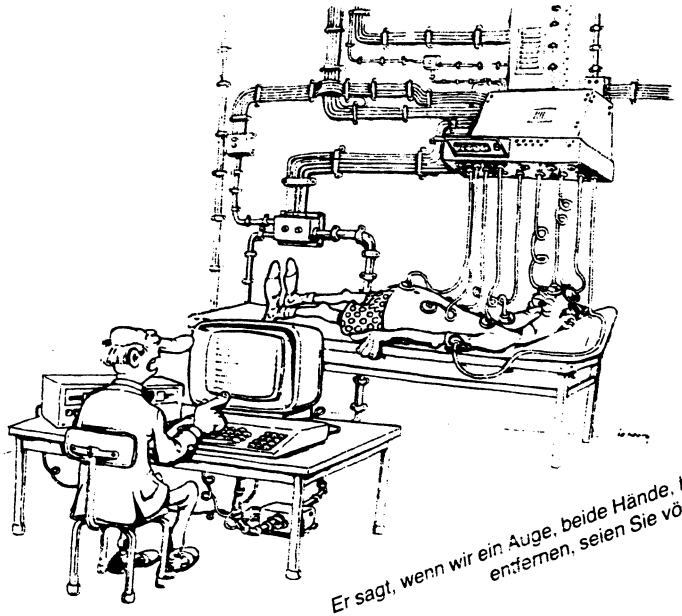
Zero-Page Seite 0; die ersten 100h im Speicherbereich; ursprünglich von Rechnern stammend, bei denen sich dieser Bereich leichter oder schneller adressieren läßt (6502); im CP/M als Schnittstelle von Anwenderprogramm zu BDOS benutzt; -> Anhang C

C) Der Speicheraufbau



Zero-Page

```
0000h         Warm Boot
0003h         I/O-Byte
0004h         User/Default-Lw.
0005h         BDOS-Ansprung
0008h-0027h RST-Vektoren 1-5 (unbenutzt)
0030h-0037h RST 6 (reserviert)
0038h-003ah RST 7 (bei Debug über DDT oder SID: Breakpoint-Entry)
003bh-003fh reserviert (unbenutzt)
0040h-004fh für BIOS reserviert (je nach BIOS-Version benutzt)
0050h-005bh reserviert (unbenutzt)
005ch-006bh 1. Filename bei Aufruf "*.COM file1 file2";
            bis 007fh: FCB zur freien Benutzung
006ch         2. Filename          - " -
0080h         Default-DMA-Buffer
```



Er sagt, wenn wir ein Auge, beide Hände, beide Füße und beide Ohren entfernen, seien Sie völlig normal.

Mein Programm sagt, wir werden hier nie Öl entdecken . . .

d B a s e I I

Eine kurze Einleitung in das Datenbanksystem

v. Harald Mand

Folge II :

4. Summieren von Feldern :

Mit dem Befehl ". S U M " lassen sich nummerische Datenfelder von Datensätzen über mehrere Sätze hinweg aufsummieren.

Beispiel : Berechnung der Summe aller Einkaufspreise

```
USE INVENTUR      =====> Eröffnung d. Datenbank
SUM EK:PREIS      =====> Aufsummierung aller
                         Datensätze
```

* Bis zu 5 Datenfelder bezw. Ausdrücke lassen sich jeweils Satz für Satz addieren und das Ergebnis getrennt in jeweils einer Speichervariablen ablegen.

Beispiel : SUM MENGE, EK:Preis, VK:PREIS, DIFFERENZ, B:PREIS

Speicherinhalte werden mit Hilfe des Befehls

" D I S P L A Y M E M O R Y " angezeigt.

5. Suchen eines Datensatzes :

Es lassen sich bestimmte Sätze mit Hilfe der folgenden Befehle auffinden :

" F I N D "

Mit diesem Befehl läßt sich ein bestimmter Satz in einer indizierten Datenbank auffinden.

* Der "FIND"-Befehl läßt sich nur anwenden, wenn der Schlüssel (z.B. Name), nach dem man suchen will, aktueller Hauptschlüssel der aktivierten Datenbank ist.

Es genügt ggf. nur den Anfangsbuchstaben des Namens einzugeben.

Der Satz, der durch einen "FIND"-Befehl lokalisert wurde, wird automatisch zum aktuellen Satz. Er läßt sich mit Hilfe des Befehls "DISPLAY" anzeigen.

Beispiel : Der Name "Meier" soll aus einer indizierten Datenbankdatei "ADRESSE1" lokalisiert werden.

```
USE B: ADRESSEN INDEX ADRESSE1
FIND MEIER
DISPLAY
```


Mit dem Befehl " L O C A T E " läßt sich ein bestimmter Satz in einer Datenbank auffinden.

* Ist der "LOCATE"-Befehl erfolgreich, erscheint der Hinweis "Satz n" auf dem Bildschirm.

Locate arbeitet am schnellsten auf nicht indizierten bezw. ohne Schlüsseldateien aktivierten Datenbanken.

Beispiel : Aus der Datenbank ADRESSEN soll die Adresse von Frau Müller gefunden werden.

```
USE B : ADRESSEN
LOCATE FOR "MüLLER"$NAME
SATZ : XX
DISPLAY
```

oder es sollen die Adressen herausgefunden werden, deren PLZ größer als 5000 ist.

```
LOCATE FOR PLZ>5000
SATZ : XX
DISPLAY
```

Der Befehl " C O N T I N U E " stellt die Fortsetzung des LOCATE-Befehls dar.

* Zwischen den Befehlen "LOCATE" und "CONTINUE" können ggf. auch andere Befehle zur Ausführung gebracht werden.

Beispiel :

```
USE B : ADRESSEN
LOCATE FOR PLZ>5000
SATZ : XXXX
DISPLAY

CONTINUE
SATZ : XXXX
DISPLAY
CONTINUE
.
.
DATEIENDE ERREICHT
```

6. Positionieren auf einen Datensatz :

Mit dem Befehl " G O " oder " G O T O " läßt sich der dbase-interne Zeiger auf einen bestimmten Datensatz neu positionieren.

* Der Befehl "DISPLAY" zeigt den neuen Datensatz an.

```
Beispiel : USE B : ADRESSEN   od.  GO TO       ===> erster Daten-
           GO TO RECORD 6                           satz
           DISPLAY
                              od.  GO BOTTOM ===> letzter Daten-
                                                  satz
```

7. Einfügen von neuen Datensätzen :

Der Befehl " I N S E R T " ermöglicht das Einfügen eines Satzes an einer liebigen Position der aktivierten Datenbank.

* Es kann mit jedem Befehl nur ein Datensatz in die Datenbank eingeben werden.
  Bei großen nicht indizierten Datenbanken ist der " I N S E R T " Befehl sehr zeitaufwendig.

Beispiel 1 : Einfügen hinter dem aktuellen Datensatz

Beispiel 2 : Einfügen vor dem aktuellen Datensatz

Beispiel 3 : Einfügen eines leeren Datensatzes

8. Zählen von Datensätzen :

Die Anzahl der Sätze in einer Datenbank werden mit dem Befehl " C O U N T " gezählt.

* dbase zeigt das Ergebnis in folgender Form an

  " Anzahl der Sätze = xxxxx"

Beispiel 1 : Die Anzahl der Sätze in der Datenbank "ADRESSEN" soll gezählt werden.

```
. USE B:ADRESSEN
. COUNT
ANZAHL DER SATZE = 00008
```

Beispiel 2 : Es sollen in der Datenbank "ADRESSEN" nur die Sätze gezählt werden, deren Postleitzahlen größer als 2300 sind.

```
.USE B:ADRESSEN
.COUNT FOR POSTLZ >2300
ANZAHL DER SATZE = 00012
```

9. Überspringen von Datensätzen :

Mit Hilfe des " S K I P " Befehls kann der Zeiger, der in dbase auf den aktuellen Datensatz zeigt, auf vorhergehende bezw. nachfolgende Sätze gesetzt werden.

Beispiel 1 :

```
.USE B:ADRESSEN
.LIST
.5
.DISPLAY
.SKIP 3
.DISP
.SKIP-6
.DISP
.SKIP 2*2
.DISP
```


10. Sortieren der Datenbankdatei :

Der Befehl " S O R T " bewirkt die Sortierung der aktivierten Datenbank.

* Sortiert wird nach dem angegebenen Datenfeld
  Ausgabedatei ist eine neue anzugebende Datei
  Die aktivierte Datenbank wird nicht geändert und bleibt weiter im Zugriff.

```
.SORT ON feld TO dateiname
```

Beispiel :

```
.USE B:ADRESSEN
.SORT ON POSTLZ TO ADRESSE1
.USE B:ADRESSE1
.LIST
```

11. Umbenennung von Datenbankdateien :

Beliebige Dateien in dbase lassen sich mit dem Befehl " R E N A M E " umbenennen.

* Die Voreinstellung für den Dateityp ist ".DBF" .
  Eine umzubenennende Datei darf nicht geöffnet (USE-Befehl) sein.

```
RENAME B:alter Name TO B:neuer Name
```

Beispiel : .RENAME B:ADRESSEN TO B:ANSCHRIFT

12. Anzeige von Datenbanken :

Mit Hilfe dieses Befehls " D I S P L A Y F I L E S " lassen sich alle auf dem angegebenen Laufwerk vorhandenen Dateien vom Typ ".DBF" anzeigen.

Beispiel 1 : alle "DBF" - Dateien von Laufwerk B sollen angezeigt werden.

```
:DISPLAY FILES ON B:
```

Beispiel 2 : alle Dateien von Laufwerk B sollen angezeigt werden.

```
DISPLAY FILES ON B: LIKE *:*
```

weitere Möglichkeiten :

```
nur Indexdateien  : ......LIKE  *.NDX
nur Reportdateien : ..... LIKE  *.FRM
```

Das war der 2. Abschnitt über dbase II . Er befaßte sich mit den notwendigen Befehlen zu Veränderungen und Sortierungen von Datenbanken. Beim nächsten Info geht es dann weiter mit der Erstellung von kpl. Programmen, sowie noch weiteren notwendigen Befehlen.

Ich wünsche Euch viel Erfolg bei der Anwendung.

Harald

Maus mit sieben Beinen

In der 22. Ausgabe des Clubinfos stellt Hartmut eine kleine Platine vor, mit deren Hilfe sich eine nicht seriell arbeitende Maus an unsere Computer anschließen läßt. Wer nur einen Funken Moral im Leibe und einen Hunni übrig hat, baut das Ding natürlich nach, so auch ich. Die hundert Mark gelten inklusive der Maus und aller Teile, versteht sich.

Der elektronische Schalter 4066 war bei meinem Dealer nicht vorrätig. Statt dessen gab er mir zwei Exemplare des 4016, der lt. Beschreibung praktisch identisch ist. Damit lief die Schaltung aber nicht. Daher möchte ich empfehlen, für die von Hartmut vorgeschlagenen Teile keine Ersatztypen zu nehmen.

Wir sehen im Info, daß Hartmut zwei Taster seiner Maus benutzt. Meine hat auch zwei, die jedoch einfach miteinander kurzgeschlossen waren. Hartmuts auch, aber er hat die Verbindung aufgetrennt und eine noch ungenutzte Leitung des Kabels damit bedröhnt. Nach dieser Anregung suchte und fand ich sogar noch zwei freie Leitungen, so daß ein weiterer Taster angeschlossen werden konnte.

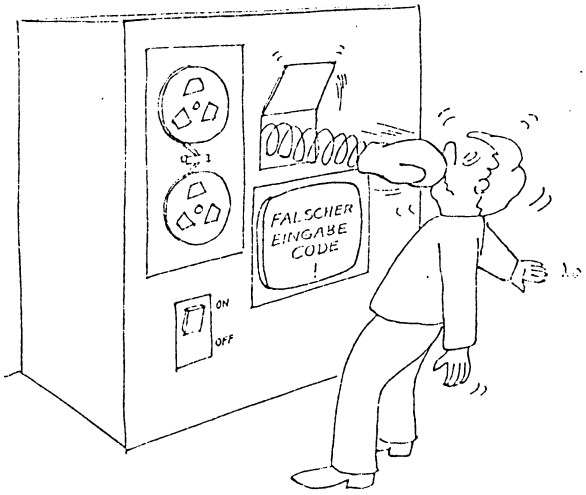
Hier war Bastelei in der Maus nötig: Zunächst mußte Hartmuts Operation durchgeführt werden, um die beiden bereits vorhandenen Tasten einzeln nutzbar zu machen. Für einen dritten Knopf fand sich Platz auf der kleinen Zusatzplatine, auf der die beiden Drucktasten sitzen. Die dafür notwendige Bohrung trennte eine Leitung auf, die jedoch mit einem Stück Draht wieder geschlossen werden konnte. Es handelt sich dabei um die Common-Leitung aller Tasten und der Rollenelektronik. Daher mußte sie auch an die neue Taste angeschlossen werden. Die letzte freie Leitung des Kabels kam an ihren zweiten Pin.

Elektronisch gab es also keine Schwierigkeiten. Mechanisch ist die Geschichte jedoch etwas heikel. Der einzubauende neue Taster braucht natürlich Raum. Glücklicherweise fand sich einer, dessen unten aus der Tastenplatine herausragender Teil mit knapper Not noch nicht die darunter liegenden Bauteile der Hauptplatine berührte. Es mußte dafür aber bereits ein wenig geschliffen werden. Sein Druckknopf steht so hoch über, daß er durch eine niederzubringende Bohrung im Gehäusedeckel gerade etwa 3 mm herausragt. Das ist eine optimale Arbeitshöhe.

Wer das nachbauen möchte, sollte sich unbedingt in seiner Maus sehr sorgfältig nach passenden Plätzen umsehen. Es ist auch nicht verkehrt, vorsichtshalber zwei oder drei verschiedene kleine Taster zur Auswahl bereitliegen zu haben. Wenn eine so glückliche Anordnung wie in meinem Exemplar nicht möglich ist, geht aber auch die einfachste Lösung: Die Taste wird über einem genügend großen Hohlraum, wovon es in allen Mäusen genug gibt, direkt in den Gehäusedeckel geschraubt. Um den Deckel ganz vom Boden trennen zu können, empfiehlt sich dann eine Steckverbindung zur übrigen Elektronik.

Leider mußte nun auch eine neue Obermann-Platine her, denn für sieben Anschlüsse hat der 7404 einen Inverter zuwenig. Ein zweiter Chip mußte sein. Außerdem mußte die Steckverbindung von dieser Platine zur Tastatur um zwei Pins verbreitert werden, die auch nicht mehr auf die sehr kleine alte Platine paßten. Im Prinzip hat sich aber nichts an dem von Hartmut vorgestellten Schaltungsvorschlag geändert, so daß die neue Schaltung hier nicht vorgestellt werden muß.

Freilich kann ich nur für meine eigene Maus (M1 von Noris Data) versichern, daß die Umbauten problemlos machbar sind. Hartmuts Maus sieht inwendig genauso aus, wahrscheinlich alle anderen auch. Man kann auf der Ladentheke schlecht dem Teil mit dem Schraubenzieher zuleibe gehen, um das zu überprüfen, aber die Chance, ein geeignetes Modell zu erwischen, dürfte bei 100% liegen.



Möchte jemand meine alte Platine haben? Sie treibt die Rollenelektronik und eine Taste. Auf dem 7404 ist noch ein Inverter, und auf dem zweiten 4066 sind noch drei Relais frei, so daß Hartmuts Trick mit der getrennten Verbindung zwischen den beiden vorhandenen Tasten ohne Anbauten noch durchführbar ist. Ich möchte die Platine aber nur unbestückt hergeben, denn in Lübeck kriege ich keinen 4066. Bitte melden!

In der Praxis stellte sich heraus, daß die Impulse von der Rolle sehr rasch aufeinander folgen. Die Maus darf daher nur sehr langsam bewegt werden. Vielleicht sollte man jedes zweite Loch in den beiden Blendenscheiben der Rollenmechanik verstopfen, so daß nur noch halb so oft ein Impuls erfolgt. Dann wäre die Garantie für die Maus endgültig verwirkt. Aber das hat den begeisterten Fummler noch selten abgeschreckt.

Es ist eine Wonne, mit der Maus zu arbeiten. Gerade eben beim Korrekturlesen dieses Textes karriökere ich damit im Bildschirm kreuz und quer, Maus in der Rechten, Bier in der Linken, Augen nur noch auf dem Text, nicht mehr auf der Tastatur. Das Teil arbeitet nämlich nicht nur mit Graphikprogrammen zusammen, sondern mit allem, was die Pfeiltasten und die anderen Tasten aus der Keyboard-Reihe 3840h bespitzelt. Also auch mit TSCRIPS. Hartmut, Deine Idee war Sahne!

Arnulf Sopp

Noch ein Druckerpatch

von Manfred Held

Nein,nein keine Angst, nicht noch ein X'der Patch im System um irgendwelche Steuerzeichen oder Grafikzeichen darzustellen.
Diesesmal ein Hardpatch ????!

