CLUB 80

Clubinfo der

TANDY - GENIE - und KOMTEK ANWENDER

25. AUSGABE

KONTAKTADRESSE : CLUB 80 / GERALD SCHRÖDER / AM SCHÜTZENPLATZ 14 / 2105 SEEVETAL 1
TEL.: 04105 / 2602

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

Tachchen!

Zuerst möchte ich mich dafür entschuldigen, daß Ihr erst jetzt das dritte Info dieses Jahres erhaltet. Wie Ihr gemerkt habt, war das Drucken ein größeres Problem als vorauszusehen war. Dafür ist die Druckqualität aber meiner Meinung nach sehr gut. Ich hoffe, daß ich in Euer aller Namen spreche, wenn ich mich bei Horst-Dieter dafür bedanke und ihn bitte, das Info weiterhin so zu drucken.

Über eins bin ich allerdings etwas enttäuscht: Eure schreibenden Tätigkeiten (bzw. die nicht stattfindenden). Obwohl zwischen dem ersten und zweiten Info dieses Jahres mehrere Monate lagen, kamen nicht viele Artikel zusammen. Ich gebe natürlich zu, daß Jens noch einige Artikel für dieses Info zurückgehalten hat, aber trotzdem könnten es mehr sein.

Also nochmal die Aufforderung: Schreibt Artikel! Ihr fragt Euch: "Worüber?" Die Antwort: "Alles!" Es ist keineswegs so, daß Arnulf und Helmut die Exklusivrechte haben oder für ihre Artikel bezahlt werden. Sie dokumentieren nur alles, was sie tun, und teilen es Euch mit, damit Ihr daraus lernt, Anregungen bekommt und auch etwas ähnliches versucht. Es wäre doch nun sehr schön, wenn Ihr den (wenigen) regelmäßigen Schreibern mal zeigt, was Ihr macht. Immerhin wäre es ja möglich, daß diese auch etwas davon interessiert.

Nun könnt Ihr zwar sagen: "Ich bin ja der einzige, der ... macht; das interessiert doch niemanden." Aber diese Ausrede gilt nicht. Seht Euch nur Arnulf an: er ist das einzige Mitglied mit einem Genie IIIs und dann schreibt er frecherweise gleich seitenlange Artikel dazu. Ich möchte diese aber nicht missen, denn allein die Assembler-Tricks und ungewöhnlichen Ideen, die er in seinen Listings offenbart, sind Gold wert. Warum sollten Eure Berichte weniger interessant sein? Ich würde gern mal etwas über Eure Erfahrungen mit verschiedenen Textverarbeitungsprogrammen hören; oder die Masse Leute, die jetzt mit einem Banker von Helmut arbeitet: was macht Ihr eigentlich damit? Hat denn niemand mal Programme dazu geschrieben oder die vorhandenen verbessert oder sie überhaupt benutzt? usw. Die Liste könnte ich endlos fortsetzen.

Meine Bitte lautet: Seht Euch an, was Ihr mit dem Rechner macht und schreibt dann darüber einen Artikel. Oder lest Euch mal den Rest dieses Infos durch und gebt den Schreiberlingen das, was die Psychologen "Feedback" nennen: Schreibt im Info, was Ihr dazu denkt, was Ihr nicht verstanden habt, was besser sein könnte.

Übrigens könntet Ihr auch eine Eure Meinung zu einem Buch äußern, sofern Ihr mal eins lest. Einen Anlauf hatte ich ja schon in Info 21 gemacht und es sind inzwischen einige wenige Besprechungen erfolgt, aber ich kann mir nicht denken, daß Ihr so wenig schmökert. Ein Steckbrief zur Suche in Euren Schränken: Bücher sind diese Dinger mit den vielen beschmutzten Seiten, die auch noch gründlich durchnumeriert sind. In unserer Clubbibliothek sollen auch einige Dutzend davon stehen (bzw. liegen und sich langweilen). Fragt mal Werner danach!

So, jetzt habe ich Euch genug aufgemuntert. Nehmt's Euch mal zu Herzen, damit das Info wieder ein paar Seiten mehr füllt.

Gerald Schröder

Happy Birthday

Servus und Grüß Gott an alle "Geburtstagskinder" des CLUB 80

als erstes möchte ich mich für die verspäteten Geburtstagswünsche entschuldigen, die einige sicherlich treffen werden. Aus diesem Grunde habe ich diesmal nicht,wie üblich, drei Monate sondern vier Monate berücksichtigt.

Zwei unserer Mitglieder haben einen besonderen Geburtstag zu feiern. Im August und September werden Arnulf Sopp und Helmut Bernhardt, unsere beiden Nordlichter, 40 Jahre alt.
Meinen allerherzlichsten Glückwunsch an beide.

Ach Gott, beinahe hätte ich einen ganz wichtigen Geburtstag vergessen. Denn man glaubt es kaum, aber mein werter Göttergatte wird 30 Jahre alt (Wer ist es? Tip: siehe Geburtsjahr bei unter stehender Liste).

Die genauen Daten der drei und alle übrigen findet Ihr nachfolgend.

Juli	:	**18.**	**Hartmann**	**Hans-Günter**
		19.	**Brandl**	**Hermann**
		30.	**Retzlaff**	**Bernd**
August	:	**02.**	**Issig**	**Günter (Neumitglied)**
		05.	**Stephan**	**Hans-Martin**
			Heidenreich	**Ulrich**
		10.	**Bernhardt**	**Helmut (1948)**
		15.	**Misioch**	**Waldemar**
		16.	**Braun**	**Harald**
		17.	**Wagner**	**Günther (Clubgründer)**
		24.	**Obermann**	**Hartmut (1958)**
September	:	**08.**	**Albers**	**Herbert**
		11.	**Rychlik**	**Andreas**
		23.	**Sopp**	**Arnulf (1948)**
		24.	**May**	**Holger**
Oktober	:	**03.**	**Schneider**	**Bernd**
		15.	**Stober**	**Reiner**
		30.	**Held**	**Manfred**

Im Namen des Vorstandes wünsche ich Euch allen ein fröhliches Feiern und weiterhin alles Gute

J u t t a

Aktuell und wichtig!

Jutta hat das alte Clubkonto aufgelöst und ein neues eröffnet. Die Bankverbindung lautet jetzt:

Hartmut Obermann Sonderkonto
Postfach 27
6209 Heidenrod

Kto-Nr.: 496 071-605
beim Postgiroamt Frankfurt am Main
BLZ: 500 100 60

Achtung!!

Wer eine Teilnahmebescheinigung für das Clubtreffen braucht, soll sich unter Beilage des Rückportos bei Gerald Schröder melden. Die Kosten für das Clubtreffen können dann von den Steuer abgesetzt werden.

-- Termine -- Termine -- Termine --

[illegible] ....29. Oktober 1988

[illegible] ....Köln 20. - 25. Oktober 1988

"Unsere Neuen!"

Hallo Clubfreunde,

hiermit stelle ich mich kurz vor,

Ich heiße Günter Issig, geb. 02.08.57, wohnhaft in Heilbronn. Beruflich habe ich nichts mit EDV zu tun und bin deshalb noch total unwissend.

Durch Zufall bin ich an einen TRS 80/II gekommen. Dies warf bis jetzt jedoch jede Menge Probleme auf.

Es wäre toll, wenn sich jemand findet, der mit mir in Kontakt treten und mir bzgl. Bedienung etc. weiterhelfen kann.

Bis "hoffentlich" bald

G. Issig

Hallo Clubfreunde !

Als neues Clubmitglied möchte ich mich kurz vorstellen.

Ich heiße Wolfgang Preußing, wohnhaft in Essen, bin 32 Jahre alt, verheiratet und habe eine vierjährige Tochter.
Über das Elektronikbasteln kam ich zur Computerei. 1981 erwarb ich einen TRS 80 Mod. 1. Im Laufe der Zeit kamen ein Expansions Interface der Fa. CE Computer Elektronik, ein Laufwerk BASF 6106 (SS,DD), ein Drucker CP 80 Type II und ein Akustikkoppler Tandy AC-3 hinzu. Selbst gelötet habe ich die I/O Karte aus 80-Micro 12/84 mit dem 8255. Z. Zt. versuche ich die RB V 24 - Karte zum Laufen zu bringen. Bisher noch ohne Erfolg.

Seit etwa zwei Jahren habe ich auch beruflich mit EDV zu tun (IBM AT unter Open Access II).

Auf gute Zusammenarbeit.

Hallo Gleichgesinnte!!

hiermit möchte auch ich mich als neues Mitglied im Club kurz vorstellen.

Mein Name ist Heinz-Dieter Meklenburg, geb.01.10.50 wohnhaft in Leck, ca. 17 Km von der Dänischen Grenze entfernt. Beruflich bin ich als Elektromaschinenbauer ausgebildet worden, und habe dann in Abendschule meinen Elektronic-Pass absolviert.

Mein erster Computer war der ZX 81 von Sinclair im Jahre 73, den ich immer noch besitze, und mit dem ich schon nach kurzer Zeit auf die Assemblersprache kam.
1979 bekam ich dann den GENI I,bei dem ich mit der gleichen Programiersprache weiter gemacht habe.

Im März 88 kaufte ich mir allerdings einen Atari 1040 ST dazu, um mit ihm speziell Schaltpläne und Layout`s zu zeichen.

Ich hoffe auf eine gute Zusammenarbeit und verbleibe hiermit Euer

H.D. M.

Heinz-Dieter Meklenburg

* *Wogegen selbst ZEUS [einer unter vielen Göttern] * *
* * vergeblich kämpft ... * *

– „Links steht HEX, rechts steht ASCII". –

– „Hier können Sie leicht ASCII, also Klartext lesen". –

Diesen und ähnlichen Unsinn kann man immer wieder lesen, sogar in Veröffentlichungen großer Meister, deren Name Gewicht hat in der Runde der Software- und Maschinensprache-Experten! [Natürlich nicht bei Clubmitgliedern...]

Der erste Satz bezieht sich z.B. auf die Sektordarstellung in SUPERZAP. In der Tat: „Links steht HEX..."
Aber was steht rechts? – Antwort: KLARTEXT – nicht „ASCII"!
„Klartext" allerdings nur, wenn es sich um ein BASIC-Programm handelt; bei Maschinenprogrammen erscheinen nur die „messages" im Klartext. (Woraus messerscharf folgt, daß alles übrige in einem Maschinenprogramm „unklar" ist? --- Aber nur für die Ahnungslosen.
Zu denen gehöre ich. Aber ich vermute einen Sinn dahinter...
[An ZEUS muß man eben glauben])

Rumpelstilzchen singt heute:
„Glaubst du wirklich, jeder weiß,
was A.-ES.-CE.-I.-I. heißt?" *)

ASCII = American Standard CODE for Information Interchange = Amerik. Code f. Ifo-Austausch.

*) Rumpelstilzchen war kein Dichter...

Der Ton liegt auf CODE, ihr Lieben!
Buchstaben, Ziffern und Zeichen, kurz: „Characters" wurden codifiziert! Das „Gegenteil" des Codes, der Klartext, das sind die Zeichen u.s.w., die „Characters".
Welcher BASIC-Freak kennt nicht die Codier-Anweisung für den Buchstaben Q:

ASC(Q) = 81_{dez} = 51H !

Dieser Freak weiß auch, daß die gegenteilige Anweisung

CHR$(81) = CHR$(&H51) = „Q"

lautet! Das ist Klartext – nicht ASCII! Klar?

Und eben das steht rechts bei einem Sektordump! – Gleichzeitig sehen wir eben, daß die ASCII-Darstellung sowohl durch HEX- als auch durch Dezimalwerte möglich ist. Im Sektordump (z.B. von SUPERZAP) steht der linke, d.h. der ASCII-Teil in HEX. Er könnte hier – rein theoretisch – auch in DEZ stehen, wenn das technisch möglich wäre. Nur die Grundstruktur (2x4 Bit) unserer EDV erlaubt das nicht. – Aber deshalb ist der Teil, der rechts steht, noch lange kein ASCII, sondern die Rückübersetzung des Codes in die Zeichensprache.
Daß diese bei Maschinenprogrammen keineswegs immer leserlicher „Klartext" ist, liegt einfach daran, daß auch die Maschinenbefehle (2^n-Bit-Codes) stur in „Zeichen" übersetzt werden [-von wem??-], nicht in mnemonics.
Und schon gar nicht – in „ASCII"!

Ich danke euch, liebe Leser, daß ich meinem „Ärger" einmal Luft machen durfte – Ärger über den Verstoß gegen unser „Grundgesetz": Die LOGIK. – Kajot

Bankeinbrüche schrecken mich nicht. Ich kann nichts dabei verlieren.
"DISAGIO" (fachmännische Aussprache: Diss-Aschi-Oh!) ist, glaube ich, ein unanständiges Wort.
"Diskont" klingt schon vertrauter für den Freak; es heißt wohl soviel wie "Diskette eingeschaltet" (? Das "t" am Schluß ist sicher wieder einer der üblichen Druckfehler.)
Aber "Rediskont"? Es erinnert mich an die Zeiten, als es noch keinen Kugelschreiber gab und man mit Stahlfedern des Typs "TO" oder "REDIS" schrieb. In der Tat: Mit der Rediskon(n)te ich viel besser schreiben als heute! Manchmal sogar "Schönschrift"! Aber ob ich's heut' noch mit der Rediskönt...? -
"Diskönt' i' nimmer!"
Schließlich fiel mir in letzter Zeit ein Begriff auf, der sich auf "Wort" reimt: der "Port"! Also, ich muß schon sagen, mein "Portemonnaie" (schreibt man's so?) ist alles andere als ein "Hafen" - und schon gar kein sicherer!!

Kurzum: Ihr versteht jetzt schon klipp und auch klar,
daß ich wohl schon immer ein *GREENHÖRNER* war,
was Bankwesen, Ports und Verwandtes betrifft.
Ich hatt' diese Klippen "erfolgreich" umschifft...
Doch seit ich Besitzer - wie kam's in den Sinn? -
von 8 (acht) beachtlichen Banken nun bin (!),
kenn' ich nur noch heimlich ein einzig' Verlangen:
Nie möchte ich mehr um die Banken mich bangen!!

Was nützen die harden und soften Experten,
die freundschaftlich Tat, Rat und Hilfe gewährten,
die man mit Fragen und Klagen gehetzt - satt! -
wenn sich im Kopf ein "Hangup" festgesetzt hat!
(Ich faß mich an die Stirne
und fühl' 'ne weiche Birne!)

Nun ist ein Mondlicht aufgegangen,
die güldnen * * * * *, ach, sie prangen
am Monitor so klar!
Der Ce-Pe-Uhu nicht mehr schweiget
und aus des Keyboards Tiefen steiget
der BANK-Direktor! . . .

WUNNERBAR!

(Nachtrag:) Das Gedicht heißt: "Die RAMDISK".

Für die meisten von euch natürlich ein alter Hut.
Aber vielleicht nützt doch dem einen oder anderen (insonderheit jenen, die nicht in IDSTEIN waren) das kleine BASIC-Programm, das ich zum Testen eines Bankers (nicht Bankerts!) geschrieben habe.

Womit ich auf keinen Fall den Experten, die sowas in Maschinensprache vollziehen, Konkurrenz machen will - Gott behüte! Weiß ich doch nicht einmal, ob das Testprogramm auch für andere Bank-Unternehmen geeignet ist. Ich fürchte sogar, daß es nur meiner Bank-Filiale schmeckt (s.LOGO). Aber trotzdem: Laßt mich doch auch einmal auf meinen Käse stolz sein! Mir schmeckt er...

Und vielleicht ist dies eine Anregung für jene Clubkollegen, die noch nie etwas hier beigetragen haben, weil sie glaubten (oder fürchteten), ihr Geistesprodukt sei Käse... Merket:

Je schlimmer für die Neese,
je besser schmeckt der Keese!!

Dieses schwöre ich bei Gott!
KaJott.

```
1 CLS:PRINT"*****************************************************
***********
2 PRINT"*   Programm-Name :  BANKEN/TST          Datum : 24.07.88
    *
3 PRINT"*   Verfasser :       K.-J.Muehlenbein, Weinheim
    *
4 PRINT"*   Das Programm testet Helmut Bernhardts Banker
    *
5 PRINT"*              ACHTUNG! Vor dem Programmlauf muss im DOS
    *
6 PRINT"*                 HIMEM = 7FFFH      gesetzt werden!
    *
9 PRINT"*****************************************************
*******
10 CLEAR260:DEFINTA-K:DEFSTRL-Z
20 PRINT:PRINTCHR$(222)"MENUE
"CHR$(222)"-----"
30 PRINT"
          1  Bank anwählen
          2  Texteingabe in die Bank
          3  Textausgabe von der Bank
          4  Ende"
40 INPUT"Wahl ";C
50 ONCGOSUB70,80,120,160
60 GOTO165
70 PRINT:INPUT"Bank-Nummer ";B:OUT236,B:RETURN
80 PRINT:LINEINPUT"Texteingabe <nicht mehr als 3 Zeilen !>

";T
90 PRINT:INPUT"ab welcher Adresse speichern <zwischen 32768 und
65280!> ";S:A=VAL(S)-65536
110 FORI=OTOLEN(T)-1:POKEA+I,ASC(MID$(T,I+1,1)):NEXT:RETURN
120 PRINT"Ausgabe aus Bank"B"<J/N>";:INPUTX
130 IFX="N"ORX="n"GOSUB70
140 CLS:PRINTCHR$(255):PRINTCHR$(255)
145 FORJ=OTOLEN(T)-1
150 PRINTCHR$(PEEK(A+J));
155 NEXT:RETURN
160 END
165 PRINT§980,"Weiter mit <ENTER>
170 Y="":Y=INKEY$:IFY=""THEN170ELSECLS:GOTO20
```

Ableitung der Negationsregel für Binärzahlen

(von K.-J.Mühlenbein, Weinheim; 06201/55052)

Vorbemerkung

Die Addition von Binärzahlen ist einfach.
Die Subtraktion geschieht durch Addition des negativen Wertes.
Den negativen Wert einer Binärzahl erhält man durch Umkehr der Setzungen; d.h. Einser werden zu Nullen und Nullen zu Einsern gemacht; anschließend wird 1 addiert.
Wie kommt es zu dieser eigenartigen "Negationsregel"?
Dies soll nachstehend abgeleitet werden.

A. Grundlagen

1) Die allgemeine Form eines nach "Stellenwertigkeit" gebildeten Zahlensystems lautet:

$$a_n B^n + a_{n-1} B^{n-1} + \ldots + a_i B^i + \ldots + a_2 B^2 + a_1 B^1 + a_0 B^0$$

worin B "Basis" genannt wird, n ganzzahlig ist und die a_i ganzzahlige Koeffizienten sind, für welche gilt: $0 \leq a_i \leq B-1$.

Die Exponenten geben den "Stellenwert" an; n kennzeichnet die höchste Stelle.

2) Der Bereich eines beliebigen, nach dem "Gesetz der Stellenwertigkeit" gebildeten Zahlensystems Z(B;S) sei durch eine maximale Stellenzahl S=n+1 begrenzt.

Folgerung: Ein Dezimalsystem (B=10) mit höchstens S=4 Stellen hat dann die Zahl "9999" als höchsten Wert. Die Zahl "10000" (und größere) ist in diesem System nicht definiert ("nicht-existent"), denn für dieses System haben nur die 4 letzten Stellen Gültigkeit. Es ist dann also "10000" = (1)0000 = Null!
Der so begrenzte Wertebereich umfaßt einschließlich der Null B^S ganze Zahlen.

Diese Einschränkung berücksichtigt die begrenzte Anzahl Bits (8 bzw. 16) bei der Zahlendarstellung durch den Computer.

3) Die Anzahl m (gültiger) Stellen in einer solchen Zahlendarstellung ist gleich der Anzahl Summanden gemäß Definition 1), d.h. es gilt: $m = n + 1$.

Beispiel: Im System mit B=2 wird die Dezimalzahl '1' dargestellt durch $2^0 = 1$. Hier ist n = 0 und m = n + 1 = 1.
Die Dezimalzahl '5' wird dargestellt durch
$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$; es ist n = 2; m = 2 + 1 = 3.

4) Im System Z(B;S) sind nach 2) alle Ausdrücke A = Null, für die $A \geq B^S$ gilt, denn bereits B^S hat gemäß 3) S+1 Stellen, ist also im System "nicht-existent".

5) Eine Zahl $\overline{X}$ heiße "komplementär zu X", wenn gilt: $\overline{X}+X = 0$.
$\overline{X} = -X$ entspricht somit der Negation. Wegen 4) kann gesetzt werden: $\overline{X} + X = B^S = 0$

6) Man findet demnach den zu X komplementären Wert $\overline{X}$ durch die Bildung:

$$\overline{X} = B^S - X$$

Da $\overline{X} = (-X)$, läßt sich als "Allgemeine Formel" schreiben:

$$\boxed{(-X) = B^S - X}$$

B. Anwendung auf das Dezimalsystem

Es sei B=10 und S=4. ein „eingeschränktes" Dezimalsystem $Z(10;4)$.
Der Wertebereich reicht dann von 0 bis $10^4 - 1 = 9999$ und umfaßt 10^4 ganze Zahlen.
Der zu X komplementäre Wert $\overline{X}$ muß die Bedingung $\overline{X} + X = B^S = 10^4$ erfüllen (Satz A5). $\overline{X}$ ist daher die Größe, die den Wert X zu 10000 "komplementiert".
10^4 hat jedoch m = n+1 = 4+1 = 5 Stellen und besitzt somit für das Zahlensystem Z(10;4) den Wert "Null": $\overline{X} + X = (1)0000$. Um im System zu bleiben, wird 10000 durch 9999+1 ersetzt. Die Gleichung schreibt sich nun:

$$\overline{X} = 9999 + 1 - X = (9999-X) + 1$$

Die "Allgemeine Formel" ist einfacher:

$$\overline{X} = (-X) = 10^4 - X$$

Beispiel:
Die zu X=5703 im System Z(10;4) komplementäre Zahl $-X$ ergibt sich wie folgt:

$$(-X) = (9999 - 5703) + 1 = „4297"$$

Bei der Subtraktion 9999-5703 wird jede Ziffer des Subtrahenden zur Ziffer 9 "komplementiert"; dies ergibt das "Neunerkomplement" 4296 als Zwischenwert. Nach Addition von 1 erhält man das "Zehnerkomplement" 4297. Im Sinne des hier definierten Dezimalsystems mit S=4 gilt daher die "Negation": $-5703 = „4297"$!

Man merke sich:
Jede Ziffer von X wird zur höchsten Ziffer im System komplementiert!
(Die höchste Ziffer ist stets B-1; im Dezimalsystem (B=10) also die "9").

C. Anwendung auf das Binärsystem

Im Binärsystem (Dualzahlensystem) des 8-Bit-Codes ist B=2 und S=8 ==> Z(2;8).
Sein Bereich umfaßt die Werte von 0 bis $B^S - 1 = 2^8 - 1 = 255$, insgesamt die Anzahl $B^S = 2^8 = 256$ ganze Zahlen (einschl.Null). Ein Wert $B^S = 2^8 =$"256" ist in diesem System jedoch nicht definiert, denn diese Zahl hätte m=n+1=8+1=9 Stellen.
Statt "256" kann aber "255+1" geschrieben werden. Den zu X komplementären Wert $\overline{X} = (-X)$ erhält man daher wieder aus der Bildung:

$$\left.\begin{array}{l}(-X) = \text{höchster Systemwert} + 1 - X \\ \text{bzw.} \\ (-X\langle bin\rangle) = 11111111\langle bin\rangle - X\langle bin\rangle + 1\langle bin\rangle\end{array}\right\} \text{(Gleichung C)}$$

Die Rechenregel für die Negation von Binärzahlen besteht also wieder darin, daß jede "Ziffer" (d.i. jedes Bit!) von X zur höchsten Ziffer des Systems (das ist hier die "1") "komplementiert" - d.h. das "Einerkomplement" gebildet - und danach 1 addiert wird!
Das "Einerkomplement" der Null, also die Ergänzung zur Eins, ist natürlich die Eins. Das "Einerkomplement" der Eins, also die Ergänzung zur Eins, ist - nun, wer errät es? .....
Richtig: Die Null!

Die Addition einer Eins ergibt dann das sogenannte "Zweierkomplement". (Eine nicht streng konsequente Bezeichnung; denn das "Ergänzende" (das Komplement) ist hierbei nicht die Zwei, sondern die addierte "1"!)

Die schnellste Lösung erhält man jedoch wieder durch die "Allgemeine Formel":

$$(-X) = 2^8 - X$$

Beispiel:
Im Dezimalsystem (mit unbegrenzten Stellen) ist -100 der negative Wert (die "Negation") von 100.
Wie lautet -100 im 8-Bit-Binärsystem ?
Die "Allgemeine Formel" liefert:
(-X) = 2^8<dez> - 100<dez> = 256<dez> - 100<dez> = 156<dez>.
Versuchen wir, dies binär darzustellen, so müssen wir die umständlichere "Formel" (Gleichung C) zuhilfe nehmen, da 256<dez> sich nicht im 8-Bit-System, also nicht mit einem Byte darstellen läßt:

Hochster Wert:	1111 1111	<bin> | -
X = 100<dez> =	0110 0100	<bin>
Einer-Komplem. =	1001 1011	<bin>
+ 1<dez> =	0000 0001	<bin>
-X = - 100 <dez> =	1001 1100	<bin> = "156"<dez>

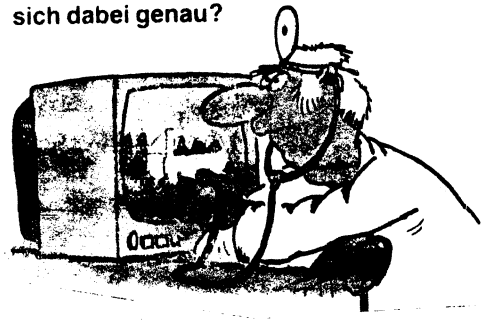
Etwaige "Overflow-Bits", die bei dieser Operation intern auftreten können, entfallen bei der "Allg. Formel" automatisch.

PS: Zahlen in Anführungszeichen bedeuten, daß nicht zwischen „Normaldarstellung" und Darstellung als „Komplementzahl" unterschieden wurde. (In der „normalen" Mathematik besorgt dies das Minuszeichen.)

Kajot

Unser Musikrätsel

Wer kennt sie nicht, die hübsche Titelmelodie aus der Fernseh-Serie „Schwarzwaldklinik"! Um was für ein Musikstück aber handelt es sich dabei genau?



A – Um die Fraktur in Zeh-Mull, Knöchelverzeichnis 256, von Ludwig van Bettpfosten?

B – Um die Akute Sinfonie in Tee-Dur, gespielt von den Wiener Vielhormonikern?

C – Um die Arie „Wie eiskalt ist dies Hehnchen", dargeboten von den Egerländer Simulanten unter der Leitung von Skalpellmeister Brinkmann?

Binärumwandlung, letzter (?) Teil

In meinem Beitrag "Binärumwandlung aus Hex und Dez und kürzere E-Routine in SYS14/SYS" ging es um ein interessantes und häufig brauchbares Programm, das Zahleneingaben in Hex oder Dez binär nach DE lädt. Aus gegebenem Anlaß habe ich mich dieser Routine noch ein paarmal gewidmet. Sie ist jetzt noch etwas kürzer geworden, hauptsächlich aber besser:

Wie gehabt zeigt HL beim Einsprung auf den Beginn des Eingabestrings. Bei der Rückkehr zeigt es auf das Zeichen nach der ASCII-Zahl. Bei DOS-Befehlen ist das z. B. 0Dh (ENTER) oder ein Trennzeichen zum nächsten Befehlsparameter (Komma oder Blank). DE enthält die Zahl in ihrer binären Entsprechung. Der Akku wird verändert. Das Flag-Register enthält Z=1, (Zero-Bedingung), wenn es nur gültige Hex- oder Dez-Ziffern oder eines der üblichen Trennzeichen gab (kein Fehler). Wenn ein Zeichen des Strings regelwidrig war, ist die NZ-Bedingung erfüllt (der Akku enthält in jedem Fall den Fehlercode "schlechte Parameter", 2Fh). BC, das nur innerhalb der Routine eine Funktion erfüllt, wird nicht verändert.

Ein Anwenderprogramm oder ein entsprechend angepaßtes SYS-File kann folgendermaßen von den Vorteilen profitieren: Zahleneingaben werden zwar auf gültige Zeichen überprüft, aber das Z-Flag kann ignoriert werden. So ist es möglich, in einer Eingabe das Trennzeichen zur nächsten Eingabe wegzulassen. Außerdem ist bei dieser Methode der Hex-Kenner "H" überflüssig, wenn es eindeutige Hex-Ziffern gibt (A-F). Wenn es aber nur Ziffern gibt, die auch das dezimale Zahlensystem kennt (0-9), ist "H" erforderlich. 27 ist nun einmal nicht dasselbe wie 27h. Aber für diese Routine ist 2A=2Ah, weil die Ziffer A nicht im dezimalen System vorkommt. Dann ist aber NZ erfüllt, damit es der Anwender, wenn er will, ganz genau nehmen kann.

Zuweilen sollte er das. Ist nämlich das dem Zahlenstring folgende Zeichen zufällig eine Hex-Ziffer (A-F), dann wird sie klaglos gelesen und ausgewertet. In solchen Fällen sollte man schon ENTER, ein Komma, ein Blank oder (bei Hex-Eingaben) H anfügen. Oder eben das Z-Flag beachten.

Es gibt einen Ausnahmefall, bei dem "kein Fehler" (Z=1) signalisiert wird, obwohl das Trennzeichen irgendwelcher Quatsch sein kann: Wenn HL auf Rubbish, bloß auf keine Dez- oder Hexzahl zeigt (gar keine Eingabe), dann ist nur scheinbar alles in Butter. DE enthält dann 0000h.

Eine ganze Reihe von SYS-Programmen und Anwender-Files akzeptieren dezimale oder sedezimale Eingaben. Ihre Umwandlungsroutinen können alle durch einen nur drei Bytes langen CALL an das irgendwo residente Programm ersetzt werden. Wo es residieren soll, muß sich der geneigte Leser je nach seiner Hardware aushecken. Im Listing ist die Standard-Adresse 5200h angegeben. In CALVA-DOS, dem Nachfolger von G-DOS 2.4 für das Genie 3s, lautet sie 3B68h.