Folgendes Problem:
Bei Newdos, Gdos ect. wird CR in LF umgewandelt. Es wird an den Drucker also nur CR (0Dh) ausgegeben. Der Drucker muss so eingestellt werden, dass nach einem CR automatisch ein Zeilenvorschub gemacht wird. Man kann dem Betriebssystem zwar beibringen, dass CR und LF getrennt zum Drucker gebracht wird, dies wurde auch schon oeffter getan, aber mit Programmen, die eigene Druckertreiber haben, steht man wieder vor demselben Problem.

Anders beim Betriebssystem CP/M. CP/M sendet CR und LF getrennt. Es wird keine Umwandlung vorgenommen. Auch die Programme behandeln CR und LF als zwei getrennte Zeichen.

Im Betriebssystem CP/M unseres Computers gibt es einen Softschalter, der wahlweise LF zu 00h umwandelt. Somit koennen wenigstens die Programme unter CP/M, in begrenztem Umfang genutzt werden.

Damit die Programme in CP/M die Faehigkeiten des Druckers voll nutzen koennen, benoetigt der Drucker ein getrenntes CR und LF. Das steht aber im Wiederspruch mit dem Drucker, der eben andes eingestellt werden muss.

Die Loesung:
Laut Druckerhandbuch gibt es einen Schalter oder Jumper (Autolinefeed oder CR(AUTO FEED XT) oder sonst irgendwie) damit der Drucker nach Empfang des CR einen Zeilenvorschub ausfuehrt.
Wichtiger ist aber der Anschluss am Drucker. Laut meinem Handbuch hat der Drucker am Pin 14 des Centronics-Steckers einen Eingang mit der Bezeichnung "AUTO FEED XT". In der Beschreibung zu den Anschlusspins des Steckers heisst es :

> Empfaengt der Drucker einen CR-Code, wenn dieses Signal L ist, schiebt er das Papier nach Druckende automatisch um eine Zeile vor.

Das heisst, wenn L-Pegel an diesem Pin anliegt, und CR empfangen wird, fuehrt der Drucker einen Zeilenvorschub aus. Liegt H-Pegel an, wird CR und LF getrennt behandelt. Der Drucker fuehrt keinen Zeilenvorschub aus, wenn CR alleine empfangen wird sondern wartet bis LF empfangen wird.
Wichtig dabei ist , dass der Schalter CR (AUTO FEED XT) auf "Drucken ohne LF" steht.

Fuer unser Problem heisst das, unter Newdos, Gdos ect. muss der Eingang von Pin 14 auf L-Pegel liegen, fuer CP/M auf H-Pegel.

Man koennte jetzt einen Schalter in den Anschlusstecker einbauen, mit dem Pin 14 einmal auf Masse geschaltet wird oder auf +5V. Das hat aber den Nachteil, vergisst man den Schalter bei Betriebssystemwechsel umzuschalten, gibt es Schrott am Drucker.

Die bessere Loesung ist, man sucht sich ein freies Port und ein freies Pin an dem Port und verbindet das Pin mit dem Eingang am Drucker, es muss nur beachtet werden, dass die Ruhelage des Ports nach einem Reset, L-Pegel ist.
Der Rest ist Software.
Ich habe im Bootsector des CP/M's ein freies Plaetzchen gefunden und vier Byte eingepflanzt, die jetzt bei Arbeiten mit CP/M den Drucker umschalten.
Somit stehen mir alle Moeglichkeiten des Druckers unter Newdos und CP/M voll zur Verfuegung.

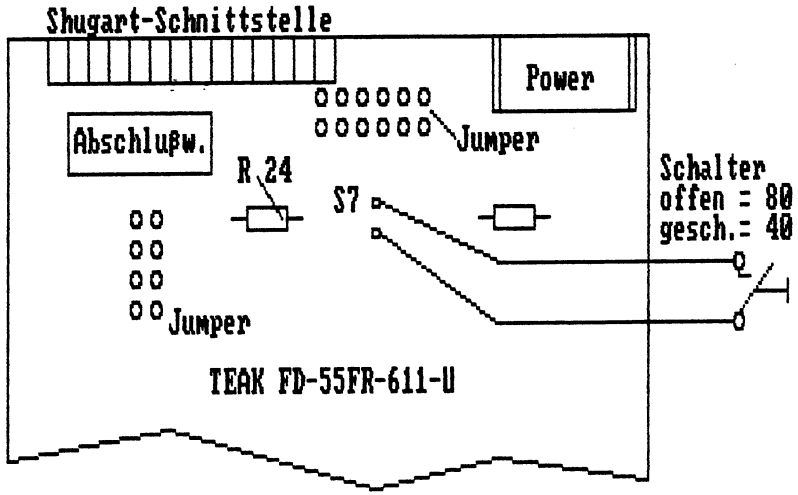
Wieder mal: 40 / 80 - Trackumschaltung diesmal: TEAK FD-55FR-611-U-Laufwerke

Die Firma TEAK scheint ständig bemüht ihre Laufwerksproduktion auf dem neuesten technischen Stand zu halten. Dadurch kommt aber mancher Käufer eines 80-Spur-TEAK-Laufwerks, von dem behauptet wird, daß es auf 40-Spurbetrieb umschaltbar ist, ganz schön in Schwierigkeiten. Immerhin erschienen allein im CLUB 80-Info schon zwei verschiedene Anleitungen für die Umschaltung von TEAK-Laufwerken.

Vor kurzem liefen mir zwei Laufwerke über den Weg, die man einem Bekannten als 40-Spurer verkauft hatte. Bei genauerer Betrachtung stellte sich heraus, daß es sich um Laufwerke mit der Bezeichnung FD-55FR-611-U handelte, die von einem anderen Anbieter als umschaltbare 80'er verkauft werden. Nach kurzer Suche war die Stelle gefunden, an der ein einfacher Ein/Aus-Schalter eingelötet werden muß, um sowohl 40 als auch 80 Tracks benutzen zu können.

Viel Spaß und Glück beim Löten, euer

Hartmut Obermann



Anmerkungen zum Club-80-ECB-Bus-Projekt

Liebe Club-Freunde, die Ihr den Mut gehabt habt, Euch an die Löterei des ECB-Bus-Projektes heranzuwagen, Euch speziell gilt der folgende Artikel zu dem Thema. Vorher möchte ich aber noch einige Bemerkungen loswerden, die verdeutlichen sollen, warum ich meine etwas abweichenden Vorstellungen eines ECB-Bus-Interfaces denen Eures Projektes entgegenstellen muß.

Die Grundvorstellung des ECB-Bus ist nicht mur das Umsortieren einer wilden Pinanordnung auf den Systembus-Steckern unserer Computer auf die noch unübersichtlichere des ECB-Bus sondern noch eine Reihe weiterer Srtrukturen, die speziell auf die besonderen Fähigkeiten der Z80-CPU abgestimmt sind und auch von einem vollständigen ECB-Bus unterstützt werden.

Dazu gehört zunächst der Interrupt Mode 2 (IM2), der voraussetzt, daß die Interrupt-auslösende Karte der CPU beim Interrupt-Acknowledge einen 8Bit-Vektor über den Datenbus mitteilt. Dieser Vektor muß vom ECB-Bus über den Datenbuffer auch zur CPU gelangen können, wenn IORQ* und M1* gleichzeitig low sind.

Außerdem müssen die Buffer aller Bussignale bidirektional sein, weil bei DMA-Zugriff auf Komponenten des CPU-Boards der DMA-Controller auf dem ECB-Bus auch die entsprechenden Baugruppen des CPU-Boards adressieren muß. Das bedeutet einen sehr viel größeren Aufwand bei der Steuerung der Treiberrichtung dieser Buffer.

Und schließlich soll der ECB-Bus nicht nur selbstgestrickte Karten aufnehmen sondern hauptsächlich den Einsatz der breiten Palette käuflicher Karten für ECB-Systeme ermöglichen. Daher verbietet sich auch die prinzipiell geniale Lösung, der Richtungssteuerung des Datenbuffers über Freigabesignale, die die ECB-Bus-Karten liefern. Kommerzielle Karten wissen nicht, daß Euer Bus solche Signale voraussetzt.

Das Konzept Eures ECB-Bus baut auf den Möglichkeiten des TRS-80-Systembus auf, die leider wirklich recht bescheiden sind. IM2- und DMA-Fähigkeit sind dort nicht vorhanden. Beim GENIE und beim Komtek 1 lassen sich aber alle Eigenschaften des ECB-Bus verwirklichen, ohne intern große Eingriffe machen zu müssen.

Um aber auch beim TRS-80 zumindest 90% der kommerziellen Karten, die ohne IM2 und DMA arbeiten, nutzen zu können, ist es unumgänglich, von der Richtungssteuerung des Datenbuffers durch die Karten selbst abzukommen. Da ist dann auch die bei mir eingeschlagene Ochsentour nötig, die Richtungssteuerung aus den Freigabesignalen der Baugruppen innerhalb des Computers herzuleiten.

Die im folgenden Artikel beschriebene ECB-Bus-Eingangkarte berücksichtigt nicht nur die o.a. Gesichtspunkte sondern bringt noch ein paar zusätzliche Leistungen, die nicht unbedingt an den ECB-Bus gebunden sind, sondern einige andere Hardware-Erweiterungen unterstützen. Diese Features sind aber nicht unbedingt wichtig und können auch ausgespart werden.

Es ist beim GENIE und Komtek ohne weiteres möglich, den ECB-Bus mit dieser Eingangskarte anstelle der alten zu betreiben. Beim TRS-80 sind noch zusätzliche Maßnahmen nötig, die einen voll funktionsfähigen ECB-Bus ermöglichen.

Helmut Bernhardt

Gepufferter DMA- und IM2-fähiger ECB-Bus für GENIE I und II

Helmut Bernhardt

Während bei Verzicht auf DMA- und IM2-Fähigkeit die Steuerung des Datenbuffers zwischen Computer und externem ECB-Bus relativ einfach zu realisieren ist (der Buffer darf nur in Richtung CPU treiben, wenn Daten von Baugruppen gelesen werden, die nicht auf der CPU-Seite des Buffers - also auf dem ECB-Bus - sitzen), sind bei der Einbindung dieser Features einige zusätzliche Aspekte zu berücksichtigen.

Der Interrupt-Mode2 der Z80-CPU setzt voraus, daß der Interrupt-Lieferant beim Interrupt-Acknowledge (M1* und IORQ* beide low) ein 8Bit-Wort auf den Datenbus legt, das die CPU als LSB einer Adresse in der Interrupt-Vektor-Tabelle interpretiert. Dieses Byte muß von einer Interrupt-Quelle vom ECB-Bus durch den Datenbuffer auch zur CPU gelangen. Der Buffer muß beim Interrupt-Acknowledge also auch in Richtung CPU treiben.

Bei übernahme der Daten-, Adreß- und Steuerbusse durch einen DMA-Controller (DMAC) auf dem ECB-Bus muß dieser auch Zugriff auf RAM, EPROM und alle CPU-seitig des Buffers liegenden Baugruppen haben. Für Adreß- und Steuerleitungen gestaltet sich diese Richtungssteuerung recht einfach. Hier genügt es, wenn durch das low aktive CPU-Signal BUSAK* (nur dann darf der DMAC arbeiten) die Buffer dieser Signale in Richtung zur CPU treiben, so daß der DMAC dort entsprechend adressieren kann. Für die Steuerung des Datenbuffers sind noch weitere Umstände zu berücksichtigen.

Wenn die CPU den Bus kontrolliert, kann beim Schreiben der Buffer immer in Richtung ECB-Bus treiben. Beim Lesen darf er nur dann nicht in Richtung CPU treiben, wenn interne Baugruppen (CPU-seitig des Buffers) adressiert werden. Sonst würde ein von außen gelesenes FFH gegen ein von der intern adressierten Baugruppe ausgegebenes Datenwort arbeiten. Abgesehen davon, daß die gegeneinander arbeitenden Komponenten sich irgendwann gegenseitig zerstören, würde die CPU Fehlinformationen lesen.

Wenn der DMAC (4,5) den Bus steuert, ist Lesen und Schreiben für die Treiberrichtung des Buffers das ganze Gegenteil. Wenn der DMAC Daten einer CPU-seitigen SystemKomponente liest, wobei er dann auch die Leitung RD* low zieht, muß der Buffer in Richtung ECB-Bus treiben. Das darf er aber nur dann, wenn er auf CPU-seitige Komponenten zugreift. Wenn er von einer Baugruppe auf dem ECB-Bus liest, muß der Buffer in Richtung CPU treiben, um dann nicht auf dem ECB-Bus zwei Datenwörter gegeneinander arbeiten zu lassen.

Für die Treiberrichtung des Buffers muß also berücksichtigt werden, ob der DMAC oder die CPU den Bus kontrolliert, ob die adressierte Baugruppe auf dem ECB-Bus oder CPU-seitig des Buffers liegt und ob gelesen oder geschrieben wird.

In Tabelle 1 sind die für verschiedene Betriebszustände aus obigen überlegungen abgeleiteten Treiberrichungen (von der CPU aus gesehen) zusammengestellt. Dabei ist zu berücksichtigen, daß zusätzlich auch noch beim Interrupt-Acknowledge die Richtung "rein" sein muß, weil der GENIE keine eigenen vektorisierten INT-Quellen hat. Diese Tabelle ist Grundlage der Schaltung in Abb.1. Die Richtungssteuerung soll ausschließlich aus Signalen, die von CPU-seitigen Baugruppen zu beziehen sind, erfolgen. Nur dann ist auch gewährleistet, daß auf dem ECB-Bus jede handelsübliche Karte funktioniert. Kommerzielle ECB-Bus-Karten legen kein aktives Signal auf den Bus, das in die Buffersteuerung einbezogen werden kann.

Tabelle 1: Treiberrichtung des ECB-Bus Databuffers

Bus-Steuerung durch	interne Baugruppe adressiert Read	interne Baugruppe adressiert Write	externe Baugruppe adressiert Read	externe Baugruppe adressiert Write
CPU	raus	raus	rein	raus
DMAC	raus	rein	rein	rein

Der Aufwand dafür ist aber nicht unerheblich und der Anwender dieser Bussteuerung muß sein System genau kennen, um die nötigen Steuersignale an die Schaltung legen zu können. Einen Anhaltspunkt dafür, welche Freigabesignale vorkommen können, mag die Tabelle 2 geben. Darin sind die prinzipiell vorhandenen Baugruppen und einige häufig benutzte Hardware-Erweiterungen und die Punkte, wo deren Freigabesignale abzugreifen sind, zusammengestellt. Diese Punkte sind mit jeweils einem der Punkte A bis H auf der Eingangskarte zu verbinden.

Tabelle 2: Interne Freigabesignale

CPU-Board	Adresse	IC	Typ	Pin	
RAM	4000-FFFF		37	367	15
	(3900-3BFF)	(7)			
ROM1	0000-0FFF		10	3001	20
ROM2	1000-1FFF		11	3002	20
ROM3	2000-2FFF		12	3003	20
ROM4	3000-36FF(37DF)	(7)	13	2716	20
Tastatur	3800-3BFF(38FF)	(7)	7	368	1
Video-RAM	3C00-3FFF		35	32	3
Video-Board					
Cassette 1	Port FF		20	32	11
Cassette 2	Port FE	bleibt unberücksichtigt #1			
Expansion-Interface	EG3014				
RAM1	8000-BFFF		37	20	10
RAM2	C000-FFFF		37	20	9
Floppy	37E0-37FF		29	139	12
Drucker	37E8 u.Port FD		31	156	9
RS 232 EG3020	Ports F8, F9		37	20	10
EXP1 von RB	37E0-37FF		18	155	2,14
	Port FD		14	32	3
RB.V24	Ports 80-88		Z80-CTC		16
			Z80-DART		35
HRG1B von RB	Ports 00-7F		?	155	2,14
Schmidtke 802.Karte	B000-BFFF		?	245	19
	Ports D0, D1		?	138	15
EG64MBA	Port DF	bleibt unberücksichtigt #1			
EG64MBA+ (6)	Ports DE, DF	"	"		"
256K-Banker (10)	Port EC	"	"		"
(u. ED mit dieser Karte)		"	"		"
CP/M-Banker (6)	Ports 50-5F	"	"		"

#1 diese Baugruppen werden nicht gelesen und brauchen deshalb nicht in die Buffersteuerung miteinbezogen zu werden, das IN A,(0DFH) des EG64MBA dient nicht der Datenaquisition, es setzt nur das 74LS259-Latch zurück, ein Datenwort wird nicht ausgegeben

Im Normalfall liegen alle memory-mapped (mm) Baugruppen CPU-seitig des Treibers, so daß dann einzig MERQ* (IC16, 74LS367, Pin5) an die Schaltung geführt werden muß. Wenn aber auch nur eine mm Baugruppe auf dem ECB-Bus steckt, müssen alle anderen mm Freigabesignale einzeln an die Schaltung geführt werden.

Durch Vorschalten weiterer AND-Gatter vor die Eingänge des 74LS30-ICs (z.B. 74LS08, 74LS11, 74LS21) kann die Anzahl der verwendbaren Eingänge für Freigabesignale beliebig erhöht werden (Abb.10).