Arnulf Sopp

```
5200            00001           ORG     5200h
                00002 ;UP liest ASCII-Zahleneingaben in Hex und Dez aus (HL) binär nach DE ein
5200  C5        00003 start     PUSH    BC              ;retten
5201  E5        00004           PUSH    HL              ;dto. Befehlszeiger
5202  010001    00005           LD      BC,0100h        ;B (- Flag: 1. Durchlauf, C (- Konst. 00
5205  CD2152    00006           CALL    cnvrtup         ;Eingabe einlesen und binär umwandeln
5208  7E        00007           LD      A,(HL)          ;letztes Zeichen nochmals einlesen
5209  D641      00008           SUB     'A'             ;binär umwandeln
520B  FE05      00009           CP      05h             ;07 wäre Hexkenner 'H' (maximal erlaubt)
520D  300B      00010           JR      NC,exit1        ;raus m. Fehlerflag, falls unerlaubtes Z.
520F  E3        00011           EX      (SP),HL         ;HL (- Stringzeiger
5210  41        00012           LD      B,C             ;Flag: 2. Durchlauf des UP cnvrtup (Hex)
5211  CD2152    00013           CALL    cnvrtup         ;Rest der Eingabe einlesen und umwandeln
5214  7E        00014           LD      A,(HL)          ;letztes Zeichen nochmals einlesen
5215  FE48      00015           CP      'H'             ;Kenner für Hex-Eingabe?
5217  2003      00016           JR      NZ,exit2        ;ohne Hex-Kenner: 'schlechte Parameter'
5219  23        00017           INC     HL              ;Befehlszeiger hinter letztes Zeichen
521A  CB48      00018 exit1     BIT     1,B             ;wurde das Segment cnvrtup erreicht?
521C  3E2F      00019 exit2     LD      A,2Fh           ;Fehlercode 'falsche Parameter'
521E  C1        00020           POP     BC              ;Stack bereinigen
521F  C1        00021           POP     BC              ;Register restaurieren
5220  C9        00022           RET
                00023
5221  51        00024 cnvrtup   LD      D,C             ;DE (- Ausgangswert 0000 für Binärzahl
5222  59        00025           LD      E,C             ;(oben MSB, jetzt LSB)
5223  7E        00026 cnvloop   LD      A,(HL)          ;Zeichen einlesen
5224  D630      00027           SUB     '0'             ;ASCII -) binär
5226  FE0A      00028           CP      0Ah             ;höher als Dez-Ziffer? (evtl. Hex-Ziffer)
5228  380A      00029           JR      C,ciphok        ;falls Dez-Ziffer (erlaubtes Zeichen)
522A  CB40      00030           BIT     0,B             ;ist es der 2. Durchlauf des UP? (Hex)
522C  C0        00031           RET     NZ              ;fertig, falls nein (1. Durchlauf, Dez)
522D  D611      00032           SUB     11h             ;falls Hex-Ziffer: 'A' -) 00 usw.
522F  FE06      00033           CP      05h+1           ;noch erlaubte Hex-Ziffer? ('F' = 05)
5231  D0        00034           RET     NC              ;Ende, falls nicht 'A' ... 'F'
5232  C60A      00035           ADD     A,0Ah           ;'A' ... 'F' -) 0A ... 0F
5234  E5        00036 ciphok    PUSH    HL              ;Umwandlung ok., Befehlszeiger retten
5235  62        00037           LD      H,D             ;HL (- Zwischenergebnis (MSB)
5236  6B        00038           LD      L,E             ;(LSB)
5237  CB85      00039           RES     1,B             ;Flag: UP cnvrtup wurde ganz durchlaufen
5239  29        00040           ADD     HL,HL           ;*2 (Zwischenergebnis links rotieren)
523A  29        00041           ADD     HL,HL           ;*4 (um zwei Stellen)
523B  CB40      00042           BIT     0,B             ;2. Durchlauf des UP? (bei Hex-Eingabe)
523D  2003      00043           JR      NZ,convdez      ;falls nein (Dez-Eingabe)
523F  29        00044           ADD     HL,HL           ;*8 (Hex-Eingabe)
5240  1801      00045           JR      convhex         ;dort weiter
5242  19        00046 convdez   ADD     HL,DE           ;plus Zwischenergebnis (Dez: *5)
5243  29        00047 convhex   ADD     HL,HL           ;Hex: *16, Dez: *10
5244  5F        00048           LD      E,A             ;DE (- letzte Ziffer (Einerstelle)
5245  51        00049           LD      D,C             ;als 16-Bit-Zahl (MSB = 0 = 00)
5246  19        00050           ADD     HL,DE           ;(bei Hex: vorläufiges) Endergebnis
5247  EB        00051           EX      DE,HL           ;DE (- Enderg. (Hex: vorl., Dez: endg.)
5248  E1        00052           POP     HL              ;Befehlszeiger
5249  23        00053           INC     HL              ;nächste Stelle im Befehlsstring
524A  18D7      00054           JR      cnvloop         ;dort weiter
5200            00055           END     start

00000 Fehler

ciphok  5234    cnvloop 5223    cnvrtup 5221    convdez 5242    convhex 5243    exit1   521A
exit2   521C    start   5200
```


Screening mit DSMBLR/CMD

Es gibt Disassembler, die selbsttätig DEFMs und DEFBs erzeugen. Sie haben jedoch alle zwei entscheidende Nachteile: DEFW kommt da nicht vor, weil der Disassembler nicht ahnen kann, was als Byte, und was als Word von Belang ist; DEFM und DEFB werden nur dann vermutet, wenn ansonsten Schwachsinn wie etwa LD A,A als Programmcode erzeugt werden müßte, oder wenn ein nicht von Zilog dokumentierter (aber u. U. durchaus ausführbarer und sinnvoller!) Befehl auftaucht. In allen anderen Fällen wird gnadenlos Programmcode erzeugt, auch wenn es sich in Wirklichkeit um eine Tabelle oder gewöhnlichen Text handelt.

Da lobe ich mir DSMBLR. Er bringt diese reichlich fragwürdige Grundintelligenz nicht mit, ist dafür aber zugunsten eines großen Source-Puffers schön kurz. Aber ihm muß dann eben in einer besonderen Disk-Datei mitgeteilt werden, wo Text (DEFM) oder Bytes (DEFB) bzw. Words (DEFW) zu definieren sind. Das ist ein ASCII-File, das man z. B. mit TSCRIPS schreiben kann. Dazu muß mit einer geeigneten Utility (z. B. FED) herausgefunden werden, wie solche Adressen lauten. Zu Fuß geht es auch: Am Record-Header kann man die Ladeadresse des ersten Bytes eines Records ablesen. Wenn man bis dorthin weiterzählt, wo der Programmcode aufhört, und Daten beginnen, kommt man auch zum Ziel.

Wie auch immer, es ist ziemlich aufwendig: FED anschmeißen, Adressen ermitteln, aufschreiben, TSCRIPS laden, Adressen eintippen. Das hat mich bei SYS4 (endlose Fehlermeldungen des DOS) fast eine Stunde gekostet. Danach tat ich den heiligen Schwur, solchen Stumpfsinn nur noch vom Computer erledigen zu lassen und schrieb ein Programm:

SCREENER lädt ähnlich wie SUPERZAP das zu disassemblierende Maschinenprogramm sektorweise in den Speicher. Ein Cursor erscheint, mit dem man beliebig im Sektor herumgurken kann. Wo DEFB, DEFW oder DEFM vermutet wird, genügt ein Tastendruck, um die jeweilige Ladeadresse automatisch in einen Puffer schreiben zu lassen. Nach getaner Arbeit kann man, wiederum mit einer Taste, die Abspeicherung des Screening-Textes auf Diskette veranlassen. Für die genaue Bedienung von SCREENER möchte ich auf die nachfolgende Kurzanleitung verweisen.

Wer das Programm haben möchte, kann sich an unseren Club-Diskothekar Werner Förster wenden. Er bekommt dann den Sourcecode, der mit ZEUS zu assemblieren ist. Wer ein G3 oder G3s besitzt, findet eine mit Sicherheit fehlerfreie Version vor. Beim Model 1 und dem Genie 1/2 bin ich ziemlich sicher, daß ebenfalls alles glattgehen müßte (ich besitze ja nur das G3s zum Austesten). Die Version für das Model 3/4 hat leider noch einen Fehler, an dem aber Hartmut Obermann z. Zt. arbeitet. Für Rückmeldungen über etwaige Fehler in der Version für das Model 1 (Genie 1/2) und für Anregungen zu einem weiteren Ausbau wäre ich dankbar.

Die Diskette von Werner enhält nur den Quellcode. Die Kurzanleitung erscheint nur hier im Info, weil mein wild gepatchtes TSCRIPS gewisse auf meinen Drucker zugeschnittene Zeichen erzeugt, die bei einer anderen TSCRIPS-Version und einem anderen Drucker möglicherweise Ärger verursachen können.

Arnulf Sopp

SCREENER ist ein Programm, mit dem bequem Screening-Files für DSMBLR erzeugt werden können. Es wird nur als Assembler-Quellcode für ZEUS weitergegeben, der für den eigenen Computer neu assembliert werden muß. Zur Anpassung an den Rechnertyp bietet der Sourcecode gleich am Anfang eine Hilfe an:

```
;Labels zur Anpassung des Programms an den eigenen Computer
m1      EQU     0       ;Model 1 usw.: 1    andere Rechner: 0
m3      EQU     0       ;Model 3:      1    andere Rechner: 0
g3      EQU     1       ;Genie 3/3s:   1    andere Rechner: 0

;Sicherung gegen falsche Label-Zuteilung (mehrfach 1 oder 3X 0)

model EQU       m1+m3+g3-1      ;das Resultat muß 0 ergeben
```

Das Label, das den eigenen Zielcomputer bezeichnet, muß zu 1 definiert werden, die beiden anderen müssen 0 lauten. Wird hierbei ein Fehler gemacht, kehrt das Programm automatisch beim Programmstart mit einer Fehlermeldung ins DOS zurück, noch bevor typspezifische Änderungen im DOS vorgenommen worden sind. Das Programm ist dann nach der Fehlerkorrektur neu zu assemblieren.

Initialisierung des Screening-Textpuffers:
Nach dem Programmaufruf wird der User gefragt, ob ein noch im Puffer stehender Text weiter bearbeitet werden, ein angefangener Text von Disk geladen oder der Speicher neu beschrieben werden soll.
<D> lädt einen Text von der Diskette,
<R> restauriert den alten Text,
<N> legt den Puffer neu an.
Wenn nach <A> kein Screening-Text im Puffer gefunden wird, wird dieser gelöscht (wie bei <N>).

Laden des Zielprogramms:
Danach wird der Benutzer nach dem Namen des zu disassemblierenden Maschinenprogramms gefragt. Die Antwort kann alle für Filespecifications üblichen Zusätze enthalten. Lautet die Extension "CMD", dann kann sie weggelassen werden (wird automatisch angehängt). Danach wird der erste Sektor des Zielprogramms angezeigt. Ein Cursor erscheint, der beim Genie 3/3s das jeweilige Zeichen invertiert, bei anderen Modellen einen Graphikblock erzeugt.

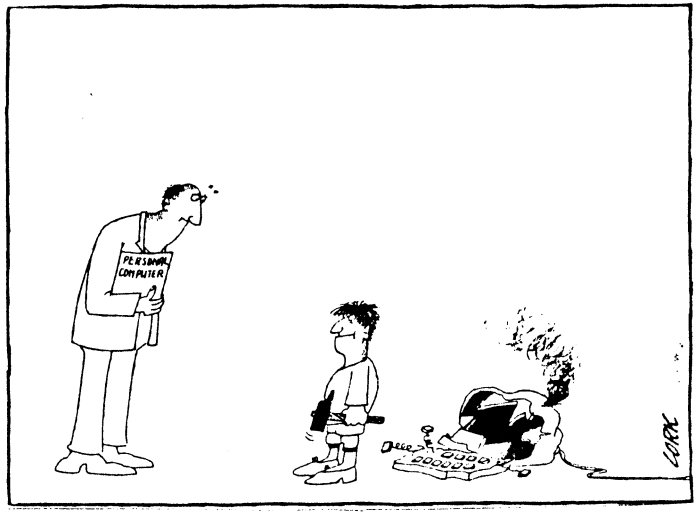
Mögliche Eingaben bei der Sektoranzeige:
Die Pfeiltasten steuern den Cursor. Mit Shift setzen sie ihn *ganz* nach oben, unten, links oder rechts.
Wenn der Cursor auf einem Byte steht, wo der Benutzer keinen Programmcode, sondern andere Daten vermutet, können Screening-Daten zur Generierung von Pseudo-Ops in DSMBLR erzeugt werden:
<M> erzeugt '$xxxx' für DEFM,
<W> " '#xxxx' für DEFW,
<B> " 'xxxx' für DEFB,
</> " '-xxxx' für "bis" (DEFB usw. "von - bis").
(xxxx ist in Hex-ASCII die Ladeadresse des markierten Bytes.)
Die errechneten Adressen werden unten links zur Kontrolle angezeigt. Die "bis"-Adresse wird nicht errechnet 1. am Textanfang (das wäre unsinnig), und 2. wenn das letzte Zeichen bereits ein "bis" war (Fehler).
Wenn der Cursor bei der Eingabe eines dieser Zeichen auf einem Record-Header oder in einem Kommentar-Record steht (kein ladbarer Code), ertönt beim Genie 3/3s ein BEEP, bei anderen Modellen unterbleibt lediglich die Anzeige der Adressen.

Edition weiterer Sektoren:
<;> (Plus ohne Shift) lädt den nächsten Sektor.
<+> (Plus mit Shift) " " letzten " .
<-> " " vorigen " .
<=> (Minus mit Shift) " " ersten " .
In allen Sektoren können in gleicher Weise Screening-Daten erzeugt werden.

Edition des Screening-Textes:
<A> löscht den Bildschirm und zeigt den bisher erzeugten ASCII-Text an. Korrekturen sind möglich:
<M> initialisiert etwaige Modifikationen und zeigt einen Cursor an (invertiertes Zeichen bzw. Graphikblock, s. o.).
Jedes eingegebene Zeichen ersetzt das Zeichen unter dem Cursor.
<CLEAR> löscht den Textpuffer von dem dem Cursor vorausgehenden Komma an (Trennzeichen der Hex-ASCII-Adressen). Der Cursor muß dabei nicht genau auf dem Komma stehen.

<BREAK> kehrt zur Anzeige des aktuellen Sektors zurück.

Verlassen des Programms:
<BREAK> kehrt jederzeit in die Befehlsebene vor dem Programmstart zurück (DOS oder BASIC).
<X> schreibt den ASCII-Text auf Diskette und kehrt in die vorige Befehlsebene zurück. Dabei wird der Screening-Text mit einem Punkt abgeschlossen, der Rest des aktuellen 256-Byte-Blocks wird ausgenullt.
Vor dem Saven des Textes wird der Benutzer nach dem Namen des Screening-Files gefragt. Alle üblichen Angaben sind möglich. Wenn die Extension "TXT" lauten soll, wird sie automatisch angehängt und kann bei der Eingabe weggelassen werden.



Kleine Hexen ohne Aufwand

Anscheinend sehe ich gelegentlich den Wald vor lauter Bäumen nicht. In meinem Artikel "Kleine Hexen bevorzugt" rede ich noch davon, daß DSMBLR/CMD die Hexziffern auf dieselbe düstere Weise erzeugt wie das DOS - aber ich komme nicht auf die Idee, das DOS einfach zu CALLen. Wer keine Arbeit hat, beschafft sich eben welche. Erinnern wir uns: Es ist lästig, bei der Arbeit mit ZEUS für Labels, die von DSMBLR erzeugt wurden, immer die Shift-Taste zu drücken, weil die Hexziffern A-F als Großbuchstaben erscheinen.

Nun ist das damals vorgestellte Konversionsprogramm aber überflüssig geworden. Wenn man die Hex-Routine des DOS nämlich CALLt, erübrigen sich einige Bytes aus DSMBLR. Wie sich zeigte, sind es exakt so viele, wie erforderlich sind, um bei den Hexziffern A-F Kleinbuchstaben erzeugen zu lassen. Der dazu nötige kleine Zusatz paßt präzise in die frei werdende Lücke, so daß ein Extrafile nicht mehr nötig ist. Ein Detail davon bedarf der Erklärung: Das DOS wähnt HL als Zeiger auf den Hex-Puffer, DSMBLR unterhält jedoch den Puffer dort, wo DE hinzeigt. Daher die beiden Befehle EX DE,HL.

Listing 1 zeigt die Originalroutine in DSMBLR. Bei den Kommentaren habe ich boshafterweise einfach bei H. Grosser in seinem "DOS-Buch" abgeschrieben. Mir scheint nämlich, daß auch er den nahezu kabbalistischen Algorithmus nicht durchschaut hat, sonst hätte er sich ausführlicher gefaßt. In Listing 2 erscheint die Modifikation, um Kleinbuchstaben für die Hexziffern A-F zu erhalten. Darunter stehen schließlich noch die ersten drei Zeilen eines Hex-Dumps mit SUPERZAP aus dem gepatchten Sektor 7 von DSMBLR. Die Änderungen sind unterstrichen.

```
                00001 ;Listing 1:     Originalversion in DSMBLR/CMD
                00002
5AFA            00003         ORG     5afah   ;Kommentare bei Grosser ("DOS-Buch", ab 4071h):
5AFA  E60F      00004         AND     0fh     ;Bits 7...4 löschen
5AFC  C690      00005         ADD     A,90h   ;in
5AFE  27        00006         DAA             ;ASCII-
5AFF  CE40      00007         ADC     A,40h   ;Code
5B01  27        00008         DAA             ;umformen
5B02  12        00009         LD      (DE),A  ;und ausgeben
5B03  13        00010         INC     DE      ;Zeiger erhöhen
5B04  C9        00011         RET


                00001 ;Listing 2:     geänderte Version für kleine Hexziffern
                00002
5AFA            00003         ORG     5afah
                00004 ;ab hier verändert:
5AFA  EB        00005         EX      DE,HL   ;tauschen, weil die DOS-Routine nach (HL) lädt
5AFB  CD7140    00006         CALL    4071h   ;das DOS tut dasselbe wie DSMBLR im Original
5AFE  EB        00007         EX      DE,HL   ;Register (Zeiger) zurücktauschen
5AFF  1B        00008         DEC     DE      ;Zeiger auf die Hexziffer zurückstellen
5B00  F620      00009         OR      20h     ;Klein- aus Großbuchst. machen (0-9 unveränd.)
                00010 ;ab hier der unveränderte Code:
5B02  12        00011         LD      (DE),A
5B03  13        00012         INC     DE
5B04  C9        00013         RET
```

```
FPS 00  3E6D 1213 1803 CD05 5BCD F15A 2B7E 0100   >m.....Ä..Z+ß..
7   10  F25A 0F0F 0F0F CDFA 5A7E EBCD 7140 EB1B   .Z......Zß..q@+.
7H  20  F620 1213 C97E FEA0 D83E 3012 13C9 CDF7   . ...ß...>0.....
```


Arnulf Sopp

Die Pratze läßt das Mausen nicht

Es mochte zunächst nach Spielkram gerochen haben, als Hartmut Obermaus die Murmeltastatur für Analphabeten (man nennt sie Maus) für die TRS-80-Kompatiblen nutzbar machte. Heute fiel mir jedoch eine weitere Anwendung für TSCRIPS ein, die das Teil richtig sinnvoll erscheinen läßt.

Beim Trennen von Texten hat man die Möglichkeit, mit "-" einen Trennungsstrich einzufügen, mit ENTER das Wort unter dem Cursor ungetrennt zu lassen, mit den waagerechten Pfeiltasten den Cursor auf eine trennbare Stelle zu setzen oder mit CLEAR den Vorgang abzubrechen. Andere Tasten werden nicht beachtet. Es fällt auf, daß bis auf den Minus- oder Trennstrich alle Zeichen in 3840h memory mapped sind. Das ist die Tastaturzeile, die die Maus bedient.

Wenn man nun außer dem Minuszeichen noch BREAK und/oder SPACE als Trennanweisung zuläßt, ist alles mit den Tasten in 3840h zu erledigen, also eine ehrenvolle Aufgabe für die Maus (oder auch den Joystick, der die gleichen Vorteile hat). Nun braucht man nicht mehr mit den Augen vom Bildschirm zur Tastatur zu wandern, aus Angst, blind könnte man sich vertippen.

Auch ohne Maus bietet diese Änderung in TSCRIPS einen Vorteil. Auf der Tastatur des alten Genie 1 liegt die BREAK-Taste direkt neben dem Minuszeichen. Wenn man es nun doch blind versucht, darf man getrost danebenhauen; man trifft dennoch eine Taste, mit der das Wort wunschgemäß getrennt wird.

Das klingt alles recht nett. Da der Joystick und die Maus von Haus aus (es lebe der Endreim!) aber nur eine Auslösetaste haben, muß auf jeden Fall gebastelt werden. Wie das geht, ohne das Teil kleinzukriegen, wurde in meinem letzten Beitrag zu diesem Thema beschrieben.

Zu der Modifikation in TSCRIPS: An der Adresse 6517h steht ein CALL-Befehl, mit dem die Tastatureingabe abgeholt wird. In dem aufgerufenen Unterprogramm wird der Tastencode gleichzeitig gegf. umgewandelt und an der Stelle (IY+1) abgelegt. Z. Zt. zeigt IY auf 7C2Fh, also lautet der Puffer für den Tastencode 7C30h.

Die CALL-Adresse kann problemlos auf eine Umleitung umgelenkt werden, wo ein Check auf BREAK und SPACE stattfindet (Label devia im Listing). Die Umleitung mit ihren 16 Bytes paßt locker in einen überflüssig gewordenen Text, der mit Cassettenoperationen zu tun hat (Adresse 7925h).

Dort wird nun zunächst auf 1Dh geprüft. Das ist der Code, den TSCRIPS aus der BREAK-Taste erzeugt. War es BREAK, dann geht es beim Label hyphan weiter: Die Minustaste wird simuliert. Dasselbe passiert, wenn Blank gedrückt wurde. Wenn es eine andere Taste war, geht es unverrichteter Dinge zurück.

Vor dem Listing des kleinen Programms möchte ich noch vorstellen, wie man an die hierzu notwendigen Informationen kommen kann, ohne eine kommentierte Source von TSCRIPS (oder irgendeines anderen Programms) zu besitzen. Der Leser wird es möglicherweise brauchen, denn es sind viele Versionen von TSCRIPS im Umlauf (meine heißt 5.4).

Wenn das Programm im Speicher steht, sucht man mit Debug nach einem signifikanten Code. In diesem Falle lautete er CP '-' (FE-2D, vergleiche den Akku-Inhalt mit dem Trennstrich). Es ist durchaus möglich, daß der Code mehrfach in einem Programm vorkommt. Deshalb ist es ratsam, alle diese Stellen aufzusuchen und in ihrer Umgebung nachzuforschen, ob der Rest für die gesuchte Routine plausible Befehle enthält.

Wenn sich der Verdacht hinreichend erhärtet hat, daß man fündig wurde, wird sicherheitshalber weitergetestet: An einer hundsgemeinen Stelle, z. B.

beim Trennstrich-Vergleich, wird F7h gepatcht. Das ist der Befehl RST 30h (Aufruf von Debug). Wenn man nun das Zielprogramm abfährt, darf sich in unserem Falle nur dann Debug melden, wenn beim Trennen eine Taste gedrückt wurde. In der Anzeige des Monitors können wir dann ablesen, was gerade im Akku steht. So läßt sich auch herausfinden, was für einen Code TSCRIPS aus BREAK macht.

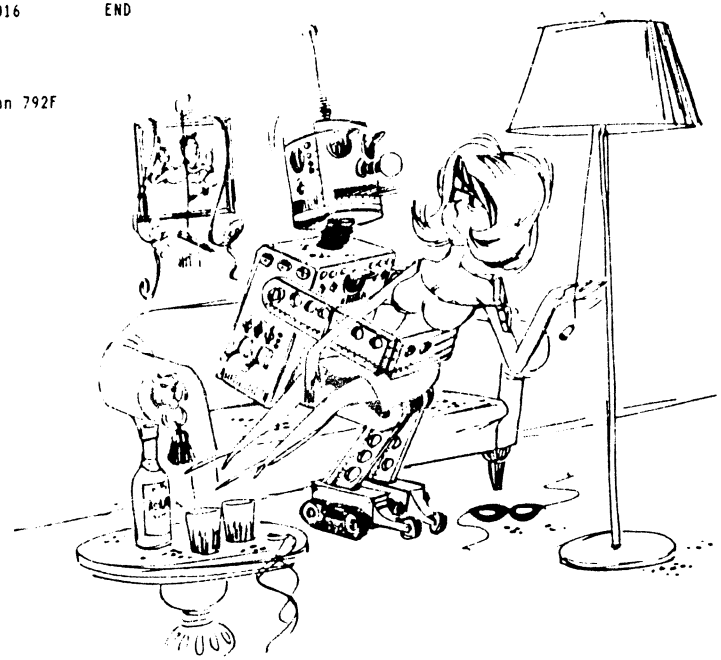
Der letzte Akt sind dann nur noch der Entwurf einer Modifikation und die Überlegung, an welcher Stelle im Programm man das Kuckucksei legen will.

Arnulf Sopp

```
                00001 ;       TSCRIPS-Patch zur Silbentrennung mit einer umgebauten Maus
                00002
6518            00003         ORG     6518h           ;dort CALL-Adresse 8446h (Tastatur lesen)
6518  2579      00004         DW      devia           ;jetzt CALL auf die Umleitung
                00005
7925            00006         ORG     7925h           ;ehemaliger Cassetten-Text (Überflüssig)
7925  CD4684    00007 devia   CALL    8446h           ;Überschriebenen Befehl nachholen: Taste
7928  FE1D      00008         CP      1dh             ;wurde BREAK gedrückt?
792A  2803      00009         JR      Z,hyphan        ;Trennung erzeugen, falls ja
792C  FE20      00010         CP      ' '             ;Blank gedrückt?
792E  C0        00011         RET     NZ              ;zurück, falls andere Taste
792F  3E2D      00012 hyphan  LD      A,'-'           ;      in Trennungsstrich umwandeln
7931  FD7701    00013         LD      (IY+1),A        ;in den Puffer
7934  C9        00014         RET                     ;erledigt
                00015 .
0000            00016         END

00000 Fehler

devia  7925   hyphan 792F
```



»ICH HABE IHNEN DOCH GLEICH GESAGT, DASS ICH ECHT UND NICHT MASKIERT BIN!«

Nochmals: Vergleich zweier Disk-Dateien

In der 22. Ausgabe unseres Kirchenblättchens habe ich mich über ein Programm verbreitet, das zwei Files auf Disketten miteinander vergleicht und die Unterschiede ausgibt. Im harten Alltag hat sich dieses Programm als zuwenig komfortabel und nicht aussagekräftig genug erwiesen. Ich möchte euch nun nicht damit langweilen, wie die Änderungen gegenüber der alten Version arbeiten. Wer sich mit Assembler auskennt, ersieht das eh' aus den Kommentaren. Wer nicht, blättert sowieso weiter. Nur die Vorteile für die Praxis seien aufgezählt:

In der Anzeige erscheinen nun nicht mehr nur die Angaben über die dateirelative Position einer Abweichung sowie die Abweichung selbst (plus das nächste Byte) im zuerst genannten File. Jetzt werden die beiden Bytes des Unterschieds aus beiden Dateien angezeigt. Zusätzlich erscheint deren ASCII-Darstellung, denn es könnte sich um Text handeln. Dabei werden solche Bytes durch einen Punkt ersetzt, die nicht anzeigbar/ausdruckbar sind. Es sind nach unten die Steuerzeichen unterhalb ASCII 20h (Blank). Die Obergrenze ergibt sich aus der SYSTEM-Option AX=n. Dabei bezeichnet n den höchsten für den Drucker noch darstellbaren Wert.

Spaßeshalber habe ich in der Source die beiden ASCII-Puffer (s. Listing, Labels ascbuf1 und ascbuf2) anstatt mit 'AB' mit '@@' ausgefüllt und diese Version in die Datei COMP/CMD assembliert. Beim Aufruf des Programms mit der Syntax COMPARE,COMPARE/CMD,COMP/CMD kam folgende Anzeige:

```
rel. Skt. / rel. Byte : Abweich. ==> 0001/35: 4142 AB - 4040 @@
rel. Skt. / rel. Byte : Abweich. ==> 0001/3F: 4142 AB - 4040 @@
```

Ein weiterer Komfort wurde eingebaut, denn die Anzeige erfolgt bei diesem Maschinenspracheprogramm natürlich rasend schnell. Solange die SPACE-Taste niedergehalten wird, passiert weiter nichts. So kann man zwischendurch stoppen und sich schon einen ersten Reim auf die Anzeige machen. Das ist aber nur von Bedeutung, wenn wegen zahlreicher Abweichungen der Bildschirm das Scrollen anfängt. Im obigen Beispiel wäre dieses Feature unnötig gewesen. Es erschienen nur diese beiden Zeilen.

Und schließlich kann mittendrin mit der BREAK-Taste abgebrochen werden. Das ist z. B. dann sinnvoll, wenn sich ins Vergleichsprogramm nur ein einziges überzähliges Byte eingeschlichen hat. Dann verschiebt sich der ganze Rest nach hinten. Deshalb gibt es von dort an nur noch Abweichungen, die aber völlig uninteressant wären.

Auch die Option einer Druckerausgabe ist nun anders gelöst. Der Bildschirm wird immer beliefert, auch wenn *zusätzlich* die Anzeige auf dem Drucker gewünscht wurde. ML-Leckermäuler mögen sich an dem Trick delektieren, mit dem das passiert.