Die DMA-Fähigkeit der ECB-Busbuffer alleine reicht nicht aus. Es muß durch Verbinden der Systembus-Leitung BUSAK* mit CCDBS/STADBS* und DODBS/ADDBS* auch dafür gesorgt werden, daß beim DMA-Zugriff die Buffer der CPU-Signale in den hochohmigen Zustand versetzt werden. Das kann entweder auf der Eingangskarte durch Querverdrahtung der entsprechenden Pins von CN2 bzw. CN3 oder durch feste Verdrahtung auf dem CPU-Board gemäß Abb.11 erreicht werden.

Da für die Richtungssteuerung des Datenbuffers ohnehin einige ICs nötig waren, und der ECB-Bus auch IM2-fähig ist, ließ sich mit etwas Mehraufwand auch gleich eine Vektorisierung der Interrupts des 25ms-Timers und des Floppy-Controllers realisieren. Dafür ist Voraussetzung, daß auf dem ECB-Bus auf einer Karte ein freier Z80-CTC-Kanal (2) vorhanden ist, über den durch Software das gemeinsame INT*-Signal des Systems umgeleitet werden kann. Der CTC-Kanal ist dann im Zähler-Modus mit der Zählkonstanten 1 und Triggern durch eine negative Flanke zu programmieren.

Die direkte Verbindung des INT*-Signals (im Schaltplan INTEXP*) von der Floppy-Baugruppe zur CPU ist dafür aufzutrennen. Das Signal INTEXP* ist stattdessen an den entsprechend bezeichneten Punkt der Schaltung und an den freien Triggereingang des CTC-Kanals zu legen. Die INT*-Leitung des ECB-Bus ist an den Punkt INTECB* zu legen und der Ausgang INTCPU* ist mit dem INT-Eingang der CPU (Z80, Pin16) zu verbinden.

Damit ist gewährleistet, daß nach dem Einschalten oder Drücken des RESET-Knopfes immer der Interrupt vom FDC-Board direkt zur CPU durchgeschaltet wird und das Betriebssystem im IM1 arbeiten kann. Wenn allerdings andere Baugruppen auf dem ECB-Bus in Echtzeit im IM2 betrieben werden sollen, müssen auch die systemeigenen Interrupts dieses Spiel mitmachen können und das geht nur dann, wenn ihnen der CTC-Kanal beim INT-Acknowledge einen Vektor zur Verfügung stellt. Die Schaltung leistet es, daß ein LD A,xxxx xx1xB; OUT (0EDH),A das INT* Signal des FDC-Boards über den CTC umleitet.

Die Software muß dann aber Rücksicht auf diesen Umstand nehmen. In der INT-Vektor-Tabelle darf nicht einfach 4012H stehen (INT-Vektor des DOS), da die Serviceroutine des DOS nicht mit RETI sondern mit RET endet und der Z80-CTC dann nicht weiß, wann er seinen IEO-Ausgang wieder auf High zurücknehmen soll. Es muß folgender Umweg eingeschlagen werden:

```
INTAB   DEFW    SERV1     ;Anfang INT-Tabelle
        DEFW    SERV2     ;nächster Vektor
        DEFW    .....     ;u.s.w.
        ....    .....
        DEFW    CTCN      ;Vektor für Serviceroutine des INT vom CTC-Kanal
        ....              ;N, über den die System-Interrupts laufen
        DEFW    LETZT     ;Ende der Vektor-Tabelle
        ....    .....
CTCN    CALL    4012H     ;Serviceroutine des DOS
        EI
        RETI
```

Außerdem bleibt es dem Anwender offen, während des IM2-Betriebes anderer Baugruppen den 25ms INT verhungern zu lassen. Wenn beim Lable CTCN einfach ein EI, RETI steht, wird 37E0H nicht mehr gelesen und daraufhin erfolgt kein weiterer Interrupt des Timers mehr. Eine Reinitialisierung des Timer-INTs geschieht durch Umprogrammieren des Vektors auf ein anderes Lable, das wieder den CALL 4012H enthält, und einmaliges Lesen von 37E0H.

Durch Zurücksetzen von D1 in Port EDH wird wieder der INT des FDC-Boards direkt an die CPU durchgeschaltet und alle Interrupts des ECB-Bus werden unterbunden.

Mit der für die Interrupt-Steuerung benötigten Portdecodierung und mit dem Latch wurden noch einige zusätzliche Funktionen realisiert. Zunächst wurden auch die Portfreigabesignale OUTECH*, OUTEEH* und OUTEFH* hergeleitet. Davon kann OUTECH* zur Freigabe des 74LS273 Latch auf dem 256K-Banker (10) benutzt werden. Dieses Signal wird direkt an den Pin11 des 74LS273 geführt, wobei dann aber die bisher dahin führende Leitung zu durchtrennen ist. Damit ist dann das 74LS30 IC auf dem Banker überflüssig und auch die Leitungen A0 bis A5 sowie IORQ* und WR* brauchen nicht mehr dorthin geführt zu werden (A6 und A7 werden dort nicht nur zur Portdecodierung sondern auch noch zur Erzeugung der 8Bit-Refreshadresse benötigt und müssen deshalb weiterhin angeschlossen bleiben).

Mit Bit 0 in Port EDH wurde ein softgesteuertes Invertieren des zum 256K-Banker führenden Signals A15 realisiert. Damit läßt sich mit D0=0 an Port EDH (Einschaltzustand) vorgeben, daß die unteren 32K Adreßbereich des Z80 gebankt werden können, während D0=1 an Port EDH das Banking der oberen 32K festlegt.

Und schließlich kann mit D2=1 an Port EDH ein gemeinsames Invertieren der Adressen A14 und A15 erreicht werden. Die dabei gewonnenen Signale A14' und A15' können anstelle der Signale A14 und A15 an die Decoderschaltung für die 16K (ROMs und mm I/O), IC25, 74LS139, Pins 14 und 13 geführt werden und gestatten dadurch ein Verlegen dieser Komponenten an das obere Speicherende 48-64K. In diesem Bereich funktionieren dann zwar die ROMs nicht mehr, wenn man aber dann noch ein 74LS32 IC spendiert und mit dem Signal ED,3 (D3 von Port EDH) ein programmierbares Abschalten der ROMs einbaut (Abb.10), hat man auch noch die Funktionen des Omikron Mappers (Kompatibler Selbstbau in (9)) für den Betrieb von CP/M realisiert. Dieser leistet (wenn auch mit einem anderen OUT-Befehl) ebenfalls das Einstellen der Systemkonfiguration:

```
0000H-F7DFH   RAM (7)   #1
F7E0H-F7FFH   Floppy, Drucker
F800H-FBFFH   Tastatur (7)
FC00H-FFFFH   Video-RAM
```

#1 der Omikron Mapper für den TRS 80 leistet selbst eine vollständige Decodierung des Bereichs des Sonder-ROMs des GENIE

Ein Patchen der CP/M Systemdiskette ist sehr einfach. Auf den Systemspuren läßt sich relativ einfach die Sprungleiste des Loader Bios finden, die aus einer Reihe von Jumps besteht (C9 nn nn C9 nn ...). Daran schließen sich die DisK Parameter Tabellen mit sehr vielen 00H an und unmittelbar danach stehen die Befehle LD A,40H; OUT (50H),A. Diese beiden Befehle sind durch die Befehle LD A,xxxx 11xxB; OUT (0EDH),A zu ersetzen. Damit läuft das CP/M des Omikron Mappers dann auch ohne denselben.

Wenn diese Eingriffe auf dem CPU-Board gemacht werden, ist der Computer aber von der ECB-Bus-Eingangskarte abhängig, weil von dieser dann lebenswichtige Signale geliefert werden. Der ECB-Bus braucht selbst nicht angeschlossen zu sein.

Von dieser Abhängigkeit kann man sich durch einen Mehrfach- (oder mehrere Wenigfach-) Umschalter lösen. Abb.9 zeigt, wie man damit entweder die ursprünglichen Signale des CPU-Boards oder die steuerbaren Signale dieser Schaltung an die entsprechenden Stellen des CPU-Boards führt, und damit das Gerät entweder wie bislang ohne ECB-Eingangskarte oder mit derselben und allen ihren Mehrleistungen laufen lassen kann.

Um den Umfang der (zusätzlich zu denen des Systembus) an die Eingangskarte geführten Leitungen etwas zu reduzieren, bietet es sich an, IC7, 74LS30 auf der Eingangskarte nicht zu bestücken und dieses IC im Computer an geeigneter Stelle huckepack mit den Pins 7 und 14 auf ein anderes Dil14 IC zu löten und die internen Freigabesignale fest dorthin zu verlegen (Pins 1-6, 11-12). Dann braucht nur noch das Signal INTERN von Pin8 des 74LS30 an die Eingangskarte (CN2, Pin25c) geführt zu werden.

Wenn die zusätzlichen Features der Eingangskarte voll genutzt werden sollen, können alle benötigten Signale (SYSRES* A15* A14' A15' ED,3 OUTECH*) über freie Verdrahtung an unbelegte Pins von CN2 gelötet werden. Im Layout sind nur die Sinale INTCPU* INTEXP* INTECB* und INTERN über CN2 geführt. Dadurch können alle Verbindungen mit dem Computer (nicht nur der Systembus) über CN2 laufen.

Wer diese Variante des ECB-Bus bauen möchte, kann bei mir für 10 DM (incl. Rückporto) eine geätzte aber ungebohrte Karte im Europaformat bekommen. Sollte irgendjemand die Karte am Komtek 1 betreiben wollen, kann ich entsprechende Informationen zur Verfügung stellen, die eine Nutzung auch dort ermöglichen. Der Anschluß an den TRS 80 ist nicht ohne Sondermaßnahmen möglich, weil die internen Datentreiber IM2-Betrieb verhindern. Ein Hardware-Patch, der dagegen hilft, ist in (8) beschrieben. Eine Herleitung der Signale IORQ* MERQ* RD* und WR* aus den daraus gewonnenen Signalen IN* OUT* MWR* und MRD* des TRS 80 Systembus mit Gattern erzeugt Signale mit nicht ganz richtigem Timing, die eventuell Schwierigkeiten bereiten können; hier ist das Abgreifen der originalenSignale eine bessere Lösung.

Aufbau und Anschließen der Karte

Entgegen der üblichen Technik, solche Karten doppelseitig geätzt zu entwerfen, wurde hier ein einseitiges Layout mit einer unverschämten Menge Drahtbrücken bevorzugt, was das Durchkontaktieren an IC-Sockel-Pins vermeidet. Damit wird beim Aufbau aber viel Fummelei und beim Testen viel Ärger erspart.

Der Stecker CN1 (VG64, a,c, Stifte, gewinkelt) wird auf den ECB-Bus gesteckt. Für den Anschluß an den Systembus wurden 2 Varianten vorgesehen. Durch Einbau einer VG64 a,c-Buchsenleiste bei CN2 von der Lötseite (Abb.8) muß durch einen entsprechenden Drahtverhau zum Gegenstück für den 50poligen Platinenrandstecker des CPU-Boards die Anordnung der Signale umsortiert werden. Wer allerdings ein ästhetisches 50poliges Flachkabel bevorzugt, kann CN2 unbestückt lassen und stattdessen in die nicht angeschlossenen Lötaugen von CN3 von der Lötseite einen geeigneten 50poligen Stecker einlöten. Dann wird aber der zum Umsortieren der Signale nötige Drahtverhau nur auf die Karte zwischen CN2 und CN3 verlagert.

Außer den reinen Bus-Signalen des Computers müssen bei Bedarf auch noch die aktiven Steuersignale der Eingangskarte an den Computer zurückgeführt werden und die im Computer abzugreifenden Feigabesignale an die Eingangskarte geführt werden. Dafür sind am oberen Platinenrand entsprechende Lötpunkte vor-

handen. Die wichtigeren Signale sind aber auch an Lötpunkte von CN2 geführt, womit dann alle Signale über CN2 (und bei entsprechender Querverdrahtung auch CN3) mit dem Computer verbunden werden können. Dafür reicht dann aber der 50polige Platinenrandstecker des CPU-Boards mit seinen 3 freien Pins nicht mehr aus. Da muß dann noch eine zusätzliche fliegende Steckverbindung herhalten, die diese zusätzlichen Signale übergibt.

Seckerbelegungen

CN1 (ECB-Bus) Reihe a	NR.	Reihe c	CN2 (vom Systenbus) Reihe a	Nr.	Reihe c
GND	32	GND	GND	32	GND
BUSAK*	31	RESET*	BUSAK*	31	RESET*
A9	30	MERQ*	CLOCK	30	nc
A13	29	CLOCK	MERQ*	29	RFSH*
RFSH*	28	A15	IORQ*	28	RD*
IORQ*	27	A12	M1*	27	WR*
#1	26	#1	HALT*	26	nc
#1	25	HALT*	nc	25	INTERN
#1	24	RD*	nc	24	INTEXP*
#1	23	#1	NMI*	23	INTECB*
-12V	#2 22	WR*	A11	22	A8
#1	21	INT*	A10	21	A14
M1*	20	NMI	A13	20	A9
#1	19	#1	A12	19	A15
A14	18	A10	nc	18	nc
#1	17	A11	nc	17	nc
#1	16	IEO	nc	16	nc
-5V	15	#1	nc	15	nc
#1	14	D1	A7	14	A6
+12V	13	#1	A5	13	A4
#1	12	A14	,A3	12	A2
BUSRQ*	11	IEI	A1	11	A0
Wait*	10	#1	BUSRQ*	10	nc
A6	9	A7	nc	9	WAIT*
A5	8	A8	nc	8	nc
A4	7	A1	D5	7	D0
A2	6	A3	D6	6	D7
D4	5	A0	D3	5	D2
D3	4	D2	D4	4	D1
D6	3	D7	nc	3	INTCPU*
D5	2	D0	nc	2	nc
+5V	1	+5V	(+5V)	1	(+5V)

Stückliste

ICs:
4 x 74LS245
1 x 74LS174
1 x 74LS139
1 x 74LS125
1 x 74LS86
1 x 74LS32
1 x 74LS30
1 x 74LS11
1 x 74LS08
1 x 74LS00

Widerstände:
2 x 10k

Sonstiges:
1 x VG64 a,c Stifte gewinkelt
1 x VG64 a,c Buchsen gerade
1 x Leiterplatte
jede Menge Draht
viel Geduld

#1 diese Pins sind zum Teil nicht belegt oder führen Signale, die hier nicht benutzt werden
#2 die Lage von -12V wird nicht einheitlich gehandhabt, diese Spannung kann auch woanders liegen (muß bei jeder gekauften Karte geprüft werden)

Literatur

(1) Verbindend - Der ECB-Bus / Johannes Assenbaum / c't 12/85, 60-61

(2) Z80 CTC - Einsatz und Programmierung / Rolf Keller / c't 4/85, 92-94

(3) Interrupt-Priority-Encoder und Vektorgenerator für IM2 mit dem Z80 CTC / H. Bernhardt / Info des GENIE/TRS 80- User-Club Bremerhaven 12/85, 10-11

(4) Direkter Speicherzugriff; Schneller als die CPU - So arbeiten DMA-Controller-ICs / Rolf Keller / c't 8/85, 80-83

(5) Z80 DMA - Schnell, aber kompliziert / Rolf Keller / c't 12/85, 76-79

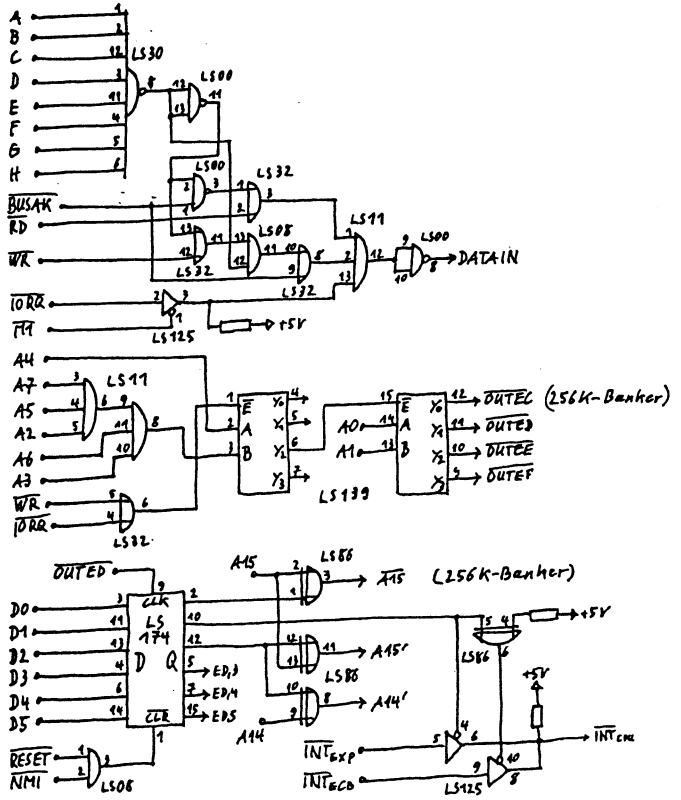
(6) Hardware-Umschaltung für den Betrieb von CP/M bei den Computern TRS 80, GENIE I und II und Komtek 1 / H. Bernhardt / Info des GENIE/TRS 80-User-Club Bremerhaven, Hardware-Sonderheft 11/85, 1-5

(7) GENIE I und II: Voll decodierter Sonder-ROM; RAM im Bereich 3900H bis 3BFFFH / H. Bernhardt / Info des GENIE/TRS 80-User-Club Bremerhaven 11/-85,....?

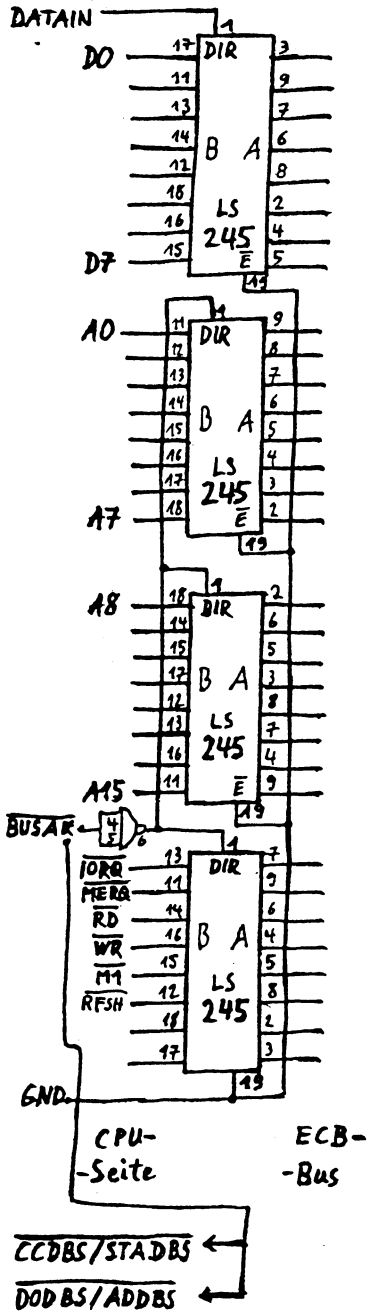
(8) Interrupt Your 80 - Without a hardware mod, your 80 is immune to rude interruptions / Douglas C. Fisher / 80 micro, January 1983, 258-266

(9) CP/M Hardware für TRS-80, GENIE I und II und Komtek 1 / H. Bernhardt, c.Ueberschaar, Info des TRS 80/GENIE-User-Clubs Bremerhaven

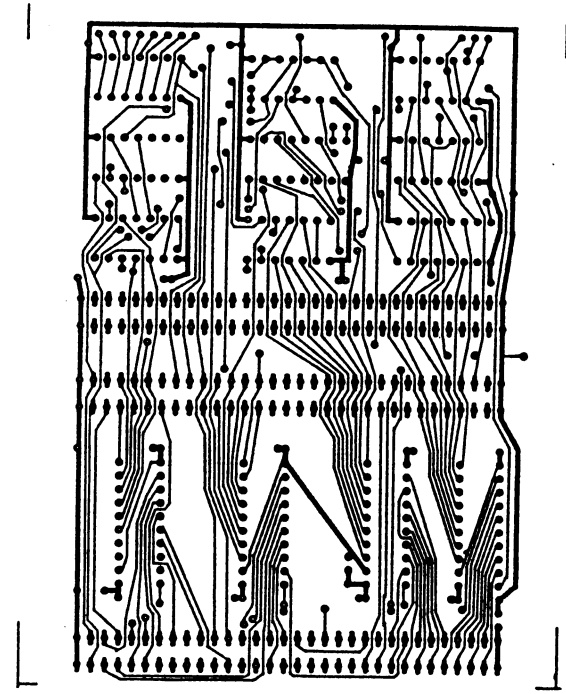
(10) 256K- (1M-) RAM für Z80-Systeme / H. Bernhardt / Info des Club 80 ,Nr.14,



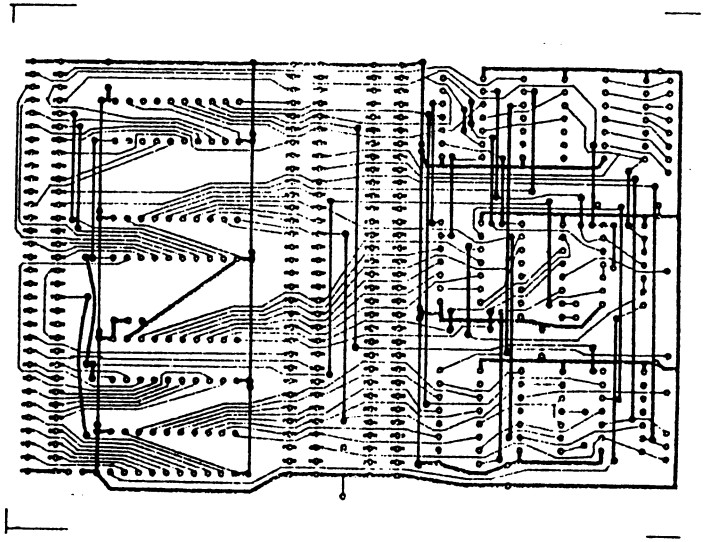
1) Steuerung des Databuffers, Portdecodierung, programmierbares Invertieren von A15 sowie A14 UND A15, Umschalten der Interrupts



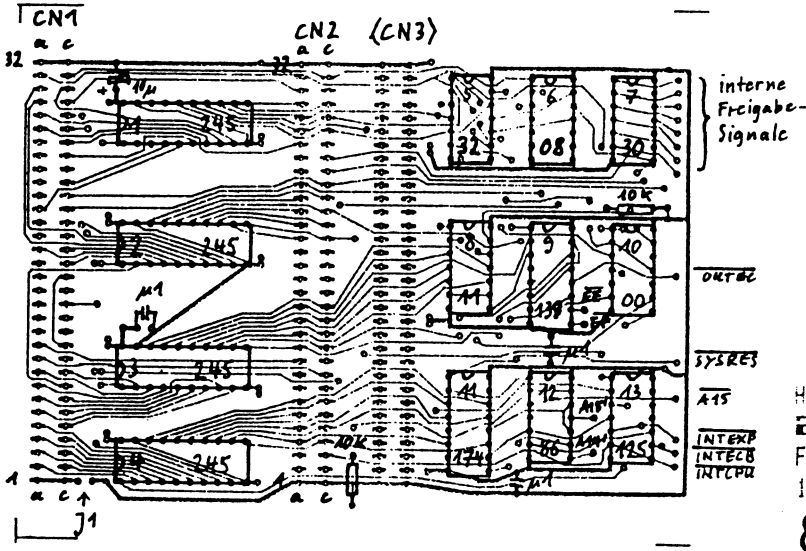
2) Pufferung des ECB-Bus



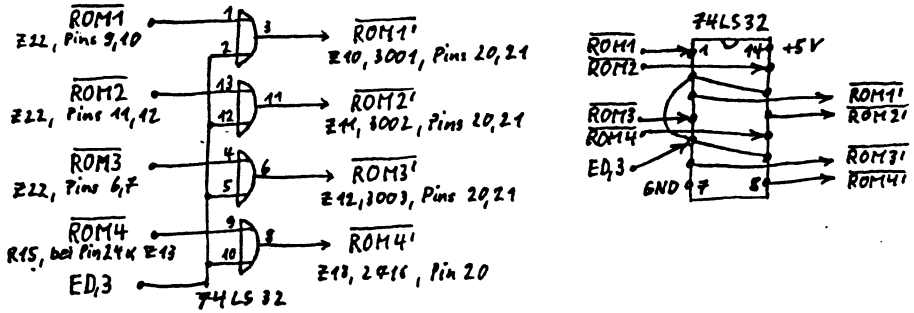
3) Layout der Eingangskarte



4) Drahtbrückenplan, die Drahtbrücken sind vor der Bestückung aller anderen Bauteile zu legen



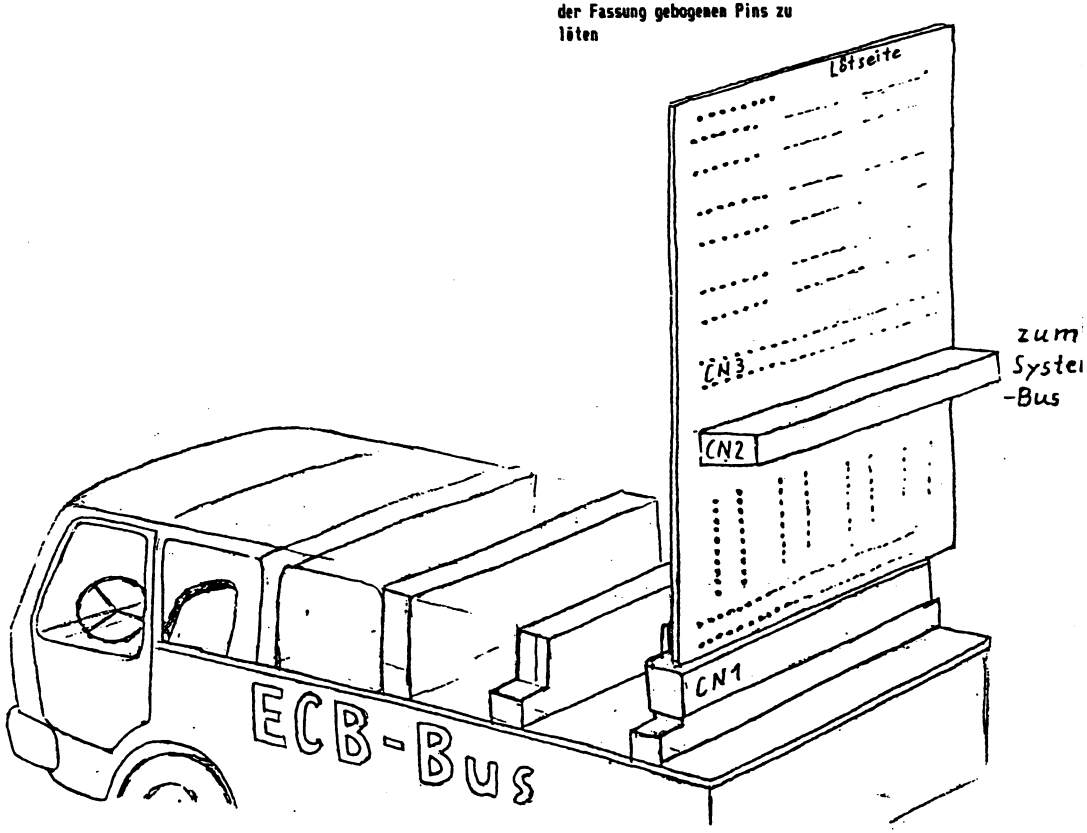
5) Bestückungsplan, J1 ist nur dann zu legen, wenn im GENIE ein stärkeres Netzteil eingebaut wurde, das den ECB-Bus mitversorgen soll



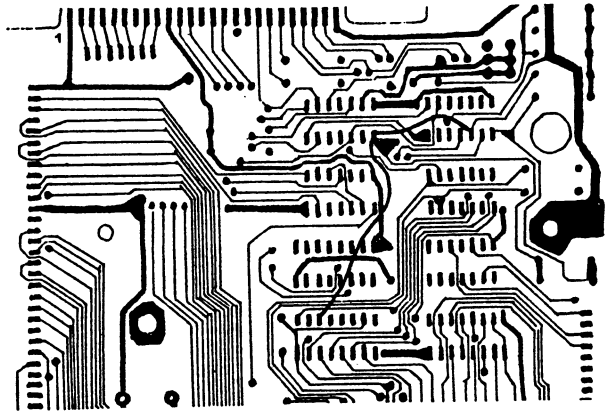
6) Beschaltung des Huckepack 74LS32 auf dem CPU-Board zum Ausblenden der ROMs

ROM1*	Z22, 74LS156, Pins 9,10	ROM1'*	Z10, 3001, Pins 20,21
ROM2*	" " " 11,12	ROM2'*	Z11, 3002, Pins 20,21
ROM3*	" " " 6,7	ROM3'*	Z12, 3003, Pins 20,21
ROM4*	R15, Seite zu Pin24 von Z13 (ROM4)	ROM4'*	Z13, 2716, Pin 20

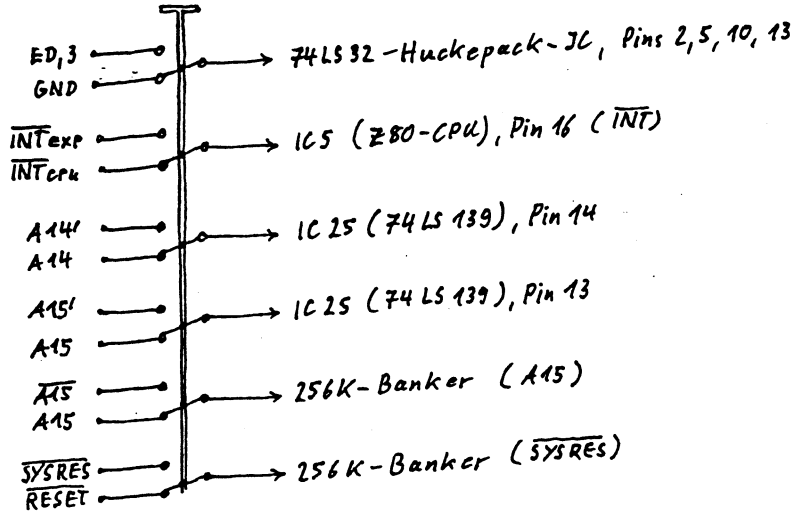
diese Signale sind an die aus der Fassung gebogenen Pins zu löten



7) Verbinden des ECB-Bus mit dem Systembus des Computers über die Eingangskarte



8) Verbinden von BUSAK* mit CCDBS/STADBS* und DODBS/ADDBS*



9) Umschalten zwischen originalen GENIE-Signalen und steuerbaren Signalen der Eingangkarte

Suche:

Wer hat ein Basic- Programm (oder kann eines schreiben) das in der Lage ist, englische Texte ins deutsche zu übersetzen. Umgekehrt muß dies auch funktionieren. Nach Möglichkeit muß der Text auf Diskette abspeicherbar sein und auf einen Drucker ausgegeben werden können. Vokabeln gebe ich selber ein, da diese auf einen bestimmten Inhalt fixiert werden.

Zu verkaufen: Maus und TRS 80 M. 100

Im Zuge der Modernisierung meines Geräteparks habe ich mir vor Kurzem einen IBM-kompatiblen Portable (Toshiba T2100) zugelegt (keine Angst, ich bleibe weiterhin voll auf Tandy eingestellt). Leider arbeitet die Atari-Maus, die ich mir vor einiger Zeit zugelegt habe, nicht mit dem Toshiba zusammen. Aus diesem Grund habe ich folgende zwei Probleme zu lösen:

1. Ich benötige eine RS 232-Maus für den PC.
2. Meine Bessere Hälfte, zugleich Finanzminister im Hause Obermann, bewilligt die nötigen Mittel für eine solches Grautier nur, wenn ich zuvor die Atari-Maus verkaufe!

Aus diesem Grunde muß (will) ich mich von der im letzten Info beschriebenen Maus incl. Interface trennen. Die Schaltung funktioniert natürlich nicht nur am 4p, sondern auch mit allen anderen TRS 80- und Video Genie-Geräten. Wer also mal seine Grafikprogramme mit Maus fahren, oder auch nur mit einem solchen Peripheriegerät experimentieren will, sollte sich bei mir melden. Meine Preisvorstellung liegt so bei ca. 60 bis 70 Märkern für Maus incl. Interface, Verpackung und Porto.

Ebenfalls im letzten Info erwähnte ich, daß ich vor einiger Zeit ein Model 100 von Tandy erstanden habe. Da ich für diesen Computer, jetzt, nachdem ich den portablen PC habe, keine Verwendung mehr sehe, möchte ich auch dieses Gerät veräußern. Das Model 100 hat 24k RAM- und 32k ROM-Speicher, eingebaute Programme für Textverarbeitung, Adressen- und Terminverwaltung sowie Kommunikation über die eingebaute RS 232-Schnittstelle. Natürlich ist auch eine Kassetten- sowie eine Druckerschnittstelle vorhanden. Das 100'ter wird komplett mit 4 Akkus sowie einem englischen und einem deutschen Handbuch geliefert. Meine Preisvorstellung für dieses wirklich sehr schöne Gerät liegt bei ca. 450,-- DM (ev. 500,-- DM mit Kassettenrecorder).

Sowohl die Maus als auch das Model 100 sind 100 %'ig in Ordnung und ich gebe eine übernahmegarantie von 14 Tagen. Wer sich für eines oder beide Geräte interessiert, kann mich entweder anrufen (ab Mitte April bis Ende Juni nur an Wochenenden) oder mir schreiben.

Hartmut Obermann
Schwalbacher Straße 6
Postfach 27
6209 Heidenrod / Kemel
Tel.: 06124 / 3913

BÖRSE -- BÖRSE -- BÖRSE

Drucker Gemini 10X

Wg. Umstieg auf 24 Nadeln stelle ich meinen 9-Nadel-Drucker einem Interessierten für schlappe DM 200,- (VB) zur Verfügung. Das Ding kann Grafik drucken und arbeitet bestens mit Tscrips und verwandten Programmen zusammen. Als Druckbeispiel mag mein CP/M-Artikel herhalten, der allerdings mit einem etwas älteren Farbband gedruckt wurde.