Arnulf Sopp

```
                00001 ;       Utility zum Vergleich zweier Disk-Dateien
                00002
5200            00003         ORG     5200h
                00004
                00005 ;Abfrage, ob die Druckerausgabe erwünscht ist, gegf. Anpassung
5200 E5         00006 start   PUSH    HL              ;Befehlszeiger retten
5201 21E352     00007         LD      HL,dvcrequ      ;Frage nach Ausgabekanal
5204 CD6744     00008         CALL    4467h           ;anzeigen
5207 CD4900     00009         CALL    0049h           ;Tastatur abfragen
520A CDB545     00010         CALL    45b5h           ;evtl. Klein- zu Großbuchstaben machen
520D FE4A       00011         CP      'J'             ;'J' für 'ja' eingegeben?
520F 2005       00012         JR      NZ,open         ;falls anderes Zeichen ( = 'nein')
5211 3ECD       00013         LD      A,0cdh          ;CALL-Opc.: LD HL,446A wird zu  CALL 446A
5213 32AB52     00014         LD      (device),A      ;den CALL-Befehl patchen
                00015
                00016 ;für beide zu vergleichenden Programme einen FCB einrichten
5216 E1         00017 open    POP     HL              ;Befehlszeiger
5217 CDD54C     00018         CALL    4cd5h           ;Trennzeichen prüfen
521A 113A53     00019         LD      DE,fcb1         ;FCB für das erste der beiden Programme
521D CD1C44     00020         CALL    441ch           ;1. Dateinamen in den FCB übertragen
5220 3E2F       00021 error1  LD      A,2fh           ;Fehlercode 'schlechte Parameter'
5222 C20944     00022 error2  JP      NZ,4409h        ;raus mit Fehl., falls z. B. ungült. Name
5225 CDD54C     00023         CALL    4cd5h           ;nächstes Trennzeichen prüfen
5228 115A53     00024         LD      DE,fcb2         ;FCB für das zweite Programm
522B CD1C44     00025         CALL    441ch           ;2. Dateinamen in den FCB übertragen
522E 20F0       00026         JR      NZ,error1       ;falls ein Fehler aufgetreten ist (s. o.)
5230 210055     00027         LD      HL,buffer2      ;Sektorpuffer für das 2. Programm
5233 05         00028         LD      B,L             ;B = 00, LRL = 256
5234 CD2444     00029         CALL    4424h           ;den 2. FCB eröffnen
5237 20E9       00030         JR      NZ,error2       ;falls ein Fehler auftrat
5239 113A53     00031         LD      DE,fcb1         ;FCB für das 2. Programm
523C 210054     00032         LD      HL,buffer1      ;dessen Sektorpuffer
523F CD2444     00033         CALL    4424h           ;den 1. FCB eröffnen
5242 20DE       00034         JR      NZ,error2       ;falls ein Fehler auftrat
5244 48         00035         LD      C,B             ;BC (- 0000, Sektorzähler ab Skt. 0000
                00036
                00037 ;einen Sektor des 1. Prg. laden - Aussprung, falls Prg. zuende o. Fehler
5245 CD3644     00038 loadlop CALL    4436h           ;einen Sektor des 1. Programms laden
5248 F5         00039         PUSH    AF              ;Fehlercode retten
5249 FE1C       00040         CP      1ch             ;'Ende der Datei angetroffen'?
524B CA2D40     00041 dosexit JP      Z,402dh         ;fertig, falls ja
524E FE1D       00042         CP      1dh             ;'hinter Ende der Datei'?
5250 28F9       00043         JR      Z,dosexit       ;dto.
5252 F1         00044         POP     AF              ;anderer Fehlercode, evtl. NZ-Flag
5253 20CD       00045         JR      NZ,error2       ;raus, falls ein anderer Fehler auftrat
                00046
                00047 ;einen Sektor des 2. Programms laden
5255 115A53     00048         LD      DE,fcb2         ;FCB für das 2. Programm
5258 210055     00049         LD      HL,buffer2      ;dessen Sektorpuffer
525B CD3644     00050         CALL    4436h           ;einen Sektor des 2. Programms laden
525E 20C2       00051         JR      NZ,error2       ;falls ein Fehler auftrat
                00052
                00053 ;ein Byte beider Sektoren vergleichen - weiter, falls gleich
5260 110054     00054         LD      DE,buffer1      ;Sektorpuffer des 1. Programms
5263 1A         00055 complop LD      A,(DE)          ;1 Byte daraus laden
5264 BE         00056         CP      (HL)            ;ident. mit dem entspr. Byte des 2. Prg.?
5265 2855       00057         JR      Z,match         ;falls ja
                00058
                00059 ;ungleich - Abweichungen in den Puffer schreiben, Puffer anzeigen
5267 E5         00060         PUSH    HL              ;nein, Sektorzeiger des 2. Prg. retten
5268 211F53     00061         LD      HL,numbuf1      ;Puffer für ASCII-numerische Angaben
526B D5         00062         PUSH    DE              ;Sektorzeiger des 1. Programms retten
526C 50         00063         LD      D,B             ;DE (- lfd. Nr. des aktuellen Sektors
526D 59         00064         LD      E,C
```



```
526E CD6340     00065         CALL    4063h           ;in Hex-ASCII in den Puffer schreiben
5271 23         00066         INC     HL              ;Pufferstelle dahinter
5272 D1         00067         POP     DE              ;DE (- Sektorzeiger des 1. Programms
5273 7B         00068         LD      A,E             ;dessen LSB als sektorrelatives Byte
5274 CD6840     00069         CALL    4068h           ;in Hex-ASCII in den Puffer schreiben
5277 23         00070         INC     HL              ;': ' im Zahlenpuffer überspringen
5278 23         00071         INC     HL              ;(2 Stellen)
5279 C5         00072         PUSH    BC              ;Sektorzähler retten
527A 012020     00073         LD      BC,'  '         ;2 Blanks
527D ED432A53   00074         LD      (secbyt1),BC    ;Platz des 2. ungleichen Bytes löschen
5281 ED433453   00075         LD      (secbyt2),BC    ;auch im Zahlenpuffer des 2. Programms
5285 012D53     00076         LD      BC,ascbuf1      ;Pufferstelle für die ASCII-Entsprechung
5288 CDCA52     00077         CALL    filbuff         ;abweichendes Byte in den Puffer schr.
528B C4CA52     00078         CALL    NZ,filbuff      ;auch nächst. B., falls noch nicht 256 B.
528E C1         00079         POP     BC              ;vorübergehend Sektorzähler restaurieren
528F E1         00080         POP     HL              ;Sektorzeiger des 2. Programms
5290 E5         00081         PUSH    HL              ;wieder retten
5291 D5         00082         PUSH    DE              ;den des 1. Programms retten
5292 EB         00083         EX      DE,HL           ;DE (- Sektorzeiger des 2. Programms
5293 213253     00084         LD      HL,numbuf2      ;Puffer für die Daten des 2. Programms
5296 C5         00085         PUSH    BC              ;Sektorzähler retten
5297 013753     00086         LD      BC,ascbuf2      ;ASCII-Puffer für das 2. Programm
529A CDCA52     00087         CALL    filbuff         ;Abweich. des 2. Programms in den Puffer
529D C4CA52     00088         CALL    NZ,filbuff      ;falls der Sektor noch nicht zuende ist
52A0 3E0D       00089         LD      A,0dh           ;CR
52A2 02         00090         LD      (BC),A          ;dort als EOS einschreiben (Textende)
52A3 C1         00091         POP     BC              ;BC (- Sektorzähler
52A4 D1         00092         POP     DE              ;Pufferzeiger des 1. Programms
52A5 21FA52     00093         LD      HL,texbuff      ;Puffer des gesamten ASCII-Textes
52A8 CD6744     00094         CALL    4467h           ;Pufferinhalt anzeigen
52AB 216A44     00095 device  LD      HL,446ah        ;Dummy-Befehl, falls keine Druckerausg.
                00096                                 ;sonst steht hier CALL 446A (drucken)
                00097
                00098 ;Warteschleife, falls SPACE gedrückt, Abbruch, falls BREAK gedrückt
52AE 214038     00099         LD      HL,3840h        ;Tastatur: Steuerzeichen
52B1 7E         00100 wait    LD      A,(HL)          ;Tastaturstatus dieser Zeile laden
52B2 07         00101         RLCA                    ;ist Bit 7 gesetzt? (SPACE)
52B3 38FC       00102         JR      C,wait          ;anhalten, falls ja
52B5 FE08       00103         CP      08h             ;ist BREAK gedrückt?
52B7 2892       00104         JR      Z,dosexit       ;fertig, falls ja
                00105
                00106 ;beide Sektorzeiger auf die Fortsetzung vorbereiten
52B9 E1         00107         POP     HL              ;HL (- Sektorzeiger des 2. Programms
52BA 1D         00108         DEC     E               ;weil später INC E folgt (Ausgleich)
52BB 6B         00109         LD      L,E             ;beide Sektorzeiger auf dieselbe Stelle
                00110
                00111 ;keine Abweichung gewesen oder Abw. angezeigt - Skt.-zeiger weiterstellen
52BC 1C         00112 match   INC     E               ;eine Stelle weiter im Sektor des 1. Prg.
52BD 2C         00113         INC     L               ;dto. 2. Programm
                00114
                00115 ;Sektoren weiter vergleichen oder nächsten Sektor vorbereiten
52BE 20A3       00116         JR      NZ,complop      ;falls der Sektor noch nicht zuende ist
52C0 113A53     00117         LD      DE,fcb1         ;sonst DE (- FCB des 1. Programms
52C3 210054     00118         LD      HL,buffer1      ;HL (- Sektorpuffer des 1. Programms
52C6 03         00119         INC     BC              ;Sektorzähler erhöhen
52C7 C34552     00120         JP      loadlop         ;dasselbe Spielchen mit dem nächsten Skt.
                00121
                00122 ;abweichende Bytes eines Sektors und/oder Punkt in den Puffer schreiben
52CA 1A         00123 filbuff LD      A,(DE)          ;aktuelles Byte eines Sektors des 1. Prg.
52CB FE20       00124         CP      ' '             ;Blank oder höher (ausdruckbar)
52CD 380A       00125         JR      C,replpnt       ;falls kleiner: durch Punkt ersetzen
52CF E5         00126         PUSH    HL              ;retten
52D0 217043     00127         LD      HL,4370h        ;dort höchster Druckercode abgelegt
52D3 34         00128         INC     (HL)            ;auf 1. nicht mehr ausdruckbaren Code
```

```
52D4  BE              CP      (HL)            ;ist der Code
52D5  35        00130 DEC     (HL)            ;Code restaurieren
52D6  E1        00131 POP     HL              ;dto. Zeiger
52D7  3802      00132 JR      C,wrasc         ;unverändert ausgeben, falls kleiner
52D9  3E2E      00133 replpnt LD      A,'.'     ;nicht ausdruckbares Z. durch Punkt ers.
52DB  02        00134 wrasc   LD      (BC),A    ;Zeichen oder Punkt in den ASCII-Puffer
52DC  03        00135         INC     BC        ;nächste Pufferstelle
52DD  1A        00136         LD      A,(DE)    ;Byte des Sektorpuffers
52DE  CD6840    00137         CALL    4068h     ;in Hex-ASCII in den Puffer schreiben
52E1  1C        00138         INC     E         ;nächste Stelle im Sektorpuffer
52E2  C9        00139         RET               ;zurück
                00140
52E3  44        00141 dvcrequ DM      'Druckerausgabe? (J/N)',0eh,0dh ;Abfrage, Cursor an, CR
52FA  72        00142 texbuff DM      'rel. Skt. / rel. Byte : Abweich. ==) ' ;Text
531F  78        00143 numbuf1 DM      'xxxx/yy: zz'   ;Puffer für Angaben des 1. Bytes der Abw.
532A  7A        00144 secbyt1 DM      'zz '           ;dto. des 2. Bytes der Abweichung
532D  41        00145 ascbuf1 DM      'AB - '         ;dto. der ASCII-Entsprechung beider Bytes
5332  7A        00146 numbuf2 DM      'zz'            ;dasselbe für die Abweichungen des
5334  7A        00147 secbyt2 DM      'zz '           ;Vergleichsprogramms
5337  41        00148 ascbuf2 DM      'AB',0dh
                00149
533A            00150 fcb1    EQU     $               ;32 Bytes Platz für den FCB des 1. Prg.
535A            00151 fcb2    EQU     $+20h           ;dto. FCB des 2. Programms
                00152
5400            00153 buffer1 EQU     $+0140h&0ff00h  ;Sektorpuffer für das 1. Programm
5500            00154 buffer2 EQU     $+0240h&0ff00h  ;dto. für das 2. Programm
                00155
5200            00156         END     start           ;dort Einsprung

00000 Fehler

ascbuf1 532D    ascbuf2 5337    buffer1 5400    buffer2 5500    complop 5263    device  52AB
dosexit 524B    dvcrequ 52E3    error1  5220    error2  5222    fcb1    533A    fcb2    535A
filbuff 52CA    loadlop 5245    match   52BC    numbuf1 531F    numbuf2 5332    open    5216
replpnt 52D9    secbyt1 532A    secbyt2 5334    start   5200    texbuff 52FA    wait    52B1
wrasc   52DB
```

Bedingt assemblieren mit ZEUS

CP/M ist das kommende Betriebssystem für die Maschinen in unserem Club, denn die NewDOS/G-DOS-Rechner sterben irgendwann aus. Solange es aber noch ein paar letzte Mohikaner (sagte da gerade jemand "Ignoranten"?) gibt, die sich dem G-/NewDOS verschrieben haben, ist ZEUS wohl der Assembler der Wahl.

Er hat ein paar supergute Features, die mir gerade jetzt bei der Erstellung von SCREENER (evtl. auch in diesem Heft vorgestellt) sahnemäßige Dienste erwiesen haben. Eins davon steht in den nachfolgenden Listings in Zeile 857. Dort wird die ORG-Adresse dynamisch gehandhabt. Das Programm kann im Lauf der Zeit beliebig weiterwachsen. Immer wird sich der Sektorpuffer (Label buffcmd) am Anfang der nächsten freien Speicher-Page hinter dem Programm befinden, ohne daß der Programmierer dafür in irgendeiner Form noch Sorge zu tragen hätte.

Hier soll von einem anderen Vorteil die Rede sein. SCREENER soll möglichst auf allen im Club vertretenen Rechnern lauffähig sein. Nicht alle Systeme sind gleich. Deshalb muß mit Bedingungen gearbeitet werden, die sich jeweils auf ein bestimmtes Modell beziehen. Solche bedingten Befehle usw. werden bei ZEUS mit IF,cond eingeleitet und mit ENIF beendet.

Die Bedingung (condition) hat die Form eines Labels. Es wird irgendwo am Beginn des Programms deklariert. Dabei ist für ZEUS von Belang ob der Wert des Labels =0 (Bedingung nicht erfüllt) oder >0 (erfüllt) ist. In unserem Beispiel heißen die Labels m1, m3 und g3 (Zeilen 14-16). Sie erhalten gemäß den Kommentaren den Wert 0 oder 1, je nach Modell. Ein zusätzliches Label schützt den Programmierer vor Fehlern beim Umgang mit dieser Technik: Das Label model ist 0, falls kein Fehler gemacht wurde. Es ist >0, wenn man sich dabei vertat. Wir werden noch sehen, welchen Vorteil das hat.

Im Listing 1 sind Auszüge aus dem Programm wiedergegeben. Mehrere IF-Statements lassen alle Möglichkeiten zur Anpassung des Programms an den eigenen Computer offen. Der Programmierer kann das alles jedoch einfach vergessen, wenn er nur das Label m1 oder m3 oder g3 auf 1 und den Rest auf 0 gesetzt hat. Was dabei jeweils passiert, zeigen die anderen Listings. Dabei ist die Version für das Model 3/4/4p nicht wiedergegeben, weil sie sich in diesem Auszug nicht von der des Model 1 unterscheidet.

Aus Listing 2 und 3 geht hervor, was letztendlich von ZEUS als Bestandteil des CMD-Programms auf die Diskette geschrieben wird, je nach dem, für welches Modell das zutreffende Label zu 1 definiert wurde. Gegenüber dem Listing 1 fehlen einige Teile, die für das eigene Rechnermodell überflüssig sind. Es muß mit dieser Technik der bedingten Assembly also nicht unnötiger Ballast im Programm stehen, der von Fall zu Fall checkt, in welchem Computer es sich gerade befindet. Das hat aber auch einen Nachteil: Das so assemblierte Programm ist nicht mehr auf alle Modelle übertragbar. Es muß jeweils neu assembliert werden.

Schließlich zeigt Listing 4, daß mit dieser Methode auch ein wirksamer Schutz gegen Vertippen realisiert werden kann. Das Programm besteht bei einem Fehler bei der Zuordnung der Modell-Labels überhaupt nur noch aus einer Fehlermeldung (die natürlich auf allen Modellen lauffähig sein muß). Dabei half ein Trick, der wegen LIST OFF nicht in den Listings erscheint: Hinter dem Statement ORG 5200h (nicht gelistet) folgt zunächst das Bedingungs-Statement IF NOT,model. Das ganze Programm kommt also nur zustande, wenn model =0 ist (kein Fehler). Am Ende des Arbeitsprogramms, in den Listings also vor der Zeile 856, steht das abschließende ENIF.

Gewisse Labels sollen für alle Versionen gelten, auch dann, wenn die jeweilige Version mit einer Sequenz beginnt, die für andere unsinnig ist. Um die Allgemeingültigkeit zu erhalten, sollte daher dem Label eine eigene Zeile ohne Programmcode gegönnt werden. Das haben wir in diesem Beispiel bei den Labels buffcmd und start. Dabei spielt start noch eine Sonderrolle: Wenn die Bedingung model erfüllt ist (Fehler), gilt das Label dieses Namens aus der Zeile 863. Wenn nicht, dann dasjenige in Zeile 871. Ihm folgt sofort ein IF-Statement, so daß die eigene Zeile notwendig ist. Andernfalls gälte es nur für das Modell, das in IF definiert ist (hier Genie 3/3s).

Arnulf Sopp

```
00011 ;       Listing 1: Source-Text, noch nicht assembliert (alle Modelle)
00012
00013 ;Labels zur Anpassung des Programms an den eigenen Computer:
00014 m1      EQU     0       ;Model 1, Genie 1/2/2s: 1       andere Rechner: 0
00015 m3      EQU     0       ;Model 3/4/4p:          1       andere Rechner: 0
00016 g3      EQU     1       ;Genie 3/3s:            1       andere Rechner: 0
00017
00018 ;Sicherung gegen falsche Label-Zuteilung (mehrfach 1 oder dreimal 0!:
00019 model   EQU     m1+m3+g3-1      ;das Resultat muß 0 ergeben
00020
00856 ;Initialisierungen im Sektorpuffer, wo sie keinen Platz wegnehmen
00857         ORG     $+0100h&0ff00h  ;in der nächsten freien Page
00858 buffcmd                        ;Sektorpuffer für das CMD-Programm
00859         IF      model           ;falls Modell-Label falsch gehandhabt
00860 ;Sicherung gegen falsche Handhabung der Labels m1, m3 und g3:
00861         DM      'Label für das Modell falsch zugeteilt!',0dh
00862         DM      'Source neu laden und assemblieren!',0dh
00863 start   LD      HL,buffcmd      ;Alarm-Text
00864         JP      texttv          ;anzeigen und Ende
00865         ENIF
00866         IF      NOT,model       ;falls alles in Ordnung ist
00867         IF      NOT,g3          ;(das G3 hat soeben ein CLS durchgeführt)
00868         DB      1ch             ;HOME CURSOR
00869         ENIF
00870         DM      'SCREENER 2.1 by The HACKTORY',1fh,0dh  ;Hello-Text
00871 start
00872         IF      g3              ;(nur beim Genie III/III s anwendbar)
00873 ;Einstellung des Videoformats
00874         LD      A,10h           ;Steuercode für 16*64 Zeichen
00875         CALL    outch           ;ausgeben
00876         LD      HL,zcmd         ;Lib-Befehl Z,R (inverse Zeichen ab 80h)
00877         CALL    doscal          ;Befehl ausführen
00878 ;Bei anderen TRS-80-Kompatiblen muß gegf. die Einstellung auf 16*64 Z.
00879 ;und inverse Zeichen ab ASCII 80h auf andere Weise gelöst werden, sofern
00880 ;sie nicht ohnehin überflüssig ist (TRS-80 (R) M1, Genie (R) I/II).
00881 ;Dieses Programm entstand auf einem Genie IIIs. FÜr das Genie III können
00882 ;noch Änderungen erforderlich sein.
00883         ENIF
00884
00885 ;Einsprung für alle Modelle gemeinsam:
00886         LD      HL,buffcmd      ;Hello-Text
00887         CALL    texttv          ;anzeigen
00888         JP      restart         ;Hauptschleife anspringen
00889         ENIF
00890
00895         END     start           ;dort Einsprung beim Programmaufruf
```



```
00011 ;       Listing 2: Source-Text, assembliert für Model 1, Genie 1/2
00012
00013 ;Labels zur Anpassung des Programms an den eigenen Computer:
0001  00014 m1      EQU     1       ;Model 1, Genie 1/2/2s: 1       andere Rechner: 0
0000  00015 m3      EQU     0       ;Model 3/4/4p:          1       andere Rechner: 0
0000  00016 g3      EQU     0       ;Genie 3/3s:            1       andere Rechner: 0
      00017
      00018 ;Sicherung gegen falsche Label-Zuteilung (mehrfach 1 oder dreimal 0):
0000  00019 model   EQU     m1+m3+g3-1      ;das Resultat muß 0 ergeben
      00020
      00856 ;Initialisierungen im Sektorpuffer, wo sie keinen Platz wegnehmen
5800  00857         ORG     $+0100h&0ff00h  ;in der nächsten freien Page
      00858 buffcmd                         ;Sektorpuffer für das CMD-Programm
5800 1C   00868         DB      1ch             ;HOME CURSOR
5801 53   00870         DM      'SCREENER 2.1 by The HACKTORY',1fh,0dh  ;Hello-Text
      00871 start
      00884
      00885 ;Einsprung für alle Modelle gemeinsam:
581F 210058 00886         LD      HL,buffcmd      ;Hello-Text
5822 CD6744 00887         CALL    texttv          ;anzeigen
5825 C30052 00888         JP      restart         ;Hauptschleife anspringen
      00890
581F      00895         END     start           ;dort Einsprung beim Programmaufruf

00000 Fehler
```

```
      00011 ;       Listing 3: Source-Text, assembliert für Genie 3/3s
      00012
      00013 ;Labels zur Anpassung des Programms an den eigenen Computer:
0000  00014 m1      EQU     0       ;Model 1, Genie 1/2/2s: 1       andere Rechner: 0
0000  00015 m3      EQU     0       ;Model 3/4/4p:          1       andere Rechner: 0
0001  00016 g3      EQU     1       ;Genie 3/3s:            1       andere Rechner: 0
      00017
      00018 ;Sicherung gegen falsche Label-Zuteilung (mehrfach 1 oder dreimal 0):
0000  00019 model   EQU     m1+m3+g3-1      ;das Resultat muß 0 ergeben
      00020
      00856 ;Initialisierungen im Sektorpuffer, wo sie keinen Platz wegnehmen
5800  00857         ORG     $+0100h&0ff00h  ;in der nächsten freien Page
      00858 buffcmd                         ;Sektorpuffer für das CMD-Programm
5800 53   00870         DM      'SCREENER 2.1 by The HACKTORY',1fh,0dh  ;Hello-Text
      00871 start
      00873 ;Einstellung des Videoformats
581E 3E10   00874         LD      A,10h           ;Steuercode für 16*64 Zeichen
5820 CD3300 00875         CALL    outch           ;ausgeben
5823 21D256 00876         LD      HL,zcmd         ;Lib-Befehl Z,R (inverse Zeichen ab 80h)
5826 CD1944 00877         CALL    doscal          ;Befehl ausführen
      00878 ;Bei anderen TRS-80-Kompatiblen muß gegf. die Einstellung auf 16*64 Z.
      00879 ;und inverse Zeichen ab ASCII 80h auf andere Weise gelöst werden, sofern
      00880 ;sie nicht ohnehin überflüssig ist (TRS-80 (R) M1, Genie (R) I/II).
      00881 ;Dieses Programm entstand auf einem Genie IIIs. FÜr das Genie III können
      00882 ;noch Änderungen erforderlich sein.
      00884
      00885 ;Einsprung für alle Modelle gemeinsam:
5829 210058 00886         LD      HL,buffcmd      ;Hello-Text
582C CD6744 00887         CALL    texttv          ;anzeigen
582F C30052 00888         JP      restart         ;Hauptschleife anspringen
      00890
581E      00895         END     start           ;dort Einsprung beim Programmaufruf

00000 Fehler
```

```
                00011 ;       Listing 4: Source-Text, assembl., alle Modell-Labels = 0 (Fehler)
                00012
                00013 ;Labels zur Anpassung des Programms an den eigenen Computer:
0000            00014 m1      EQU     0       ;Model 1, Genie 1/2/2s: 1       andere Rechner: 0
0000            00015 m3      EQU     0       ;Model 3/4/4p:          1       andere Rechner: 0
0000            00016 g3      EQU     0       ;Genie 3/3s:            1       andere Rechner: 0
                00017
                00018 ;Sicherung gegen falsche Label-Zuteilung (mehrfach 1 oder dreimal 0):
FFFF            00019 model   EQU     m1+m3+g3-1      ;das Resultat muß 0 ergeben
                00020
                00856 ;Initialisierungen im Sektorpuffer, wo sie keinen Platz wegnehmen
5300            00857         ORG     $+0100h&0ff00h  ;in der nächsten freien Page
                00858 buffcmd                         ;Sektorpuffer für das CMD-Programm
                00860 ;Sicherung gegen falsche Handhabung der Labels m1, m3 und g3:
5300  4C        00861         DM      'Label für das Modell falsch zugeteilt!',0ah
5327  53        00862         DM      'Source neu laden und assemblieren!',0dh
534A  210053    00863 start   LD      HL,buffcmd      ;Alarm-Text
534D  C36744    00864         JP      texttv          ;anzeigen und Ende
                00890
534A            00895         END     start           ;dort Einsprung beim Programmaufruf

00000 Fehler
```

Wo die Tomaten fliegen

Es war eine dieser typischen Parties, auf denen die Gäste sich gegenseitig mit den ewig gleichen Geschichten langweilen und man blitzschnell ein paar Gläser in sich hineinschüttet, um den Abend leichter zu überstehen.
Mit ein paar Drinks war sogar Barbara zu ertragen, die zum hundertsten Mal eine Anekdote von »Käpten Nemo« erzählte. Das war ihr »süüüßer Hund«, den sie in ihren Erzählungen immer als eine Kreuzung zwischen Dogge und Dobermann darstellte. In Wirklichkeit war das Biest ein giftiger, überfressener Scotch-Terrier, der seinen Bauch kaum noch über die Türschwelle wuchten konnte, aber in jeden Schuh biß, der ihm zu nahe kam.
Barbara redet im Durchschnitt ununterbrochen 30 Minuten, bis sie zum erstenmal wieder Luft holt. Da ihr danach meist nicht mehr einfällt, worüber sie eben noch gesprochen hat, hilft in solchen Momenten ihr treusorgender Ehemann Wolfgang, indem er sagt: »Erzähl doch mal die Geschichte, wie Käpten Nemo bei Richters auf den Perser gekotzt hat – also das war wirklich zum Totlachen.«
Doch diesmal hatte Wolfgang seiner Frau gar nicht zugehört. Er beugte sich zu Gastgeber Jürgen herüber und redete in einer Fremdsprache, die so klang: »Da sind mir doch glatt 50 Kilobyte ins ROM gerutscht. Und jetzt konnte ich sehen, wie ich die File wieder auf die Disc bekam.«
Jürgen machte Augen, als hätte er eben die heißeste Sex-Story seines Lebens gehört, und antwortete aufgeregt: »Warum hast du nicht versucht, sie mit dem Turbo-Pascal wieder übers Back-up zu retten.«
Ich war sprachlos. Hier unterhielten sich zwei völlig normale Leute, als seien sie Datenexperten des CIA.
Ausgerechnet Jürgen sprach über Computer. Dabei ist der eingeschworene Junggeselle technisch so unbegabt, daß er sich schon auf der Suche nach dem Werkzeugkasten den Daumen quetscht.
Doch jetzt war er wie weggetreten. Er schwärmte plötzlich von »seiner Kleinen« und deutete in Richtung Schlafzimmer. Dabei strahlte er, als warte dort Bo Derek unter der Decke.
Was war los mit ihm? Jürgen begegnete zwar mindestens einmal pro Woche der »Frau fürs Leben«, aber er hätte sie nie vor uns im Schlafzimmer versteckt. Im Gegenteil, er ließ sonst keine Gelegenheit aus, seine neuesten Eroberungen stolz im verheirateten Freundeskreis vorzuführen.
Ich wartete noch einige Minuten, nachdem die beiden in besagtem Zimmer verschwunden waren, und ging dann unauffällig hinterher. Ich war immerhin so höflich, vorher anzuklopfen, aber es tat sich nichts, und so öffnete ich einfach die Tür.
Der Anblick war wirklich erschütternd: Da saßen zwei erwachsene Männer auf dem Bett und starrten auf ein Fernsehgerät. In den Händen hielten sie kleine Plastikgriffe, auf denen sie wie wild rumdrückten. Dann jagten kleine Männchen über den Bildschirm und gingen vor irgendwelchen herabstürzenden Tomaten in Deckung. Dazu machte es ununterbrochen »Pfiuuuu« und »Donkdonkdonkdonk«.
»Das ist aber nicht die Sportschau«, war das einzige, was mir in meiner Fassungslosigkeit einfiel, und die beiden zuckten zusammen wie zwei Buben, die beim Marmeladenaschen erwischt wurden.
»Nein«, sagte Jürgen und starrte wieder auf den Schirm, auf dem die Tomaten inzwischen die Oberhand gewonnen hatten, »hier siehst du den neuen XP-75 in voller Aktion.« Er deutete auf eine telefonbuchgroße Tastatur, die vor ihm auf dem Nachttisch lag und mit dem Fernseher verbunden war.
Ich wußte zwar nicht, was der neue XP-75 ist, aber ich war

→ S. 44

Im letzten Beitrag zu diesem Thema ging es darum, mit IF-Anweisungen an ZEUS ein Programm sehr universell verwendbar zu machen. Ein und derselbe Sourcecode kann z. B. für alle möglichen Computer gültig sein, wenn nur ein Bedingungs-Label entsprechend angepaßt wird.

So leistungsfähig dieses Feature ist, so umständlich ist es aber, für jede denkbar auftretende Möglichkeit eigene Sequenzen zu programmieren, die zwischen IF und ENIF stehen. Diese Methode ist auch fehlerträchtig, wenn man nicht höllisch aufpaßt. Sollte einmal ein ENIF vergessen werden, wird der ganze Rest nicht assembliert, falls die mit IF formulierte Bedingung nicht zutrifft. Die Möglichkeit der Verschachtelung von IF-ENIF-Blocks (ganz ähnlich wie in BASIC) birgt zusätzliche Risiken bei unkonzentriertem Arbeiten.