Schmidtke-80-Zeichen

Über die Karte habe ich ja schon viel gelästert, aber sie tut es noch. Aufgebaut ist sie für einsteckfertig für ein Genie IIs mit Z80-Prozessor. Bei mir arbeitet sie mit einem HD64180 (9 Mhz) nicht mehr zusammen. Sie müßte sich durch etwas Bastelei auch in eine für andere Genies taugliche Form bringen lassen. evtl. sogar für TRS-80. Für DM 50,- kann es jemand damit probieren. Ein angepaßtes Tscrips und die wichtigsten Routinen, um sich ein CP/M dafür selbst basteln zu können, sind vorhanden. Die Karte erledigt übrigens auch das Banking. wenn es sein muß, also kann ohne andere Zusätze CP/M gefahren werden.

Gerald Schröder, 04105 / 2602

VERKAUFE:
=========

- TANDON Floppy, 40 Tr, SS/DD, 80,-- DM

- TANDY Model 1, 150,-- DM

- Literatur:
  Personal Computer Lexikon, Markt und Technik Verlag
  Microcomputer Technik, Blomeyer
  Das DOS Buch, Hartmut Grosser
  Programmieren in Maschinensprache mit Z 80, C. Lorenz
  Programmieren mit TRS 80, M. Stubs
  Basic, G. Abeldt
  The First Book of 80 US
  Basic Disk I/O Faster and Better, L. Rosenfelder
  TRS-80 Disk and other Mysteries
  komplett 120,-- DM

- Texas Instruments, TI-58, mit Handbuch/Literatur, 60,-- DM

- TANDY Modell 4P, 900,-- DM

Klaus Hermann, Tel. 07127/70024

BÖRSE -- BÖRSE -- BÖRSE

Betr. Computer für DDR
Ich habe kostenlos folgende Geräte abzugeben :

- Sharp PC 1211 mit Drucker und Kassettenspeicher, Betriebsanleitung und versch. Literatur.

- Junior-Computer in der Basisausführung mit Netzteil und Handbücher.

Sollte dafür Bedarf bestehen, so bin ich gerne bereit die Geräte dem Club zur weiteren Verwendung zuzusenden.

Das wäre alles für heute.
Mit freundlichen Grüßen,

Helmut Bode

Farbband für
EPSON-Drucker
ab DM 10.95 in Weinheim
erhältlich!
Bestellung (+Porto)
bei „Ka-Jot"
(bis auf
weiteres)

Achtung!!!

An alle Clubmitglieder, die am 23. April den Flohmarkt in Nürnberg besuchen, ich habe einen Tisch reservieren lassen und wenn jemand von euch ein Gerät zu veräußern hat, kann es mitbringen.

Ich habe außerdem ein Stringy-Floppy zu verkaufen mit I/O Program und vielen Kassetten. Preis Verhandlungssache.

Eckehard Kuhn

Z80 Hochintegrations CPUs in CMOS nutzen bestehende Entwicklungssysteme und Software

Unter den Bezeichnungen TMPZ84C011/013/015 bietet TOSHIBA **Hochintegrationsprodukte mit Z80-Kern** an. Die Produkte sind in 4 und 6 MHz-Version lieferbar und von −40 °C bis +85 °C spezifiziert. TMPZ84C011/015 haben ein 100-Pin Flat-Pack-Gehäuse, während TMPZ84C013 im 84-Pin PLCC gefertigt wird. Gemeinsam ist allen 3 Produkten neben der CPU ein On-Chip-Clock Generator und Controller.

TMPZ84C011 ist durch Ausstattung mit 5 Ports für I/O-intensive Aufgaben ausgelegt, während TMPZ84C013/014 mit 4 bzw. 2 USART-Kanälen für kommunikationsintensive Applikationen entwickelt wurden. Beide Produkte verfügen über einen integrierten Watch Dog Timer, TMPZ84C015 darüber hinaus über 2 Timer/Counter.

Bei der Entwicklung der Produkte wurde besonders Wert darauf gelegt, daß vorhandene Entwicklungssysteme für die Emulation verwendet werden können.

Für den Anwender bedeutet dies, daß der Einsatz dieser Produkte ohne größere Investitionen in Hard- und Software erfolgen kann.

Mit ihrem sicheren Gespür für Produkte, die der Markt in großen Stückzahlen braucht, haben die Japaner unseren bewährten Z8Ø um mehrere integrierte Zusatzfunktionen erweitert. Gedacht sind die neuen Schaltkreise für Controller-Anwendungen, bei denen der Mikroprozessor für fest vorgegebene Steuerungaufgaben mit Programmen aus dem ROM betrieben wird. Automaten, Hausgeräte und Verpackungsmaschinen sind Beispiele für den Controller-Einsatz. Werden größere Stückzahlen gefertigt, muß der Hersteller mit dem Pfennig rechnen. Sockel, Lötverbindungen und der Platz auf der Leiterplatte kosten Geld. Je mehr mechanische Verbindungen eine elektronische Schaltung hat, desto größer ist ihre Fehleranfälligkeit und um so schwieriger ist das Testen.

Unverständlich ist, daß Zilog die Entwicklung hochintegrierter Contoller-CPU's nicht schon längst betrieben hat. Andere Halbleiterhersteller, wie z.B. Intel mit seiner 8Ø51-Serie, haben einen großen Marktanteil erobert, ohne daß Zilog die Chance mit seinem weltweit eingeführten Z8Ø-uP und den dafür vorhandenen Entwicklungssystemen genutzt hätte. Der für einfachere Anwendungen vorgesehene Z8 ist nie so recht zum Zug gekommen und stand im Schatten seines erfolgreichen älteren Bruders. Nachdem der Z8Ø immer noch der in der größten Stückzahl verkaufte Mikroprozessor ist, kann man erwarten, daß Toshiba mit seinen hochintegrierten Schaltkreisen am Markt offene Türen einläuft. Hätte man diese Produkte schon vor drei oder vier Jahren (von Zilog) bekommen können, wäre heute wahrscheinlich der Z8Ø mit seinen Abkömmlingen der Industriestandard auch bei den Low Cost Controllern.

Heinrich Betz

Testbericht Drucker PEACOCK D 1018 (Baugleich mit Panasonic KX 1083) Nadeldrucker

Ende des vergangenen Jahres habe ich mich entschlossen meinen alten Drucker gegen ein besseres Exemplar einzutauschen.

Bei der Firma Computer-Service (W.Grundmann) konnte ich meinen alten Nadeldrucker in Zahlung geben und erstand ein Vorführmodell der Firma PEACOCK, den D 1018, der baugleich mit dem Panasonic KX 1083 ist. Mit diesem Gerät bin ich sehr zufrieden und die Testergebnisse der Zeitschrift "Warentest von Januar 1988" gaben dem Drucker die Note "GUT". Schwierigkeiten gab es zuerst mit den Farbandkassetten, aber mittlerweile habe ich ohne Schwierigkeiten vom Händler genügend Ersatz bekommen.

Technische Daten des Nadeldruckers mit 9 Nadeln : (Epson kompatibel)

Zeichenarten : Entwurf, Pica-Schönschrift, komprimierte Schrift, Elite Schönschrift, Proportional-Schönschrift.

Zeichensatz : 96 ASCII-Zeichen, 96 ASCII-Kursivzeichen, 32 Internationale Zeichen, 64 Block-Grafik-Zeichen, 132 IBM-PC-Sonderzeichen.

Punktmatrix : Punktdurchmesser 3/254 Zoll (0.3 mm)

Größe der Zeichen : Standardzeichen : 0.078(B)*0.095(H)Zoll
Hoch-/tiefgestellte Zeichen :
0.078(B)*0.053(H)Zoll

Anzahl der Zeichen pro Zeile :

Pica (Entwurf, Schönschrift)	80 Zeichen/Zeile	(10 Zeichen/Zoll)
Elite(Entwurf, Schönschrift)	96 Zeichen/Zeile	(12 Zeichen/Zoll)
Komprimiert	137 Zeichen/Zeile	(17 Zeichen/Zoll)
Pica gedehnt	40 Zeichen/Zeile	(5 Zeichen/Zoll)
Elite gedehnt	48 Zeichen/Zeile	(6 Zeichen/Zoll)
Kompimiert und gedehnt	68 Zeichen/Zeile	(8.5Zeichen/Zoll)

Druckgeschwindigkeit :

Entwurf Pica	180 Zeichen/s
Entwurf Elite	180 Zeichen/s
Schönschrift	33 Zeichen/s

Druckrichtung : Ausdruck von Text : bidirektional
Bitmuster : eine Richtung (links ==>rechts)

Zeit für einen Zeilenvorschub : ca 100 ms

Papierzufuhr : Traktorantrieb für Endlospapier
Friktionsantrieb für Enzelblattpapier

Kopf Lebensdauer : 100 Million Zeichen bei Entwurfs-Qualität

Farbband : Kassette mit nahtlosem Farbband

Wie geagt das war ein kurzer Steckbrief der technischen Daten. Obrings lassen sich die einzelnen Schriftarten per Tastendruck von vorne bequem einstellen, ebenso die Länge des Blattformates.

Für meine Wordstar-Version habe ich mir eine eigene Druckeranpassung gemacht und so verfüge ich über zusätzlich weitere Schriftarten.

Hier eine kurze Übersicht der Druckarten per Softwareansteuerung :

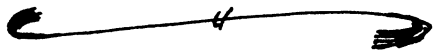
PEACOCK-Druckeranpassung

Worte und Leerzeichen unterstreichen
Schmalschrift
Kursiv
Riesenbreitschrift
fuer diverse Funktionen:
Epson-Fettdruck
Epson-Doppeldruck
Epson-Elite; Epson-Pica-Schrift
Proportionalschrift fuers Auge

schmale Breitschrift
kursiv Breitschrift
fette Breitschrift
kursiv schmale Breitschrift

Das war es wieder einmal in Kürze. Für interessierte, der Drucker kostet im Handes Zwischen 948,- und 1100,- DM.

Harald Mand



Reingefallen mit Rheintec

Liebe Clubfreunde,
sicher seit auch Ihr hier und dort darauf angewiesen, einmal Zubehör bei einem Händler zu kaufen. So war dies bei mir auch im letzten Jahr, als ich nach einer Speichererweiterung für meinen Rechner (PLANTRON XT/20 diesmal; mein GENIE steht im selben Zimmer) suchte und mich an eine Düsseldorfer Firma vertrauensvoll gewendet habe. Im Ergebnis ist jetzt mein Ärger groß, weil nichts klappte und die Firma am Schluß sich unfähig zeigte, wirklich zu helfen und unverschämter Weise auch die defekten Karten nicht zurück nehmen wollte.
Ich muß für mich feststellen, daß ich bei dieser Firma nicht mehr kaufen werde.
Paul-Jürgen Schmitz
7. März 1988

Echte Programmierer meiden PASCAL

(Aus DATAMON 7/83, uebersetzt von Bernd Leeds)

Vor langer Zeit, in der goldenen Aera der Computer, war es noch einfach, die Maenner von den Memmen zu trennen (mitunter auch "Echte Maenner" und "Muesli-Fresser" genannt). Echte Maenner programmierten Computer und Muesli-Fresser liessen es bleiben. Ein echter Computerprogrammierer sagte Dinge wie "DO 10 I=1,10" oder "ABF1D", und der Rest der Welt quengelte "Computer sind mir zu kompliziert" oder "Ich kann zu Computern keine gefuehlsmaessige Bindung aufbauen - sie sind zu unpersoenlich". Dabei zeigt schon Remy Eyssen's Buch "Echte Maenner moegen kein Muesli" (Heyne TB 6290), dass Echte Maenner zu nichts und niemanden eine 'gefuehlsmassige Bindung' aufbauen, und dass sie auch keine Angst haben, unpersoenlich zu sein.

Aber die Zeiten aendern sich. Heute stehen wir einer Welt gegenueber, in der kleine alte Damen vollcomputerisierte Mikrowellenherde kaufen koennen, in der 12 Jahre alte Dreikaesehochs gestandene Maenner bei ASTROIDS und PACMAN sattmachen, und in der jeder seinen eigenen Homecomputer kaufen und sogar verstehen kann. Der Echte Programmierer ist gefaehrdet, von Studenten mit einem igITT 2020 (deutsche Version des ABFALL-II, Anm. d. Uebers.) im Gepaeck ersetzt zu werden!

Es gibt allerdings einige Unterschiede zwischen dem typischen, PAC MAN-spielenden Gymnasiasten und einem Echten Programmierer. Die Kenntnis dieser Unterschiede wird den Heranwachsenden ein Ziel geben nach dem sie streben koennen - ein Vorbild, eine Vaterfigur. Ausserdem schuetzt sie die Echten Programmierer vor Arbeitslosigkeit.

Der einfachste Weg, um einen Echten Programmierer zu erkennen, fuehrt ueber die von ihm benutzte Programmiersprache. Echte Programmierer benutzen FORTRAN. Muesli-Fresser benutzen PASCAL. Niklaus Wirth, der Schoepfer von PASCAL wurde einmal gefragt, wie man seinen Namen ausspreche. "You can either call me by name, pronouncing it 'Veert' or call me by value, 'worth'", antwortete er. Diese Bemerkung zeigt sofort, dass Wirth ein Muesli-Fresser ist. Der einzige Parameter-Uebergabe-Mechanismus, den Echte Programmierer akzeptieren, ist call-by-value-return (call-by-result, Anm. de. Uebers.), wie er in den IBM/370 FORTRAN-G- und -H-Compilern implementiert ist. Echte Programmierer brauchen schliesslich keine abstrakten Konzepte, um ihre Arbeit zu erledigen; sie sind vollkommen gluecklich mit einem Lochkarten-Stanzer, einem FORTRAN-IV-Compiler und einem Bier. Echte Programmierer erledigen Listenverarbeitung, Zeichenketten-Manipulation, Graphikdarstellungen (wenn ueberhaupt) und kuenstliche Intelligenz in FORTRAN. Was sie mit FORTRAN nicht machen koennen, machen sie in Assembler, was sie in Assembler nicht machen koennen, lassen sie veraechtlich liegen.

Akademische Computerspezialisten sind in den letzten Jahren auf's Abstellgleis der Strukturierten Programmierung geraten. Sie behaupten, dass Programme verstaendlicher werden, wenn bestimmte Sprachkonstrukte und Programmiertechniken benutzt werden. Sie koennen sich natuerlich nicht einigen, welche Konstrukte am besten geeignet sind, und die Beispiele, an denen sie ihren speziellen Standpunkt aufzeigen wollen, passen ausnahmenslos auf eine einzige Seite irgend eines obskuren Journals.

Als ich aus der Schule kam, dachte ich, ich sei der beste Programmierer der Welt. Ich konnte ein unschlagbares TIC-TAC-TOE-Spiel (Vier-in-einer-Reihe, Anm. d. Uebers.) schreiben, beherrschte 5 verschiedene Programmiersprachen und schrieb fehlerfreie 1000-Zeilen-Programme Dann kam ich in die Wirklichkeit. Meine erste Aufgabe bestand darin, ein 200.000-Zeilen-FORTRAN-Programm zu lesen, zu verstehen und um den Faktor 2 zu beschleunigen. Jeder Echte Programmierer wird einem versichern, dass die gesamte strukturierte Programmierung der Welt in einem solchen Fall nicht hilft - hier braucht man wirklich Talent.

Einige Beobachtungen zum Thema "Echte Programmierer und Strukturierte Programmierung":

- Echte Programmierer haben keine Angst vor GOTO's.
- Echte Programmierer schreiben 5 Seiten lange DO-Schleifen, ohne durcheinander zu geraten.
- Echte Programmierer lieben das arithmetische IF-Statement (das mit den 3 Ausgaengen, Anm. d. Uebers.), weil es den Code interessanter macht.
- Echte Programmierer schreiben selbstmodifizierende Programme, speziell, wenn sie damit in einer kleinen Schleife 20ns einsparen koennen.
- Echte Programmierer brauchen keine Kommentare; das Programm ist schliesslich selbstdokumentierend.
- Da FORTRAN strukturierte IF-, REPEAT UNTIL- oder CASE Anweisungen nicht kennt, braucht sich der Echte Programmierer nicht zu sorgen, dass er sie nicht benutzt. Ausserdem kann man sie noetigenfalls ueber "ASSIGNED GOTO's" simulieren.

Datenstrukturen sind in letzter Zeit in gewissen Kreisen populaer geworden. Wirth, der Muesli-Fresser, verfasste sogar ein ganzes Buch (Algorithmen und Datenstrukturen", Teubner 1975), in dem er behauptete, dass man Programme schreiben koenne, die auf Datenstrukturen aufbauen, statt es umgekehrt zu machen. Wie jeder Echte Programmierer weiss, gibt es nur eine wirklich nuetzliche Datenstruktur, das Array. Zeichenketten, Listen, Records und Mengen sind allesamt Sonderfaelle von Arrays und koennen auch so behandelt werden, ohne dadurch die Sprache zu verkomplizieren. Das Schlimmste an den ganzen schoenen Typen ist ausserdem, dass man sie deklarieren muss, waehrend Echte Programmiersprachen, wie man weiss, den Typ anhand des ersten Buchstabens eines maximal 6 Zeichen langen Bezeichners implizit festlegt.

Welches Betriebssystem der Echte Programmierer benutzt? CP/M? Gott bewahre! Das ist doch im Grunde ein Spielzeug-Betriebssystem. Selbst kleine alte Damen und Hauptschueler koennen CP/M benutzen und verstehen.

UNIX ist natuerlich schon viel komplizierter - der typische UNIX-Hacker weiss nie, wie das PRINT-Kommando diese Woche heisst - aber wenn man es genau nimmt, ist UNIX auch nur ein verherrlichtes Telespiel. Niemand arbeitet auf UNIX-Systemen an ernstzunehmenden Dingen - man schickt kleine Witzchen ueber USENET rund um die Welt, oder man schreibt ein neues Adventure-Spiel oder Forschungsberichte.

Nein, der Echte Programmierer benutzt OS/370. Ein guter Programmierer kann die Beschreibung des Fehlers IJK3051 in seinem JCL-Manual (Job Control Language, Batch-Kommandosprache, Anm. d. Uebers.) finden und interpretieren. Ein sehr guter Programmierer erkennt die Fehler, ohne in sein Manual zu sehen. Ein wahrhaft ausserordentlicher Programmierer kann Fehler in einem 6-Megabyte-Hexdump finden, ohne hierfuer einen Taschenrechner zu benutzen.

OS/370 ist ein wirklich bemerkenswertes Betriebssystem. Mit einem einzigen falsch plazierten Leerzeichen kann man die gesamte Arbeit mehrer Tage zerstoeren, was die Wachsamkeit im Programmierteam ungemein foerdert. Der beste Weg zum System ist der Kartenstanzer. Zwar behaupten einige Leute, es gaebe ein Timesharing-System unter OS/370, aber nach sorgfaeltigen Nachforschungen bin ich zu dem Schluss gekommen, dass sie sich irren.

Welche Werkzeuge ein Echter Programmierer benutzt? Nun, theoretisch koennte er seine Programme ueber die Maschinenkonsole eingeben und laufenlassen. In den fuehen Tagen der Computerei, als Computer noch Maschinenkonsolen hatten, wurde dies auch gelegentlich getan. Der typische Programmierer wusste den System-Urlader Bit fuer Bit auswendig und tastete ihn ein, sobald er von seinem Programm zerstoert worden war. Damals war Speicher auch noch Speicher - der war nicht einfach leer, wenn der Strom ausfiel. Hauptspeicher von heute hingegen vergessen entweder Dinge, die sie behalten sollten, oder halten Informationen, die schon lange weg sein sollten. Aber zurueck zum Thema. Die Legende sagt, dass Seymour Cray, der Erfinder des Cray-I-Supercomputers und der meisten anderen Rechner von Control Data, selbst das erste Betriebssystem fuer die CDC 7600 an der Maschinenkonsole eingetastet hat, als sie das erste Mal eingeschaltet wurde. Cray ist selbstverstaendlich ein Echter Programmierer.

Einer der Echten Programmierer, die ich am meisten bewundere, arbeitete als Sytem-Programmierer fuer Texas Instruments. Eines Tages erhielt er ein Ferngespraech von einem Benutzer, dessen System mitten in einer wichtigen Arbeit abgestuertzt war. Der Typ reparierte dann den Schaden ueber's Telefon. Er brachte den Benutzer dazu, System-Tabellen in Hexadezimal zu reparieren und Registerinhalte ueber's Telefon durchzugeben. Die Moral von der Geschichte: obwohl ein Echter Programmierer normalerweise Kartenlocher und Schnelldrucker benutzt, kommt er im Notfall auch mit Maschinenkonsole und Telefon aus.

In einigen Firmen besteht die Programmeingabe allerdings nicht nur aus 10 schlangestehenden Ingenieuren, die auf einem 029-Locher warten. In meiner Firma z.B. steht kein einziger Kartenstanzer. Der Echte Programmierer muss in diesem Falle seine Arbeit mit einem Text Editor erledigen. Auf den meisten Rechnern stehen verschiedene Editoren zur Verfuegung, und der Echte Programmierer muss aufpassen, dass er einen erwischt, der seinen persoenlichen Stil wiedergibt. Viele Leute glauben, dass die besten Editoren der Welt am Xerox Pablo Alto Research Center geschrieben wurden und auf Alto- oder Dorado-Computern laufen. Ungluecklicherweise wuerde jedoch kein Echter Programmierer einen Computer mit einem Betriebssystem benutzen, das SmallTalk heisst, und sicherlich auch nicht ueber eine Maus mit einem Rechner kommunizieren.

Einige Konzepte der Xerox-Editoren sind mittlerweile in Editoren eingeflossen, die unter sinnvoller benannten Betriebssystemen arbeiten, so wie EMACS oder VI. Das Problem mit diesen Editoren ist, dass Echte Programmierer das Konzept des "Du kriegst, was Du siehst" fuer schlecht halten. Der Echte Programmierer will einen "Du hast es so gewollt, da hast Du's"-Editor, einen der kompliziert ist, skeptisch, leistungsfaehig, gnadenlos und gefaehrlich. TECO, um genau zu sein.

So wurde beobachtet, dass TECO-Kommandofolgen dem Leitungsrauschen aehnlicher sind als lesbarer Text. Eins der unterhaltsameren Spiele, die mit TECO moeglich sind, besteht darin, den eigenen Namen als Kommando einzugeben und zu raten, was dann passiert. So ungefaehr jeder moegliche Tippfehler kann dank TECO das gerade editierte Programm zerstoeren, oder schlimmer noch, kann kleine mysterioese Fehler in einstmals funktionierende Unterprogramme einbringen.

Aus diesem Grunde editieren Echte Programmierer nur sehr widerwillig Programme, die schon fast laufen. Sie finden es viel einfacher, den binaeren Objektcode direkt zu aendern, fuer gewoehnlich mit einem wundervollen Programm, das SUPERZAP heisst (auf Nicht-IBM-Rechnern entsprechend anders). Dies funktioniert so gut, dass viele laufende Programme auf IBM-Systemen keine Aehnlichkeit mit den urspruenglichen FORTRAN-Quellprogrammen haben. In einigen Fallen ist nicht einmal mehr das urspruengliche Quellprogramm vorhanden. Wenn dann der Zeitpunkt gekommen ist, so ein Programm zu aendern, wuerde kein Manager auch nur daran denken, einem geringeren als einem Echten Programmierer diese Arbeit zu uebertragen - kein Mueslifressender strukturierter Programmierer wuesste auch nur, wo er mit der Arbeit anfangen soll. Man nennt das Arbeitssicherungsmassnahme.

Hier eine Liste der wichtigsten Programmierhilfen, die der Echte Programmierer nicht benutzt:

- FORTRAN-Praeprozessoren wie MORTRAN oder RATFOR. Diese Haute Cuisine der Programmierung eignet sich hervorragend, um Muesli zu produzieren.
- Quellcode-orientierte Debugger. Echte Programmierer lesen Hexdumps.
- Compiler, die Code fuer Array-Indexpruefungen zur Laufzeit erzeugen. Sie ersticken jede Kreativitaet, zerstoeren die meisten der interessantesten Anwendungen der EQUIVALENCE-Vereinbarungen, und machen Aenderungen des Betriebssystems mit Hilfe negativer Indizes unmoeglich. Und schlimmer noch, solcher Code ist ineffizient.
- Programm-Pflege-Systeme. Ein Echter Programmierer haelt seine Software als Kartonstapel unter Verschluss, denn dies zeigt, dass der Besitzer seine wichtigsten Programme nicht unbewacht lassen kann.

Wo der typische Echte Programmierer arbeitet? Welche Art von Programmen derart talentierter Individuen wuerdig ist? Nun, man kann sicher sein, dass man nie einen Echten Programmierer beim Schreiben von Buchhaltungsprogrammen in COBOL erwischen wird, oder gar beim Sortieren der Abonnentenadressen des 'Spiegel's. Nein, ein Echter Programmierer braucht Aufgaben von weltbewegender Bedeutung.

Echte Programmierer arbeiten fuer das Los Alamos National Laboratory und schreiben dort Atomkriegs-Simulationen auf Cray-I-Supercomputern, oder sie arbeiten bei der National Security Agency und entschluesseln russische Funksprueche. Nur weil tausende Echte Programmierer fuer die NASA gearbeitet haben, waren 'unsere' Jungs eher auf dem Mond als die Kosmonauten. Die Computer im Space Shuttle wurden von Echten Programmierern programmiert, und auch die Betriebssysteme der Cruise Missiles der Firma BOEING wurden von diesen echten Professionals entworfen.

Einige der ehrfurchteinfloessendsten Echten Programmierer kennen das gesamte Betriebssystem der Pioneer- und Voyager-Sonden auswendig. Mit einer Kombination aus grossen, bodengebundenen FORTRAN-Programmen und kleinen, von Sonden mitgefuehrten Assemblerprogrammen vollbringen sie unglaubliche Kunststuecke der Navigation und Improvisation. So treffen sie nur 10 Kilometer grosse Fenster nahe Saturn nach 6 Jahren Flug durch den Weltraum, oder reparieren bzw. umgehen defekte Sensoren, Sender oder Batterien. Angeblich soll es einem Echten Programmierer sogar gelungen sein, in ein paar Hundert Byte ungenutzten Speichers innerhalb der Voyager-Sonde ein Mustererkennungsprogramm zu pressen, das einen neuen Mond des Jupiters suchte, fand und fotografierte.

Fuer die Galileo-Sonde ist vorgesehen, dass sie auf ihrem Weg zum Jupiter entlang einer schwerkraftgelenkten Bahn am Mars vorbeizieht. Diese Bahn fuehrt in einer Entfernung von 80 +/- 3 km an der Marsoberflaeche vorbei. Kein Mensch wuerde diese Art von Navigation einem PASCAL-Programm oder gar -Programmierer anvertrauen.

Viele Echte Programmierer dieser Welt arbeiten fuer die amerikanische Regierung, meist fuer das Verteidigungsministerium. So soll es sein. In letzter Zeit allerdings erscheinen dunkle Wolken am Horizont der Echten Programmierer. Es scheint, als haetten einige einflussreiche Muesli-Fresser im Verteidigungsministerium entschieden, dass in Zukunft

alle Verteidigungsprogramme in so einer Art von grosser vereinheitlichter Programmiersprache namens ADA geschrieben werden muessten. Lange Zeit schien es, als laege ADA's Bestimmung im Verstoss gegen alle Regeln der Echten Programmierung. Es ist eine Sprache mit Strukturen, Datentypen, strenger Typenbindung und Semikoli. Kurz, sie ist wie geschaffen um die Kreativitaet de typischen Echten Programmierers zu verkrueppeln.

Gluecklicherweise hat die jetzt vom DoD (Department of Defence) noch genuegend interessante Eigenschaften, um dem Echten Programmierer eine Annaeherung zu ermoeglichen: sie ist unglaublich komplex, sie enthaelt Moeglichkeiten, um mit dem Betriebssystem herumzumachen und Speicherbereiche neu zu verteilen, und Edgar Dijkstra mag sie nicht. Dijkstra ist wie man wissen sollte, der Autor von "GOTO's considered harmful", einem Meilenstein der Programmiermethodologie, der von PASCAL-Programmierern und Muesli-Fressern gleichermassen bewundert wird.

- Und ausserdem, ein zu allem entschlossener Echter Programmierer kann in jeder Sprache FORTRAN-Programme schreiben -

Der Echte Programmierer kann allerdings auch Kompromisse in Bezug auf seine Prinzipien eingehen und an etwas geringeren Aufgaben als der Vernichtung des Lebens arbeiten, sofern er dafuer entsprechend bezahlt wird. Viele Echte Programmierer schreiben z.B. Videospiele fuer ATARI, allerdings spielen sie nicht damit. Ein Echter Programmierer weiss, wie er die Maschine jedesmal schlagen kann, und damit ist es keine Herausforderung mehr. Jeder bei Lucas-Film ist ein Echter Programmierer, denn es waere doch verrueckt, das Geld von 50 Millionen STARS-WARS-Fans auszuschlagen.

Der Anteil der Echten Programmierern im Bereich der Computer-Graphics ist etwas niedriger als anderswo, was wahrscheinlich daran liegt, dass noch niemand irgendeinen Nutzen der Computer-Graphics entdeckt hat. Andererseits werden Computer-Graphics ueberwiegend in FORTRAN abgehandelt, daher gibt es einige Leute, die so das Schreiben von COBOL-Programmen vermeiden.

Im Allgemeinen spielt der Echte Programmierer wie er arbeitet - mit Computern. Er ist staendig darueber erheitert, dass sein Arbeitgeber ihn tatsaechlich fuer etwas bezahlt, was er nur so zum Spass sowieso tun wuerde - allerdings achtet er darauf, dieser Meinung nicht zu laut zu aeussern. Gelegentlich kommt der Echte Programmierer auch aus seinem Buero heraus, um sich ein wenig frische Luft und/oder zwei Bierchen zu genehmigen.

Hier daher einige Hinweise, wie man Echte Programmierer ausserhalb des Computerraumes erkennt:

- Auf Parties stehen Echte Programmierer in einer Ecke und diskutieren ueber Sicherheitsmassnahmen von Betriebssystemen und wie man um sie herumprogrammiert.
- Bei Fussballspielen vergleicht der Echte Programmierer die Ergebnisse mit seinem auf gruenliniertem Leporello-Papier gedruckten Computer-Simulations-Ergebnissen.
- Am Strand zeichnet der Echte Programmierer Flussdiagramme in den Sand.
- Ein Echter Programmierer geht in die Disco, um sich die Lichtorgel anzusehen.
- Bei Begraebnissen sagt der Echte Programmierer typischerweise: "Armer Hans-Helmut. Er war mit seinem Sortierprogramm schon fast fertig, als ihn der Herzinfarkt erwischt hat."
- Im Supermarkt besteht der Echte Programmierer darauf, seine Bierdosen selber ueber das Fenster des Barcodelesers zu schieben, weil er keinem Kassierer zutraut, dies beim ersten Versuch richtig zu machen.

In welcher Umgebung der Echte Programmierer am besten funktioniert? Nun, dies ist eine sehr wichtige Frage fuer die Manager von Echten Programmierern. Wenn man bedenkt, wie teuer es ist, einen von ihnen ihm Betrieb zu halten, dann sollte man ihn oder sie in eine optimale Arbeitsumgebung versetzen.

Der typische Echte Programmierer lebt vor einem Computerterminal. Rund um dieses Terminal liegen Ausdrucke von jedem Programm, an dem er je gearbeitet hat, sie stapeln sich grob chronologisch geordnet auf jeder ebenen Flaeche des Bueros. Im Zimmer verteilt finden sich ueber ein Dutzend mit kaltem Kaffee mehr oder weniger gefuellte Tassen. Gelegentlich schwimmen Zigarettenkippen darin herum, in einigen Faellen auch Reste von Orangenschalen. Irgendwo liegen Kopien des OS JCL-Manuals und der "Principles of Operation" herum. Ueber den Boden verteilt liegen Reste der Verpackungen von gefuellten Keksen (der Typ, der schon in der Fabrik so furztrocken gebacken wird, dass er auch bei laengerem Liegen im Automaten nicht schlechter wird).

Schliesslich, in der linken oberen Schublade des Schreibtisches, unter der Schachtel mit Muntermachern, liegt eine Schablone fuer Flussdiagramme, die sein Vorgaenger dort vergessen hat. Echte Programmierer schreiben Programme und keine Dokumentation; das ueberlaesst man den Typen von der Wartung.

Der Echte Programmierer ist in der Lage, 30, 40, ja sogar 50 Stunden in einem Rutsch zu arbeiten, und das unter hohem Zeitdruck. Genaugenommen mag er es so am liebsten. Schlechte Antwortzeiten regen den Echten Programmierer nicht auf - sie geben ihm die Chance, zwischen 2 Kommandos ein bisschen Schlaf zu ergattern. Wenn die Planung nicht genug Zeitdruck bereithaelt, dann tendiert der Echte Programmierer dazu, seine Arbeit herausfordernder zu machen, indem er sich die ersten neun Wochen mit einem kleinen, aber sehr interessanten Teil des Problems befasst, um dann in der letzten Woche seine Aufgabe in zwei oder drei 50-Stunden-Marathonsitzungen zu beenden. Dies beeindruckt nicht nur den Manager, sondern schafft gleichzeitig eine hervorragende Entschuldigung fuer das Fehlen einer Dokumentation.

Und ueberhaupt: kein Echter Programmierer arbeitet von 9 bis 5, ausser denen von der Nachtschicht. Echte Programmierer tragen keine Schlipse. Echte Programmierer tragen keine hochhackigen Schuhe. Echte Programmierer kommen zur Arbeit, wenn andere zum Mittagessen gehen. Ein Echter Programmierer vergisst vielleicht den Vornamen seiner Angetrauten, aber niemals den Inhalt der gesamten ASCII- (oder EBCDIC-) Tabelle. Echte Programmierer koennen nicht kochen. Da Supermaerkte um 3 Uhr morgens selten geoeffnet sind, muessen sie sowieso von Kaffee und Keksen leben.