Eine Alternative eröffnen die arithmetisch-logischen Fähigkeiten von ZEUS. Je nach dem Wert eines Bedingungs-Labels können Adressen, Werte usw. *errechnet* werden. In unserem Beispielprogramm (auf dessen Sinn und Zweck ich zum Schluß eingehen möchte) ist z. B. in Zeile 10 das Label drv (Laufwerksnummer) zu 0 definiert. Das bedeutet, daß sich die nachfolgenden Operationen auf das Laufwerk 0 beziehen. Der Block mit den PDRIVE-Parametern für Lw. 0 steht an 4371h. Jeder dieser Blocks ist 10 Bytes lang. Gem. der Label-Gleichung pdrive EQU 10*drv+4371h ergäbe sich daher für Lw. 1 (drv EQU 1) die Adresse 437Bh usw. Dasselbe gilt analog für alle Werte, die sich auf die drei zuerst definierten Bedingungs-Labels wr, ds und drv beziehen.

Die Syntax solcher Operanden ist genau festgelegt. Man möge sich dazu das ZEUS-Manual ansehen. Als Faustregel kann man sich merken, daß die Elemente wie bei einem sehr dummen Taschenrechner eingegeben werden müssen. Es gibt keine Klammern, der jeweils nächste Operator verwendet immer das bisherige Zwischenergebnis, das sofort errechnet wird. Z. B. die Definition des Labels spt in Z. 13: In der Form ds+1*5 kommt je nach Laufwerkstyp 5 oder 10 dabei heraus. Stünde dort aber 5*ds+1, so ergäben sich je nach dem ein oder sechs Sektoren pro Spur, beides völlig absurde Werte.

Verzichtet man nun konsequent auf IF-Zeilen, tritt ein Nachteil dieser Methode zutage. Was zum Teufel steht denn nun eigentlich in B, wenn dieses Register in Zeile 32 mit -wr*2+spt geladen wird? Oder wohin geht der darunter stehende CALL wr*10h+4630h? Diese Art der Programmierung kann sehr unübersichtlich sein, wie man sieht. Eine gesunde Mischung aus IF und Rechnerei wird wohl das Optimum sein.

Zum Nährwert des Programms: Bei allen Computern unserer Rechnerfamilie ist die erste Diskettenspur von einfacher Dichte. Die einzige Information, die sie enthält, ist der Code von BOOT/SYS (GDOS/SYS), davon aber nur die ersten beiden Sektoren. Der ganze Rest der Spur ist vollkommen leer und steht für jede Schandtat zur Verfügung. Bei einseitigen Laufwerken sind das immerhin noch drei Sektoren, bei doppelseitigen sogar noch acht, nach denen kein Hahn kräht. Da paßt allerhand hinein, das z. B. in einem SYS-File keinen Platz mehr fände.

Dieses Programm, das hier aber nur die Form eines allgemeingültigen Schemas hat, kann dort etwas einschreiben oder auch auslesen. Dabei sollten aber die ersten beiden Sektoren, die ja unbedingt zum Booten gebraucht werden, nicht verändert werden. Deshalb wird der Sektorzähler in Z. 32 zum Schreiben nur mit zwei Sektoren weniger beladen, die Sektoradresse erhält in Z. 30 beim Schreiben den Anfangswert 0002h.

Es wäre interessant, demnächst an dieser Stelle zu lesen, welche sinnvolle Verwendung sich jemand für diesen freien Diskettenplatz ausgedacht hat.

Arnulf Sopp

```
5200                    00001           ORG     5200h
                        00002
                        00003 ;Label wr muß für RD (lesen) bzw. WR (schreiben) definiert werden
0000                    00004 wr        EQU     0       ;Sektoren schreiben:    1       lesen:    0
                        00005
                        00006 ;Label ds muß dem eigenen Laufwerktyp angepaßt werden
0001                    00007 ds        EQU     1       ;Laufwerk doppelseitig: 1       einseitig: 0
                        00008
                        00009 ;Label drv muß den Wert des gewünschten Laufwerks haben
0000                    00010 drv       EQU     0       ;Laufw. 0: 0    Laufw. 1: 1     usw.
                        00011
                        00012 ;sonstige Labels (brauchen nicht angepaßt zu werden)
000A                    00013 spt       EQU     ds+1*5  ;doppelseitig: SPT=10           einseitig: SPT=5
0040                    00014 pds       EQU     ds*40h  ;doppelseitig: PD-Wert 40h      einseitig: 00h
4371                    00015 pdrive    EQU     10*drv+4371h    ;Adresse des PD-Parametersatzes für Lw. x
8000                    00016 buffer    EQU     8000h   ;Puffer für zu lesende/schreibende Sektoren
                        00017
5200 3E00               00018 start     LD      A,drv           ;gewünschte Laufwerksnummer
5202 CD5B44             00019           CALL    445bh           ;Laufwerk selektieren und Motor starten
5205 217543             00020           LD      HL,pdrive+4     ;dort SPT im PDRIVE-Block
5208 7E                 00021           LD      A,(HL)          ;Wert laden
5209 F5                 00022           PUSH    AF              ;retten
520A 360A               00023           LD      (HL),spt        ;SPT auf 5 bzw. 10 Skt. pro Spur setzen
520C E5                 00024           PUSH    HL              ;Adresse im PDRIVE-Block retten
520D 217843             00025           LD      HL,pdrive+7     ;dort TI und TD im PDRIVE-Block
5210 7E                 00026           LD      A,(HL)          ;s. o.
5211 F5                 00027           PUSH    AF
5212 3640               00028           LD      (HL),pds        ;5.25", SS bzw. DS, SD, Spur ab 0 zählen
5214 E5                 00029           PUSH    HL              ;s. o.
5215 110002             00030           LD      DE,wr*2         ;ab Skt. 0 (RD) bzw. 2 (WR) (diskrelativ)
5218 210080             00031           LD      HL,buffer       ;Pufferadresse
521B 060A               00032           LD      B,-wr*2+spt     ;Zähler der zu lesenden/schreibenden Skt.
521D CD3046             00033 loop      CALL    wr*10h+4630h    ;diskrel. Lese- bzw. Schreibrout. des DOS
5220 2004               00034           JR      NZ,exit         ;falls ein Fehler auftrat
5222 24                 00035           INC     H               ;sonst Pufferzeiger eine Page weiter
5223 1C                 00036           INC     E               ;DE auf den nächsten Sektor stellen
5224 10F7               00037           DJNZ    loop            ;bis 5 bzw. 10 Sektoren bearbeitet sind
5226 E1                 00038 exit      POP     HL              ;ursprüngliche Werte restaurieren
5227 C1                 00039           POP     BC              ;Flags würden sich bei POP AF ändern
5228 70                 00040           LD      (HL),B
5229 E1                 00041           POP     HL
522A C1                 00042           POP     BC
522B 70                 00043           LD      (HL),B
                        00044 ;vorläufiger Notausgang (im Ernstfall RET, JP 402D oder ähnlich)
522C F7                 00045           RST     30h             ;zum Begucken mit BEGUG
                        00046 ;bei NZ-Bedingung: aufgetretener Fehler in A (Debug im X-Modus fahren)
                        00047
5200                    00048           END     start

00000 Fehler

buffer 8000   drv   0000   ds    0001   exit  5226   loop  521D   pdrive 4371   pds   0040
spt    000A   start 5200   wr    0000
```


Der Teufel steckt oft im Detail!!

Dieses Sprichwort hat mich heute, an einem wirklich schönen und sonnigen Sonntag, der eigentlich zu allem andern taugt als dazu am Computer zu sitzen, mehr als drei Stunden wertvoller Freizeit gekostet. Um anderen solche Ungemach zu ersparen, hier kurz ein paar wertvolle Tips.

Wer wie ich mit mehr als einem Rechner arbeitet (TRS 80 Model 100, Model 4p und Toshiba T2100) kommt manchmal in die Verlegenheit, Daten (sprich Dateien) zwischen den einzelnen Systemen hin und her kopieren zu wollen/müssen. So ging es mir heute mit den Programmen zum Thema HRG-Bildübertragung (siehe an anderer Stelle in diesem Info), die ich unter MS-DOS gespeichert habe, die nun aber auf CP/M (für Andreas Rychlik) und auf NewDOS (für Werner Förster) kopiert werden sollten. Das hört sich schlimmer an, als es in Wirklichkeit ist.

Das Programm HyperCross, welches es sowohl unter NewDOS (für Model 1 und 3/4/4p) als auch unter TRSDOS 6.x (für TRS 80 Model 4/4p) gibt, leistet in einem solchen Fall gute Dienste und beherrscht zudem sowohl die MS-DOS-Formate als auch das von mir benutzte CP/M-Format (Montezuma 80/DS/DD). Ohne viel Zaudern wurde also das 4p angeworfen und zunächst alle Dateien von der MS-DOS-Disk auf eine CP/M-Diskette übertragen. Probleme? Keine!

Auch die Übertragung der Dateien von MS-DOS auf NewDOS hatte mir bisher, ich habe das ja nicht zum erstenmal gemacht, keine Probleme bereitet. Jetzt aber wollte HyperCross plötzlich nicht so, wie ich wollte und schon bei der ersten Datei, die ich übertragen wollte, erhielt ich einen ominösen TRSDOS Error 11. Hätte ich zu diesem Zeitpunkt in das TRSDOS-Handbuch geschaut, mir wäre wahrscheinlich einiges erspart geblieben. So aber versuchte ich eine andere Methode, die zunächst auch anstandslos funktionierte.

Nun benutzte ich DBLCROSS, ein Programm, welches nur auf dem TRS 80 Model 4/4p unter Montezuma-CP/M 2.2 lauffähig ist, und kopierte die Dateien von der CP/M-Diskette auf eine TRSDOS 1.3 Disk, die ich dann mit NewDOS zu lesen beabsichtigte. Und diesmal klappte auch alles, jedenfalls bis zu dem Zeitpunkt, wo ich mir die erste Datei zur Bearbeitung vornehmen wollte. NewDOS behauptete hartnäckig, die von mir angegebene Datei sei nicht vorhanden, führte sich aber selbst ad absurdum, denn bei DIR zeigte es die Datei an und zuvor hatte es ja auch bei "COPY,1,0,,nfmt,cbf" die Datei(en) von der TRSDOS 1.3-Disk ordnungsgemäß kopiert.

Die Sache hat mich wirlich sehr viel Nerven gekostet und so will ich euch nun nicht länger auf die Folter spannen und die Sache aufklären. Der Teufel, der ja (siehe Überschrift) im Detail steckt, verbarg sich in meinem Fall hinter einem simplen Bindestrich. Diesen hatte ich, zur Verbesserung der Lesbarkeit der Dateinamen unter MS-DOS eingeführt und unter CP/M damit natürlich keine Probleme. HyperCross warnte mich wohl mit einem TRSDOS-Error 11 schon, DBLCROSS kopierte aber den Bindestrich lustig in das NewDOS-Direktory und kümmerte sich nicht im mindesten darum, daß NewDOS damit überhaupt nichts anfangen kann.

Und die Moral von der Geschicht: traue Deinem Betriebssystem nicht (und erst recht nicht seinem Bediener)!!!

Hartmut Obermann

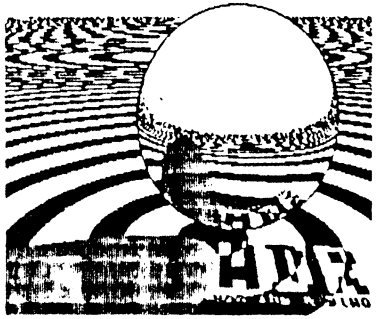
HRG - Bildübertragung

Vor mehr als zwei Jahren erschien im Clubinfo 14 auf den Seiten 24 bis 28 schon einmal ein Artikel mit dieser Überschrift. Im Grunde beschäftigte er sich auch mit genau dem gleichen Thema, nämlich der Übertragung von Bildern zwischen nichtkompatiblen Systemen bzw. über Modem/Akustikkoppler.

Das Thema haben Alexander Schmid und ich, während einer zwei Tage (die Nacht nicht zu vergessen) dauernden Session wieder aufgewärmt. Angefangen hat es damit, daß Alexander für sein Genie IIs einen Grafiktreiber unter CP/M geschrieben hat, der zu dem, den ich für das TRS 80 Model 4p verfasst habe, befehlskompatibel ist. Dadurch laufen Grafik-Programme, die ich unter CP/M in Turbo-Pascal geschrieben oder von PC's konvertiert habe, ohne weitere Änderungen auf dem IIs. Danach ging es darum Grafikdateien, die Alexander auf seinem IIs bzw. ich auf meinem 4p geschrieben haben, auf dem jeweils anderen Rechner zu laden und anzuzeigen. Auch dieses Problem wurde recht schnell gelöst. Ebenfalls ohne Probleme können wir inzwischen Grafikdateien der Atari ST-Serie (kopiert bei Gerald Schröder) lesen (und demnächst hoffentlich auch erzeugen. Darüber aber mehr in einem weiteren Artikel.

Zu guter Letzt probierten wir das von Alexander selbst gebaute Modem aus und verirrten uns dabei u.a. in die WDR-Mailbox, womit wir wieder beim Thema wären. In dieser Mailbox gibt es nämlich eine Rubrik "Bildübertragung", die nochmals unterschieden wir in Übertragung komprimierter und nicht komprimierter Bilder. Die nicht komprimierten Bilder behandelte der oben schon erwähnte Artikel im Info 14, hier soll auf die komprimierten Bilddateien eingegangen werden.

Wenn man in der WDR-Mailbox die Rubrik "Übertragung komprimierter Bilder" anwählt, erhält man folgende Auswahl. Unter (1) findet man einen sehr ausführlichen Erklärungstext, der im Anschluß zu meinen Ausführungen abgedruckt ist. Weiterhin gibt es eine recht große Anzahl von Bildern, von denen Alexander und ich, zur Freude des Herrn Postminister und zum Schrecken meiner(s) Frau (Finanzministers), sechs Stück in unseren Besitz brachten.



Hier das Menü der Rubrik "Übertragung komprimierter Bilder" aus der WDR-Mailbox (vom 12.08.1988)

```
 1  Erlaeuterungen
 2  Testbild
 3  Verschlusszeitschaltung
 4  Sesamstrasse
 5 *Space-Shuttle
 6 *WDR - Kugel
 7 *3 - D Funktion
 8  Eisenbahn-Interface 1
 9        "          2
10  Interface-Stueckliste
11  Joystick-Interface 1
12  Joystick-Interface 2
13  Corvette
14 *Weltkarte
15  Triangle
16  Dragons
17  Schach
18 *Papagei
19 *Snoopy
20  Morse-Interface (CC 10/86)
```


Hier nun zunächst der unter "1 Erläuterungen" abrufbare Text.

Bildübertragung

Den meisten wird das bisher verwendete Prinzip der Bildübertragung bereits bekannt sein :

Ziel ist es ein Grafikbild in Zeichen umzuwandeln, die ohne Informationsverlust das komplette Bild repräsentieren, und ohne teure Konvertierungssoftware anderen Rechnern zugänglich gemacht werden können oder z.B. über Telefon übertragen werden können (wofür man dann einen Akustikkoppler und ein DfÜ-Programm haben muß).

Ein Grafikbild besteht natürlich aus Punkten, die in Zeilen organisiert sind. Um in einem möglichst unproblematischen Bereich der Ascii-Tabelle (Zahlenmäßige Verschlüsselung der Bildschirmzeichen incl. Steuerzeichen) zu bleiben werden immer nur 6 Punkte in ein Zeichen zusammengefaßt. Das ist deshalb nötig, weil nur in den seltensten Fällen Zeichen mit einem Ascii-Code, der größer als 126 oder kleiner als 32 ist, unverändert übertragen werden. Man hat also pro Zeichen einen Bereich von chr$(32) bis chr$(126), also 95 Werte. Da man allerdings immer nur ganze Punkte repräsentieren kann, interessiert nur die nächst kleinere 2-er Potenz also 64 = 2 hoch 6 ==> macht 6 Punkte.
Man faßt also immer 6 horizontal nebeneinander liegende Punkte zu einer Zahl zusammen; und zwar so, daß die 6 Punkte von links nach rechts die Wertigkeiten 1,2,4,8,16 & 32 erhalten und die Wertigkeiten all derjenigen Punkte, die nicht schwarz, sind addiert werden.
Addiert man zu dieser Zahl 32 hinzu, so ist die kleinste mögliche Zahl (alle Punkte Schwarz) 32 und die größte mögliche (kein Punkt schwarz) 95. Damit bewegt man sich im Bereich von 32 bis 95; die entsprechenden Ascii-Zeichen sind also alle unproblematisch.
Reiht man nun alle Ascii-Zeichen für je 6 Punkte von links nach rechts aneinander, so erhält man eine Zeile, die genau den gleichen Informationsgehalt, wie die durch sie repräsentierte Punktreihe (Zeile von Punkten im Grafikbild), besitzt.
Alle Zeichenzeilen aus den Zeilen des Grafikbildes, von oben nach unten zusammengefaßt in einem Textdatei entsprechen dem Grafikbild bis auf den letzten Punkt, lassen sich im Gegensatz dazu allerdings ohne weiteres übertragen.
Wenn man noch ein paar zusätzliche Konventionen, zur Erleichterung der Verarbeitung solcher Dateien einhält, ist das Grundgerüst der Bildübertragung komplett :

- *Jede Zeichenzeile wird mit Crlf (chr$(13),chr$(10)) abgeschlossen.*
- *Am Ende der Datei sind zwei Zeilen, in denen keine Informationen enthalten sind, also auch keine Leerzeichen. (d.h. die Zeilen sind komplett leer)*
- *Bevor die eigentliche Information beginnt, müssen einige Informationen in Klarschrift eingetragen werden :*
  1. *Zeile : Hier muß das Wort BILD stehen. (Anfangsmarke)*
  2. *Zeile : Hier muß der Name des Rechners, auf dem das Bild erstellt wurde, stehen.*
  3. *Zeile : Hier muß die Anzahl der horizontalen Bildpunkte, die eine Zeile tatsächlich enthält (Breite des Bildes in Bildpunkten) in der Form 'Hnnn' stehen. (H steht für horizontal, um Verwechslungen auszuschließen).*
  4. *Zeile : Hier muß die Anzahl der Zeilen, die das Bild hat, in der Form 'Vnnn' stehen.*
  5. *Zeile : Hier kann das Wort KOMPRESS stehen. (nähere Erläuterungen - siehe weiter unten)*
  6. *Zeile : Diese Zeile dient evtl. späteren Erweiterungen.*
- *Ab der 7. Zeile sind dann die eigentlichen Bildinformationen zu finden.*

Ziel der Bildkompression ist es, die Größe der Text-Bild-datei (speziell bei Schaltbildern) zu verringern, ohne daß dabei Informationen verloren gehen. Dies ist möglich, indem man Bereiche, die vollständig schwarz oder weiß sind, nicht mehr Punkt für Punkt repräsentiert, sondern als Bereichsbeschreibung zusammenfaßt.

Um eine Aufwärtskompatibilität mit früheren, nicht komprimierten, Dateien zu erhalten, gelten die gleichen Konventionen mit folgenden Erweiterungen :
In der 5. Zeile der Datei muß jetzt für komprimierte Dateien das Wort KOMPRESS stehen.
Die Zeichen chr$(96) bis chr$(126) bekommen jetzt eine besondere Bedeutung :
chr$(96) - chr$(99) : Hier sind mindestens 18 schwarze Punkte nebeneinander. Um die Kompression / Dekompression programmtechnisch zu vereinfachen, wird allerdings nicht die genaue Anzahl der nebeneinander liegenden schwarzen Punkte bestimmt, sondern die Anzahl der Zeichen mit dem Ascii-Code 32, die in einer nicht komprimierten Datei in einer Zeile nebeneinander zu liegen kommen würden.
Das bedeutet, daß man 6-er Gruppen von schwarzen Punkten zählt und die Anzahl dieser in zwei aufeinander folgenden Zeichen verschlüsselt. Um möglichst viele 6-er Gruppen in zwei Zeichen darstellen zu können, wird folgendes Verfahren benutzt :
Das erste Zeichen enthält sowohl die Information, daß es sich hier um 6-er Gruppen von schwarzen Punkten handelt, als auch einen Teil der Zahleninformation zur Darstellung der Anzahl dieser 6-er Gruppen. Dazu darf das erste Zeichen nur den Ascii-Code 96 bis 99 besitzen. Die verschiedenen Codes liefern verschiedene Offsets : 96 entspricht dem Offset 0
97 " " " 95
98 " " " 190
99 " " " 285
Das zweite Zeichen kann dann natürlich den gesammten gültigen Bereich von chr$(32) bis chr$(126) benutzen und stellt den zweiten Teil der Zahlenrepräsentation dar. Die genaue Anzahl der 6-er Gruppen errechnet sich dann nach
Anzahl = Offset + Ascii-Code(zweites Zeichen) - 32.

chr$(100) - chr$(103) : Hier sind mindestens 18 weiße Punkte nebeneinander. Die Repräsentation dieser entspricht genau der der schwarzen Punkte, mit einem kleinen Unterschied in der Offset-tabelle : 100 entspricht dem Offset 0
101 " " " 95
102 " " " 190
103 " " " 285.

chr$(104) - chr$(126) : Dieser Bereich dient zur Verschlüsselung größerer leerer Bereiche. Sollten in einem Bild eine oder mehrere Zeilen vollständig aus schwarzen Punkten bestehen, so läßt sich die Anzahl derer wieder in zwei Zeichen zusammenfassen.
Da es sich hierbei um Einzelzeilen und nicht um 6-er Gruppen handelt, ist der Bereich der Ascii-Codes von 104 bis 126 auch fast 6 mal so groß wie die anderen beiden Bereiche. Die Anzahl der leeren Zeilen berechnet man genau so, wie die der 6-er Gruppen (also auch mit Hilfe von zwei Zeichen), mit dem Unterschied, daß die Offset-tabelle wesentlich größer ist. Anstatt sie explizit anzugeben, kann man sie (genau wie die anderen beiden Offset-tabellen auch) berechnen. Dabei gilt :
*Offset(Zeichen) = (Ascii-Code(Zeichen) - 104) * 95.*
Entsprechend ist natürlich für die anderen beiden Tabellen die 104 gegen 96 bzw. 100 auszutauschen.

Je nach der Natur des zu komprimierenden Bildes ist die Ersparnis an Speicherplatz und Zeit (und bei Übertragung via Telefon auch an Geld) nicht unerheblich. Ein weitgehend leeres Bild z.B. kann im Extremfall mit zwei Zeichen dargestellt werden. Ein Bild, das keine durchgehend weiße oder schwarze Bereiche besitzt, kann dagegen nicht komprimiert werden, d.h. es ist nach der Kompression genau so groß wie vorher.

Die im Programm-Service der DFÜ enthaltenen Programme zeigen die Technik des Abtastens und Wiedergebens sowie der Kompression und Dekompression von Bildern in Basic und Turbo-Pascal für verschiedene Rechner wohl deutlich genug, damit man sie an den eigenen Typ anpassen kann. Außerdem wird gezeigt, wie man Bilder von einem Rechner beliebigen Typs auf einem Nadeldrucker vollständig wiedergeben kann.

Wie im letzten Absatz zu lesen, enthält die Mailbox nicht nur Bilder im WDR-Format, sondern auch Programme in verschiedenen Sprachen und für verschiedene Rechner, mit denen diese Bilder auf einem Grafikbildschirm angezeigt oder einem Drucker ausgedruckt werden können. Auch hier haben wir uns bedient und zunächst zwei Turbo-PASCAL- und danach noch ein BASIC-Programm herübergezogen. Die beiden Turbo-PASCAL-Programme sind ursrpünglich für einen IBM-PC geschrieben worden, ließen sich aber in weniger als fünf Minuten an den oben schon erwähnten 4p/IIs-Grafiktreiber anpassen. Das Programm IBM-BILD.PAS dient dazu, sowohl komprimierte als auch unkomprimierte Bilder einzulesen und auf dem Bildschirm sichtbar zu machen. KOMPRESS.PAS macht, der Name verrät es schon, aus einem nicht komprimierten ein komprimiertes Bild im WDR-Format. Was das bringt, will ich an einem kleinen Beispiel verdeutlichen.

Das nebenstehende Bild stammt von einem Apple II+ und hat horizontal 280 und verikal 192 Punkte. Als unkomprimiertes WDR-File ist es genau 9600 Bytes lang. Komprimiert bringt es weniger als die Hälfte "auf die Waage" und schlägt nur mit 4268 Bytes zu Buche. Im Klartext: die Übertragung des komprimierten Bildes über die Telefonleitung kostet nur halb soviel, wie die des unkomprimierten!!! Ein weiterer Vorteil besteht darin, daß bei der Darstellung komprimierter Bilder Linie- statt Point-Befehle verwendet werden können und sich so die zur Darstellung benötigte Zeit auf ca. die Hälfte reduziert (gemessen auf dem IIs von Alexander).

Auch das BASIC-Programm, welches wir der Mailbox entnommen haben, dient dem Anzeigen von WDR-Bildern, und zwar sowohl von unkomprimierten als auch von komprimierten. Es ist für den TRS 80 Model 1 und seine Verwandschaft geschrieben, welche mit einer HRG 1b von RB-Elektronik ausgerüstet sind. Die drei Programme an dieser Stelle abzudrucken halte ich nicht für sonderlich sinnvoll, da sie (genau wie die in der oben angeführten Liste, der in der WDR-Mailbox vorhanden Bilder mit einem '*' gekennzeichneten Grafiken) ab sofort auch in den Programmsammlungen des CLUB 80 (bei Werner Förster für NewDOS und bei Andreas Rychlik für CP/M) zu haben sind. Wer sich für das Thema interessiert, kann sich an diese beiden, oder auch direkt an mich, wenden. Sollte jemand weitere Bilder aus der WDR-Mailbox herauskopieren, wäre ich dankbar, wenn er diese Dateien ebenfalls der Clubdiskothek zur Verfügung stellen würde.

Hartmut Obermann

Es wurden schon viele Vorschläge für einen einheitlichen Grafik-Standard im Club gemacht, aber die Resonanz war leider immer sehr gering. Ein kleiner Schritt in diese Richtung war auch der Artikel von Gerald Schröder, wo er sowas für das Genie II und IIs gebracht hat. Auf dem Clubtreffen konnte ich nun die Grafikfähigkeiten des Model IV bewundern, für das u.a. Hartmut Obermann einen Treiber unter Turbo-Pascal hatte. Um die bestehenden Programme nicht umschreiben zu müssen, lag es für mich nahe, die Prozeduren zu übernehmen und auf mein Genie IIs zu übertragen. Natürlich ist die Grafik hier ganz anders organisiert und so war ich erstmal auf der Suche nach einem Treiber, der die ganze Rechnerei übernimmt. Geralds fast vergessenes Werk war nun wie dafür geschaffen, diese Rolle zu übernehmen. Er erwartet nur die Koordinaten der Punkte in den Registern HL und DE und die Nummer der gwünschten Funktion in A, den Rest macht er von allein. Das ganze Problem war nur noch, den Treiber irgendwo unterzubringen und vom Turbo-Pascal anspringen zu lassen. Um z.B. einen Punkt zu setzen reicht folgendes (pseudo) Programmchen:

```
PROZEDUR Set(X,Y)
  Lade HL mit X;
  Lade DE mit Y;
  Lade A mit 1;
  CALL Treiber
END
```

Am besten wird sowas als INLINE direkt mit ein paar Bytes Maschinencode erschlagen.

Wer jetzt traurig ist, daß er kein IIs hat, den kann ich trösten, es gibt nämlich genau den gleichen Treiber auch für die HRG 1b, womit wohl die meisten bedient sein dürften.

Um Euch einen kurzen überblick über die Möglichkeiten zu geben, habe ich hier das Definitions-Modul der Modula-Einbindung eingebaut:

```
(*****************************************************************************)
(* GRAFMOD1 macht die HRG des Genie IIs auch unter Turbo-Modula nutzbar.     *)
(*                                                                           *)
(* Folgende Prozeduren werden zur Verfuegung gestellt:                       *)
(*                                                                           *)
(* HIRES                              -> schaltet die HRG ein und den        *)
(*                                       Textschirm aus                      *)
(* TEXTMODE                           -> schaltet die HRG aus und den        *)
(*                                       Textschirm ein                      *)
(* BEIDE                              -> wie HIRES (hier nicht anders        *)
(*                                       möglich)                            *)
(* CLSH                               -> löscht den Grafikschirm             *)
(*                                                                           *)
(* INVERS                             -> invertiert den Grafikschirm         *)
(*                                                                           *)
(* PLOT(X,Y,COLOR)                    -> setzt, löscht oder invertiert       *)
(*                                       einen Punkt                         *)
(* DRAW(X1,Y1,X2,Y2,COLOR)            -> zieht eine Linie von (X1,Y1) nach   *)
(*                                       (X2,Y2)                             *)
(* DOTLINE(X1,Y1,X2,Y2,COLOR,MUSTER)  -> wie DRAW, aber gestrichelt           *)
(*                                                                           *)
(* CIRCLE(X,Y,R,COLOR)                -> zieht einen Kreis mit Radius R      *)
(*                                       um (X,Y)                            *)
(* ELLIPSE(X,Y,ACHSE1,ACHSE2,COLOR)   -> zeichnet eine Ellipse               *)
```



```
(*                                                                           *)
(* MOVETO(X,Y)                        -> setzt aktuelle Koordinaten auf (X,Y)*)
(*                                                                           *)
(* LINETO(X,Y)                        -> zieht Linie von den letzten Koordi- *)
(*                                       naten nach (X,Y) mit der aktuellen  *)
(*                                       Farbe                               *)
(*                                                                           *)
(* POINT(X,Y)                         -> Abfrage, ob (X,Y) gesetzt           *)
(*                                                                           *)
(* Bedeutung von COLOR:  0 - löschen                                         *)
(*                       1 - setzen                                          *)
(*                       2 - invertieren                                     *)
(*                                                                           *)
(*               MUSTER: BITSET (beliebiger Typ, der mit 16 Bits dargestellt *)
(*                       werden kann, z.B. INTEGER-Zahl oder direkt als Bit- *)
(*                       muster z.B. mit {1,2,3,4,5,10,11,12,13}  )          *)
(*                                                                           *)
(*****************************************************************************)

DEFINITION MODULE GRAFMOD1;

  TYPE set15=BITSET;

  CONST XMAX=479;       (* maximale X-Koordinate *)
        YMAX=191;       (* maximale Y-Koordinate *)

  PROCEDURE HIRES;
  PROCEDURE TEXTMODE;
  PROCEDURE BEIDE;
  PROCEDURE CLSH;
  PROCEDURE INVERS;
  PROCEDURE PLOT(X,Y,Color:INTEGER);
  PROCEDURE DRAW(X1,Y1,X2,Y2,Color:INTEGER);
  PROCEDURE DOTLINE(X1,Y1,X2,Y2,Color:INTEGER; Muster:set15);
  PROCEDURE CIRCLE(X,Y,Radius,Color:INTEGER);
  PROCEDURE ELLIPSE(X,Y,A,B,Color:INTEGER);
  PROCEDURE MOVETO(X,Y:INTEGER);
  PROCEDURE LINETO(X,Y:INTEGER);
  PROCEDURE POINT(X,Y:INTEGER):INTEGER;

END GRAFMOD1.
```

Damit wären wir eine recht ansehnliche Gemeinde (Model IV, Genie I/II und Tandy mit HRG 1b und Genie IIs...) und es würde sich endlich lohnen, sogar größere Grafik-Programme entweder anzupassen oder selber zu schreiben. Von Hartmut Obermann habe ich z.B. etliche Programme aus den CHIP-Specials, die ich, von ihm schon für sein Model IV umgebaut, ohne Probleme laufen lassen konnte (Apfelmännchen (natürlich!), Funktionenplotter...).