Die Zukunft betrachtend machen sich eine ganze Reihe von Echten Programmierern Sorgen, dass die juengste Programmierergeneration nicht mehr mit der gleichen Lebensperspektive aufwaechst wie sie selbst. Viele der Juengeren haben noch nie eine Computer mit einer Maschinenkonsole gesehen. Kaum ein Schulabgaenger kann heute noch hexadezimal rechnen, ohne einen Taschenrechner zu benutzen. Die Programmierung durch symbolische Debugger oder Texteditoren, die Klammern zaehlen, und benutzerfreundliche Betriebssysteme. Und das Schlimmste ist, einige von ihnen werden auf die Menschheit losgelassen ohne je FORTRAN zu lernen! Sind wir dazu verdammt, eine Industrie von UNIX-Hackern und PASCAL-Programmierern zu werden?

Nun, aus meiner Erfahrung heraus glaube ich behaupten zu duerfen, dass das Schicksal den Echten Programmierern wohlgesonnen ist. Weder OS/370 noch FORTRAN zeigen irgendwelche Symptome des Aussterbens, trotz aller Anstrengungen der PASCAL-Programmierer. Selbst subtilere Tricks wie das Hinzufuegen strukturierter Schleifen zu FORTRAN sind fehlgeschlagen. Sicher, einige Computerhersteller liefern FORTRAN-

77-Compiler, aber jeder einzelne von ihnen laesst sich ueber eine einzige Compiler-Option in einen FORTRAN-66-Compiler verwandeln - mit DO-Schleifen wie von Gott geschaffen.

Selbst UNIX scheint fuer den Echten Programmierer nicht mehr so schlecht zu sein wie frueher. Die neueste UNIX-Version hat das Potential eines Betriebssystems, das eines Echten Programmierers wuerdig ist. Sie hat zwei verschiedene, leicht inkompatible Benutzer-Schnittstellen, einen geheimnisvollen und komplizierten Teletype-Treiber und virtuellen Speicher. Und wenn der Echte Programmierer die Strukturierung ignoriert, kann er sich sogar mit C anfreunden. Schliesslich gibt es keine Typenbindung, Bezeichner sind sieben (zehn? acht?) Zeichen lang und man hat Zeiger als Bonus. Das ist, als haette man die besten Teile von FORTRAN und Assembler vereint, von den kreativeren Moeglichkeiten des #define ganz zu schweigen.

Nein, die Zukunft ist nicht voellig schlecht. So hat sich in den vergangenen Jahren die populaere Presse sogar ueber clevere neue Brut von Computer-Schraten und -Hackern geaeussert, die Plaetze wie Stanford oder das MIT zugunsten der Wirklichkeit verlassen haben. Allen Anzeichen nach lebt der Geist der Echten Programmierer weiter in diesen jungen Maennern und Frauen. Und solange es schlecht beschriebene Ziele, bizarre Fehler und unrealistische Zeitplaene gibt, solange wird es Echte Programmierer geben, die bereit sind einzuspringen und das Problem zu loesen, und die sich die Dokumentation fuer spaeter aufheben.

LANG LEBE FORTRAN !

ED POST, WILSONVILLE

*** Brought to you by BIRNE for TORNADO BBS ***

PublicDomain für Modell 4/4P
==========

Im letzen INFO (Nr. 22) wurden mehrere PublicDomain Programme für das Modell 4/4P vorgestellt. Hierbei ist zu berücksichtigen, daß ein Teil der Programme mit doppelseitigen Laufwerken nicht arbeitet. Im einzelnen sind dies:

RESTORE/CMD - RUN4/BAS - ZAP4/CMD - SPACEMAP/CMD

Das Programm WD/CMD (Windows) arbeitet nur mit der US-TRSDOS Version.

Klaus Hermann

Hallo liebe Clubkameraden !

Wie im letztem Club-Info schon beschrieben, haben wir jetzt auch eine große CP/M 2.2-Software Bibliothek !

Ich habe bislang nur drei Zuschriften bekommen !!! Ist denn die CP/M 2.2-Software nicht gefragt ?

Damit Ihr einen kleinen Eindruck von der ganzen Biblothek bekommt, habe ich das Inhaltsverzeichnis der CP/M-USER-GROUP ausgedruckt:

Disk-Nr.:
---

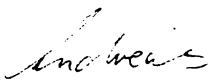
01 = "Various CP/M Utilities - 0"
02 = --> see CPMUG Vol. 10 11
03 = "BASIC-E Games and Programs"
04 = "ACTOR, ML80 and Examples of FORTRAN-80 Code"
05 = "BASIC-E Programs and Microsoft BASIC programs"
06 = "Maillist Program and CP/M Utilities"
07 = "PILOT - Programmed Enquiry Learning Teaching"
08 = "Various CP/M Utilities - 1"
09 = "General Ledger from Interface Age Vol.2 No.10"
10 = "LLBASIC: Lawrence Livermore BASIC interfaced to CP/M"
11 = "TINIBASIC and Processor Technology BASIC/5"
12 = "PILOT Interpreters interfaced to CP/M"
13 = "BASIC-E, CBASIC Microsoft BASIC Programs"
14 = "Various CP/M Utilities - 2"
15 = "Utilities and non-BASIC Games"
16 = "Assemblers, Other Utilities and FOCAL"
17 = "Utilities, Denver Tiny BASIC non-BASIC Games"
18 = "Maths Routines, Monitors and CASUAL (no CP/M I/O yet)"
19 = "Various CP/M Utilities - 3"
20 = "BASIC-E, CBASIC Programs Pictures"
21 = "Microsoft BASIC Games Programs"
22 = "Monsterous StarTrek Games in BASIC"
23 = "STOIC - Stack Oriented Interactive Compiler"
23B= "STOIC - Overflow Programs From Vol 23" (in CPMUG 25)
24 = "CP/M Utilities, Macro Libraries RATFOR"
25 = "Various Assembler Utilites"
26 = "Microsoft BASIC FORTRAN Games and Utilities"
27 = "Microsoft BASIC Games"
28 = "BASIC-E Games Utilities, ALGOL-M"
29 = "Assembler Games Utilities, BASIC-E Source"
30 = "BASIC-E Ver 1.4 Source Code in PL/M"
31 = "Tarbell BASIC - 1"
32 = "Tarbell BASIC - 2"
33 = "Search and Rescue Programs"
34 = "SAM-76 Language"
35 = "FELIX - Graphics Animation System"
36 = "Assemblers, Editors and Utilities"

37 - "CBASIC2 Games, Utilities and CAI Programs"
38 - "Speed Up and Tarbell Disk Controller Utilities"
39 - "Music Programs for SOL CP/M"
40 - "Various Utilities, Disk Catalog System"
41 - "Ham Radio, Chess and FORTRAN programs"
42 - "Disassemblers, Diablo Clock Routines"
43 - "OSBORNE A/P,A/R Business Software I"
44 - "OSBORNE G.Ledger, Budget,ROBO, Business Software II"
45 - "OSBORNE Payroll Business Software III"
46 - "CP/M utilities, CPM LABEL, DU, DIR, REASM"
47 - "CP/M Utilities, MODEM, Copy, BMAP, PROM"
48 - "BDS C Sampler Disk, Comp. by Leor Zolman"
49 - "Rational FORTRAN (RATFOR), precompiler"
50 - "PASCAL PASCAL Compiler , Prgs. for UNIX"
51 - "STAGE2 Macro Processor by Dick Curtiss"
52 - "Copyfast vers. 3.5, BATCH/VARBATCH"
53 - "BDS-C Adventure disk"
54 - "BASIC Games, CAI Programs"
55 - "Adventure run-time disk"
56 - "Adventure source code implemented for CP/M"
57 - "Expanded Adventure supercedes CPMUG 56"
58 - "Miscellaneous CP/M Utilities"
59 - "8080 8085 memory and ICOM Controller diagnostics"
60 - "6502 Simulator, 6502 Zapple Monitor"
61 - "Bulletin Board, File Transfer CP/M Utilieties"
62 - "PASCAL Comunication programs"
63 - "Misc. CP/M Utilities"
64 - "Games, Disassembler, North Star Basic, CDOS simulator"
65 - "MITS-CP/M file conversion, PHELP, Utilities, FIG-Forth"
66 - "HELP File system for MBASIC, M80, CBASIC other"
67 - "CPMUG CATALOG disk vol. 01 - 42 with comments"
68 - "Miscellaneous CP/M Utilities"
69 - "Miscellaneous CP/M Utilities"
70 - "Miscellaneous CP/M Utilities"
71 - "Miscellaneous PASCAL-Z Programs - 1"
72 - "Miscellaneous PASCAL-Z Programs - 2"
73 - "Miscellaneous PASCAL-Z Utilities - 3"
74 - "Miscellaneous PASCAL-Z Utilities - 4"
75 - "PASCAL-Z Programs - 5, MBASIC Disassembler, Date routines"
76 - "Miscellaneous PASCAL-Z Utilities - 6"
77 - "PASCAL-Z Utilities - 7, Datebase System, Disk sort Program"
78 - "Miscellaneous CP/M Utilities"
79 - "MODEM Programs for MMI SMARTMODEM"
80 - "CROMENCO structured BASIC Programs, Mail list, Spelling"
81 - "Miscellaneous CP/M Utilities, Submit, Editor, Hard Disk"
82 - "North Star BIOS Routines"

Wie Ihr seht, gibt es eine ganz schöne Auswahl !

Wenn Ihr wollt, dann könnt Ihr noch mehr Informationen auf Disketten anfordern. Das Diskettenformat müßt Ihr allerdings mit angeben !

Bis Bald



CLUB 80 — Bücherei

Hallo Freunde,

viel erfreuliches gibt es wieder von der Bücherfront zu berichten. Zunächst wäre da zu vermelden, daß es Jens endlich gelungen ist, zwei Sammelordner mit den "alten" Infos Nr. 1 bis 15 zusammenzustellen. Die Infos sind auf Karton vervielfältigt, so daß sie (hoffentlich) auch mehrere Einsätze auf dem Kopierer überleben.

Die Ordner werden wohl vornehmlich für Neuzugänge interessant sein, die sich noch nicht die älteren Ausgaben des CLUB 80-Info besorgt haben (oder besorgen konnten). Die Ausleihzeit beträgt, da eine rege Nachfrage zu erwarten ist, maximal drei Wochen. Die Besteller werden, wenn beide Ordner unterwegs sind, in eine Warteliste aufgenommen, so daß zuerst malt, wer zuerst kommt! Selbstverständlich geht das Porto zu Lasten des Ausleihers, so daß der Bestellung gleich drei DM (ev. auch in Briefmarken) beigelegt werden sollten.

Die Infos 16 bis 21 werden demnächst als Sammelband verfügbar sein, so daß Neuzugänge immer auch das schon dagewesene zur Verfügung haben. Ich möchte die Ausleiher an dieser Stelle bitten, die Ordner möglichst schonend zu behandeln, damit sie noch möglichst lange den Neuen zeigen können, wie aktiv ein Club mit nur ca. 60 Mitgliedern sein kann (und hoffentlich noch lange sein wird)!

Ebenfalls erfreulich ist die Tatsache, daß der Heise-Verlag wieder mal seine Lager räumt und der CLUB 80 dadurch sehr hochwertige Bücher zu äußerst günstigen Preisen erwerben kann. Welche der bestellten Bücher tatsächlich geliefert werden ("nur solange Vorrat reicht" stand auf dem Angebot), werde ich euch erst im nächsten Info mitteilen können. Wenn aber alles von einer solchen Qualität ist wie das Buch, das schon jetzt als Neuanschaffung in der Bibliothek auzuleihen ist, können wir sehr zufrieden sein.

Erst gestern habe ich, ganz kurzfristig und durch Zufall, eine Neuerwerbung für die CLUB 80-Bücherei einkaufen können. Es handelt sich dabei (Model- 1 und III- sowie Video Genie-Besitzer bitte nicht neidisch werden) um das Model 4/4p Technical Reference Manual. Dieses Buch vereint technische Informationen über das Gerät und über das Betriebssystem TRSDOS 6.x. Von "Model 4/4p Theory of Operation" über die genaue Beschreibung der verwendeten Spezial-IC's (CRT-Controller MC6835, Baud Rate Generator BR1941, Floppy Disk Controller FD 179x-02 usw.) bis zur detaillierten Erläuterung der Diskettenorganisation und den, für Assemblerprogrammierer sehr wichtigen, SVC's (Supervisor Calls) ist alles vorhanden, was das Herz begehrt.

Wieder einmal ans Tageslicht befördern, und euch in Erinnerung rufen, will ich die Sammelmappen, die Günter Wagner in seiner Zeit als Vorsitzender einmal zusammengestellt hat. Es handelt sich dabei um Kopieen von Artikeln aus verschiedenen Computerzeitschriften (von ELCOMP bis Computer persönlich), die sich alle mit dem TRS 80 bzw. Video Genie beschäftigen. Die Sammlung dürfte vor allem für die Mitglieder interessant sein, die sich zur Erscheinungszeit der Artikel (1980 bis 1983) noch nicht mit Computern beschäftigt haben.

Damit genug zu diesem Thema. Bestellen könnt ihr die Sammelmappen und Bücher direkt bei mir (Rückporto nicht vergessen)

Hartmut Obermann

- I M P R E S S U M -

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

2. Vorsitzende
Gerald SCHRöDER
Am Schützenplatz 14
2105 Seevetal 1
☎ 04105 /2682

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

Diskothekar
Werner FöRSTER
Christoph-Krebs-Straße 9
8720 Schweinfurt
☎ 09721 /21841

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

Autoren
Die Redaktion bedankt sich bei den im INHALTSVERZEICHNIS genannten Autoren für die Mitarbeit an der Club-INFO.

Druck
Peter Spieß
Trugenhoferstraße 27
8859 Rennertshofen 1
☎ 08434 /454

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

Das INFO erscheint zweimonatlich.

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

S C H L U S S

Hallo Club-80er,

zum Abschluß des ersten 1988'er Info's darf ich mich wieder zu Wort melden. Diesmal bin ich mit dem INFO fast termingerecht fertig geworden. Leider habe ich auch eine negative Nachricht für Euch. Dazu gleich, zuerst aber meinen Dank an die fleisigen Autoren unserer INFO.
- Wieder sind wir um die hundert Seiten stark geworden.
- Macht weiter so!

Wie Ihr im Info schon habt lesen können, hat der Peter Spies seine Offsetdruckerei aus beruflichen Gründen aufgegeben. Ich hoffe, daß es mir gelingt rechtzeitig Ersatz für Ihn zu finden, so daß Ihr weiterhin Euer INFO bekommen könnt. Gleichzeitig bitte ich Euch um eure Mithilfe. Vielleicht hat jemand von Euch einen "Drucker" in der Hinterhand? -Offset natürlich!- Wer mir helfen kann melde sich bitte bei mir.

Wobei ich auch schon beim nächsten Thema bin.
Die Redaktionsadresse ändert sich !!
Ich bin zwar noch nicht "körperlich" umgezogen aber "postmäsig". Der Nachsendeantrag läuft für ein halbes Jahr. Trozdem bitte ich Euch alle, meine neue Adresse in Eure Listen aufzunehmen und ab sofort an den neuen (Redaktions-)Briefkasten zu schreiben:

Jens NEUEDER
Rudolf-Then-Straße 32
7178 Gschlachtenbretzingen

Telefonnummer bleibt gleich! Ihr könnt mich weiterhin unter 0791 / 42877 erreichen.