Wer sich für nähere Einzelheiten interessiert, soll sich bei mir melden, für Fragen stehe ich gerne zur Verfügung. Den kompletten Treiber hier abzudrucken halte ich für ziemlich witzlos, da es eine elende Tipperei wäre, die sich mit Sicherheit keiner antun würde. Außerdem treten die Probleme normalerweise sowieso erst beim Einbauen auf und sowas läßt sich dann wesentlich besser individuell lösen und nicht global im Info.

Alexander Schmid

Z-CP/M, eine deutliche Verbesserung

von Rüdiger Sörensen.

1. Einführung.

Jeder kennt CP/M, das Control Program for Microcomputers. Der Name und das Programm stammen von der Firma Digital Research. CP/M ist eines der ältesten Betriebssysteme für Micros und bestimmt eines der schlechtesten. Ursprünglich wurde es für die Intel-8080 Familie von Prozessoren entwickelt, also I8080, I8085 usw. Als Zilog mit dem Z80 auf den Markt kam, wurden auch für diese Maschinen CP/M Systeme erstellt, denn der Z80 ist ja bekanntlich Software-verträglich mit dem I8080. Unter anderem auch durch diese Portierung auf den Z80 wurde CP/M eines der meist verwendeten Betriebssysteme überhaupt. Es wurde auf einer Unzahl der verschiedensten Rechner implementiert. Heutzutage gibt es noch eine Reihe von interessanten Rechnern, die voll auf CP/M aufbauen, die meisten davon sind in den USA zu finden (Northstar, Kaypro, Osborne und verschiedene Bus-Systeme wie S100, SB180 oder in Europa ECB-Bus). Die Software-Entwicklung unter CP/M war jahrelang geradezu unglaublich, praktisch jede Hochsprache wurde unter CP/M implementiert, die ersten Spreadsheets, Datenbanken, Textverarbeiter ebenfalls. Jeder kennt Programme wie DBase, WordStar, Multiplan und Turbo-Pascal. (Anm. Wenn ich von CP/M spreche, meine ich CP/M 2.2)

Ein Zeichen für die mangelhafte Benutzerunterstützung durch CP/M ist die Tatsache, daß praktisch alle Programme eigene Shells haben und die meistgebrauchten Aufgaben wie das Kopieren, Löschen von Files, Ausgabe eines Directories usw. selbst erledigen. Darüber hinaus werden vielfach diese Ausgaben in Utilities zusammengefaßt, man denke an CPMPOW, Sweep oder so was. CP/M selbst kennt nur 6 interne Befehle : DIR, ERA, REN, USER, TYPE und SAVE. Sie besorgen das Directory-Listen, Löschen, umbennenen von Files, Wechseln der USER-Ebene (eine Art Sub-Directory) und den Dump des Speichers auf die Floppy. das ist nicht gerade viel. Dazu gibt es die CP/M Standard-Utilities PIP, STAT, LOAD, ED, ASM, SUBMIT. Damit ist unter anderem das Kopieren von Dateien möglich, Ausgabe des freien Diskettenplatzes und der Länge von Files, Laden von Intel-HEX-Files, Erstellen und Ändern von Text-Dateien und Assemblieren von I8080-Quelldateien. Bedingt durch ihr Alter leiden die Utilities an übermäßiger Einfachheit. Deshalb gab es immer einen großen Markt für Verbesserungen jeder Form. Wegen der großen Verbreitung des Betriebssystems wurden neben kommerziellen auch eine Unzahl von public domain Programmen erstellt und der Öffentlichkeit zugänglich gemacht. Die letzte Zählung in einem der größten PD-Archive ergab über 8000(!) PD-Pakete. Dabei ist die Qualität der Programme zum Teil sehr hoch. Diesen Text schreibe ich auf einem PD-Wordprocessor.

Kurz gefaßt, zu CP/M kann man sagen : das Betriebssystem taugt nichts. Die Qualität und Anzahl der verfügbaren Programme übersteigt das Angebot von anderen Betriebssystemen. (Mir tut es leid, daß ein excellentes System wie TRSDOS 6.x so den Bach runtergeht, weil kein Mensch mehr Software entwickelt.)

2. Komponenten.

CP/M besteht aus vier Bestandteilen. Der erste ist das BIOS (Basic Input/ Output-System). Das BIOS ist ein Programm, welches den einzigen(!) Hardware-abhängigen Teil des Betriebssystems darstellt. Das BIOS enthält eine Sprungleiste, meist am Anfang, die in die einzelnen Routinen verzweigt. Die Reihenfolge der Sprünge und die Antwort des BIOS sind die einzigen standartisierten Teile des BIOS. Für eine genaue Beschreibung schaut in Geralds Artikel (wie man sich ein BIOS strickt) nach. Aufgaben des BIOS sind unter anderem Ein/Ausgabe Terminal, Drucker, RS232, Diskette. Der zweite Teil ist das BDOS, das eigentliche Disketten-Betriebssystem. Hier sind die Programmteile zum Lesen und Schreiben von Files gelagert. Das BDOS unterliegt dem Copyright von DRI. Der Code wurde meines Wissens niemals veröffentlicht. Der dritte Teil ist der CCP (Console Command Processor). Das ist das Ding, mit dem sich der Benutzer herumärgert. Der CCP hat die Aufgabe, Benutzereingaben zu interpretieren und auszuführen, falls kein Fehler auftritt. Der Funktionsablauf kann so skizziert werden :

```
wiederhole
Lies Benutzereingabe
trenne Kommando und optionale Parameter
wenn eingebautes Kommando
        dann führe Kommando aus
        sonst
                wenn
                Kommando = ausführbare Datei
                dann lade Datei nach 100H und
                starte Programm bei 100H
                sonst gib Fehlermeldung aus.
für  immer.
```

Aufgrund der wenigen eingebauten Kommandos ist einzusehen, das CP/M reichlich auf den Scheiben rumkratzt. Hinzu kommt eine andere Eigenart: Jedes Programm darf den CCP überschreiben, sogar das BDOS (wenn z.B. kein I/O stattfindet). Nach Beendigung des Programms werden deshalb der CCP und das BDOS von der Diskette neu geladen und das System neu initialisiert. Das nennt sich Warmstart (warm boot). Wenn man nur kleine Programme hat, wird das ganz schön langweilig. Weiter oben habe ich die Utility SUBMIT erwähnt. Damit ist eine Batch-Verarbeitung möglich. Dabei ist es wirklich eine Kunst, keine grauen Haare zu bekommen. SUBMIT liest eine Textdatei, expandiert ev. Parameter und schreibt die einzelnen Kommandos in umgekehrter Reihenfolge in einen File namens $$$.SUB. Nach jedem Warmstart schaut nun der CCP nach, ob diese Datei existiert. Dann wird der letzte Datensatz gelesen, gelöscht und verarbeitet. Dann wieder Warmstart. Hhhhhh... Die Speicherorganisation unter CP/M ist genial einfach. Die ersten 256 Byte sind für Systemaddressen reserviert. Ab 0100H fängt der für Benutzerprogramme freie Bereich an (die TPA). dann der CCP, der auf eine Seitengrenze (Addresse XX00H) anfangen muß. Der CCP ist 800H Byte lang. Dann das BDOS, ebenfalls 800H Byte lang. Dann das BIOS mit unbeschränkter Länge. Die Addressen des BIOS und des BDOS stehen bei 0006H bzw. 0001H. Dadurch kann jedes Programm die notwendigen Addressen selbst ermitteln. Die letzte, vierte Komponente sind die System-Utilities.

3. Z-Systeme.

In unserem Verein hat jeder einen Rechner mit Z80 oder HD64180. Im Prinzip ist also CP/M auf jedem Rechner lauffähig, der über einen ungestörten RAM-Bereich von 0000H an verfügt. Spaß macht es erst ab 64K Speicher. Wie gesagt, ist aber das Betriebssystem für I8080-Maschinen gebaut. Warum also nicht die Zusatz-Power unserer Apparate nutzen? Im BIOS wird das ohnehin gemacht, aber BDOS und CCP könnten geändert werden (d.h. neu geschrieben) um leistungsfähigere Systeme zu entwickeln. Die Idee scheint gut, und glücklicherweise haben andere sie auch gehabt. Da das BIOS allein hardwareabhängig ist, sind Z80-Versionen der anderen Bestandteile prinzipiell auf jeder Z80-Kiste lauffähig (das gilt eingeschränkt für HD64180, wegen Illegals). Also, ans Werk! Z-Systeme sind in den USA Mailboxen, Server und Händler, die Software für Z80-basierende CP/M-Systeme anbieten. Über die Uni habe ich Zugang zu den Servern. Die dort angebotene Software ist in der Regel Public Domain, neuerdings meist Freeware. Public Domain heißt, jedermann zugänglich, änderbar und auch in geänderter Fassung beliebig weiterzugeben. Freeware macht eine Einschränkung : Programme sind in den meisten Fällen mit Copyright des Autors versehen, dürfen aber für nicht-kommerzielle Zwecke in ungeänderter Version benutzt und weitergegeben werden. Es gibt BDOS-Fassungen für Z80. Diese heißen allerdings nicht BDOS (DRI!) sondern P2DOS, TURBODOS oder Z80DOS. Meistens sind die Versionen schon älter und von vielen Leuten getestet, erkannte Fehler korrigiert. Es wäre kein wesentlicher Fortschritt, die DOS-Teile nur in Z80-Code umzusetzen. Das DOS wird zwar kompakter und unwesentlich schneller (wg. Disk-Zugriffen), leistet aber doch nicht mehr. Es ist deshalb Ziel der Programmierer, neue DOS-Funktionen einzubauen. Dabei geht aber die Kompatibilität von Programmen verloren, die diese neuen Features nutzen. Es gibt in den Z-Systemen deshalb Normierungsbestrebungen. Eine beliebte Erweiterung ist das Nutzen einer Hardware-Uhr, um Dateien mit Zeit/Datum Information zu versehen. Alle mir bekannten Z80-Systeme bieten das als Option an. Weitere Möglichkeiten sind beschleunigte Disketten-Zugriffe durch Disk-Caching oder public files, das heißt, Files, die in einer bestimmten USER-Ebene stehen, sind von jeder anderen USER-Ebene aus zugänglich. Ein CP/M-Eigenart wird ebenfalls meist abgeschafft : Wenn unter CP/M eine Diskette gewechselt wird, weigert sich BDOS, auf diese zu schreiben, bis sie durch einen Warmstart angemeldet wird. Neuere Z-Dos Versionen erkennen Disk-Wechsel und melden die Disketten automatisch an. Das spart Ärger. Ein typischer Fall unter CP/M : starte Turbo-Pascal und schreibe ein größeres Programm. Dann versuche, es zu sichern. Disk full. Diskette wechseln, Versuch zu saven, und CP/M sagt : "BDOS Err on x:, disk R/O". Feierabend. Wie man sieht, kann ein Z80-DOS nicht von Nachteil sein. Was ist mit dem CCP? Hier schlägt die Stunde des Programmierers. Läppische sechs residente Kommandos... Es gibt eigentlich nur einen Ersatz für den CCP, und der heißt ZCPR3x. Un der ist der Grund für diesen Artikel. An anderer Stelle (wahrscheinlich in diesem Info) habe ich die Installation beschrieben. Hier soll eine Beschreibung der Konzepte und Möglichkeiten folgen.

4. ZCPR3x.

Für ein Programm macht es keinen Unterschied, ob es unter ZCPR3 oder CCP gstartet wird. Alle (lt. Aussage des Autors, R. Conn) käuflichen Programme laufen unter ZCPR3 genau wie unter CCP. Das ist wichtig und beruhigend. ZCPR3 fügt eine Reihe von wesentlichen Verbesserungen ein. Teilweise sind sie von Großrechner-Systemen abgeguckt (UNIX, MULTICS, VMS, NOS). Anlehnungen an MSDOS finden sich ebenfalls. Fangen wir an :

Mich hat immer geärgert, das unter CP/M zwar Directories existieren in Form von 32 USER-Ebenen, daß es aber für den Normalbenutzer sinnlos ist, damit arbeiten zu wollen. Zum Beispiel ist es sehr schwierig, Dateien von einer Ebene in eine andere zu kopieren. Programme haben in der Regel keinen Zugriff auf andere Ebenen. Man kann sich kaum einen Überblick verschaffen über den Inhalt der einzelnen Ebenen, es sei denn, man loggt sich in die Ebene ein und startet DIR. Unter ZCPR3 sind alle Ebenen gleichberechtigt (manche mehr..). Im Prompt wird neben dem Laufwerk auch die Userebene ausgegeben. Mit den ZCPR3x Utilities ist Zugriff auf andere Ebenen problemlos möglich. Weiter verwaltet ZCPR3 sogenannte Named Directories. Man kann einer Ebene einen symbolischen Namen geben, unter dem sie ansprechbar ist. Das sieht aus wie MSDOS, nur gibt es keine hierarchische Struktur. Jede Ebene kann mit Password versehen werden. Beim Einloggen in eine Ebene kann ein Initialisierungsprogramm anlaufen. CP/M sucht eine ausführbare Datei nur in der aktuellen USER-Ebene. Optional kann die Laufwerksbezeichnung mit angegeben werden. ZCPR3 unterstützt einen sog. PATH, das heißt, einen Suchpfad, auf dem nach Dateien geguckt wird, wenn sie nicht im aktuellen DU zu finden sind (DU heißt DRIVE/USER). Genau wie die Named Directories kann auch der PATH jederzeit geändert werden. Entlang dem PATH werden nicht nur Programme gesucht, sondern auch Systemdateien. ZCPR3 hat eine Multiple Command Line Buffer (MCL). Damit kann eine Kommandozeile mehrere einzelne Kommandos enthalten, die durch ";" getrennt werden. Diese Fähigkeit ist eine der wichtigsten, auf sie bauen andere Nützlichkeiten auf. Z.B. die ALIASe. Ein ALIAS ist ein .COM-File, das als solches gestartet wird und nichts weiter macht, als eine Reihe von Kommandos in den MCL zu schieben. Dabei werden Parametervariablen expandiert, wie in SUBMIT. Beispiel für eine Kommandozeile:

```
A0:BASE>asm:;zas test;zlink test;test
```

Macht folgendes :
Wechselt in DU asm:
Assembliert TEST.Z80
Linkt TEST.REL
Startet TEST.COM

Als Alias würde man das so schreiben:

```
ASM:;ZAS $1;ZLINK $1;$1
```

Dabei wird der Alias mit einem Namen versehen, z.B. MAKE.COM, und dann so aufgerufen : A0:BASE>make test Was dann entsprechend expandiert wird, d.h. $1 wird durch den String "test" ersetzt. Um ein ALIAS anzulegen, wird eine Z3-Utility benötigt. Bis jetzt bin ich noch nicht auf die residenten Kommandos eingegangen. ZCPR3 hat, wie CCP, einige eingebaute Kommandos. Das sind in der Regel weniger als 6. Dafür existiert ein Speicherbereich, der RCP (resident command package), wo sich beliebige Pakete von residenten Kommandos hinladen lassen. Meine Lieblinkskonfiguration verfügt über

die Kommandos

GO, SAVE (CPR-resident)
H, DIR, ERA, REN, CP, TYPE, LIST, SP, NOTE, ECHO (RCP-resident)

Nicht schlecht, was? LIST gibt auf den Drucker aus, H gibt Liste der Kommandos, SP gibt Diskettenplatz aus, NOTE ist Kommentar, ECHO gibt sein Argument am Terminal aus. Die ist EINE Möglichkeit, einen RCP zu bauen. Es gibt viele andere, da noch andere Kommandos zur Verfügung stehen, zum Beispiel PEEK und POKE zum Ändern des Speichers, REG zum Ändern von ZCPR3-Registern u.a. Es ist durchaus möglich, während einer Sitzung dén RCP zu ändern, indem enfach ein neuer geladen wird. Neben den RCP's gibt es die FCP's. Das sind Flow Command Packages, eine Art Programmiersprache für den Command Processor. Auch sie sind resident und können jederzeit geladen werden. In Verbindung mit ALIAS und Batch-processing eine tolle Sache. Hier mein FCP :

IF, ELSE, FI, IFQ, ZIF

Zum FCP gehört ein transientes Programm namens IF.COM. Es ist einleuchtend, daß ein IF allein noch keine Steuerung des Kommandoflusses bewirken kann, dazu gehört auch noch eine Bedingung. Wenn der FCP eine Bedingung nicht kennt (meiner kennt gar keine), dann lädt er das transiente IF und sucht dort nach. Als Bedingungen kann man z.B. prüfen ob eine Datei existiert, ob ein Parameter gleich irgendwas oder nicht besetzt ist, ob ein Fehler aufgetreten ist, ob eine Shell aktiv ist, ob der Batch-Prozessor läuft und....

In Verbindung mit dem obigen ALIAS benutze ich folgende Konstruktion:

```
asm:zas $1;if input;zlink $1;$1;fi
```

Wenn der Assembler fertig ist, wartet der ALIAS auf eine Tastatureingabe. Falls der Assembler fehlerfrei gearbeitet hat, gebe ich <RETURN> oder <T> ein, um den Flow State auf TRUE zu setzen. Dann wird der ALIAS weiter fortgesetzt. Sollte ein Fehler aufgetreten sein, gebe ich <BREAK> oder <F> ein. Dann wird der Flow State FALSE, und bis zum "fi" (=endif) wird kein Kommando mehr ausgeführt.

ZCPR3 erlaubt seinen Utilities (und inzwischen auch vielen anderen Programmen) den Zugriff auf sein sog. Environment. Darin steht unter anderem, ob ein Fehler aufgetreten ist oder ob ZEX läuft (s.u.). 80 byte dieses Environment Blocks werden von TCAP benutzt, der ZCPR3 Terminal Capability. Darin ist codiert, wie das Terminal anzusprechen ist, also Sequenzen zum Bildschirm löschen, Clear to EOL, Cursor positionieren, Belegung der Cursortasten. Das heißt, daß jede Utility davon Gebrauch machen kann, ohne daß sie speziell installiert werden müßte. (Das stimmt nicht ganz, ZCPR3-Utilities müssen schon installiert werden. Das geschieht mit dem Programm Z3INS. Es erledigt die Installation aller ZCPR3-Utilities (COM-Files, über 80 Stück) in wenigen Minuten. Dabei wird in jedes Programm die Addresse des Environment-Descriptors eingetragen, also ein 2-Byte Wert.)

4.1 ZCPR3 Utilities.

Einige davon habe ich schon angesprochen, hier eine lose Liste (es gibt über 100 ZCPR3 Utilities) :

ALIAS, VALIAS erstellen ALIAS-Scripts (.COM-Files)
XDIR gibt directory aus
MCOPY kopiert Gruppen von Files
MLOAD macht aus HEX-Dateien .COM-Files.
UNERASE restauriert gelöschte Dateien.
RENAME benennt Gruppen von Files um, RENAME *.CMD=*.COM
VLU Video orientierte Library Utility
HELP gehört zum help-System. Das kann man nicht beschreiben, das muß man gesehen haben.
DU3 Disk Utility & Editor
MU3 Memory Utility & Editor
DIFF untersucht Dateien auf Identität
DPROG erlaubt mittels einer speziellen Sprache, I/O-devices zu programmieren.
CLEANDIR räumt directories auf, entfernt unbenutzte Einträge und sortiert nach alphabetischer Reihenfolge.
FINDF sucht auf dem angegebenen ode auf allen Laufwerk(en) nach den spezifizierten Files. Wildcards sind möglich. Die Suche erstreckt sich über alle 32 DU's.

4.2 Shells.

Eine Shell ist ein Kommando-Prozessor, der dem RCP vorgeschaltet ist. Es gibt verschiedene und sie lassen sich überlagern. Eine Beschreibung im Einzelnen :

o SH.COM

Die Abkürzungs-Shell. Mit den Utilities SHVAR oder SHDEFINE wird eine Datei namens SH.VAR angelegt. In ihr befinden sich Definitionen von Abkürzungen, z.B. W = TEST.Z80. Beim Starten von SH ändert sich der PROMPT:

```
A0:BASE>> TYPE %W
```

%W wird von SH in TEST.Z80 expandiert. Shell-variablen werden durch das "%" gekennzeichnet. Sie können verschachtelt werden.

o VFILER.COM

Die Directory-Maintenance-Shell. Full-Screen löschen, umbenennen, kopieren, unsqueezen von Files und Gruppen von Files. Information über Filesize. Macro-fähig. Aus der Shell können natürlich ZCPR3-Kommandos abgesetzt werden.

o MENU.COM, VMENU.COM

Damit lassen sich einfach Menus bauen. Zum Beispiel eine TURBO-Pascal Umgebung für den Assembler/Linker/Editor. VMENU erlaubt Full-Screen Möglichkeiten.

o HSH.COM,NHSH.COM

Die History-Shell. Sie ist für den Benutzer unsichtbar und verwaltet eine Liste der 20 letzten Kommandos. Auf Druck der CursorUp Taste erscheint das letzte Kommando und kann editiert und abgeschickt werden. (DOSEDIT läßt grüßen)

Alle Shells können gleichzeitig übereinander betrieben werden.
(4 Stück max.)

4.2 ZEX

ZEX ist der ZCPR3-Batch-Prozessor. Er kann alles, was SUBMIT kann, und viel mehr. Seine hervorstechende Eigenschaft ist, daß er im Speicher arbeitet, und nicht auf der Disk. Dadurch wird er viel schneller und flexibler. Natürlich werden FCP-Kommandos verarbeitet und mit GOTO besteht die Möglichkeit, Schleifen bedingt zu durchlaufen. Prinzipiell werden Dateien vom Typ .SUB und vom Typ .ZEX verarztet. Bei der Installation von ZCPR3 hat man die Möglichkeit, SUBMIT wahlweise zu aktivieren. Diese Eigenschaft habe ich zugunsten von ZEX abgeschafft. Er erledigt größere Aufgaben zuverlässig und weitaus schneller als SUBMIT.

4.3 CMDRUN

Das ist der gängige Name für einen ECP (External Command Processor)
Wenn der CPR mit einem Kommando aktiviert wird, passiert folgendes:

```
Wenn Kommando Bestandteil des CPR, führe es aus.
sonst
Wenn Kommando Bestandteil des RCP, führe es aus
sonst
Wenn Kommando = .COM-File im PATH, führe es aus
sonst (ECP starten)
Wenn Kommando = ALIAS definiert in ALIAS.CMD
expandiere Parameter
     führe ALIAS aus
sonst
                    starte ERROR-Handler
```

Diese Beschreibung betrifft meinen ECP. Im Prinzip kann jedes Programm zum ECP gemacht werden, z.B. ZEX. Falls der oben beschriebene Weg also fehlschlägt, würde ZEX den PATH absuchen, ob eine .SUB oder .ZEX Datei gefunden wird, die dann zur Ausführung gelangt.

4.4 ERRORx

Wenn auch der ECP nichts findet oder gar keiner installiert ist, gelangt der ERROR-Handler zum Zuge. Im einfachsten Fall gibt er nur ein Fragezeichen aus, wie CCP, er kann aber auch zum Beispiel die Kommandozeile zum Editieren und wiederverwenden bereitstellen.

5. Zusammenfassung.

Wer bis hierher gelesen hat, ist sicher ein potentieller ZCPR3x Anwender. Wie an anderer Stelle beschrieben, ist die Installation allerdings nicht ohne Tücken. Die meisten Leute werden mit der Auto Install Version Z-DOT-COM zufrieden sein. Andere mögen sich an die Installationshinweise halten. Von den PD-Servern habe ich inzwischen eine ganze Reihe von ZCPR33-Utilities bezogen, die zur Verfügung stehen. Man frage den Diskothekar. Falls Probleme auftreten sollten, stehe ich telefonisch oder schriftlich zur Verfügung.

6. Abkürzungen.

DRI	Digital Research
CP/M	Control Program for Micro Computers
ECP	External Command Processor
RCP	Resident Command Package
FCP	Flow Command Package
NDR	Named Directories
TCAP	Terminal Capability
CCP	Console Command Processor
ZCPR	Z80 Command Processor Replacement
CPR	Command Processor Replacement
BIOS	Basic Input Output System
BDOS	Basic Disk Operating System

Ich wünsche allen, die den Mut und die Zeit haben, allen denkbaren Erfolg.

Rüdiger.

---

mir ganz sicher, daß meine Freunde noch vor ein paar Tagen völlig normale Leute waren; liebenswerte Menschen, die auf Parties nicht in dunklen Schlafzimmern verschwanden, sondern sich ordentlich betranken, um anschließend ausfälliges Zeug zu reden und sich mit anderen Gästen zu streiten.
»Das hier ist die Zukunft«, sagte Wolfgang und schoß noch eine Tomate ab. »Eines Tages wird jeder so ein Ding haben.«
»Computer werden so selbstverständlich sein wie ein Telefon.« Jürgen sah mich mit verklärtem Blick an. »So ein Kasten kann dir jede Arbeit abnehmen, wirklich jede. Der macht alles für dich.«
Die Kiste hatte es tatsächlich geschafft. Sie hatte meine beiden Kumpel innerhalb kürzester Zeit um den Verstand gebracht. Denn wenn einer, der denkt, er würde impotent, wenn er auch nur eine Nacht ohne Frau im Bett verbringt, plötzlich nachts vor einer Maschine hockt und auf Tomaten schießt, dann muß irgend etwas schrecklich schiefgelaufen sein.
»Hattest du Probleme mit der Kleinen aus der Fotoagentur?« fragte ich vorsichtig, und dann versöhnlich: »Weißt du, bei mir klappt's manchmal auch nicht.«
»Du hast nichts begriffen«, sagte Jürgen, und seine Stimme klang richtig mitleidig. »Du versuchst, die Dinge zu ignorieren, aber das hier ist die Zukunft«, dabei drückte er ein paarmal auf den Plastikgriff in seiner Hand, und das Männchen auf dem Bildschirm gab ein lautes »Pfiuuuu« von sich.
»Schon in den nächsten Jahren«, ereiferte sich Wolfgang, »werden unsere Arbeitsplätze alle so aussehen. Und wenn du dich dagegen sperrst, verlierst du den Anschluß. Dann ziehen die jungen Leute vorbei, daß du kein Land mehr siehst.«
Zugegeben, mein Job als Steuerberater war nicht immer besonders spannend, aber ich konnte mir nicht vorstellen, daß ich eines Tages meine Bürozeit damit verbringen würde, auf vorbeifliegende Tomaten zu schießen.
Dann machte ich einen Fehler. Ich fragte ganz harmlos: »Kann der noch was anderes?«
Die beiden fielen über mich her wie zwei eifernde Sekten-Prediger bei der Jagd auf ein neues Opfer. Sie redeten von Bit und Chip, von RAM und ROM, Basic und Pascal und schoben zwischendurch immer neue kleine Plastikscheiben in den Computer, der darauf immer neue verwirrende Grafiken vorführte. Sie sprachen ununterbrochen und gleichzeitig. Meine Fragen interessierten sie nicht, sie waren glücklich, ihr Wissen loszuwerden.
Ich verstand nur Bahnhof, aber eines war sicher: Es hatte die beiden schwer erwischt.
Ich hatte schon viel von Computern gehört und von den schrecklichen Folgen, die diese Elektronengehirne für unsere Gesellschaft haben sollen. Aber ich konnte mir bisher nichts darunter vorstellen.
Für mich kam so eine Maschine sowieso nicht in Frage. Ich besaß nichts zu Hause oder auf dem Bankkonto, das sich lohnen würde zusammenzuzählen. Ich legte auch keine Fahndungskarteien über meine Freunde an, und schon beim Wort »Daten« zuckte ich angeekelt zusammen. Datenverarbeitung war etwas für Beamte mit Sockenhaltern und glänzenden Hosenböden.
Jetzt wurde mir die Computeritis zum erstenmal in ihrer entsetzlichen Tragweite vor Augen geführt. Sie hatte meine engsten Freunde befallen, und ich mußte hilflos zusehen, wie es mit ihnen bergab ging. Denn die Nacht auf der Party war nur der Beginn eines tragischen Leidensweges.
In dieser Zeit verloren sie nicht nur ihre Frauen, den Job und den Respekt der Nachbarn. Es wurde ihnen sogar der Überziehungskredit gestrichen und der Fernseher gepfändet. Es vergingen viele Monate, bis sie sich von diesem Schock wieder erholten. Ich mußte diese Tragödie in allen Einzelheiten miterleben. Da entschloß ich mich, etwas zu tun, um zu helfen, diese Landplage des 20. Jahrhunderts zu stoppen. Auch wenn es weltfremd klingen mag, ich bekenne mich zu meinen Gefühlen und sage auch jedem, der es nicht wissen will:

»Ich hasse Computer!«

Ich hoffe, daß ich mit diesem Buch die eine oder andere Seele retten kann. Und daß es denen Kraft und Argumente gibt, die sich auch in Zukunft ihre Software nicht im Computer-Shop, sondern in der Kneipe um die Ecke holen wollen ...

→ F.F. Die Redaktion

ZCPR33 - Installation auf Model 4 (und anderen).

Von Rüdiger Sörensen.