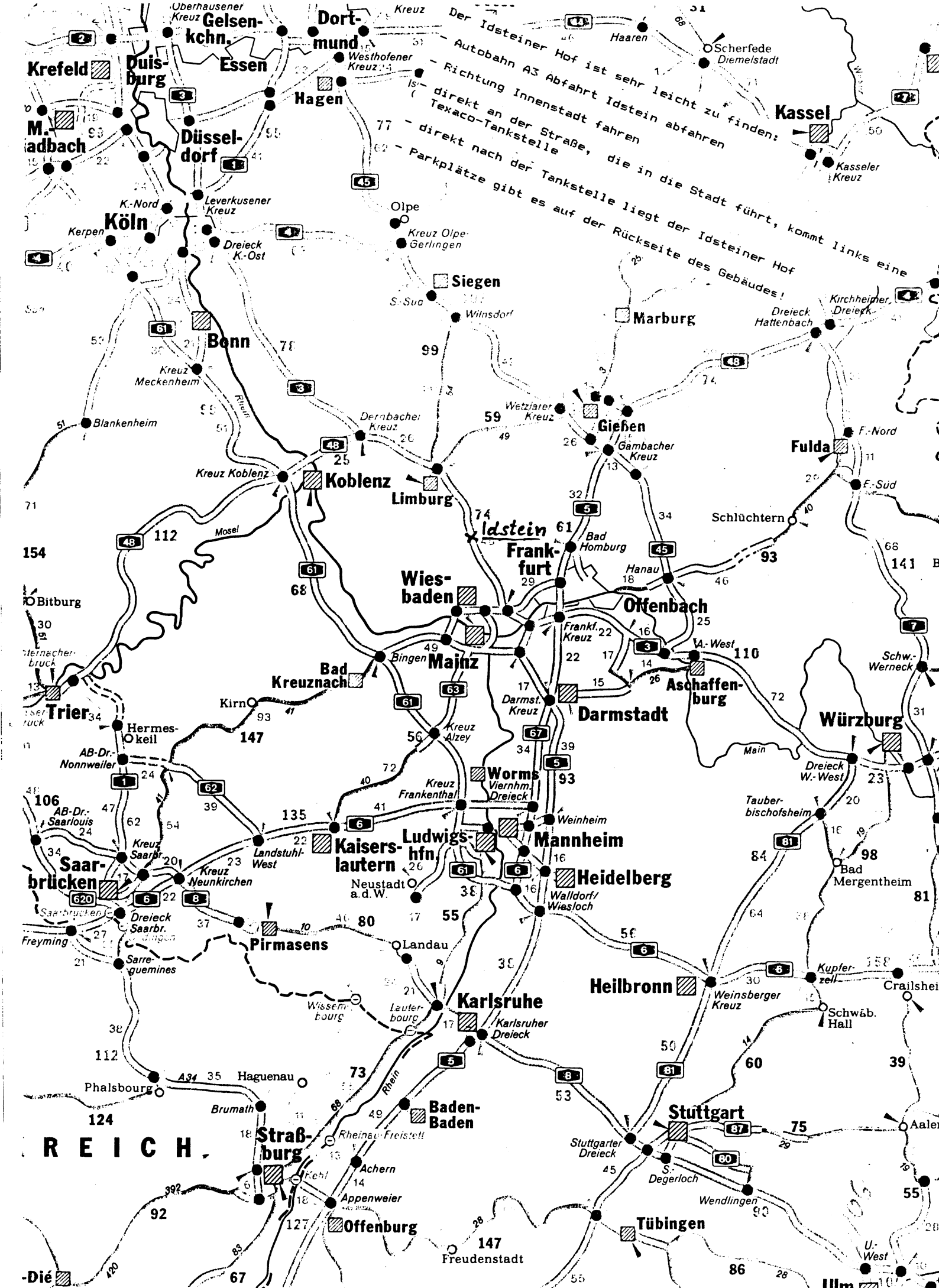
So, das war nun einmal wieder. Als Beilage findet Ihr in diesem INFO ein Infoinhaltsverzeichnis bis 1. 3. 1988 von KaJot.
Als weiterer Anhang ist noch ein Kartenauszug dabei. Ich bin sicher, daß Ihr damit die beste "Anflugschneise" nach Idstein finden werdet.
Eine neue Adressenliste habe ich diesmal nicht mit beigefügt. Es gibt noch einige säumige Zahler unter uns. Bis zum nächsten mal ist das sicher geklärt. Dann kommt auch eine neue Liste.

Bis zum Clubtreffen oder nächsten INFO grüßt Euch

Jens

STICHWORT	TITEL	VERFASSER	INFO-#/SEITE
ASCII	ASCII-TABELLE IMMER ZUR HAND	SOPP	19/19
ASSEMBLER	KLEINSCHRIFT fuer den EDTASM+	HEIDENREICH	20/37
	SCHUETZEN VON ASSEMBLERPROGRAMMEN	KONRAD J.	5
	HALBAUTOMATISCHE FESTLEGUNG VON ORG	KONRAD, J.	9/3
	MEIN DRITTES ASSEMBLERPROGRAMM	MUEHLENBEIN	16/33
	MEIN ERSTES ASSEMBLERPROGRAMM	MUEHLENBEIN	14/9
	"INPUT"	SCHROEDER	11/5
	ASSEMBLER CONTRA LISP	SCHROEDER	17/19
	DIE WERKZEUGE DES ASSEMBL.PROGRAMMIERERS	SCHROEDER	21/9
	EINIGE ASSEMBLER-ROUTINEN	SCHROEDER	14/17
	LMOFFSET with DISABLE DOS	SCHROEDER	9/9
	WERKZEUGE DES ASSEMBLER-PROGRAMMIERERS	SCHROEDER	21/9
	LPRINT ALLES einfacher! (Teil 3)	SOPP	9/11
	SCHOENER LISTEN MIT ZEUS	SOPP	18/41
	WER HAT ANGST VOR ASSEMBLER?	SOPP	5
	ZEUS MIT 64 UND 80 ZEICHEN	SOPP	18/36
	BASIC RUFT ASSEMBLER	WOLLSCHLAEGER	21/11
	TIP FUER SCHNELLERES ARBEITEN MIT EDTASM	ZWICKEL	6/67
BANKER	SYSCOPY1 und RAMDISK f.HELMUT's BANKER	SCHROEDER	16/32
BANKING	256K-RAM fuer Z80-SYSTEME	BERNHARDT	14/91
BASIC	EXTENDED BASIC(RESTOREn u. LPRINT alles)	OBERMANN	11/33
	EIN MYSTERIOESER BASIC-BEFEHL: IsA	SOPP	5
BERICHT	GENIE IIs - Erfahrungsbericht	SCHROEDER	9/67
BILDFORMAT	FUTTER fuer GENIE IIIs CRTC	HEIDENREICH	20/22
BILDSCHIRM	BILDSCHIRMAUSGABE mit INVERSEN ZEICHEN	?	12/79
	BIN ICH STRAHLENGEFAEHRDET?	BOIKAT, Ute	20/109
	DAS PROJEKT (FERNSEHER --> MONITOR)	MUELLER, K.	18/61
	NOCH EIN BILDSCHIRMSCHONER	SCHMID, A.	22/26
	SCHWIMMENDE BILDER	SCHMID, A.	22/66
	80-ZEICHENKARTE X80, SEDIT und TSCRIPS	SCHROEDER	13/17
	DIE 80-ZEICHEN-KARTE	SCHROEDER	12/63
	EIN SCHONER fuer die BILDROEHRE	SOPP	20/20
	EIN WEITERER BILDSCHIRMSCHONER	SOPP	22/28
	FUTTER fuer den CRTC	SOPP	19/40
	MEHR FERNSEHEN fuer's GELD (f.GENIE IIIs	SOPP	14/21
	NOCH MEHR FERNSEHN fuers GELD	SOPP	19/39
	VIDHEX - HEXANZEIGE d.BILDSCHIRMS m.HRG	SOPP	9/13
CHIP	BEI SIEMENS HEISST ES: CHIP CHIP HURRA!	ZEITSCHRIFT	18/76
	ERSTER MEGABIT-CHIP IN SERIE	aus Zeitschrift	18/18
	DER LEIDENSWEG DER CMOS-CHIPS	aus c't	19/67
	NANOHENRY BLOCKIEREN DEN FORTSCHRITT	aus c't	19/70
CODE	BASICODE-INTERFACE	?(aus ELEKTOR)	3
	BASICODE-2	?(aus Elektor)	3
COMP.-MUSIK	MEIN ZWEITES ASSEMBLERPROGRAMM	MUEHLENBEIN	15/15
COMP.WECHSEL	UNTERSCHIEDE (zw. MODEL 1 und 3 bzw. 4)	OBERMANN	14/55
COMPILER	BASCOM-KURZANLEITUNG	MUEHLENBEIN	13/38
	RPNL - DIE ANDERE ART ZU PROGRAMMIEREN	MUELLER, KURT	15/57
	RPNL-BESCHREIBUNG	MUELLER, KURT	16/37
	CALL mit BASCOM	OBERMANN	12/33
COMPUTER	MODEL 4p ROM-IMAGE-LOADER	OBERMANN	15/43
	TANDY<--> SCHNEIDER	OBERMANN	17/49
	CP/MAC-eine BRUECKE zw.G-/NEWDOS u.CP/M	RENSCH	18/44
	WIE PHOENIX AUS DER ASCHE (4 COMP.in 1)	SCHROEDER	17/9
	DEIN GIIIs... (NACHTRAG)	SOPP/BERNHARDT	17/54
	TRS-80 MODELL 4p, DER SCHOENSTE PORTABLE	WOLLSCHLAEGER	12/69
	TANDY - BETTER AGAIN	ZEITSCHRIFT	16/40
	32 BIT zum NIEDRIGPREIS (TANDY 4000)	aus - ? -	22/79
CONTROLLER	FLOPPYCONTROLLER-KOMPATIBILITAET	RETZLAFF	19/47
CP/M	WAS HABT IHR GEGEN CP/M ?	BERNHARDT	21/53
	GENIE, CP/M UND 80 ZEICHEN UND HRG???	HELD	21/43
	CMD ==> COM ohne CPMac	SCHROEDER	21/23
	F.HEIMWERKER: WIE BAUE ICH MIR EIN CP/M?	SCHROEDER	21/25

STICHWORT	TITEL	VERFASSER	INFO-#/SEITE
CP/M	Wie baue ich mir ein CP/M ?	SCHROEDER	21/25
DATEI	INDEX-DATEIEN	HERMANN	19/15
DATENBANK	BDBS - BASIC DATENBANK SYSTEM (MODEL 4)	HERMANN	22/37
	dBASE II - eine Einfuehrung	MAND	21/33
DATENUEBERTR	VIDEODAT 300	?	13/51
DDE	DDE fuer das GENIE IIIs	SOPP	12/17
DEF.FUNCTION	BASIC-PROGRAMMIERBARE FUNKTIONEN	HERMANN	19/26
DENKAUFGABEN	MATHEMATIK fuer FEINSCHMECKER	WEBER	20/101
DFUe WDR	DATENUEBERTRAGUNG p.FERNSEHBILD/WDR-CLUB	EDDE u.HERIBERT	12/62
DOS	R-BEFEHL MSDOS-LIKE (ein Editor f. DOS)	HEIDENREICH	20/44
	LS-DOS 6.3 - KOPIERSCHUTZ	HERMANN	21/55
	EXTENDED NEWDOS 80	OBERMANN	10/5
	NEWDOS/80+10 VERSION 2.4 (26.10.85)	OBERMANN	11/25
	SUPERDOS	OBERMANN	2
	DIR/SYS ODER INHALT/SYS?	RENSCH	15/21
	FILTER UNTER TRSDOS 6.x	SOERENSEN, R.	18/29
	MDOS im INFO???????	SOERENSEN, R.	20/11
	DOS - (FAST) OHNE FLOPPY	SOPP	14/69
DOS-FEHLER	FEHLER IM NEWDOS-BEFEHL 'BOOT'	RETZLAFF	11/21
DOUBLER	SELBSTBAU-DOUBLER fuer EXP1		17/45
DRUCKEN	DREISPALTIG DRUCKEN	SOPP	21/15
	EIN FEHLER IM SYS28/SYS	SOPP	22/33
	LPRINT MIT MODERNEN GENIES	SOPP	18/19
DRUCKER	SPEED WRITING (fuer EPSON MX-DRUCKER)	LINDSEY	12/65
	...WEIL KUERZE SO WUERZIG IST...	MUEHLENBEIN	20/35
	DIE DRUCKERABFRAGE	SCHMITZ	18/12
	VON 8 AUF 24 PINS (= DRUCKNADELN)	SOPP	22/20
DRUCKERGRAF.	GRAPHIK-JKL, TSCRIPS UND DER NEC P6	SOPP	16/27
DRUCKZEICHEN	AM BILDSCHIRM DEFINIERT/EDITOR f.DRUCKER	KLUGE	9/19
ECB-BUS	ECB-ADAPTER-PLATINE	?	15/65
	VERBINDEND	ASSENBAUM	11/39
	DIE EINGANGSPLATINE	DROWAELDER	15/73
	ECB-BUS-GRUNDEINHEITSAUFBAU	NEUEDER	15/71
	ECB-BUS-PROJEKT	NEUEDER	14/101
	ECB-BUS-TRS80-PROJEKT	NEUEDER	13/39
	ECB-BUS fuer MODEL 3/4!???	OBERMANN	15/84
ECHTZEITUHR	ACH DU LIEBE ZEIT!	SOPP	15/32
	REAL TIME BLACK BOX?	SOPP	15/53
ERFAHRUNG	PC STATT GENIE? (Ueber weitere PC's)	SCHMITZ	20/99
ERROR	"PRUEFZAHLFEHLER BEIM LESEN"	BERNHARDT	22/71
FILEVERGLCH	VERGLEICH ZWEIER DISK-DATEIEN	SOPP	22/53
	VERGLEICH ZWEIER PROGRAMME: noch'n TOOL	SOPP	22/57
FLOPPY	FLOPPY-TRICKS	OBERMANN	13/41
FORMATIERUNG	40 TRACKS? WARUM NICHT 42?	OBSCHERNINGKAT	18/11
GENAUIGKEIT	MATHEMATISCHER UEBERFLUSS	MUEHLENBEIN	20/17
GRAFIK	ERGAENZUNG ZU GRAFIK-SHORTY	HERMANN	19/32
	TRS80 M1/3 SCHNELLE GRAFIK IN BASIC	HORNUNG	2
	HRG-BILDUEBERTRAGUNG OHNE HRG (WDR-FILES	OBERMANN	15/11
	SCHNELL BEWEGTE GRAFIK AUF DEM TRS80	ROECKRATH	2
	GRAFIK-STANDARD: ALLEIN GEHT'S AUCH	SCHROEDER	19/11
	EINF.GRAFIKPROGRAMM: PLOT/BAS	SCHROEDER /ARPS	14/29
	LINIENMODELLE IN 3 DIMENSIONEN	SCHROEDER/ ARPS	14/35
	GRAFIK-CLS fuer das GENIE IIIs	SOPP	13/35
	STRINGANZEIGE MIT VOLLDAMPF	SOPP	6/41
GRAFIK-JKL	GRAFIK-JKL fuer das GENIE IIIs	SOPP	13/32
HAENDLER	TANDY-VERTAGSHAENDLER		19/79
	TANDY SCHLIESST COMPUTER-CENTER	ZEITSCHRIFT	18/75
HARDCOPY	HARDCOPY AUCH OHNE INTERRUPT	RYCHLIK	20/15
	FLOPCOPY AUCH OHNE INTERRUPT	SOPP	21/14
HARDWARE	DATENFERNUEBERTRAGUNG		6/19
	RELAIS-INTERFACE	?	9/47
	AUSLESEN DES GRAFIKSPEICHERS	BERNH./SEELMANN	20/65


Der Idsteiner Hof ist sehr leicht zu finden:
- Autobahn A3 Abfahrt Idstein abfahren
- Richtung Innenstadt fahren
- direkt an der Straße, die in die Stadt führt, kommt links eine Texaco-Tankstelle
- direkt nach der Tankstelle liegt der Idsteiner Hof
- Parkplätze gibt es auf der Rückseite des Gebäudes!
Idstein
Krefeld
Duisburg
Essen
Gelsenkchn.
Dortmund
Hagen
Düsseldorf
M.-adbach
Köln
Leverkusener Kreuz
K.-Nord
Dreieck K.-Ost
Kerpen
Olpe
Kreuz Olpe-Gerlingen
Siegen
Wilnsdorf
Marburg
Dreieck Hattenbach
Kirchhainer Dreieck
Bonn
Kreuz Meckenheim
Blankenheim
Dernbacher Kreuz
Wetzlarer Kreuz
Gießen
Gambacher Kreuz
Fulda
F.-Nord
F.-Süd
Schlüchtern
Kreuz Koblenz
Koblenz
Limburg
Bad Homburg
Frankfurt
Hanau
Wiesbaden
Offenbach
Frankf. Kreuz
A.-West
Aschaffenburg
Darmstadt
Darms. Kreuz
Bad Kreuznach
Bingen
Mainz
Kirn
Trier
Hermeskeil
Kreuz Alzey
Worms
Viernhm. Dreieck
Würzburg
Dreieck W.-West
Main
Tauberbischofsheim
Bad Mergentheim
AB-Dr. Nonnweiler
AB-Dr. Saarlouis
Kreuz Frankenthal
Weinheim
Ludwigshfn.
Mannheim
Heidelberg
Walldorf/ Wiesloch
Kreuz Saarbr.
Saarbrücken
Landstuhl-West
Kaiserslautern
Kreuz Neunkirchen
Neustadt a.d.W.
Dreieck Saarbr.
Pirmasens
Landau
Freyming
Sarreguemines
Wissembourg
Lauterbourg
Karlsruhe
Karlsruher Dreieck
Heilbronn
Weinsberger Kreuz
Kupferzell
Schwäb. Hall
Crailsheim
Phalsbourg
A34
Haguenau
Brumath
Rhein
Baden-Baden
Rheinau-Freistett
Straßburg
Kehl
Achern
Appenweier
Offenburg
Freudenstadt
Stuttgart
Stuttgarter Dreieck
S.-Degerloch
Wendlingen
Tübingen
U.-West
Ulm
Aalen
Schw.-Werneck
Haaren
Scherfede
Diemelstadt
Kassel
Kasseler Kreuz
Kreuz
Mosel
Rhein
REICH