Seit einiger Zeit läuft auf meinem Model 4 ZCPR3 als Kommando-Prozessor unter CP/M. Das Ding war die sogenannte Z-DOT-COM Version, die sich selbst installiert. Das hat gehörige Vorteile, vor allem im Aufwand, den man betreiben muß, um das System an's Laufen zu kriegen, aber auch, was die Speicheraufteilung angeht. Z.COM kann nämlich abgeschaltet werden, und dann ist nur noch der Standard - CP/M - CCP (Command Processor, ZCPR heißt Z80-Command-Processor-Replacement) vorhanden und man verfügt wieder über den vollen Speicher. Z.COM nimmt etwa 8K in Anspruch, was in meinen Augen eine ganze Menge ist.
Aus diesem Grunde, und weil das Z.COM System so langsam bootet, habe ich eine manuelle Installation von ZCPR in's Auge gefaßt und mich für die neue Version 3.3 entschieden. Leider läuft das System noch nicht hundertprozentig, aber es läuft, und deshalb will ich versuchen zu erklären, wie's geht. Zunächst eine Beschreibung der Fähigkeiten von ZCPR33. (Nicht aus dem Auge verlieren : für die meisten Anwenderprogramme sind die Fähigkeiten nicht sichtbar, aber wer in seiner Kiste selbst rumstrickt, wird begeistert sein.)

1) Alle USER-Ebenen (bis 31) können benutzt werden und auch mit Namen versehen werden. Man kann Directories (USER-Ebenen) verstecken und mit Passwordschutz versehen. Beim Einloggen in ein neues Directory kann automatisch eine Einlogsequenz ablaufen.

2) Man kann einen Suchpfad definieren, auf dem der ZCPR nach *.COM-Files sucht.

3) Man kann Abkürzungen (ALIAS) für Kommandofolgen definieren und Parameter (bis 10 Stück) an sie übergeben.

4) Die Länge der Kommandozeile ist benutzerdefinierbar. Mehrere Kommandos können in einer Zeile abgesetzt werden.

5) Die residenten Kommandos (RCP's, das sind die, die der ZCPR direkt ausführt ohne eine .COM datei zu laden) können vom Benutzer definiert werden. Es ist auch möglich, eigene Kommandos zu entwickeln und einzubinden oder während einer Sitzung andere residente Kommandos zu laden.

6) Es gibt bedingte Ausführung von Kommandos mit den FCP's, das sind Flow Command Packages. Wieder benutzerdefinierbar. Beispiele sind IF, ELSE, FI (endif), XIF (alle IF's beenden), die Bedingungen, die überprüft werden können, sind z.B. Existenz eines Files, Fehler bei einem Programm, leerer File. Negation ist möglich.

7) ZCPR unterhält einen Buffer, in dem Informationen über das benutzte Terminal vorhanden sind. Alle ZCPR-Utilities (über 70 Stück) können darauf zugreifen und man verfügt dann über Cursor-Addressierung, Schirm löschen, inverse Darstellung und dergleichen.

8) ZEX.
Das ist für mich fast das beste. ZEX ist ein Ersatz für SUBMIT, kann aber viel mehr und läuft resident im Speicher. Dadurch ist ZEX viel schneller als SUBMIT. ZEX kann aber auch *.SUB-Files ausführen.

9) Shells. Davon gibt es verschiedene. Eine Shell ist eine Art vorgeschalteter Kommando-Prozessor, der Eingaben des Benutzers filtert und eigene Aktionen starten kann. Beispiel : VFILER, eine Shell zur Diskettenverwaltung. Oder MENU, damit kann man Menus auf ZCPR - Ebene aufbauen, zum Beispiel um eine TURBO-PASCAL Umgebung für den Lieblingsassembler zu bauen.

10) Input/Output-Packages erlauben die Umleitung von I/O auf beliebige Geräte. Viel flexibler als IO-BYTE.

Soweit ein kleiner Lobgesang. Der Nachteil soll aber auch nicht verschwiegen werden : Ein voll ausgebautes ZCPR33 System kostet ca. 5K TPA. Bei mir habe ich immer noch 48K TPA, das langt bis jetzt für alle Anwendungen. Außerdem ist die Installation schwierig, unter Umständen sogar unmöglich. Ihr könnt prüfen, ob es bei euch geht, wenn ihr die folgende Liste der Notwendigkeiten anschaut.

Hardware: Z80 - System mit mindestens einem Laufwerk (ich empfehle 2-3 80T/DS oder Harddisk)

Software:

Sourcen zu den ZCPR3/33 Programmen und Utilities. ZCPR3/33 ist ein Freeware-System, das bedeutet, man kann es zu seinem persönlichen Gebrauch benutzen, ändern und weitergeben, aber nicht verkaufen.

Sourcecode des BIOS. Das dürfte das größte Problem sein. Das BIOS muß ziemlich gründlich geändert werden, zumindest die Kaltstartroutine.

SYSGEN, MOVCPM, DDT oder ähnliches, MAC. Wenn die Utilities geändert werden sollen, auch noch M80 und L80. Wenn ZCPR33 benutzt werden soll, ZAS (oder man editiert 128 KB Quellcode für M80).

nützlich, aber nicht unbedingt notwendig : MLOAD und ein Full-Screen Editor wie PMATE oder WORDSTAR.

Als erstes muß man sich hinsetzen und planen, was für Fähigkeiten der ZCPR bekommen soll. Meine Empfehlung ist, alles einzubauen, außer vielleicht den IO-Packages. Manche Sachen sind unbedingt notwendig, andere sehr zu empfehlen. Man benötigt folgende Speichergrößen (opt = optional, rec = empfohlen, req = muß sein):

2K	RCP	resident Command Package (rec)
1.5K	IOP	IO-Package (opt)
0.5K	FCP	flow command package (if, else..) (rec)
256 B		Environment descriptor (req)
128 B		Shell Stack (rec)
80 B		Message Buffers (req)
48 B		external FCB (rec)

128 B	NDR	named directory buffer (rec)
208 B		Command Line Buffer (req)
48 B		external Stack (req)
11 B		external path (req)
1 B		Wheel Byte (opt)

Das einfachste ist, oberhalb des BIOS Platz zu machen, indem man sein System mittels MOVCPM auf 59K verkleinert. Wenn das IOP eingebaut werden soll, dann muß man darauf achten, daß es auf einer Page Boundary beginnt, also bei einer HEX-Adresse, die mit 00 endet.
Wenn der Platz besorgt wurde, alle Addressen notieren.
Dann muß das BIOS geändert werden, und zwar die Kaltstartroutine.
Folgendes wird gefordert :

> Alle ZCPR-Buffer (RCP,FCP,IOP etc.) müssen beim KALTSTART initialisiert werden, d.h. mit 00H gefüllt werden.
>
> Der Command Line Buffer wird initialisiert mit Information über die Länge des Buffers, dem String 'STARTUP' und der Länge dieses Strings. Dieser String wird nach dem Booten als Kommando interpretiert und ausgeführt. STARTUP sollte ein ZCPR-ALIAS sein, der mit dem Programm LDR alle System-Segmente (ENV, IOP, FCP, RCP, NDR, Z3T) in die Buffer lädt.

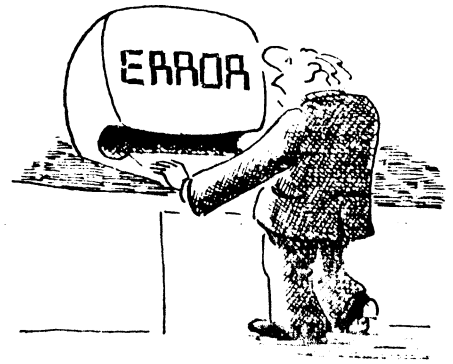
Wenn das passiert ist, kann das neue System getestet werden. Dazu einen Kaltstart ausführen (booten) und nachsehen, was im Speicher bei den zugehörigen Addressen steht. Wenn alles richtig initialisiert wurde, kann man daran gehen, ZCPR selbst zu entwickeln. Dabei müssen 2 Dateien editiert werden : Z3BASE.LIB und Z3HDR.LIB. In erstere sind die Addressen der Buffer einzutragen, nebst der Addresse des CCP, die sich ja durch MOVCPM geändert hat. In Z3HDR.LIB muß man sich die features aussuchen, die direkt in den ZCPR eingebaut sein sollen. Ich empfehle SAVE, GET, GO, JUMP. Alle anderen sind leicht in die RCP's zu stecken. Dann mit MAC assemblieren und auf die Systemspuren schreiben via DDT und SYSGEN. (Bei mir funktionierte SYSGEN nicht, wenn ZCPR33 mitspielte. Deshalb mußte ich ein kleines Programm schreiben, das einen COM-File liest und auf die Sektoren der Diskette schreibt, die vom CCP belegt sind. Der CCP ist 2K lang, macht 16 Sektoren zu 128 Byte.)
Sodann muß als erstes SYS.ENV gebaut werden. Dazu ist SYSENV.ASM zu assemblieren, in einen COM-File zu verwandeln und in SYS.ENV umzubenennen. Dasselbe gilt sinngemäß für SYSRCP, SYSFCP, SYSIOP, SYSNDR und SYS.Z3T. Dann müssen die Utilities installiert werden. Jede ZCPR3-Utility enthält einen Pointer auf den Environment Descriptor oder den Descriptor selbst. Das Eintragen macht Z3INS. In eine Datei, sagen wir SYS.INS, wird eine Liste aller Utilities eingetragen und dann, falls das gültige Environment SYS.ENV heißt, das Kommando

```
Z3INS SYS.ENV SYS.INS
```

gegeben. Das Ganze dauert, bei 70 Utilities, nur ein paar Minuten. Dann kann mit LDR SYS.ENV,SYS.NDR,... das System hochgefahren werden und das ALIAS STARTUP mit der ALIAS oder VALIAS Utility erstellt werden. Das System ist fertig.
Beim Booten müssen natürlich STARTUP und die zu ladenden Packages jeweils auf der bootenden Scheibe sein. Das erste zu ladende Package ist notwendigerweise SYS.ENV. Ansonsten spielt die Reihenfolge keine Rolle.

Auf dem Model 4 gibt es leider (noch) ein Problem. Von den Utilities führen die meisten zum Absturz des Systems, obwohl sie richtig installiert worden sind. Die Ursache ist mir völlig unklar, zumal alle Utilities einwandfrei arbeiten, wenn ich sie aus einem Debugger heraus lade und starte. Wenn einer von euch eine Idee hat, woran es liegen könnte, möge er sich bitte umgehend melden. Ich bin am Ende meiner Weisheit, habe aber die US-CP/M-Gemeinde um Rat gebeten und hoffe von dort auf Antwort.
Wenn sich einer von euch an die gewiß nicht einfache Aufgabe machen will, sein System auf ZCPR33 umzustellen, kann er ja bei mir mal fragen. Was das Model 4 angeht, kann ich ein vollständiges BIOS anbieten, inklusive deutschem Tastaturtreiber, modifizierter Kaltstartroutine und verbessertem Videotreiber (gibt Cursorposition zurück und schaltet Cursor an und aus). Ein Z80-BDOS ist ebenfalls vorhanden (Public Domain). Das Resultat ist ein System, das, neben der dramatisch erhöhten Leistungsfähigkeit; auch vollständig von Copyrights der Firma Digital Research frei ist. Außerdem liegt das BDOS und BIOS im Quellcode vor, und man kann sich seine Erweiterungen dazustricken. Bei mir, der ich noch 192K Speicher brach liegen habe, soll noch Track Buffering eingebaut werden. Vielleicht auch noch ein paar Grafikroutinen oder so was. Ich wünsche viel Spaß und Erfolg,

PS.
Das zuletzt geschilderte Problem (Utilities) ist inzwischen behoben. Es lag am Stack, der wohl entweder zu klein war, oder, weil ZCPR33 einen externen Stack im oberen Speicher erwartet, vom BIOS gelegentlich überschrieben wurde. Ich habe den Stack wieder nach ZCPR33 hinein gesetzt, denn dort war noch etwas Platz (ca. 140 Byte). Komischerweise wissen das die Utilities gar nicht, scheinbar greifen sie nicht direkt darauf zu. Für ZCPR33 gibt es eine Reihe von neuen, verbesserten Utilities. Seit ein paar Tagen "downloade" ich wie wild...



Assembler unter CP/M

Seit CP/M auf meinem Rechner läuft, beschäftigen mich fast nur noch CP/M-Programme, und ich mußte feststellen, daß es sehr, sehr viele davon gibt. Erdrückend ist vor allem die Masse der Assembler, die für mich (und noch einige andere Assembler-Freaks im Club) besonders interessant sind. Bis jetzt konnte ich mir keinen Überblick verschaffen. Jeder schwört auf einen anderen, scheinbar denjenigen, den er als ersten zu sehen bekam oder zu dem er eine Anleitung besitzt.

Welche Assembler gibt es? Diese Frage kann ich namentlich nicht beantworten, aber ich möchte die Klassen nennen, in die sich Assembler einordnen lassen, wobei ein Assembler durchaus in mehrere Klassen fallen kann. Diese Klassen sollen Euch (und mir) helfen, einen Überblick zu bekommen und mögliche Bewertungskriterien für die Leistung eines Assemblers zu bestimmen. Ich würde mich freuen, wenn Ihr mal was darüber schreibt, welchen Ihr bevorzugt und warum (bzw. was der leistet und was nicht).

a. "Paketgröße"

Der erste Leistungsunterschied ist die Anzahl der Programme, die zu einem "Paket" gehören, mit dem dann möglichst wirkungsvoll Programme erstellt werden können.

Es sind zuerst die integrierten Editor-Assembler im Stile des ZEUS aus Newdos zu erwähnen. Der Editor zum (alleinigen) Editieren von Assembler-Programmen und der Assembler bilden eine unzertrennliche Einheit. Der Assembler erstellt gleich einen lauffähigen COM-File (entspricht den CMD-Files unter Newdos). Einen Editor-Assembler habe ich zwar unter CP/M noch nicht gefunden, aber sicherlich existiert er irgendwo. Vorteile: kurze Programme können schnell erstellt und geändert werden; die Syntax wird bereits beim Eintippen geprüft (Tippfehler EX HL,DE gleich erkannt); der Quelltext wird evtl. platzsparend mit Tokens gespeichert (wie bei BASIC oder ZEUS). Nachteile: größere Programme müssen aufgeteilt werden, wobei bei jeder Änderung das ganze Programm neu übersetzt wird (das dauert!); der Editor gefällt evtl. dem Benutzer nicht; andere Editoren oder Assembler können bei Verwendung von Tokens den Text nicht bearbeiten.

Als zweite Kategorie möchte ich die Stand-alone-Assembler nennen. Hier wird ein beliebiger ASCII-Editor (unter CP/M oft WORDSTAR) zum Schreiben des Quelltextes genommen. Dieser Text wird dann von dem Assembler in einen COM-File übersetzt. Vorteile: jeder kann seinen Lieblingseditor einsetzen, um die Quelltexte zu erzeugen; sowohl beim Editieren als auch beim Assemblieren ist mehr Speicher frei, weil das jeweils andere Programm fehlt; verschiedene Assembler verstehen evtl. die gleichen Quelltexte. Nachteile: beim Eintippen wird die Syntax nicht geprüft; die Abfolge Eintippen-Übersetzen-Fehlerkorrektur-Übersetzen dauert länger, weil zwei Programme beteiligt sind; der Quelltext belegt ohne Tokens mehr Platz.

Die dritte Form sind Assembler-Linker. Wieder wird kein Standard-Editor vorgesehen, sondern es kann ein beliebiger Editor genommen werden. Dann wird der erzeugte Text von dem Assembler in eine Zwischenform übersetzt (z.B. REL-File), die wiederum von dem Linker zu einem lauffähigen COM-File umgesetzt wird. Der Linker hat vor allem die Aufgabe, mehrere verschiedene REL-Files zu einem COM-File zusammenzufügen (linken). Dabei können die REL-Files auch von Pascal, Fortran oder sonstigen Compilern stammen. Vorteile wieder ist ein beliebiger Editor verwendbar; evtl. sind Assembler und Linker austauschbar; größere Programme lassen sich gut aufteilen und dann schnell ändern, denn es muß meist nur ein Quellfile geändert und neu assembliert werden (das Linken geht dann relativ flott); Assembler-Routinen können mit Hochsprachen-Programmen gemischt werden. Nachteile: keine Syntaxprüfung beim Eintippen; es dauert lange, bis die Programme als COM-File vorliegen (editieren, assemblieren, Fehler korrigieren, assemblieren, linken); es wird noch ein File (der REL-File) mehr erzeugt, der natürlich Platz wegnimmt.

Es können natürlich auch andere Arten auftreten, aber das halte ich für unwahrscheinlich (nennt mir Gegenbeispiele!). Es kann z.B. sein, daß ein Editor-Assembler keine Syntax-Prüfung vornimmt, weil das bei Makros nicht geht, oder daß er keine Tokens erzeugt. Allerdings schwinden dann fast alle Vorteile dieser Art von Assemblern.

Zur Beurteilung sollte noch der Begriff "Turn-around-Zeit" erwähnt werden. Meines Wissens werden dabei immer zwei Zeiten betrachtet: Erstens die Zeit, die man braucht, um ein Programm zu erstellen (mit Fehlerfindung und -korrektur), und zweitens die Zeit, die bei einer Änderung des Programms gebraucht wird. Ich werde in der folgenden Zusammenfassung zwischen den Turn-around-Zeiten für kurze und lange Programme unterscheiden. Kurze Programme sind solche, deren Quelltext vollständig in den Hauptspeicher paßt. Hier ist vor allem die Erstellzeit wichtig. Längere Programme verteilen sich auf mehrere Quellfiles, wobei hier der Aufwand für Änderungen betrachtet werden muß.

Zusammenfassung "Paketgröße":

1. Editor-Assembler:
   Vorteile:
   - Syntaxprüfung beim Eintippen/Editieren
   - kurze Turn-around-Zeit bei kleinen Quelltexten
   - platzsparende Speicherung mit Tokens möglich
   - es wird nur ein Programm benötigt

   Nachteile:
   - der Editor ist fest
   - das Programm nimmt viel Speicherplatz weg
   - lange Turn-around-Zeiten bei großen Quelltexten
   - Tokens verhindern evtl. Austauschbarkeit von Quelltexten
2. Stand-alone-Assembler:
   Vorteile:
   - beliebiger Editor
   - mehr Speicher frei
   - Quelltexte oft austauschbar

   Nachteile:
   - keine Syntaxprüfung beim Editieren
   - Turn-around-Zeiten bei kurzen und langen Quelltexten hoch
   - mehr Speicherplatz verbraucht (keine Tokens)
   - immer zwei Programme (Editor und Assembler)
3. Assembler-Linker:
   Vorteile:
   - beliebiger Editor
   - mehr Speicher frei
   - evtl. austauschbare Assembler/Linker
   - kurze Turn-around-Zeiten bei langen Quelltexten
   - Mischung mit Hochsprache möglich

   Nachteile:
   - keine Syntaxprüfung beim Editieren
   - sehr hohe Turn-around-Zeiten bei kurzen Quelltexten
     immer drei Programme (Editor, Assembler, Linker)
   - immer drei Files (Quelltext, Zwischenform, COM-File)

Welche Assembler sind zu empfehlen? Das kommt ganz auf den einzelnen an. Arnulf schwört auf das Editor-Assembler-Konzept des ZEUS. Ich würde es auch auf jeden Fall für Anfänger empfehlen, denn die Syntaxprüfung ist eine große Hilfe. Es kann mit BASIC verglichen werden, denn dort ist auch eine Syntaxprüfung beim Editieren möglich und Tokens werden zur Speicherung eingesetzt.

Allerdings hat ein Editor-Assembler seine Grenzen bei Makros (hier wird die Syntaxprüfung schwierig) und bei der Entwicklung größerer Programme. Dann wird eine Strukturierung und Aufteilung nötig, die sich am effektivsten mit der Assembler-Linker-Kombination zu lösen ist. Ich kenne bisher jedoch nur einen Assembler dieser Klasse (M80). Er hat ähnliche Sprachelemente wie Modula, das auch zur Entwicklung größerer Systeme hervorragend geeignet ist bzw. diese besonders unterstützt (Modul-Konzept, Import/Export-Listen, getrennte Assemblierung/Comilierung).

Allerdings gibt es heute weitaus am meisten Stand-alone-Assembler. Sie ähneln Pascal: Eine Strukturierung ist nur bei kleinen Programmen gegeben, größere Entwicklungen (mit mehreren Programmierern) sind schwierig. Ebenso wird bei Pascal-Compilern auch meist streng zwischen Editor und Compiler getrennt (selbst Turbo-Pascal kann keine Syntaxprüfung beim Eintippen liefern).

b. Fähigkeiten des Assemblers

Mit Fähigkeiten meine ich das, was der Assembler (und evtl. Linker) kann bzw. versteht. In den Anleitungen findet Ihr diese Fähigkeiten meist unter "Pseudo-Ops", also Befehlen, die für den Assembler gedacht sind und nicht (direkt) in Maschinensprache übersetzt werden. Ich habe hier nur einige aufgezählt, die mir besonders wichtig erscheinen bzw. die ich kenne. Euch fallen bestimmt noch mehr ein; schreibt was dazu!

1. Z80 <-> 8080: CP/M wurde für Systeme mit dem Prozessor Intel-8080 entwikkelt. Erst später brachte Zilog den leistungsfähigeren Z80 heraus. Außer den zusätzlichen Befehlen entwickelte ZILOG auch eine neue Schreibweise, z.B. heißt der Befehl LD (Load) bei Intel MOV (Move). Alte Assembler kennen daher oft nur die Intel-Schreibweise, jüngere nur die von Zilog. Am besten ist eine Umschaltung wie ".Z80" und ".8080", so daß beide Schreibweisen benutzt werden können.

2. Labels: Bei den Labels halte ich es für wichtig, daß der Assembler zumindest alle Buchstaben akzeptiert (Groß- und Kleinbuchstaben) und möglichst viele Sonderzeichen zuläßt. Ob er Groß- und Kleinschreibung auch unterscheiden soll, ist eine andere Frage; ich bin dafür, allerdings sind leicht Tippfehler möglich ("Loop"<>"loop"). Auf jeden Fall sollte die Länge der Labels nicht begrenzt sein. Ein Abschneiden nach 6 Buchstaben ist nicht zeitgemäß. Einige Assembler benötigen unbedingt einen Doppelpunkt als Abschluß eines Labels. Auch hier sehe ich den Sinn nicht ein.

3. Bedingte Assemblierung (IF .. THEN .. ELSE .. ENDIF) Diese Anweisung ist vor allem bei der Entwicklung von Programmen wichtig, die auf verschiedenen Systemen oder mit unterschiedlicher Leistung assembliert werden sollen. Bei ZEUS lautet die mögliche Syntax nur "IF..ENDIF", was auch bei anderen Assemblern leider oft der Fall ist, so daß immer zwei Blöcke "IF x ... ENDIF" und "IF NOT,x ... ENDIF" erforderlich werden. Außerdem bieten die Assembler bei der Bedingung (für "x") durchaus unterschiedliche Leistungen. Komplexe Vergleiche und vom bisherigen Übersetzungsverlauf abhängige Entscheidungen sind vor allem bei Makros (s. Punkt 4) [illegible]

4. Makros: Mit Makros kann man sich eigene Befehle schneidern. Bei der Assemblierung werden die Makro-Befehle, die wie normale Befehle aussehen, durch andere "echte" Befehle ersetzt. Dies ist z.B. nötig, wenn der neue Prozessor HD64180 effektiv eingesetzt werden soll. Die bei ihm vorhandenen Befehle gehen über den Intel- und Zilog-Standard hinaus und sind keinem Assembler bekannt. Sie müssen dem Assembler erst per Makro "beigebracht" werden. Aber auch als Abkürzung für bestimmte wiederkehrende Anweisungenfolgen (z.B. viermal RLCA) sind sie nützlich. Damit die Makros voll ausgenutzt werden können sind zwei andere Fähigkeiten des Assemblers nötig: Erstens müssen Argumente, die beim Aufruf übergeben werden, gut ausgewertet werden können, indem leistungsfähige Sprachelemente wie IFs zur Verfügung stehen (z.B. sollte ein Aufruf "RLCAx 4" zum Erzeugen von vier RLCAs möglich sein). Zweitens müssen lokale Labels unterstützt werden. Wenn also in einem Programm zweimal ein Makro aufgerufen wird, in dem das Label "Loop" vorkommt, darf dieses nicht zweimal im Programm auftauchen, sondern muß durch z.B. "Loop001" und "Loop002" ersetzt werden.

5. Import/Export von Labels: Wer die Hochsprache Modula kennt, weiß sofort, was ich meine. Es geht darum, daß ich mehrere getrennte Programmstücke habe, die getrennt entwickelt werden. Dem einen Programmstück muß nicht jedes Label aus dem anderen bekannt sein, sondern es sind bei guter Aufteilung nur sehr wenige nötig. In jedem Programmstück werden zwei Listen angegeben: Export = welche Labels den anderen Programmstücken sein dürfen (falls diese welche benötigen); Import = welche Labels aus einem anderen Programmstück in diesem benötigt werden. Damit existieren klare Schnittstellen zwischen den Programmstücken, welche nun Module genannt werden. In jedem Modul kann jetzt das Label "LOOP" auftreten und es gibt keine Probleme, wenn dieses nicht von beiden ex- und importiert wird. So wird das getrennte Entwickeln von Modulen und das spätere Ändern erleichtert.

6. Trennung von Daten und Befehlen: Meist lassen sich Programme in zwei Bereiche gliedern: die Befehle (das ablaufende Programm) und die Daten (Variablen, Konstanten, Buffer...). Die Trennung dieser Bereiche kann nützlich sein, wenn z.B. zwei Module getrennt entwickelt werden. Im Endprodukt, dem COM-File, sollen aber trotzdem die Befehle der beiden Module in einem Bereich und die Daten alle in einem anderen Bereich stehen. Wenn "sauber" programmiert wird, d.h. wenn sich das Programm während des Ablaufs nicht selbst verändert, kann der Befehlsteil später in ein EPROM gebrannt werden und dort ablaufen, während alle Daten in den RAM-Bereich gelegt werden. Allerdings ist es für Assembler-Freaks schwer, sich solche "unsauberen", aber extrem effektiven Tricks abzugewöhnen (das sind eben "echte" Programmierer").

7. Funktionen und Zahlensysteme: Der Assembler sollte möglichst viele Operatoren verstehen können, sowohl mathematische (+, -, *, /, MOD) wie logische (AND, OR). Dazu kommen noch solche Operationen wie das Shiften und andere Verknüpfungen (XOR). Bei Zahlensystemen sollte eine Umstellung des Defaults möglich sein und zumindest die binäre, dezimale und hexadezimale Darstellung unterstützt werden. Daß auch ASCII-Zeichen (in Anführungsstrichen o.ä.) angegebbar sein sollten, ist fast selbstverständlich. Alle diese Dinge erleichtern das Verständnis des Quelltextes doch sehr, denn es kann sonst später schlecht nachvollzogen werden, welche Rechnungen mit dem Taschenrechner oder einer Tabelle ausgeführt wurden, um nun gerade auf diesen Wert zu kommen.

8. Daten-Pseudo-Ops: Darunter fallen Anweisungen, die Platz für Bytes, Worte, Strings oder Buffer freihalten. Meist heißen sie DB (Define Byte), DW (Define Word), DS (Define Storage), wobei DB auch für Strings gelten sollte. Leider fehlt manchmal DC (Define Constant), welches einen Speicherbereich festgelegter Länge mit einem Wert füllt. Das ist statt DS dort nötig, wo beim Starten des Programms ein bestimmter Wert in einem Buffer erwartet wird; ansonsten muß dies umständlich mit DBs nachgebildet werden.

9. Meldungen: Während des Assemblierens ist es oft sinnvoll, wenn eigene Meldungen ausgegeben werden, z.B.: "Erzeuge TRS-80-Model-I-Version". Außerdem
ist es manchmal nötig, eigene Fehlermeldungen anzuzeigen, wie "Sie haben das Label xyz nicht richtig initialisiert", denn solche semantischen (inhaltlichen) Fehler kann der Assembler nicht erkennen (er findet nur syntaktische).

10. LORG: Es kann vorkommen, daß ein Programm zuerst geladen wird (z.B. nach 100h), sich dann selbst verschiebt (oder einen Teil), und dann dort weiter ausgeführt wird. Solche Situationen muß der Assembler oder Linker besonders berücksichtigen und den entsprechenden Teil des Programms so assemblieren, als würde er von Anfang an dort stehen. Eine Anweisung LORG (oder so ähnlich) würde dem Assembler/Linker sagen: "Paß auf, dieser Teil wird nicht an dieser Stelle, sondern woanders ausgeführt." Wenn diese Anweisung nicht existiert, wird die Sache unter CP/M schwierig, denn Programme werden nicht wie unter Newdos einfach beim Laden irgendwo in den Speicher plaziert, sondern immer ab 100h abgelegt.

11. Listing: Zu einem Quelltext möchte ich auch immer einen Ausdruck haben, der den Quelltext und das daraus erzeugte Maschinensprache-Programm zeigt. Schön ist es noch, wenn Drucker-Parameter eingestellt werden können und einige Anweisungen existieren, die das Listing übersichtlicher machen, indem Seitennummern, Titel und Untertitel erzeugt werden. Anweisungen zum Unterdrücken von Abschnitten des Listings (LIST ON/OFF) sind auch nötig. Eine Auflistung der Labels mit ihren Werten und dem Vorkommen (Cross-Reference) hilft ungemein bei der Fehlersuche.

c. Utilities

Unter Utilities verstehe ich Programme, die beim Erstellen von Assembler-Programmen nicht nötig, aber sehr nützlich sind. Sie helfen vor allem bei der Fehlersuche und der Erstellung größerer Systeme.

1. Debugger: Ein Debugger sieht nicht unbedingt so aus wie der des Newdos, obwohl der schon nicht schlecht ist. Unter CP/M gibt es symbolische Debugger, bei denen auch Labels eingegeben werden können. Dazu muß der Assembler oder ein anderes Hilfsprogramm einen File mit den verwendeten Labels (Symbolen) und ihren Werten erstellen, so daß diese dann beim Debuggen des COM-Files verfügbar sind. Außerdem ist eine Disassemblierung von Speicherstellen fast schon Standard (aber oft nur in 8080- oder Z80-Mnemonics). Einzelschritt-Abarbeitung und das Setzen von Breakpoints ("Unterbrechungspunkten" hört sich nicht so gut an) ist selbstverständlich. Einige Debugger bieten noch viel mehr, aber mir kommt es vor allem auf eine einfache und übersichtliche Darstellung des Speicherinhalts an, was leider vernachlässigt wird.

2. Disassembler: Mit denen habe ich unter CP/M die wenigste Erfahrung. Ich habe einen gesehen, der halbautomatisch Datenbereiche herausfand und mit dem das Programm gleich kommentiert und den Labels vernünftige Namen gegeben werden konnten. Allerdings war die Bedienung zu umständlich und komlex, so daß ich ihn mir nicht weiter angesehen habe. Eine Kennzeichnung bzw. halbautomatische Erkennung von Daten sollte aber immer möglich sein. Evtl. kann auch ein Symbol-File für einen Debugger erstellt werden.

3. Library-Manager: Vor allem bei der Entwicklung größerer Programme ist es sinnvoll, auf schon vorhandene Routinen zurückzugreifen. Wenn diese geeignet zusammengefaßt (d.h. in einem Library-File verzeichnet) sind, sollte ein Programm sich beim Assemblieren oder Linken per Import oder anderer Anweisungen die benötigten Routinen daraus hervorholen können. Dies ist natürlich besonders bei der Assembler/Linker-Kombination der Fall. Damit ist nicht gemeint, daß der Linker einfach alle Routinen (Module) aus der Library einbindet, sondern es sollten nur die benötigten genommen werden. Dieses Prinzip wird übrigens auch bei Modula verfolgt.

4. File/Disk-Editoren: Von diesen nützlichen Helfern gibt es zwar einige und die Erstellung sollte aufgrund der Normung von CP/M auch nicht schwer sein, aber an Superzap und FED reicht kein mir bekannter heran. Alle werden mit kryptischen Kommandos bedient, die Hilfsmenüs sind immer dann nicht sichtbar, wenn ich sie benötige, und das Ändern eines Sektoren scheint unmöglich zu sein. Glücklicherweise brauche ich die Editoren nur, wenn eine Diskette wirklich böse zerschossen ist, was aber sehr selten vorkommt.

d. Bedienung

Leider mußte ich feststellen, daß die Bedienung der Programme unter CP/M noch chaotischer als unter Newdos ist. Oft müssen mit dem Aufruf des Assemblers (oder anderen Programmen) gleich alle Parameter eingegeben werden. Wenn dies nicht geschieht, passiert auch nichts und es kommen selten ordentliche Fehlermeldungen. Einige Programme melden sich einfach mit einem Prompt (meist "*") und harren dann der Dinge, die da kommen (eingegeben werden) mögen. Leider erfährt man selten, was für eine Eingabe erwartet wird. Optionen müssen oft mit kryptischen Abkürzungen wie "/S" oder "-S" eingegeben werden und haben nie die erwartete Wirkung, falls die überhaupt bekannt ist.

Heutzutage (1988, der Begriff "Benutzerfreundlichkeit" ist schon länger in aller Munde) ist ein Hilfesystem oder eine Menüsteuerung nicht utopisch. Zwar bieten dies viele Programme, aber es beschränkt sich dann auf eine Auflisting aller Kommandos. Das ist ungefähr so effektiv wie ein Fahrkartenautomat mit Tasten ohne Aufschrift: Jeder darf ausprobieren, was wohl bei dieser oder jener Taste (bzw. diesem oder jenen Kommando) passiert. Anzahl und Form der Parameter werden sowieso nicht angegeben und abgefragt werden sie auch nicht. Die Meldungen "Syntax Error" oder "Command Error" scheinen als einzige im Wortschatz der Programmierer vorhanden zu sein. Na, dann gute Nacht!

Gerald Schröder

Freiheit für den Bus!
(des Genie IIs)

Das Genie IIs ist eigentlich nicht mehr als ein normaler TRS-80, aber mit einem Bus und mehreren Steckkarten. Dadurch ist die Erweiterung des Systems zwar leicht, aber die Signale auf dem Bus sind ganz auf die Nachbildung eines TRS-80 zugeschnitten und halten sich nicht im geringsten an den eigentlichen Standard ECB. Das kann ich zwar nicht ändern, aber mir stinkt es gewaltig, daß einige Signale (fast) nie benötigt werden. Die kann ich besser verwenden. Aber zuerst gilt es, diese Signale von ihren ursprünglichen Verbindungen abzuschneiden und lokal (auf den Karten) zu ersetzen.

Ich habe bei mir mit dem Signal $\overline{32C}$ angefangen. Es dient zur Umschaltung auf die 32-Zeichen-Darstellung, die wohl nie jemand benutzt. Eigentlich wäre 64C als Bezeichnung besser gewesen, denn High bedeutet, daß 64 Zeichen angezeigt werden. Aber irgendwie scheinen die Hardware-Freaks auf invertierte Logik zu stehen, weiß der Teufel (bzw. Uwe Böker) warum.

Dieses Signal tritt nur auf zwei Karten auf: Auf der I/O-Karte wird es erzeugt (durch Schreiben in Port FF) und kann abgefragt werden (durch Lesen von Port FF). Auf der Video-Karte wird eine Umschaltung damit vorgenommen.

Der Umbau:
1. I/O-Karte:
   a) U10, Pin 4 -> +5V (schätzungsweise mit Widerstand 1kΩ)
   b) U11, Pin 14 rausbiegen
      alternativ: Leiterbahn zum Bus-Signal A8 abtrennen
2. Video-Karte:
   a) U8 rausnehmen (LS157)
   b) an der Fassung von U8 (unten) folgende Verbindungen herstellen:
      Pin 12 <-> 13
      Pin 3 <-> 4
      Pin 6 <-> 7

Neben dem LS157 ist jetzt auch U7, Pin 2-7 frei. Das freie Bussignal wird bei mir dazu benutzt, um ein Signal $\overline{0-32R}$ von der CPU- zur I/O-Karte zu leiten. Dieses Signal wird benötigt, um bei Einsatz des Bankers zu erkennen, ob jetzt der Memory-mapped-I/O-Bereich oder eine höhere Bank angesprochen wird.

Was Ihr jetzt mit dem freien LS157 machen sollt? Jedenfalls nicht wegwerfen! Evtl. zeige ich Euch im nächsten Info, wie man damit die Memory-Map auf Portbetrieb umstellt (nützlich unter CP/M). Das nennt man dann Hardware-Recycling.

Gerald Schröder

[illegible] und das elfte Gebot

Ich war so unvorsichtig, das elfte Gebot (Du sollst nicht schwärmen...) nicht zu beachten und habe Gerald Schröder von meiner Solid-State-Floppy erzählt. Zur Belohnung hat er mir natürlich sofort einen Artikel auf's Auge gedruckt, damit sich jeder selbst ein Bild über den Sinn oder Unsinn von sowas machen kann.

Die Karte ist im Grunde eine ganz gewöhnliche Speichererweiterung, aber sie hat ein paar kleine Besonderheiten, die sie sehr interessant machen. Einmal muß man im Rechner absolut nichts umbauen und sie ist keine Erweiterung auf, sondern up max. 256kB, d.h. sie ist unabhängig von dem schon vorhandenen Hauptspeicher. Die völlige Systemunabhängigkeit wird dadurch erreicht, daß die Karte über Ports angesprochen wird und zwar fast wie ein richtiger Floppy-Controller. Man gibt also über einen Port die 'Spurnummer' aus und auf einem anderen die 'Sektornummer', dann werden die Daten einfach per IN oder OUT hin und her geschoben, um die interne Adressierung auf der Karte muß man sich also überhaupt nicht kümmern. Wenn man will, kann man sogar seinen PC oder Atari damit beglücken. Bestückt wird sie wahlweise mit statischen RAMs (akkugepuffert) oder EPROMs, wobei der Clou an der Sache der ist, daß man die Karte auch gleich als EPROMMER verwenden kann. Leider braucht man dazu aber ein spezielles Programm, von dem ich nur einmal unter CP/M was gehört habe. Da bei mir der Inhalt aber sowieso nur ein paar Tage lang konstant bleibt, habe ich mich lieber gleich für die RAM-Version entschieden. Der große Vorteil ist, daß ich gleich beim Einschalten alle meine wichtigen Utilities griffbereit habe und das Laden völlig geräuschlos und sehr schnell geht. Wenn man sich erst mal an sowas gewöhnt hat, möchte man es nicht mehr hergeben. Stellt Euch vor, Ihr arbeitet viel mit Superzap, dann werdet Ihr es wahrscheinlich auf fast allen Disketten irgendwo haben, damit es immer zur Hand ist. Das braucht nicht nur viel Platz, auch wenn mal ein Zap fällig ist oder es gegen ein anderes Utility ausgetauscht werden soll, (bei Superzap wohl weniger, aber bei anderen Programmen) müßt ihr alle Disketten durchackern, wo überall eine Kopie ist. Ich brauche das nur einmal auf der RAM-Floppy zu tun und bin fertig. Am schönsten ist es aber, wenn man dort Programme hat, die viel mit Overlays arbeiten. Früher habe ich immer geflucht, wenn dauernd die Floppy angelaufen ist, heute merkt man das Nachladen überhaupt nicht mehr.

Erst habe ich Euch den Mund wässrig gemacht, jetzt kommt eine kleine Abkühlung. Nichts gegen die c't, aber dieses Projekt teilt das Schicksal so vieler: es kann (wird) größere Probleme geben. Sowohl bei Helmut Bernhardt als auch bei mir hat es längere Zeit gedauert, bis die Karte lief. Der Aufbau selber ist nicht weiter schlimm, aber irgendwo muß sie sehr empfindlich sein, wie es aussieht kann schon ein IC vom 'falschen' Hersteller das Aus bedeuten. Wer sich eine längere Fehlersuche nicht zutraut, oder keinen kennt, der einen kennt..., der sollte doch lieber vorsichtig sein.

Wem das noch immer nicht reicht, der soll sich mal ausrechnen, was das kostet. Für die Leerkarte+Porto muß man DM 78 anlegen, für die CMOS-RAMs (256kBit) zwischen 20 und 40 Mark, je nachdem wo man sie kauft (EPROMs ca. 10.-). Die Fassungen und das ganze Kleinzeug kommen so ungefähr auf 50.-, d.h. für statische 256kB kommt man auf ca. 300 Eierchen.

Hat noch jemand bis hierher weitergelesen ? Gut, wenn noch Interesse besteht stehe ich gerne zur Verfügung. Der Artikel in der c't war übrigens in der Nummer 5/86, bei Bedarf schicke ich Euch gerne eine Kopie.

Bei mir läuft die SSD unter CP/M und Arnulf Sopp hat mir mal geschrieben, daß er an einem Treiber für GDOS/NEWDOS bastelt; die Softwarefrage ist also keine mehr.

Was tun die Mutigen, wenn es heißt 'Freiwillige vor' ? Richtig, sie machen Platz, damit die Freiwilligen durchkommen. Also, wer meldet sich ?

Alexander Schmid

Prof 180x und mc68000-ECB

Helmut Bernhardt

Der mc68000-ECB aus mc 1-3/87 ist ein Coprozessor-Board mit einem Motorola 68000 Prozessor auf einer Europakarte für den ECB-Bus. Diese Karte kann in jedem DMA-fähigen ECB-Bus-Computer eingesetzt werden, wobei dann abwechselnd oder auch gleichzeitig Z80- und 68000-Software gefahren werden kann. Im einfachsten Fall kann das 68000-Board aber auch nur dem Z80 eine bis zu 2MB große RAM-Floppy zur Verfügung stellen.

Wenn der Computer im 68000-Modus läuft, spielt der Z80 (oder im Prof 180x der HD64180) I/O-Knecht für den 68000. Der 68000 bedient sich des BIOS des CP/M 80. Der 8Bit-Prozessor wartet nur auf Aufgaben, die ihm der 68000 stellt.

Die Kommunikation zwischen den beiden CPUs ist für beide unterschiedlich zu handhaben. Der 8Bit-Prozessor kann dem 68000 einen 1Byte-Befehl in ein Latch schreiben, während der 68000 per DMA in den Speicher des Wirts-Computers schreiben und auch daraus lesen kann. In den 16MB-Adreßraum des 68000 ist ein Fenster von 2MB Größe gelegt, in dem die vermeintlichen 64KByte des Z80-Systems erreichbar sind. Zur Adressierung im Z80-System werden dann die Adressen A1-A16 des 68000 auf A0-A15 des ECB-Bus geschaltet und BUSAK* low gezogen. Außerdem werden auch die Z80-kompatiblen Signale MERQ*, IORQ*, RD* und WR* erzeugt und der Datentreiber des 68000-Boards zum ECB-Bus wird freigegeben (für D8-D15 des 68000).

Der Prof 180x hat aber nicht nur 64KByte RAM; zur Adressierung innerhalb der 512K des Prof 180x müssen noch die Adressen A16-A18 mit festen Pegeln für den Bereich 40000H-4FFFFH erzeugt werden. In diesem physikalischen Adreßbereich liegt im Prof die TPA, in der ein Kommunikationsprogramm den Datenaustausch mit dem 68000 erledigt und von wo der 68000 sich Daten abholt und wohin er seine Daten an das 8Bit-System überträgt. Das läßt sich noch recht einfach mit einem 74LS367 Tristate-Treiber, der A16-A18 des ECB-Bus bedient, auf dem 68000-Board erledigen. Dieser Treiber wird mit den Pins 1, 8 und 16 auf IC13, 74LS367 aufgelötet. Die Pins 3, 5 und 7 werden mit A18, A17 und A16 des ECB-Bus verbunden und an Pin 2 wird ein Widerstand von 4k7 gegen +5V angeschlossen. Die Pins 4 und 6 werden an GND gelegt.

Das eigentliche Problem war weder vorher abzusehen noch durch einen einfachen Patch zu beheben - so schien es zumindest im ersten Augenblick. Die DMA-Fähigkeit des Prof 180x brücksichtigt zunächst die Steuerung der Treiber zum ECB-Bus. Durch ein PAL 14L8 ist die Bussteuerung ohne den sonst für die Berücksichtigung der DMA-Fähigkeit nötigen TTL-Aufwand mit folgender PAL-Gleichung für das Steuersignal des Datentreibers gelöst:

```
/DIR = /IOE * /RD * /ADR * A7 * /A5    ;Treiberrichtung beim Lesen interner Ports immer
                                       ;zum Bus hin
     + /ME * /RD * A19                 ;Zum Bus beim Lesen der internen 512K RAM, A19 ist
                                       ;ein im gleichen PAL erzeugtes Signal zur Unter-
                                       ;scheidung zwischen internem und externem RAM
     + RD * LIR * BUSAK                ;zum Bus auch, wenn weder ein Interrupt-Acknowledge
                                       ;noch DMA-Zugriff noch eine Schreiboperation erfolgt
```

Die Treiber für den Adreßbus und Steuerbus werden einfach durch den Pegel von BUSAK* gesteuert. Wenn BUSAK* low aktiv ist, treiben diese die entsprechenden Signale vom Bus zur CPU-Karte.

Damit kann ein externer DMA-Controller zwar auf das CPU-Board zugreifen aber noch nicht den Speicher lesen und schreiben, denn das flexibel bei jeder CPU-Taktrate funktionierende Timing der RAM-Steuersignale RAS*, MUX und MUX* und CAS*, das aus HD64180-spezifischen Signalen abgeleitet wird, kann von einem DMAC nicht unterstützt werden. Die im Prof 180x verwendete Schaltung (Abb.2) zeigt, daß aus dem MERQ* mit der nächsten steigenden Flanke des Systemtaktes die Signale MUX und MUX* erzeugt werden und daß ein vom Systemtakt abhängiges HD64180-spezifisches Signal E für das Timing des CAS*-Signals herangezogen wird. Ein asynchron arbeitender DMAC kann diese Signale garnicht bedienen, E ist kein ECB-Bus-Signal und der CPU-Takt ist kein Tristate-Signal, das beim DMA-Zugriff vom DMAC bedient werden kann.

Es galt nun, möglichst unter Beibehaltung des eleganten und flexiblen Timings der RAM-Steuerung durch den HD64180, einem externen DMAC eine weitere Möglichkeit zu schaffen, ausschließlich mit MERQ* (=ME* beim HD64180) und Verzögerungsgliedern die getimeten RAM-Signale zu erzeugen. Wenn dies z.B. mit Gatterlaufzeiten gemacht wird, sind dann allerdings der Bandbreite möglicher Taktraten des DMAC engere Grenzen gesetzt. Die in Abb.3 dargestellte Schaltung erlaubt das durch BUSAK* gesteuerte Umschalten zwischen ursprünglichem Timing des Prof 180x und über Gatterlaufzeiten geregeltes Timing bei Kontrolle durch den externen DMA-Controller. Die Anzahl Gatter für die Verzögerung gilt für einen mit 8MHz getakteten 68000. Bei Verwendung entsprechend schneller RAMs und EPROMs auf dem 68000-Board macht diese Schaltung aber auch bei einem mit 16MHz getakteten 68000 mit.

Es sei noch erwähnt, daß im DMA-Betrieb MUX und CAS* nicht gleichzeitig kommen, wie die Schaltung beim ersten Hinsehen erwarten läßt. Das im PAL aus E erzeugte CAS* ist gegenüber E nochmals um die Gatterlaufzeit des PALs verzögert.

Die Schaltung sieht auf den ersten Blick nach sehr viel Flick- und Sägearbeit am CPU-Board aus. Das ist aber durch Verwenden eines kleinen Adapterplatinchens, das in den Sockel von Z9, 74S74 gesteckt wird und Z9 selbst mit aufnimmt, nicht der Fall. Außer dem 74S74 bringt das Platinchen noch ein 74LS367 (6 Tristate-Treiber) und ein 74LS368 (6 invertierende Tristate-Treiber) unter. Über den Sockel von Z9 werden fast alle benötigten Signale (MERQ*, CLK, RFS*) dem Platinchen zugeführt und auch die jeweils erzeugten Signale MUX und MUX* abgeleitet. BUSAK* muß durch freie Verdrahtung von J9 und E von Pin60 des HD64180 bezogen werden. Das Ausgangssignal E wird an Pin7 des PALs gelegt. Damit die bisherige Verbindung zwischen Pin60 des HD64180 und Pin7 des PALs unterbrochen wird, muß Pin7 des PALs aus der Fassung gebogen werden.

Eine ähnliche Herleitung von MUX und/oder MUX* weisen einige moderne TRS80-Kompatible Computer auf, bei denen eine Taktumschlatung zwischen 1,77MHz und einer anderen, deutlich höheren Geschwidigkeit möglich ist. Leider ist all diesen Geräten ein nicht DMA-fähiger Bus konstruiert worden. Ein externer DMAC kann hier nur direkt mit den entsprechenden Signalen der CPU verbunden werden. Unverständlicherweise werden dann aber die Treiber für Adreß- und Steuerbus doch durch ein invertiertes BUSAK* oder BUSREQ* freigegeben. Offensichtlich haben ganze Generationen von Kostrukteuren von einem Urtrottel abgeguckt. Deshalb muß man dann zusätzlich noch die Freigabe dieser Treiber mit GND anstelle mit BUSAK bzw. BUSRQ einrichten.

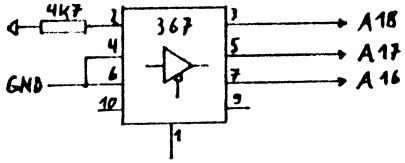


Abb.1: Erzeugen der richtigen Pegel von A16-A18 für die Adressierung der TPA beim Prof 180x bei physikalisch 40000H-4EFFFH

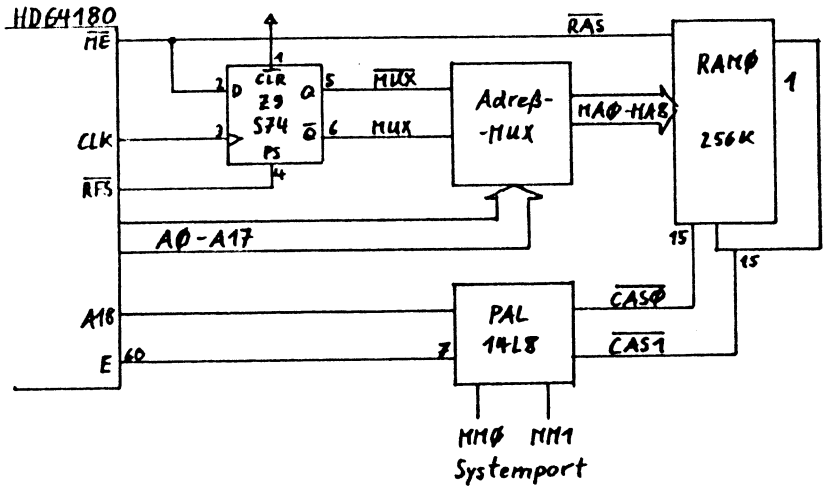


Abb.2: das für jede Taktrate funktionierende Timing der RAM-Steuersignale im Prof 180x

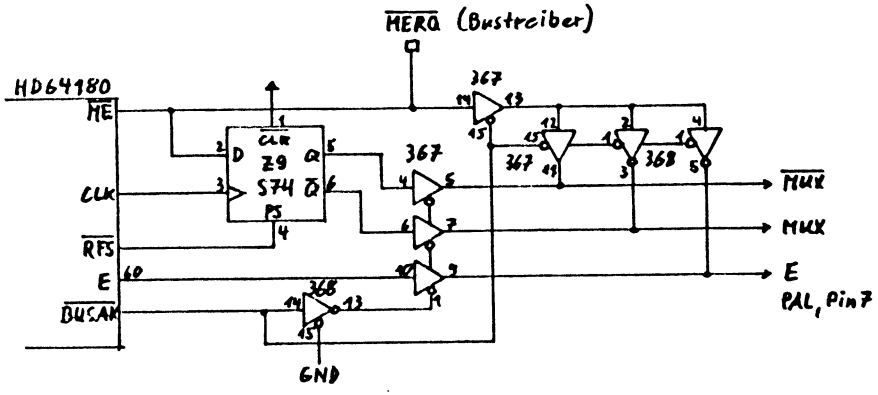


Abb.3: auf einer Adapterplatine im Sockel von Z9, 74S74 wird die Umschaltung zwischen den im Prof 180x erzeugten und den aus MERQ* abgeleiteten RAM-Steuersignalen untergebracht, BUSAK* regelt die Umschaltung

TRS 80 Japan Modell aufbohren

auf
1,77/3,55MHz Taktumschaltung
CP/M-Fähigkeit
256K gebanktes RAM

H. Bernhardt

Aufgrund eines mir zunächst üppig erschienenen, sich aber nachträglich als keinesfalls dem Aufwand adäquat erwiesenen Honorars ließ ich mich breitschlagen, in einem vermeintlichen TRS 80 Modell 1 einen 256K-Banker, einen CP/M-Banker und eine einfache Taktumschaltung einzubauen. Bei übernahme des Gerätes stellte sich aber heraus, daß es sich um ein Japan-Modell handelt, bei dem alle meine Unterlagen zum TRS 80 unbrauchbar sind. Es galt also zunächst, dessen Schaltung zu analysieren, um dann die nötigen Änderungen für die Erweiterungen vorzunehmen und zu sehen, an welchen Punkten die benötigten Signale abzugreifen sind. Solches hatte ich vor Jahren beim Komtek 1 schon mal durchgeführt, so daß mir der Mut nicht ganz verlorengegangen war.

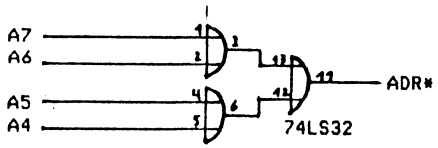
Eventuell besitzt hier im Club noch irgendjemand diese Variante des TRS 80 (vielleicht, ohne es selbst zu wissen, denn äußerlich sehen Original und Fälschung gleich aus), dem die von mir extrahierten Informationen etwas nützen. Deshalb will ich es riskieren, einige Seiten des kostbaren Platzes in diesem Info für diesen Zweck zu verbrauchen. Den 256K-Banker habe ich vor längerer Zeit schon einmal im Club 80 Info beschrieben, so daß hier nur noch dessen Einbau und Anschluß am CPU-Board breitgetreten werden soll.

Den CP/M-Banker (für den es ein angepaßtes CP/M 3.0 BIOS gibt, zu dem auch die Source-Codes vorliegen) habe ich vor noch längerer Zeit zusammen mit Christof Ueberschaar entwickelt und im Info des mittlerweile eingegangenen Bremerhavener TRS 80/GENIE-Clubs veröffentlicht. Um die zahlreichen Mitglieder des Club 80, die dort auch mal Mitlied waren, nicht noch einmal damit zu langweilen, soll nur die Schaltung dieses CP/M-Bankers und eine Skizze mit den Anschlußpunkten auf dem Platinchen vorgestellt werden. Wer darüber hinaus Interesse an dem Artikel hat, kann sich mit 1,90DM Rückporto im Brief bei mir melden.

Der CP/M-Banker setzt voraus, daß im Grundgerät 64K oder 256K RAM vorhanden sind. Durch Ausgabe des Bytes 00H an den Port 50H wird der ROM-Bereich abgeschaltet und im Bereich 0000H-37DFH wird RAM verfügbar gemacht. Diese Einstellung läßt sich durch Software nicht wieder rückgängig machen. Nur ein RESET stellt wieder die urspünglichen ROMs bereit. Durch ein kleines Maschinenprogramm kann der ROM-Inhalt nach oben kopiert werden, dann der Umschaltbefehl ausgeführt und schließlich der ROM-Inhalt an seine ursprüngliche Adresse zurückgeschoben werden. Damit ist das Level 2 ROM weich geworden und kann beliebig gepatcht werden.

Wenn an den Port 50H der Wert 40H (D6=1) ausgegeben wird, ist anschließend auch noch der memory mapped I/O-Bereich 37E0H-3FFFH nach F7E0H-FFFFH verlegt worden, so daß ein fast 62K großer RAM-Bereich ab 0000H vorliegt, womit CP/M leben kann.

Es sei noch gesagt, daß die Verwendung des Ports 50H (redundant decodiert auf 50H-5FH) bei gleichzeitig vorhandener HRG1B wegen deren Verschleiß an Portadressen (00H-7FH) eine feinere Decodierung der HRG1B erfordert. Prinzipiell genügt es, anstelle von A7 (an die HRG1B) ein aus A7 und A6 OR-verknüpftes Signal an die Karte zu führen. Damit reduziert sich deren Portadreßhunger auf den Bereich 00H-3FH. Da dann aber ohnehin ein zusätzliches IC 74LS32 nötig ist, läßt sich die HRG1B auch noch auf die Portadressen 00H-0FH beschränken.



anstelle von A7 an die HRG1B

Aber zurück zum Anschluß der Banker an das CPU-Board des Japan-Modells. Zunächst sollen die Änderungen besprochen werden, die den Einsatz von 64K bzw.256K RAM-Chips auf dem CPU-Board ermöglichen.

Auswechseln der 4116-RAMs

Dafür müssen die unsinnigerweise eingelöteten 4116-RAMs entfernt werden. Es werden alle Pins aller RAMs (Z15-Z22) möglichst hoch abgekniffen und die verbleibenden Stümpfe der Pins einzeln ausgelötet und schließlich die Lötlöcher mit einer Entlötpumpe freigemacht.

Vor dem Einlöten der DIL16-Fassungen für die neuen RAMs sind noch ein paar Änderungen nötig, um die +12V-Versorgung von den Pins 8 der RAMs und die -5V-Versorgung von den Pins 1 der RAMs zu entfernen und die +5V-Versorgung von den Pins 9 zu den Pins 8 der RAMs zu verlegen, und um an die Pins 9 ein neu zu erzeugendes gemultiplextes Signal MA7 (und bei 256K-Chips an die Pins 1 ein MA8) zu führen. Dafür sind folgende Arbeiten durchzuführen:

Alle Kondensatoren (100nF) oberhalb und unterhalb der RAMs, die mit einem Beinchen Kontakt mit den Pins 1 oder 9 der RAMs haben, werden ausgelötet.
Die bestückungsseitige, breite, die Pins 1 der RAMs verbindende Leiterbahn wird auf Höhe der 330hm-Widerstände (R101) durchtrennt (-5V an den Pins 1 der RAMs).
Die lötseitige, breite, um RAM Z15 und die 33-Ohm-Widerstände herum zu Pin 16 von Z14, 74LS157 führende Leiterbahn wird auf Höhe der 33-Ohm-Widerstände (R101) durchtrennt (+5V an den Pins 9 der RAMs).
Die vom Kondensator C15 auf der Lötseite zu Pin 8 von Z19 (RAM) verlaufende, breite Leiterbahn wird neben C15 durchtrennt (+12V an den Pins 8 der RAMs).
Eventuell müssen lötseitig noch einige zusätzliche Kondensatoren von 100nF über die Pins 8 und 16 der RAMs gelegt werden. Optimal ist ein 100nF-Kondensator je RAM. Diese Kondensatoren sollten nach dem Einlöten der DIL16-Sockel für die neuen 4164- bzw. 41256-RAMs lötseitig an die Pins 8 und 16 der Sockel gelötet werden. Es ist dafür zu sorgen, daß keine Kontakte zu anderen Pins entstehen (Isolierhülsen auf die Beinchen der Kondensatoren).
Für den Einsatz von 4164-RAMs mit 7Bit-Refresh reicht es, wenn A15 von Z48, Z80-CPU, Pin 5 an Pin 13 von Z13, 74LS157 und A14 von Z48, Z80-CPU, Pin 4 an Pin 14 von Z13, 74LS157 verdrahtet wird und der Pin 12 von Z13, 74LS157 mit den untereinander verbundenen Pins 9 der RAMs verdrahtet wird.

Für den Einsatz von 4164-RAMs mit 8Bit-Refresh oder 41256-RAMs (mit grundsätzlich 8Bit-Refresh) muß A7 von Pin 3 von Z14, 74LS157 abgetrennt werden. Die lötseitige Leiterbahn zu Pin 3 von Z14 wird direkt neben dem Pin 3 durchtrennt. An diesen Pin 3 wird dann A14 von Z48, Z80-CPU, Pin 4 verdrahtet. A15 von Z48, Z80-CPU, Pin 5 wird mit Pin 13 von Z13, 74LS157 und Pin 12 von Z13 wird mit den Pins 9 der RAMs verdrahtet. A7' von einem 8Bit-Refresh-Generator (z.B. auf dem 256K-Banker) wird an Pin 14 von Z13, 74LS157 gelegt.

Im Expansion-Interface des Japan-Modells (schwarzes Blechgehäuse) werden alle RAMs 4116 aus den Fassungen gezogen und verworfen.
Neben den RAMs befinden sich 2 Stück 74LS244-Treiber. Bei beiden ICs wird die bestückungsseitig von den Pins 1 wegführende Leiterbahn direkt neben den Pins 1 durchtrennt.
Auf der Lötseite wird die die Pins 19 der 74LS244-ICs verbindende Leiterbahn neben der Durchkontaktierung beim 74LS32-IC durchtrennt. Bei beiden 74LS244-ICs wird eine Verbindung zwischen den Pins 1 und 19 hergestellt.
Auf der Bestückungsseite wird an einem der 74LS244-ICs ein Widerstand von 2k2 von Pin 1 nach Pin 20 gelegt.

Taktumschaltung

Nach dieser Änderung sollte bei Verwendung von 4164-RAMs mit 7Bit-Refresh der Computer normal mit 48K RAM arbeiten. Da die langsamen RAMs im Expansion-Interface nun nicht mehr benutzt werden, läßt sich eine Speed-Umschaltung zwischen 1,77MHz und 3,55MHz einbauen. Die hier mit einem einfachen Umschalter realisierte Variante schaltet einen der beiden Takte zur CPU durch. Das Umschalten ist also nicht während des Betriebs möglich (sollte es zumindest nicht, hier hat es in den meisten Fällen ohne Systemabsturz geklappt). Für diese Umschaltung sind folgende Operationen durchzuführen:

Auf der Bestückungsseite
wird die Leiterbahn, die zwischen Pins 13 und 14 von Z65, 74LS92 herauskommt und zwischen Pins 2 und 3 von Z66, 74LS367 verschwindet, durchtrennt.
Die Pins 12 und 13 von Z63, 74LS74 werden miteinander verbunden.
Die Pins 8 und 9 von Z65, 74LS92 werden an die Außenkontakte eines Umschalters gelötet. Der Mittelkontakt des Umschalters wird mit Pin 14 von Z66, 74LS367 verbunden (möglichst kurze Leitungen).

Auf der Lötseite
wird die Verbindung von Pin 12 von Z63, 74LS74 zur Durchkontaktierung unter dem IC durchtrennt.

CP/M-Banker einbauen

Für den Anschluß des CP/M-Bankers sind die in Abb.3 dargestellten Änderungen in der Adreßdecodierung der unteren 16K auf dem CPU-Board vorzunehmen. Im einzelnen bedeutet dies folgende Arbeiten:

Bestückungsseite:
Die Leiterbahn, die zwischen den Pins 2 und 3 von Z65, 74LS92 herauskommt und zwischen den Pins 11 und 12 von Z64, 74LS32 verschwindet, wird durchtrennt.
Der Pin 3 von Z66, 74LS367 wird mit Pin 13 von Z64, 74LS32 verbunden.
Der Pin 6 von Z64, 74LS32 wird mit Pin 12 von Z44, 74LS00 verbunden.
Die von Pin 12 von Z49, 74LS244 zur ca. 2 cm entfernten Durchkontaktierung führende Leiterbahn wird durchtrennt (A14).
Die daneben verlaufende (zwischen den Pins 11 und 12 von Z49, 74LS244 herauskommende) Leiterbahn wird durchtrennt (A15).
Die an Pin 10 von Z61, 74LS139 führende Leiterbahn wird neben dem Pin durchtrennt.

Lötseite:
Die Leiterbahn von Pin 11 von Z64, 74LS32 zur Durchkontaktierung bei Pin 8 von Z64, 74LS32 wird durchtrennt.
Die Leiterbahn von Pin 13 von Z44, 74LS00 zur Durchkontaktierung wird durchtrennt.
Die Verbindung zwischen den Pins 2 und 6 von Z64, 74LS32 wird durchtrennt.
Die Leiterbahn von der Durchkontaktierung auf Höhe der Pins 5 und 10 von Z64, 74LS32 zur Durchkontaktierung auf Höhe der Pins 3,4,11 und 12 von Z44, 74LS00 wird durchtrennt.

Auf dem CP/M-Banker:
Das IC Z10, 74LS32 wird nicht bestückt und der Jumper J5 bleibt offen. Die Jumper J1, J2, J3 und J4 werden gelegt.
Die Anschlüsse SYSRES*, ROM3*, ROM4*, ROM3'*, ROM4'* sowie 37EX* bleiben unberücksichtigt.

Der CP/M-Banker wird gemäß folgender Tabelle mit dem CPU-Board verbunden:

Signal auf dem CP/M-Banker	Anschluß auf dem CPU-Board		
A4	Z68	74LS244	Pin5
A5	"	"	16
A6	"	"	3
A7	"	"	18
A8	Z49	74LS244	Pin14
A9	"	"	16
A10	"	"	18
A11	"	"	3
A14	"	"	12
A15	"	"	9
12-14(16)K*	Z61	74LS139	Pin15
OUT*	Z30	74LS367	Pin7
RESET*	Z46	74LS04	Pin12
NMI*	Z45	74LS02	Pin10
ROM1*	Z60	74LS11	Pin8
ROM2*	Z61	74LS139	Pin6
RAM*	Z64	74LS32	Pin11
KB*	Z61	74LS139	Pin10
ROM1'*	Z42	ROM A	Pin20 diese Pins werden aus
ROM2'*	Z43	ROM B	Pin20 der Fassung gebogen
RAM'*	Z49	74LS367	Pin15
KB'*	Keyboard-Stecker Pin10		
A14'	an die Durchkontaktierung, deren Verbindung zu Pin 12 von Z49, 74LS244 durchtrennt wurde		
A15'	an die Durchkontaktierung direkt daneben		

256K-Banker einbauen

Um den Anschluß des 256K-Bankers vorzubereiten, wird auf Z14, 74LS157 noch ein weiteres IC 74LS157 mit den Pins 1, 8, 15 und 16 huckepack-gelötet; alle anderen Pins dieses ICs werden waagerecht abgebogen. An dieses IC Z14' werden die Signale A7', A15', A16 und A17 des 256K-Bankers angeschlossen.

Signal des Bankers	Pin von Z14'
A7'	Pin14
A15'	" 13
A16	" 11
A17	" 10

Die Pins 9 der RAMs werden mit Pin 12 von Z14' und die Pins 1 der RAMs mit Pin 9 von Z14' verbunden. Die eventuell beim Umbau auf 64K RAM (event. mit CP/M-Banker) hergestellte Verbindung zwischen Pin 12 von Z13 und den Pins 9 der RAMs wird wieder entfernt. Das anstelle der durchtrennten Leiterbahn (A7 der Z80-CPU) an Pin 3 von Z14 geführte Signal A14 (Pin 4 der Z80-CPU) bleibt angeschlossen.

Die Eingangssignale für den 256K-Banker werden an folgenden Punkten auf dem CPU-Board entnommen und an die entsprechend bezeichneten Punkte des 256K-Bankers geführt (freie Verdrahtung).

Signal	CPU-Board IC	Typ	Pin	256K-Banker Anschlußpunkt
D0	Z23	245	17	g
D1	"	"	18	f
D2	"	"	15	e
A0	Z68	244	9	8
A1	"	"	12	9
A2	"	"	7	1
A3	"	"	14	2
A4	"	"	5	10
A5	"	"	16	4 und 5
A6	"	"	3	6 und m
A7	"	"	18	7 und n
A15	Z49	244	9	h
SYSRES*	siehe unten			a
OUT*	Z30	367	7	l
RFSH*	Z49	Z80-CPU	28	o

Bei "nur" 256K RAM werden die Eingangssignale D3 und D4 sowie CASEN* und die Ausgangssignale CASEN1* bis CASEN4* nicht angeschlossen.
Das Signal SYSRES* wird auf dem CP/M-Banker aus RESET* und NMI* erzeugt und kann von dort bezogen werden. Wenn der CP/M-Banker nicht eingebaut wird, muß RESET* (von Z46, 74LS04, Pin 12) oder NMI* (von Z45, 74LS02, Pin 10) stattdessen verwendet werden.

Bei den spartanischen Platzverhältnissen im TRS 80 Modell 1 muß beim Einbau von Erweiterungen von vornherein geplant werden, wo die zusätzlichen Platinchen in der Schachtel Platz finden. Das CPU-Board liegt mit der Bestückungsseite nach unten im Gehäuse. Erweiterungen, die darunter liegen sollen, müssen also von der Bestückungsseite her angeschlossen werden. Platz ist unter dem CPU-Board nur in der hinteren Hälfte des Gehäuses vorhanden. Die Anschlußdrähte sollten also gleich so lang gewählt werden, daß diese Einbaulage möglich ist. Allerdings ist in digitalen Schaltungen prinzipiell das Einhalten kurzer Leitungslängen anzustreben. Man sollte also auch nicht zu großzügig mit der zu verwendenden isolierten Litze umgehen.

Obwohl so mancher rechtschaffener Standard-Z80 die 3,55MHz besser verträgt als mancher A-Typ, sollten doch die paar Mark und funfzig für einen auf 4MHz ausgelegten Z80 investiert werden. Dagegen ist die Investition für die neuen RAM-Chips bei den heutigen Preisen durchaus die Überlegung wert, ob man das dem alten Möbel noch angedeihen lassen will.

Außerdem ist (auch wenn für einen potentiellen Nachbauer der Aufwand für die Analyse des Japan-Modells entfällt) der Arbeitsaufwand und die Wahrscheinlichkeit, einen Fehler zu machen, recht hoch. Der eigene Performance-Ehrgeiz sollte also ruhig überschlafen werden. Auch die CP/M-Fähigkeit ist bei einem nach wie vor auf 64*16 Zeichen beschränkten Bildschirm begrenzt tauglich und Tasten für deutsche Umlaute hatte ein TRS 80 Modell 1 noch nie.

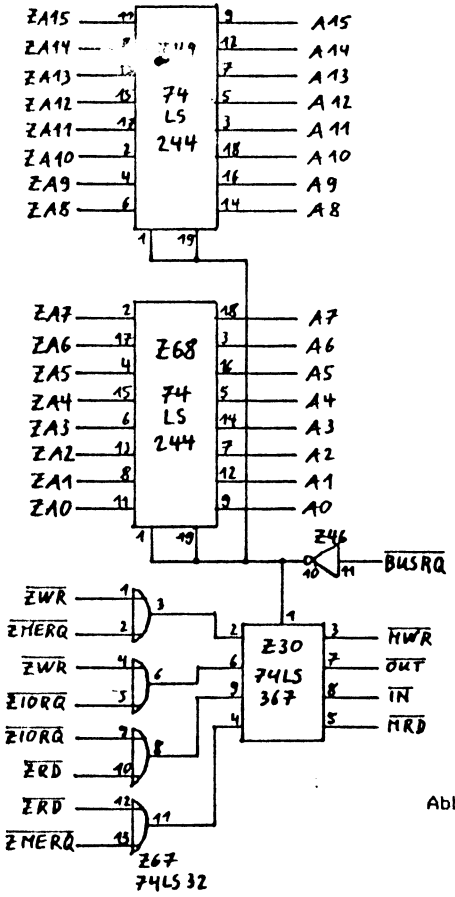
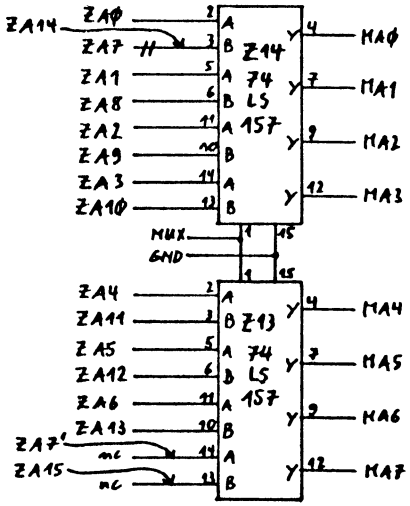


Abb.1: Treiber für Adressen und Daten des Z80 (die mit Z.. bezeichneten Signale sind die des Z80)
Erzeugen der Signale MWR*, MRD*, OUT* und IN*
Adreßmultiplexer für die RAMs (eingezeichnet sind die Änderungen für 4164--RAMs mit 7- oder 8-Bit-Refresh; für 8-Bit-Refresh-RAMs muß ZA7' von einem Generator für 8-Bit-Refresh-Adressen kommen)



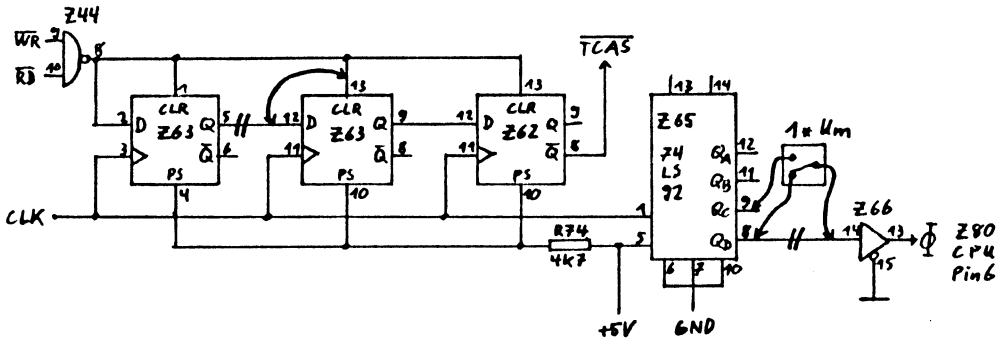


Abb.2: Änderungen für die Umschaltung des Systemtaktes zwischen 1,77 und 3,55 MHz

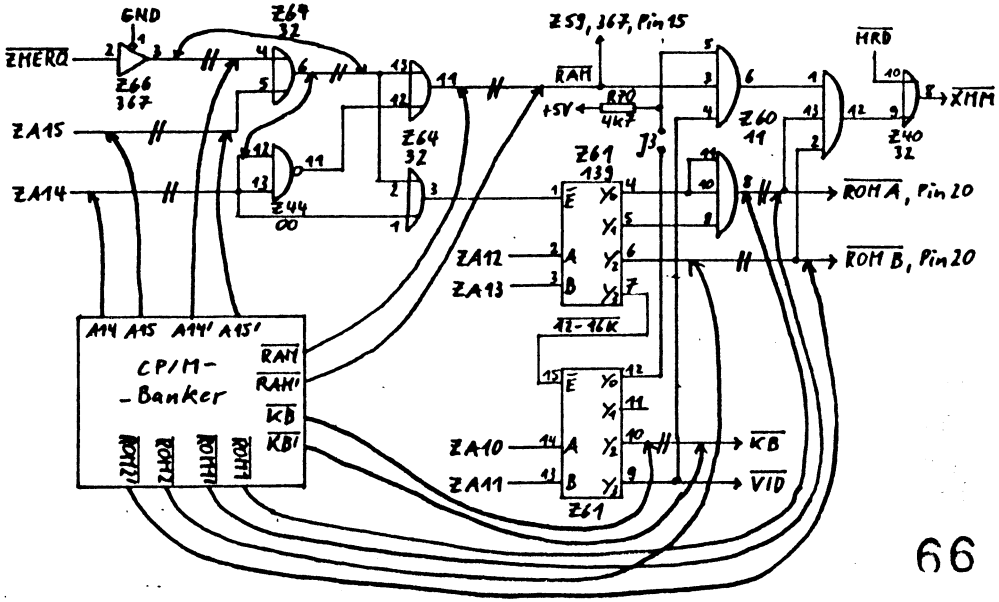


Abb.3: Änderungen an der Adreßdecodierung des CPU Boards und Anschluß des CP/M-Bankers

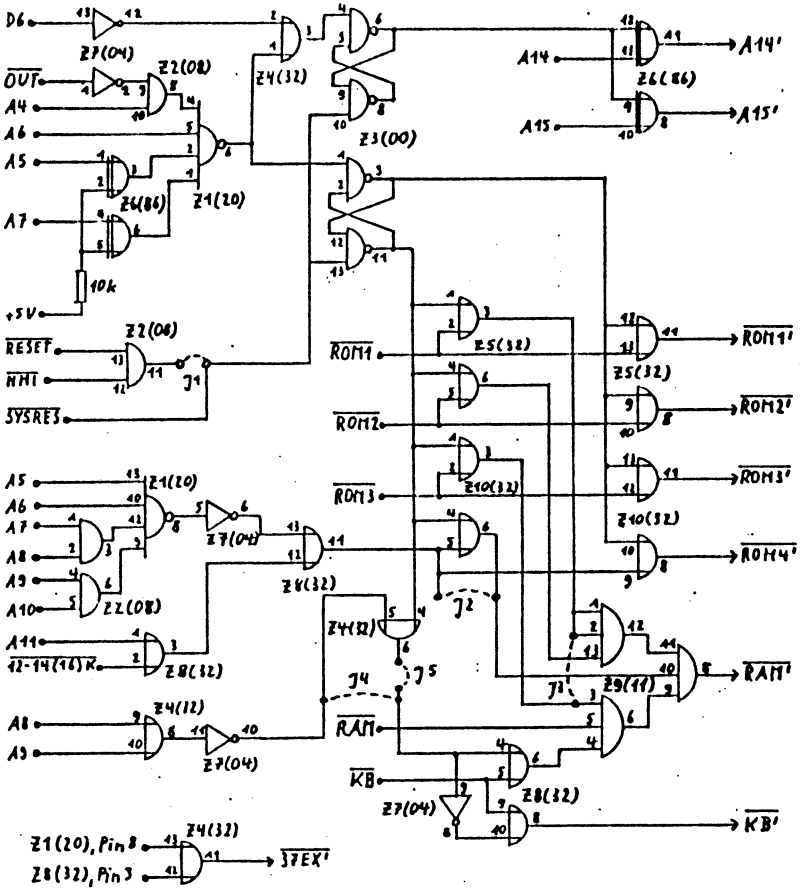


Abb. 4: Schaltung des CP/M-Boards

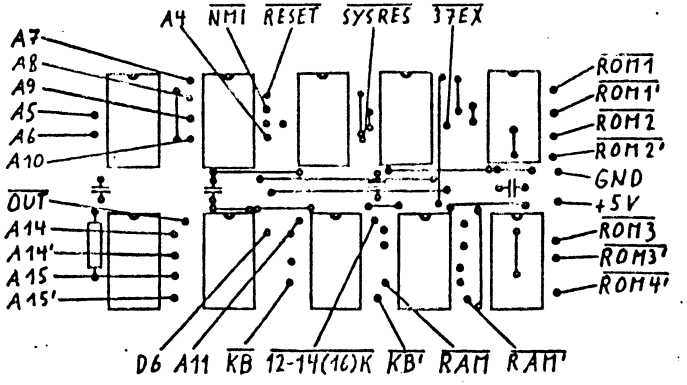


Abb. 5: Anschlüsse an das CPU-Board

Verkaufe
2x Model II
Heiko, 04169/1366

Folgende Dinge sind in meinem Inventar überflüssig und deshalb will ich sie losschlagen:

1 x Diskettenlaufwerk, einseitig, 40 Track, 2 Jahre alt, ohne Gehäuse und Stromversorgungsstecker.
Preis: DM100.--

5 Jahrgänge mc: 1981...1985 jeweils als Bücher gebunden. Gute Verarbeitung, da von mir selbst gebunden
Preis: DM25.--/Stück

1 x Matrixdrucker, Typ Heath H14. 8 Jahre alt aber sehr gut erhalten. Bisher keinerlei Ausfälle ! Serieller Anschluß 110...4800 Bd., mit Druckerkabel.

Wer interesse hat, rufe mich an oder schreibe mir. Ich bin abends ab 18.30 Uhr zu erreichen (am Arbeitsplatz nicht immer !)

Kurt Müller, Soltaustr. 24a, 2050 Hamburg 80, Tel: 040/7 24 60 83

- I M P R E S S U M -

1. Vorsitzende
Gerald SCHRÄDER
Am Schützenplatz 14
2105 Seevetal 1
☎ 04105 /2602

2. Vorsitzende
Bernd RETZLAFF
Kleiner Sand 98
2082 Uetersen
☎ 04122 /43551

Hardware-Koordinator
Eckehard KUHN
Im Dorf 14
7443 Frickenhausen 1
☎ 07022 /45417

Bibliothekar
Club-Bücherei
Werner FÖRSTER
Christoph-Krebs-Straße 9
8720 Schweinfurt
☎ 09721 /21841

Redaktion
Jens NEUEDER
Panoramastraße 21
7179 Michelbach /Bilz
☎ 0791 /42877

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

Druck
Horst-Dieter Schroers
Breslauer Straße 9
8016 Feldkirchen
☎ 089 / 9032615

Das INFO erscheint zweimonatlich.

Hallo Club-Ler,

Kurz bevor ich mich ein paar Wochen ins Ausland verdrücke, habe ich es doch noch geschafft, ein INFO zu erstellen. Ich hatte ja versprochen, gleich nach Erscheinen des Letzten mit der neuen INFO anzufangen.

Nach meinem Urlaub hoffe ich, daß wir endlich mal wieder die regulären Erscheinungstermine einhalten können. Vielleicht klappt es bis zum übernächsten INFO. Ich werde es auf alle Fälle versuchen.

Weiterhin kamen Anregungen für ein neues Layout unserer Titelseite. Da wir ja nun ein gültiges Club-Emblem haben ist es richtig und sinnvoll, dieses auch auf der Titelseite zu verwenden. Leider ging es bei dieser Ausgabe noch nicht, da ich schon vorgearbeitet hatte. Ab Heft 26 haben wir dann ein neues "Outfit".

Bis dahin wünsche ich Euch noch eine schöne Zeit und unbegrenzte "Rechnerzeit".

Es grüßt Euch Euer

Jens

Edles aus ELCOMP

serviert von KJ

Müde, stets Gedrucktes zu lesen, lieber Leser?
OK! Hier ist Geschriebenes aus erster Hand!
[Sogar HRG: Himmlisch Rührende Grafik...]

Und ihr anderen, Enttäuschten! Habt Nachsicht - 7 (sieben) Jahre lang nahm mein "Model 1" einen wesentlichen Platz auf meinem Schreibtisch ein - viele Jahre lang verwandelten TSCRIBS + EPSON meine finsteren Gedanken in noch finstere Druckerschwärze, -- nun plötzlich ist der Platz leer, verwaist, traurig... und ich täusche euch mit einer Druckschrift, die doch nur eine Typ-schrift ist, Marke xj.

„Ausgestiegen"? - Oh, no no no no no: Mein Chauvi - Peiniger - Unhold - kurz: CPU- befindet sich an einer ersten Adresse und wird z.Z. in einen Jungbronnen getaucht... Wenn er (sie) aus der Verjüngungskur wieder auftaucht: macht euch auf was gefaßt - (worauf, weiß ich selbst noch nicht!)

In der Zwischenzeit übe ich mich in der edlen Kunst des Schreibens - nachdem Ulrich so fleißig viele Artikel aus ELCOMP ab 1979 gesammelt und kopiert hat, war es meine verdammte Pflicht, diese ebenso fleißig auszumisten, zu ordnen und zu kommentieren. Nun schlummern sie in einer Mammelsappe bei Werner, dem Bibliothekar, und harren der Abrufe ...

Artikel aus ELCOMP [Stand: Mai 88]

Gruppe	Titel	Kurzkommentar / Verf.	Heft/Seite
CP/M	Das Standard-Interface für Microcomputer	← dto.	4/81/29-31
"	Erfahrung mit dBASE II	← "	6/84/56-57
EDITOR	XEDIT - Ext. Ed. f. TRS80/1	Leistet mehr als TANDY's Ed. [Wollschl.]	4/82/9-15
"	SCREDIT - ein Bildschirm-orientierter Ed. f. L2	[Heidenreich]	1/83/42-46
FUNK	TRS80 u. Funkfernschreiben	Für Amateurfunker (m. Listing)	7/9/81/10-22
"	RTTY + CW mit dem TRS80	RTTY-Interface der Fa. Funk & Comp.	7/9/80/114
GRAFIK	Pixelgrafik in PASCAL	Grafik f. PASCAL m. TRS80 [Wollschl.]	9/82/30-33
"	Hardcopy m. TRS80/GENIE	Eine Bit Image Grafik m. allen Druckern	7/9/82/43-47
"	TRS80 - MX80 u. Grafik	Progr. f. d. Grafik auf dem APPLE	2/82/46-48
"	Hardcopy f. TRS-80-Grafik	Ein (langsamer!) Ersatz für ZKL	11/12/80/103-105
HARDWARE	Motorsteuerung bei Minifloppies	(m. Schaltbild) [Keller]	1/83/186-[illegible]
"	Den Comp. richtig nutzen	Verwendung der RS232-Schnittstelle	4/5/84/88-92
"	Datenübertragung im Zeitmultiplexverfahren	f. GENIE	7/8/82/76-81
"	EPROM für TRS80	mit BASIC u. M.-Programm → s. „Progr."	
MATHE	Berechng. generat. Funktion	Ein BASIC-Progr. hierfür	4/80/10-12
"	Suchlauf	Darstellg. von parametr. Kurven	3/79/6-12
"	Kurvenuntersuchg. m. PASCAL	Bestimmg. v. Extrema u. Nullstellen	1/83/88-91
"	PRINTUSING als Rundungshilfe	Einfach „nur" ein Trick	1/82/46
"	Math. Funktionen f. PASCAL-Anfänger	(mit TRS-80) [Klaus]	1/83/104-108
"	TRS-80 als elektron. Kurvenlineal. - BASIC-Progr. zur Approximation		1/80/5-6
"	Iterative Verfahren	BASIC. - u. a. Lösung d. KEPLER-Glchg.	1/81/20-21
"	Ein Problem von Kopf um Kopf	-?→ eben etwas f. Köpfchen	7/8/82/119-126-129
"	Die Mandelbrotmenge	Mathe + Kunst v. Comp. (Randstruktur)	3/4/86/9-14
"	Rundungsfunktion in BASIC	Minimiert Rundungsfehler	7/8/82/123
"	Kalenderrechnung	Ewiger Kal. v. 15.10.1582 - 22.1.3268 (!)	-/-/11-25

Gruppe	Titel	Kurzkommentar / Verf.	Heft/Seite
Masch.-Sprache	Z80-Assembler/Bsp.	- erklärt am nütel. Bsp. /Wollschl.	10/82/44-50
"	Programmieren in M.-Spr.	Kurs in 8 Teilen von Flögel	6/81 - 3/82
"	Relokatierprogr. f. Z80	Listing mit Flußdiagr. + HEX-Dump	11/12/81/42-44
"	One Pass 8080-Macrouss. [MS]	für den TRS-80 /Matthaei	11/12/82/107-120
"	ACCEL 3 - Comp. f. gehobene	Ansprüche " -	10/82/55-56
"	HEX-Binär-Wandler m. Z80	Ein bißchen Masch.-Spr./Wollschl.	6/81/18-29
Mysteries	Vgl. CP/M u. MS-DOS	MS-DOS bei Dateien schneller; „Piping"	6/84/42
"	RAM + ROM im TRS-80	Das Programm „EXPLOR"/Wollschl.	11/12/82/38-48
"	BASIC hinter d. Kulissen geschaut	Was nie im Handbuch steht	4/82/41-42
"	DOS unter die Lupe genommen	" " " " "	4-8/82 [4 Teile]
"	Vgl. TRS-80 Monitore	Vgl. RSM-2 u. TBUG; Vorteile v. RSM	9/80/62-64
"	TRS-80 Tastaturspion	ASCII wird jedesmal in DEZ u. HEX gezeigt	7/8/80/4-5
"	Prüfsummen üb. ROM-Bereich	Zeigt Summen von je 256 B. ROM an	2/81/56
Programm-mierung	NEWDOS80: 3f. Didkette, 1,5x schnell	Erklärg. der Filetypen/Wollschl.	-/83/54-59
	SWAP - virt. Vertauschen	numerischer Arrays / Wollschl.	10/84/57-59
"	Chung/Yuen "ting"-PASCAL	auf dem TRS-80, M.1	7/8/81/54-58
"	APL80-Interpreter f. TRS80	Progr.-Sprache f. TRS80 (3 Teile)	7-10/81
"	FLIST	Formatiertes Listing u.a. (Woll.)	2/81/44-59
"	Kopplg. eines Masch.-Progr.	mit einem BASIC-Progr. (DATAs)	10/81/63-69
"	Sprungentf. b. rel. Adressg.	Ein BASIC (!)-Progr. z. Berechg.	3/79/15-16
"	Laufzeitreduzierung	beim BASIC-Interpreter	3/79/16-18
"	Neue BASIC-Befehle	5 neue Befehle (mit Listing)	4/5/84/46-49
"	Disk-BASIC in Level 2	Befehlserweiterungen f. L2 (Wollschl.)	7/8/81/45-48
"	Programmiertrick f. TRS-80	bei Kassettenbetrieb; Vor- u. Nachteile	1/80/23-25
"	EPROM-Programmer f. TRS80	Schaltplan + Programme in BASIC u. M-Spr.	10/82/4-17
"	Programmieren in FORTH	Ein Kursus in 7 Teilen	6/82 - 6/84

Gruppe	Titel	Kurzkommentar / Verfasser	Heft/Seite
Programm-mierung	BASIC-Monitor	System-Programme nach BASIC	3/81/34-37
	BASIC-Monitor D	DATA ... für Masch.-Progr. in "	2/82/37-39
"	ABASIC	Rücksprung ins BASIC nach M.-Progr. [Wollschl.]	10/81/24
"	Schnelles Zählen in BASIC	z.B.: A+1 = -NOTA	10/80/14
"	Was ist FORTH ?	Geschichte und Beschreibung	5/82/55-57
Utilities	Protokoll v. Bildschirmausg.	TRACE-Fkt. auch auf Drucker	5/80/21-22
"	Bildschirmausdruck o. Probl.	d.h. auch bei NMI ! (z.B. Spiel)	6/84/53
"	Eingabe v. BASIC-Bef. m. SHIFT	Eine andere Art "Shorthand"	11/12/80/44-49
"	TRS80-READDISK/BAS	Lesen beliebiger, auch geschützter Sektoren	
"	Hardcopy Print f. TRS-80	Eine [motor.] Ergänzung f. Level II	11/12/79/112
"	Ein-/Ausgabeprogrammierg.	Hard- u. Softwarebeschreibung	10/79/1-5
"	TRS-80 als Timer/Wecker	M.-Progr. = 192 Bytes f. alle Zwecke! [Woll.]	7/8/82/32-34
"	Simul. v. 6502 auf TRS80/GENIE	Emulat.-Progr. simuliert 6502-M.Proz.	7/8/82/10-15
"	Meldeton für TRS80	Schaltplan u. Listing (RS232 C-Schnittst.)	7/8/80/5-6

Platz für Nachträge: