SHK KLUB-INFO

1/88

ASSEMBLER-ECKE

BASIC-TIPS

DIT & DAT

SEHEN

STEREOKOPISCHES

PROGRAMM

BILDSCHIRM TIPS

PROGRAMM DER MONAT

SUPERCOPY

PROGRAMM LANGZAHL

VIDEO

INFORMATIONEN

MZ 80B

PROGRAMM WUERFEL

BASIC

E- UND A-

SHARP USER CLUB

VERWALTUNG MZ 80B

SHARP
HISOFT-PASCAL
KLUB E.V.
c/o Viola Petersen
Behringstr. 23
2000 Hamburg 50

ARCHIVEXEMPLAR

BITTE ZURUCK AN DEN KLUB

WIR ÜBER UNS
TEIL 3

Liebe Leser,

wir haben heute Anlaß uns zu freuen. Nicht nur darüber, daß das erste KLUB-INFO für dieses Jahr fertig geworden ist, sondern weil es gleich mehrere gute Gründe gibt, die zum Jubel Anlaß geben.

Wir können Geburtstag feiern, denn unser
SHARP HISOFT-PASCAL KLUB IST IN DIESEN Tagen 1 Jahr alt geworden. Am 2.3.1985 wurde er aus der Taufe gehoben und hat seit dem Tage seine Arbeit erfolgreich wahrgenommen.
Statt der Torte, die sich sehr schlecht versenden läßt, haben wir unser KLUB-INFO etwas weiter "aufgeputzt" und hier eine buntgemischte Palette an Informationen und Programmen zusammengestellt, von denen wir uns wünschen, daß diese Mischung allen gefällt, ankommt und "schmeckt".
Unser Titelblatt hat sicher schon signalisiert, daß hier ein volles Programm zu erwarten ist.
Es ist immer recht erfreulich, was so alles an Informationen zusammenkommt, da doch Computer-Zeitschriften hier so überhaupt nichts mehr für SHARP-Computer und für deren Besitzer zu berichten wissen.
Dank den vielen fleißigen Mitarbeitern, die uns diese Sachen zur Verfügung gestellt haben.

In Sachen HISOFT-PASCAL und MZ 800er haben wir noch eine besonders gute Nachricht:
Ich habe die erste Version laufen gesehen, die mit 80-Zeichen und den Grafikmöglichkeiten des 800er arbeitet. Es funktionierte einfach großartig. Ein Klubmitglied hat es geschafft, die benötigten Routinen für die 80-Zeichen und das Ansprechen der Grafik zu schreiben. Mehr darüber - wie, wann und wo - bringen wir in der nächsten KLUB-INFO Ausgabe.
Für heute wünsche ich beim Lesen und Ausprobieren viel Spaß.
Mit freundlichen Grüßen

Ihre

Viola Petersen

DIT & DAT

Liebe Sharp-Hisoft-Pascal-Freunde!

Ich benutze mit Hisoft-Pascal den 800er Modus (800er Modus, Hisoft-Pascal laden, Monitor wählen, nach Reset drücken, J 1200 eingeben, Compiler meldet sich).
Nun wie kann ich jetzt die 80-Zeichen und die hochauflösende Grafik des MZ -800 nützen?

P.K.Wien

Wenn Du nach dem Laden des Hisoft-Compilers RESET eingibst, bist Du automatisch wieder im 700er Modus. Hisoft-P. benutzt den ab Adr 0000 bis 1000H liegenden ROM-Monitor fuer seine Kommunikation mit der Aussenwelt. Dieser ROM-Monitor wird nach dem Start den Compilers in das parallel dazu liegende RAM kopiert und dort leicht modifiziert. Dieser Monitor ist leider nur ein 700er Monitor. Er benutzt das VideoRAM ab Adresse D000H und nicht wie der 800er ab 8000H. Auch ist die Verwaltung des Bildschirms viel einfacher. Soll z.B. links oben der Buchstabe "A" dargestellt werden, so wird auf Adresse D000H die Hexziffer 01 (Videocode!) geschrieben. Dieser Wert wird nun von einem separat ablaufenden System als Adresse aufgefasst, die in den Charakterzeichenbereich zeigt. Ab Adresse 01 (in diesem Beispiel, fuer "A") multipliziert mit 8 (jeweils 8 Bytes fuer ein Zeichen) stehen 8 Bytes, die untereinander auf den Bildschirm gebracht das Zeichen "A" ergeben. Im Gegensatz zum MZ700 stehen diese 8 Bytes nicht mehr im ROM sondern im RAM ab Adresse C000H. Dort koennen sie auch modifiziert werden, das heisst, man kann sich seinen eigenen Zeichenvorrat selbst schaffen. In den Bereich ab C000H (Video) kannst Du wie folgt hineinschreiben:

```
IN A,(E0H)        umschalten
LD A,FFH
LD (C008H),A      waagerechter Balken, da alle 16 Bits gesetzt
IN A,(E1H)        zurueckschalten
```

Nach dieser Aktion hat der Buchstabe "A" oben einen waagerechten Querbalken.

Die hochaufloesende Grafik wird voellig anders betrieben. Wie gesagt, liegt das VIDEO-RAM dazu ab Adresse 8000H. Jedes Byte, das dort hineingeschrieben wird, wird bitweise abgebildet. So zum Beispiel wuerde das Hineinschreiben von 85 in die Adresse 8000 oben links so aussehen: -.-.-.-. (Strich:nichts, .:Pixel)
85 ist hexadecimal 55, als0 01010101.
Vor dieser Pokerei muss jedoch das ganze Betriebssystem auf 80 Zeichen umgestellt werden. Das kann man durch Ansteuerung des Ports CCH machen. Die Ausgabe LD A, 06 und out (CC),A stellt um. Nur muessen alle Bi;dschirmausgaberoutinen jetzt neu gefasst werden. Das geht nicht mit der vorhandenen Software im ROM-Monitor.

Es tut mit also leid, Dir vorerst sagen zu muessen, dass die 80 Zeichendarstellung mit Hisoft noch nicht moeglich ist. Ich bin jedoch dran, den Monitor neu zu schreiben, damit dies dann moeglich ist. Fuer Klubmitglieder steht er dann zur Verfuegung. Es wird aber wohl noch 3 Monate dauern, da ich selbst erst jetzt mit dem 800 angefangen habe. 3 Jahre benutzte ich den MZ80K und ein Jahr den MZ 700.

Bruno Volkmer
D-2940 Wilhelmshaven,
Bernauer Weg 8

DIT & DAT

JOYSTICK

Dieses Kapitel widmet sich der Bedienung des Joysticks. Der Joystick wird hinten an die Erweiterungsleiste angeschlossen. Über den Befehl IN A,(FOH) wird in den Akku die Information eingelesen, ob der Joystick bedient wurde oder nicht. Es ist möglich, festzustellen, ob der Joystick in acht verschiedene Richtungen bewegt wurde. Dabei liefert bei nicht gedrückter Schußtaste der Joystick folgende Werte:

```
                  oben

          FAH       FEH       F6H
links     FBH       FFH       F7H        rechts
          F9H       FDH       F5H

                  unten
```

Nun das Schaubild bei gedrückter Schußtaste.

```
                  oben

          EAH       EEH       E6H
links     EBH       EFH       E7H        rechts
          E9H       EDH       E5H

                  unten
```

Auszug aus dem Buch
Alles über den M2-800
von BBG. Sehr zu empfehlen.
Dirk

Man kann das Bitmuster des Akkus auschlüsseln, um festzustellen, ob der Joystick bedient wurde oder nicht.

Bit 0: Enthält 0, wenn Joystick nach oben gedrückt wurde

Bit 1: Enthält 0, wenn Joystick nach unten gedrückt wurde

Bit 2: Enthält 0, wenn Joystick nach links gedrückt wurde

Bit 3: Enthält 0, wenn Joystick nach rechts gedrückt wurde

Bit 4: Enthält 0, wenn Schußtaste gedrückt wurde

Bit 5-7: Enthalten immer 1.

Wenn also die Schußtaste und der Joystick nach links gedrückt wurden, sind Bit 2 und 4 zurückgesetzt, der Rest ist 1. Man hat dann folgendes Bitmuster: 11101011 = EB Hexdezimal (siehe Tab.)

Dann wollen wir noch ein Programm schreiben, daß den Joystick vom Basic aus analysiert. Es kann mit USR($5700) aufgerufen werden.

In folgenden Adresse sind die Werte für die Tasten des Joysticks enthalten:

5800H: Wert für Joystick hoch 1=hoch 0= nicht bedient

5801H: Wert für Joystick runter 1=runter 0= nicht bedient

5802H: Wert für Joystick links 1=links 0= nicht bedient

5803H: Wert für Joystick rechts 1=rechts 0= nicht bedient

5804H: Wert für Schußtaste 1= gedrückt 0= nicht gedrückt

Hier nun das dazugehörige Programm, daß als Maschinenunterprogramm aufgerufen werden muß.

```
5700  DB F0     IN   A,(F0H)      (Wert von Joystick 1)
5702  2F        CPL               (komplementieren des Wertes)
5703  21 FF 57  LD   HL,57FFH
5706  06 05     LD   B,06         (Schleife für 5 Werte)
5708  23        INC  HL
5709  F5        PUSH AF           (Rette Akku)
570A  E6 01     AND  01           (Letztes Bit maskieren)
570C  77        LD   (HL),A
570D  F1        POP  AF
570E  0F        RRCA              (Rotiere Akku)
570F  10 F7     DJNZ 5708
5711  C9        RET
```

Natürlich können 2 Joysticke abgefragt werden. Das Prinzip zur Abfrage des zweiten Joysticks ist analog zum Abfragen des ersten Joysticks. Jedoch muß statt PORT F0H der PORT F1H benutzt werden.

DIT & DAT

"...und plötzlich war das Trennzeichen weg!"

Im Augenblick ist das Wetter ja nicht umwerfend und man arbeitete wieder mehr am Rechner.Ich saß also stundenlang da und verbesserte eines meiner früheren Programme,aber als ich es abspeichern wollte passierte nichts.Der Befehl P N1,N2,Name funktionierte nicht! Scheibenkleister!
Nachdem ich mir mit 'V' die Voreinstellung angeschaut habe,haute es mich fast um,anstatt eines Kommas war nichts mehr da und damit funktionieren alle Befehle nicht mehr,die bei der Eingabe ein Trennzeichen brauchen,also 'P','W','N' usw..
Laut Handbuch kann das nicht passieren,tuts aber doch.Mein Fehler war, daß ich wahrscheinlich einmal 'S CR' eingegeben hatte und das ist beim System mit dem ich an der UNI arbeite ein Editorbefehl,hier führt es aber dazu,daß das Trennzeichen gelöscht wird.Die Macht der Gewohnhe.. hatte also zugeschlagen.
Also war das ganze schöne Ändern für die Katz!
Aber ich war nun so richtig schön wütend und wenn ich wütend werde, dann passiert was.Ich fing also damit an,zu suchen wo das Trennzeichen im System abgespeichert wird.
Meine Suche war mit Erfolg gekrönt.
Beim Quick-Disk Pascal ist das die Speicherstelle 4680H,in ihr steht der SHARP-ASCII Code des Trennzeichens.Falls einem also das gleiche wie mir passieren sollte,keine Panik!
Mit dem Befehl 'B' aus dem Pascalsystem aussteigen,im Monitor den Befehl M4680 eingeben und dann 2C <CR> <SHIFT BREAK> und mit J1221 ins Pascal zurück gehen und alles ist wieder in Ordnung,das Trennzeichen ist wieder da.Ich hoffe,daß es anderen jetzt nicht mehr so geht wie mir.

Peter Stöhr
Prinz-Konstantin-Str. 9

8000 München 83

Dieser Fehler passiert schon mal und es ist wirklich recht ärgerlich, wenn man sich nicht zu helfen weiß, und das Programm dadurch vielleicht auch noch verloren geht.
Normalerweise kommt man mit dem S-Befehl (s.S. 66 im Handb.) nur in berührung, wenn man das vordefinierte Trennzeichen KOMMA durch ein anderes ersetzen möchte z.B. den Bindestrich, wie man es von Basic her gewohnt.
Nun kann es aber passieren, daß man ausversehen, den S-Befehl anwählt, z.B. im Zusammenhang mit dem F-Befehl für Suchen und das Dilemmer ist da. Nichts geht mehr. Doch das stimmt auch nicht ganz, denn mit dem V-Befehl hat man die Möglichkeit, sich das derzeitige aktuelle Trennzeichen ausgeben zulassen und mit diesem könnte man dann ohne weiteres weiterarbeiten. Nur weil es dann sehr ungewohnt ist, hält man dieses nicht lange durch. Die Lösung oben ist schon die elegantere und damit eine Prima Hilfe, falls einem mal das

DIT & DAT

Problem über die Rechengenauigkeit im HISOFT-PASCAL

Im HISOFT-PASCAL werden REAL-Zahlen mit einer Mantissenlänge von 22 Ziffern dargestellt. Diese 22 Ziffern sind aber Binärziffern, im Dezimalsystem sind das aber nur ungefähr 7,5 Stellen!
Also kann man mit ungefähr 7 gültigen Stellen arbeiten, wenn bei der Ausgabe mehr Stellen angezeigt werden, dann sind die letzen Stellen gemogelt, der Rechner schwindelt dann eine nicht vorhandene Genauigkeit vor.
Was kann man dagegen machen ?
Am Compiler können wir nichts ändern, aber man kann in Pascal ja jeden beliebigen Typ selber erzeugen.Man muß sich also 'nur' den Typ LONGREAL und die dazugehörigen Rechenoperationen selbst definieren. 'Nur' ist in diesem Fall etwas untertrieben, den Typ LONGREAL kriegt man noch ganz leicht hin, aber bei den Rechenoperationen wird es dann schon etwas kitzlich, spätestens wenn man SIN,COS,SQRT,LN und andere mathematischen Funktionen selberschreiben will, ist der normale Hobbyprogrammiere überfordert.
Es gibt zwar die Möglichkeit,daß ganze über Reihenentwicklungen zu machen,die Rechenzeiten gehen dann aber leider ziemlich schnell in Zeitberreichen,die eigentlich nicht mehr tragbar sind.
Im großen und ganzen muß man sich also mit der Rechengenauigkeit begnügen.Für die meisten Fälle reicht sie aus, nur wenn man Numerikaufgaben macht reicht sie leider nicht, da bleibt einem nichts anders übrig, als zu jemand zu gehen, dessen Compiler besser rechnet, oder den Rechner an der UNI oder Schule zu benutzen.

Peter Stöhr
Prinz-Konstantin-Str. 9

8000 München 83

Anfrage von J.R. (Bremen) über Rechengenauigkeit bei PASCAL: Auch unter PASCAL arbeitet der Computer "einwandfrei", man muß nur die interne Darstellung der Zahlen berücksichtigen. Hisoft-PASCAL benutzt für eine Realzahl die Darstellung mit 4 Bytes, also mit 32 Bits. "Darstellung" heißt, daß intern alle Zahlen in normierter Form wiedergegeben sind der Art

$$X=0.nnnn*2^{p}$$

wobei nnnn für eine Ziffernfolge aus 0 oder 1 (im Dualsystem !) steht. Für die Hochzahl p werden (natürlich wieder im Dualsystem) 8 Stellen reserviert, das erste Bit gibt das Vorzeichen an, und so bleiben 23 Bits für die "nnnn" übrig. Diese Zahl wird - wie vom Logarithmieren her bekannt - **Mantisse** genannt. Die größte Mantisse ist demnach 2 hoch 23, d.i. 8 388 608 (jetzt im Zehnersystem), die kleinste ist 2 hoch 0 , also 1. Der Unterschied zwischen 2 Zahlen muß also größer sein als das Verhältnis 1/8 388 608, das ist etwa $1.1921*10^{-7}$. Anders ausgedrückt, in der siebenten Stelle einer Zahl wird es ungenau, und das erklärt dann das erhaltene Ergebnis. Möchte man genauer rechnen, muß man mehr Bytes für die Darstellung benutzen; solche Rechenprogramme arbeiten dann mit "Doppelter Genauigkeit" (oder wie immer es auch in Prospekten noch heißen mag). Es gab einmal ein Gerücht, daß Hisoft-PASCAL in Zukunft soetwas vorsehen will - Näheres darüber weiß ich aber nicht.

Prof.Dr.Snatzke

(Weiter hinten im Info finden wir noch das Programm LANGZAHL, was mit diesem Problem beschäftigt.)

DIT & DAT

Klub Info Nr 4
Anfrage von M.K. aus Oeynhausen

Gleich zwei Probleme: Grafikpokerei und Joystick.
1. Grafik
Hisoft Pascal bedient sich des ROM-Monitors, und der ist im 700er Modus geschrieben. Von hier aus kann nur auf das VideoRAM ab Adr D000H zugegriffen werden. Man kann zwar Werte in den Bereich ab 8000H einschreiben, jedoch haette dies auf die Bildschirmwiedergabe keine Wirkung. Abhilfe ist nur moeglich, wenn Pascal echt im 800er Modus arbeitet, und das ist noch nicht moeglich. Also: Grafik mit Hisoft: bitte warten! Es kommt!

1. Joystick
Der Joystick wird ueber die Ports $F0 (Stick 1) und Port $F1 (Stick 2) eingelesen. Die Werte scheinen dann aber noch etwas unlogisch zu sein. Macht aber nichts, die Hauptsache ist, man weiss, welcher Wert welcher Stickstellung zuzuordnen ist. Dazu ein kleines Beispielprogramm (in Pascal selbstverstaendlich):

```
program joy;
var wert:integer;

 procedure  stick(var wert:char);
   begin
    wert:=inp(#f0);
   end;

begin
  repeat
    stick(wert);
    write(ord(wert));
  until false;
end.
```

Klub-Info Nr 4
Anfrage der Herrn D.G. aus Ploen

Problem: Mit dem 800er lassen sich keine mit dem 700er geschriebene Quelldateien laden, wenn sie mit einer aelteren Hisoft-Version geschrieben wurden.

Hierzu folgende kurze Erlaeuterungen: Pascal kopiert den vom benutzten Monitor aus dem ROM in das adressparallel dazu liegende RAM. Dort wird der Monitor leicht fuer Belange des Pascal manipuliert.

Beim 700er laufen im RAM liegende Programme schneller als im ROM liegende. Die Schleifenzeiten muessten vergroessert werden, damit ein RAM-Program auch Programme lesen kann, welche vom ROM aus abgespeichert wurden. Der Grund liegt an WAIT-Zyklen, die die CPU einlegt, wenn sie mit dem ROM arbeitet.

Andersherum: Ein vom ROM aus geschriebenes Programm kann auch nicht von einem nur im RAM laufenden Programm gelesen werden.

Der 800er hat nun mit dieser Differenzierung Schluss gemacht. Es gibt hier keine Unterschiede mehr. Die 700er Quellprogramme sind nun etwas zu schnell fuer den auf dem 800er laufenden Pascal-Compiler. Herr G. soll versuchen, falls er die Quellprogramme unbedingt braucht, sie auf den 800 zu transferieren. Hierzu muss er den Monitor ins RAM kopieren und dann die betreffenden Zeitschleifewerte um ca 20% verkuerzen.
Die Werte sind zu finden auf den folgenden Adressen:

```
0A4B     52 steht dort, mit 41H versuchen,
09AA     73  ---*---  ,  *  60H ----*----;
075A     1B  ---*----  ,  *  16H ----*----;
```

HISOFT PASCAL

Programm erstellen:

```
Zeilen-Nr. Leertaste,dann Text,mit CR beenden
                              oder Automatik Modus
I erste Zeile, Schrittgröße CR.
                  Automatische Zeilennummerierung.
CTRL X löscht die angefangene Zeile wieder!
DEL(ete)      nur zurück löschen.
CTRL C löscht den Automatik Modus.
L CR Programm listen. L Nr,Nr CR
                     von Zeilen Nr. bis Zeilen Nr.
K Anzahl CR.    ? Zeilen gleichzeitig listen.
```

Editieren

```
E Zeilen Nr. CR holt die Zeile z. Editieren!
Space _ mit dieser Taste  vorangehen.
CTRL I  achter Sprünge vorwärts.
C       + überschreiben
          mit CR beenden.
I       * einfügen (insert) auch mehrere Zeich.
          mit CR beenden.
X       * Sprung ans Ende der Zeile,  (anfügen)
                     (oder mit DEL löschen    )
          mit CR beenden.
K       _ Zeichen löschen (kill) nicht sichtbar
Den Editiervorgang mit nochmaligem CR beenden

Q Abbruch des Editiervorgangs.Ohne Änderung!
L baut den Rest der Zeile auf (Cursor ist
                                  wieder vorne).
Z CR löscht alles(inclusiv Curser)nach rechts
R Neustart des Editiervorgangs (restaurieren)

M n,m  (move) Zeile dublizieren m gleich n
D n,m Delete von n bis m einschließlich
N erste Zeile, Schrittgröße CR.
                     (renumber) Neunummerierung.
Zeilennummer CR, Zeile löschen!
-------------------------------------------------
C CR   Compilieren
   bei Fehler E letzte Zeile editieren.
               P vorletzte Zeile    "
Y Programmstart  R (replai) wiederholen
```

Ein kleine Hilfe, die die wichtigsten Befehle in HISOFT-PASCAL anzeigt, falls man noch nicht so sicher ist, oder auch dadurch daß man auch in anderen Sprachen programmiert, nicht immer alle Befehle Auswendig behält.

ERNST KUNZE

ASSEMBLER-ECKE

Assembler-Ecke:

Wie wohl jedem bekannt, ist ein Computer ein Gerät, das in vielen tausenden von Einzelzellen einen von 2 Zuständen eindeutig festhalten kann (z.b.: magnetisiert oder nicht magnetisiert; Spannung oder keine Spannung;positive oder negative Ladung; etc.), und diese 2 Zustände bezeichnet man mit 0 und 1, mit H und L, oder sonst in geeigneter Form. Um größere Zahlen wiedergeben zu können, da fasst man mehrere solcher Einzelzellen (BITs) zu einem BYTE zusammen (8,12,16,32 oder noch mehr), und um damit zu hantieren muß man eine solche mehrstellige "BINAER-Zahl" in ein geeignetes Arbeitsregister (im allgemeinen AKKUMULATOR genannt und mit A oder AC abgekürzt) bringen, wo man es näher untersucht. Meist sind an Operationen möglich: Einschreiben, Auslesen, Addieren, "Vernichten",man kann den Inhalt eines weiteren BYTEs dazu addieren, man kann die AND- und die OR-Operation ausführen, und man kann die Einzelbits im AC hin und herschieben.

Alle entsprechenden Befehle dafür sind wieder als solche Zahlen von der Länge eines BYTEs einzugeben, und das ergibt nicht mehr merkbare Zahlenkolonnen. Man hat daher für häufig wiederkehrende Ziffernkombinationen einer Zahl Codes erstellt, die meist dreibuchstabig waren, und die Abkürzungen englischer Befehle waren. Solche Codes kann man sich schon eher merken, und man nennt sie - aus dem Griechischen abgeleitet - Mnemonics. Eine Sprache, die diese Mnemonics benutzt, ansonsten aber ganz "maschinennah" ist, heißt ein Assembler.

Das Schreiben von Programmen in Assembler ist sehr zeitraubend, und so hat man bald "höhere Sprachen" wie FORTRAN, COBOL, BASIC etc. entwickelt, die der (englischen!) Alltagssprache näher standen, und mit deren Hilfe man leichter verständliche Programme schreiben konnte; ein Interpreter oder Compiler für die "Hochsprache" übersetzt dann den "hochsprachlichen" Ausdruck in eine Folge von Assembler-Befehlen. Warum aber dann sich noch die Mühe machen und in Assembler programmieren?

Es gibt mehrere Gründe dafür. Z.B. sind nicht alle von einem Zentralprozessor ausführbaren Befehle durch Befehle in der Hochsprache ansprechbar, und Assembler-Programme sind im allgemeinen ökonomischer als solche in einer Hochsprache. Ökonomie gibt's hier in zweierlei Hinsicht, und beide schließen sich gegenseitig aus: Ökonomie im Zeitablauf, und Ökonomie im Platzbedarf. Beispiel: schreibt man zwanzigmal hintereinander einen Befehl, kostet dies viel Platz im Speicher, doch das Programm ist sehr schnell. Schreiben wir den Befehl in eine Schleife (FOR I:=1 TO 20 DO BEGIN .....END) dann nimmt dies viel weniger Platz ein, doch muß immer ein Schleifenzähler gebildet werden, der nach jedem Durchlauf inkrementiert werden muß, und den man am Schleifenende auch auf "Ende" abfragen muß. So sind z.B. viele Befehle erforderlich, um eine Multiplikation von zwei Ganzzahlen in Assembler zu schreiben, und diese ganze Prozedur wird immer ablaufen müssen, wie einach oder kompliziert die Multiplikation auch sein mag.Es ist andererseits ganz einach, mit 2 zu multiplizieren: man braucht nur den Inhalt des Akkumulators um eine Stelle bitweise nach links zu verschieben - also ein einziger Befehl in Assemblerprogrammen, aber die gleiche Vielzahl von Einzelschritten bei Übersetzung mit einem Compiler oder Interpreter.

Erfolgreiches Assemblerprogrammieren setzt voraus, daß man wenigstens etwas davon weiß, wie ein Computer funktioniert, außerdem muß man eine Reihe von ungewohnten Befehlen lernen (oder zumindest wissen, wo man schnell darüber Information herbekommt). Schließlich soll man das SHARP-Betriebssystem kennen, um die vielen dort bereits vorhandenen Unterprogramme für sich nutzbar machen zu können. Es ist naturgemäß unmöglich, hier einen Assembler-Kurs abzuhalten. Wer interessiert daran ist, der muß schon eines der Bücher durchstudieren, die sich damit beschäftigen. Aber kleine "Kostproben" kann man schon von Zeit zu Zeit geben!

PASCAL kann Assemblerprogramme integrieren, und es sind hier vornehmlich die beiden Befehle USER und INLINE, die das leisten (siehe dafür das Handbuch). USER steht für den Assemblerbefehl CALL, ruft also (mit absoluter Adresse!) ein Unterprogramm auf. INLINE hingegen ist kein Befehl, sondern einfach die Anweisung an den PASCAL-Compiler, das, was dahinter kommt, direkt als Maschinenprogramm-Code zu übernehmen. Wann kann man soetwas gut brauchen? Stellen wir uns z.B. das Problem, die Codes für Zeichen (ASCII, 1 BYTE), Ganzzahlen (2 BYTEs) und für Realzahlen (4 BYTEs) in Binärdarstellung auszuschreiben.

ASSEMBLER-ECKE

Die Ablage im Computer geschieht also immer binär, und das Handbuch beschreibt auch genau, wie das passiert. Jedes BYTE enthält hintereinanderliegend 2 Hexadezimalzahlen, und die kann man natürlich nach mathematischen Vorschriften in Binärform darstellen (z. B. durch eine Folge von Divisionen durch 2 etc.).Bedenken wir aber, daß die Ablage im BYTE bereits in binärer Form passiert, dann bietet sich eine viel einfachere Lösung an: Man holt sich BIT für BIT aus dem BYTE heraus und prüft, ob es 0 oder 1 ist, und das schreibt man dann aus. Ein passender Befehl dafür ist RLC (HL). Er bewirkt, daß der Inhalt der Speicherzelle, die durch HL adressiert wird, nach links rotiert wird (BIT 7 fällt links heraus und wird als BIT 0 eingeschrieben, BIT 0 geht nach BIT 1, 1 nach 2, usw.)Zusätzlich wird BIT 7 aber auch noch in die Übertragskennung (Carry-Flag) eingeschrieben. Diese wiederum wird durch verschiedene Sprungbefehle abgefragt, so auch durch JR in der Form JR C bzw. JR NC. Da wir "0" oder "1" ausschreiben wollen, so geben wir ASCII-Code #30 (für 0) oder #31 (für 1) vor. Bei 1 in CF soll auf 31, bei 0 in CF auf 30 gesprungen werden. Dazu laden wir z.B. 30 oder 31 in den Akkumlator A und schreiben dessen Inhalt in eine freie Speicheradresse, woraus wir sie dann später mit einem peek-Befehl wieder herausholen können. Als solche intermediäre Speicherzelle kann z.B. Adresse #11F0 dienen.

Was noch bleibt: wir müssen die Adresse des betreffenden BYTEs nach HL schreiben, und dies erreichen wir am einfachsten dadurch, daß wir die Adresse aus dem aufrufenden Programm als Ganzzahlparameter übergeben. Im UP liegt solch ein Parameter dann in den Adressen IX+2 und IX+3 (im Handbuch steht auch dies genau beschrieben !), und von dort holen wir die Adresse ins Registerpaar HL. Das kann dann in Assembler so aussehen:

```
    Befehl       Code          Bedeutung
LD L,(IX+2) #DD,#6E,2    Inhalt der Adresse IX+2 nach L laden
LD H,(IX+3) #DD,#66,3    Inhalt der Adresse IX+3 nach H laden
LD A,#30    #3E,#30      A mit #30 laden (ASCII für "0")
RLC (HL)    #CB,6    Inhalt der durch HL adressierten Speicherzelle linksrotieren
JR NC 2     #30,2        falls CF=0 dann 4 Plätze überspringen
LD A,#31    #3E,#31      A mit 31 (ASCII von '1') laden; dies wird bei NC über-
                                                              sprungen!
LD (11F0),A #32,#F0,#11   Inhalt von A nach Speicherplatz #11F0 laden
```

Diese Codes schreibe man einfach mittels INLINE ins Programm, und dann geht es wieder mit PASCAL weiter. D.h., noch nicht mit "richtigem" PASCAL-Code, denn dort kennt man nicht den Ansprung direkter Adressen. Also ersteinmal der Befehl WRITE(PEEK(#11F0,CHAR));
und damit wird 0 oder 1 ausgeschrieben. Das Ganze macht man 8 mal, dann ist auch der Inhalt der angesprochenen Speicherzelle wieder so, wie er ursprünglich war, und zur besseren Formatierung ist nach 4 BITs noch ein Leerschlag eingefügt. Das UP UEBERTRAG (als Procedure geschrieben) enthält diesen Assembler-Teil, alles andere im Programm dient dem "Rundherum". Wissen sollte man noch (auch aus dem Handbuch zu enträtseln, leichter aber durch Probieren zu finden), daß die vier BYTEs, die eine Realzahl darstellen, in der Reihenfolge 4/3/1/2 abgelegt werden! Dies ist wieder einer der Vorteile von Hisoft-PASCAL, daß Assembler-Code einfach über INLINE eingebaut werden kann, ohne dafür irgendwelchen Platz zu reservieren oder andere Umwege gehen zu müssen.

Prof.Dr.Snatzke

INFORMATIONEN

Hallo Viola!

Hier melde ich mich mal wieder. Diesmal mit ein paar Informationen über den Kern des Hisoftcompilers. Gemeint sind die Runtimeroutinen. Alle Beispiele und Adressen beziehen sich auf meinen Rechner, einen MZ-80B. Ich hoffe allerdings, daß sich bei anderen Rechner prinzipiell nichts ändert. Ich habe übrigens die Version 1.5, die Cassettenversion. Man sollte zum Verstandnis der Routinen sich die entsprechenden Seite des Handbuches über den Gebrauch der internen Register durch den Compiler mal durchlesen. Interessant ist da das A, und HL Register. Mein Compiler fängt bei 1220h an. Alle Adr. sind HEX-Zahlen.

Wozu die Routinen? Eine Person, die in Assembler programmiert, kann sich durch diese Routinen eine Menge Arbeit ersparen. Die Routinen erledigen für ihn die Eingabe und die formatierte Ausgabe. Ebenso kann der Benutzer bequem auf die Arithmetikroutinen zugreifen. Gerade diesen Routinen gilt der größte Teil meines Beitrages: Bei der Arithmetik unterscheidet man zwischen der Integer und der Real-Arithmetik. Schauen wir uns zuerst die Realarithmetik an: Im 'B' findet man die Routinen an den folgenden Stellen:

SIN	20CE
COS	20B0
TAN	22F4
EXP	1F83
LN	2029
ARCTAN	2223
FRAC	1F48
SQRT	1E04
SQR	19F0

Der Eingangswert dieser Funktionen liegt in HLDE. Das Ergebnis ebenfalls. Recht einfach, nicht wahr?! Bei den Grundoperationen braucht man zwei Operanden. Ich nenne sie a und b. In der Form a Operation b gilt folgendes: a steht auf dem Stack (PUSH HL,PUSH DE), b in den Registern HLDE. Das Ergebnis liegt wieder in HLDE. Die Lage der Routinen:

+	193E
*	19F4
/	1A6A

Die '-' Routine benutzt ebenfalls die '+' Routine, der Compiler macht aus b: (-b) und addiert dann den wert. Die Umkehrung des Vorzeichens geht so:

```
LD A,80h
XOR H
LD H,A
```

Nun zur Integerarithmetik. Allgemein gilt aber auch, daß Integer auch die Routinen der Realarithmetik benutzen können. Vorher müssen die Zahlen umgewandelt werden.

Integer in HL CALL 1AE2 Real in HLDE

Es gibt keine spezielle '+' oder '-' Routine, der Compiler addiert direkt.

```
OR A      ADC HL,DE      CALL PE,1599
```

Bei der Subtraktion wird das Komplement addiert.

DIV,MOD

diese Funktionen werden immer gleichzeitig durchgeführt. In der Form a op b gilt hier: a steht in DE, b steht in HL, Aufruf mit CALL 16C7, der MOD-Wert steht in HL, der DIV-Wert in DE Die '*' Routine steht bei 168A, die werte in DE,HL. Ergebnis in HL.

Als letzte Funktion nun noch die ABS-Funktion: Bei Integer gilt:

```
Wert in HL    CALL 16BC      ABS wert in HL
```

Bei REAL gilt:

```
Wert in HLDE  RES 7,H
```

Die 'HALT' Funktion liegt auch im Runtimesystem. Es erfolgt aber hier immer ein Rücksprung in den Compiler. Diese Adresse muß man finden und abändern. Beim 'B' bei CALL 15C6. Nun zur Ausgabe der Werte:

REAL:

```
          Pascalform  Erläuterungen
          WRITE(A)  CALL 1CF0
          WRITE(A:m)  Zahl in HLDE, m in A, CALL 1CEB
          WRITE(A:m:n)  Zahl auf Stack, m in DE, n in HL, CALL 18C4
```

Integer:

```
          WRITE(A)  Zahl in HL, CALL 14D8
          WRITE(A:m)  Zahl in HL,m in A, CALL 14AC
          WRITE(A:m:H)  Zahl in DE, m in HL, CALL 1551
```

Strings:

```
          WRITE(A)  Stringanfang in HL, Länge in B, CALL 1530
```

Boolean:

```
          WRITE(A)  logische Wert in A, True=1, False=0, CALL 1538
```

allgemein:

```
          WRITELN   CALL 1527
```

Die Eingabe von Werten:

```
          READ(REAL)  CALL 1E2D, Ergebnis in HLDE
          READ(INT)  CALL 184F, Ergebnis in HL
          READLN    CALL 1750
```

Die INCH-Funktion erfolgt durch CALL 1229. Der ASCII Wert steht in A.Jetzt noch schnell ein kleines Beispiel für die Anwendung:

```
          WRITELN(SQRT(A*A+B*B))
          CALL 1E2D  holt A herein
CALL 1F0  PUSH HL    A=A
          PUSH AF
          PUSH DE
          CALL 1E2D  holt B herein
          CALL 19F0  B*B
          CALL 193E  A*A+B*B
          CALL 1E04  SQRT(A*A+B*B)
          CALL 14D8  WRITE(SQRT(A*A+B*B))
          CALL 1527  WRITELN
```

Das ist doch nun eine einfache Anwendung der Routinen, oder? Nun noch ein paar Anmerkungen: Die NEW Routine ist auch in den Runtimeroutinen vertreten. Dabei spielt beim 'B' die Adresse 2407 eine wichtige Rolle. Hier speichert der Compiler den Anfangswert der letzten Zeigervariablen! MARK und DISPOSE werden direkt in das Listening compiliert. NEW funktioniert so: HL zeigt auf die Adresse des Zeigers. BC enthält den Wert 0-SIZE(Variable). CALL 1923. Jedem Pascalprogramm wird ein CALL 1475 vorangestellt. Die genaue Bedeutung ist mir nicht klar. Sie ändert einige Sprungadressen am Anfang der Routinen ab. So damit erst einmal Schluß. Demnächst noch einige Informationen über Vergleiche mit dem Runtimeroutinen. Dafür gibt es leid keine Unterroutinen.

Marten Feldmann

[illegible]

SUPERCOPY

Schnelles Laden von Systemprogrammen mit dem SHARP MZ 700

Wer mit dem Sharp-MZ 700 arbeiten will, muß erst ein Programm , sei es Basic, Pascal oder Fortran, von der Cassette laden. Wen hat es nicht schon geärgert, daß er zum Laden z.B. des S-BASIC ganze 193 Sekunden warten muß. Das nachfolgend beschriebene Programmchen hilft, diese Wartezeit auf etwa die Hälfte zu verkürzen.

Zum besseren Verständnis soll zuerst kurz SHARP's Ladeverfahren beschrieben werden:
Jedes Byte, das auf Cassette geschrieben werden soll, wird bitweise auf 0 oder 1 getestet. Wird eine 1 erkannt, wird 460 Mikrosekunen lang ein H-Impuls gesendet, gefolgt von 460 Mikrosek. L-Zustand. Jede 0 wird als Folge von 230 µs H- und 230 µs L-Zustand über Bit 1 des Port C des 8255 geschrieben.
Jedem Byte geht dabei noch ein Startbit 1 voran.

Beim Wiedereinlesen wird solange abgefragt, bis ein übergang von 0 auf 1 gefunden wird. Dies ist eine laufende Synchronisation. Ist dieser übergang gefunden, wird 320 us später der Zustand wieder abgefragt. Ist er immernoch H, so handelt es sich un eine 1, ist er L, so ist eine 0 gefunden. Diese Bits werden wieder in ein Register geschoben und nach Erreichen eines Bytes im Speicher abgelegt.

Die Zeitschleifen, die zum Schreiben von 0 oder 1 und zum Einlesen im Monitor-ROM eingebaut sind, können nun manipuliert weden. Bei dem langen Impuls für eine 1 wird das Reg A 89 mal dekrementiert, bei einem kurzen Impuls für eine 0 wird 2mal von 21 heruntergezählt.
Zum Wiedereinlesen wird ein Zähler benutzt, der etwa 75% der 1-Schleife und etwa 150% der 0-Schleife lang ist. Hier wird die Zahl 63 benutzt.
Parallel zum Monitor-ROM liegt RAM-Bereich. Hierhin kann der ROM-Inhalt mit ein paar Maschinenbefehlen geswappt werden. Von diesem RAM-Monitor läßt sich aber kein Programm mehr lesen. Nachrechnungen ergaben, daß die genannten Zeitschleifen um ca 20% verlängert werden müssen. Der Grund dafür mag darin liegen, daß das ROM langsamer ist und evtl WAIT-Zyklen eingefügt werden. Das Handbuch ist hier nicht aussagekräftig genug. Egal warum, es klappt jedenfalls. Die zu dekrementierenden Werte sind jetzt in der Reihenfolge 1,0,Lesen: 105,25, 75.

Das kleine Programm, dessen Hexdump abgedruckt ist, macht nun folgende Schritte:
1. Laden des zu kopierenden Programmes. Dabei wird geprüft, ob es früher schon im Schnellverfahren abgespeichert wurde.
2. Abspeichern im Normalverfahren mit S wie SAVE oder C wie COPY (wenn die Parameter noch vorhanden sind,

SUPERCOPY

andernfalls wird wie gehabt nach Anfang, Ende etc gefragt)
3. Abspeichern im Schnellverfahren mit QS = QuickSave oder QC = QuickCopy
Hierbei wird zuerst das Programmende errechnet, wohin ein kleines Vorprogramm gelegt werden soll. Dieses muß ja vom normalen Monitor nach dem Einschalten des Computers gelesen werden können. Diese Vorpogramm wird jetzt ganz normal abgespeichert.
Ist dies nach dem bekannten Dialog geschehen, werden der Monitor ins RAM kopiert und dort die Zeitschleifenwerte auf die Hälfte herabgesetzt. Anschließend wird der Header wieder hergestellt und das Hauptprogramm im Schnellverfahren abgespeichert. Am Schluß wird wieder der ROM-Monitor aktiv, es kann erneut geladen werden.
Bei guten Bändern können die Zeitwerte weiter verringert werden. Das Verhältnis der in den Adressen A2BB, A2C8 und A2C5 stehenden Werte sollte immer ca. wie 3 Teile zu einem Teil zu 4 Teilen sein. Allerdings wird der Zeitvorteil bei kleineren Systemprogrammen immer geringer, da daß Vorprogramm allein ca 25 Sekunden zum Laden benötigt.

Das Laden des Programmes vom Monitor aus geschieht in umgekehrter Reihenfolge: Laden und Starten des Vorprogrammes, dadurch Swappen und korrigieren der Zeitwerte und Nachladen des Hauptprogrammes. Diese Programme lassen sich von anderen Kopierprogrammen nicht so ohne weiteres vervielfältigen, bieten also einen bescheidenen Kopierschutz.
Werden von einem so geladenen Programm Files erstellt, werden diese ebenfalls schnell abgespeichert. Alte Files können nur gelesen werden , wenn nach dem Start ein Reset mit anschließendem Warmstart vom Monitor aus gemacht wird. Die gilt nicht für das S-BASIC, da dieses Pgm einen eigenen Monitor enthält.

Die entscheidenden Routinen aus dem Schnellkopierpgm. sind QCOPIE und SWAPP. Diese sind daher mit Erläuterungen beigfügt .

(R Volkmer)

P.S. Für HISOFT-PASCAL Benutzer, die von Pascal aus auf dem Monitor zugreifen wollen, gilt, nach dem Einladen von SUPERCOPY+Pascal, bei Systemmeldung. Mit E CR zurück in den Monitor und mit J121E CR wieder ins Pascal hinein. Dann erst können Programme eingeben werden.

```
*****************************************************************
Routine Swapp wird doppelt genutzt. Einmal werden von diesem Pgm
die Monitordaten verandert, zum zweiten werden Teile davon in das
Vorprogramm kopiert.
*****************************************************************

SWAPP:  DI              ;DISABLE INTERRUPT
        OUT (E1H),A     ;VIDEOBEREICH IST RAM
        LD DE,D000H     ;ZIEL DES ROM-MONITORS
        LD HL,0000H     ;MONITORANFANG
        LD BC,1001H     ;LAENGE DES MONITORS
        PUSH BC         ;LAENGE AUFHEBEN
        LDIR            ;TANSFER NACH D000H
        OUT (E0H),A     ;0000-OFFF IST RAM
        POP BC          ;LAENGE ZURUECK
        EX DE,HL        ;TAUSCHE QUELLE UND ZIEL
        LDDR            ;UND MONITOR N. UNTEN INS RAM
        OUT (E3H),A     ;D000.. IST WIEDER VIDEO U. TASTATUR
        LD A,1          ;1 MAL ABSPEICHERN
        LD (05B1H),A    ;1 * VERIFY
        LD (0512H),A    ;1 * LESEN
        LD (048EH),A    ;1 * SCHREIBEN
        LD A,36H        ;EINLESESCHLEIFE
        LD (0A4BH),A    ;KORRIGIERE
        LD A,13H        ;SCHLEIFE BYTE = 0
        LD (075AH),A    ;KORRIGIERE
        LD A,48H        ;SCHLEIFE BYTE = 1
        LD (09AAH),A    ;KORRIGIERE

*************************************************
bis hierher ist SR zugleich Teil des Vorprogrammes
*************************************************

        EI              ;ENABLE INTERRUPT
        RET             ;FUER SWAPP-ROUTINE

*************************************************************
dieser Teil korrigiert nach dem Start des Vorpgm den Header
*************************************************************

RA2:    LD HL,(1167H)   ;QUELLE DER ORG. PARAMETER
        LD DE,1102H     ;ZIEL
        LD BC,6         ;6 WERTE
        LDIR            ;TRANSFER
        EI
        CALL 0024H      ;LAEDT PGM NACH START DES VORPGM NACH
        JP C,0005H      ;BEI ERROR ZUM MONITOR
        LD HL,(1102H)   ;STARTADRESSE
        JP (HL)         ; UND SELBSTSTART
*****************************************
```


```
*****************************************************************
Routine OCOPIE auf Adr. A230 schreibt Ersatzheader, hangt Vorprogramm
an das Hauptprogramm an, speichert dieses ab, restauriert Header,
und schreibt Pgm mit veranderten Zeitschleifen auf Band.
*****************************************************************

OCOPIE: LD HL,(1104H)   ;PROGRAMMANFANG
        LD DE,(1102H)   ;PROGRAMMLAENGE
        ADD HL,DE       ;ERRECHNE PROGRAMMENDE
        INC HL          ;ABSTAND
        LD (1108H),HL   ;LEGE ENDE AB
        LD DE,1167H     ;HIERHER WERDEN PARAMETER GERETTET
        LD HL,1102H     ;QUELLE DER PARAMETER
        LD BC,6         ;6 WERTE
        LDIR            ;SCHIEBE ORIGINALWERTE NACH 1167H
        LD HL,(1108H)   ;HOLE PGM-ENDE
        LD (1104H),HL   ;ENDE WIRD ANFANG DES VORPROGRAMMES
        LD (1106H),HL   ;UND AUCH START
        LD HL,128       ;128 BYTES LAENGE
        LD (1102H),HL   ;DORT EINTRAGEN
        LD DE,(1108H)   ;HOLE ANFANG DES VORPROGRAMMES
        LD HL,SWAPP     ;QUELLE DER EINZUTAGENDEN DATEN
        LD BC,49        ;49 DATEN, OHNE DAS RET UND EI AM ENDE
        LDIR            ;UEBERTRAGE
        LD HL,RA2       ;ZWITER TEIL DES VORPROGRAMMES
        LD BC,22        ;22 WERTE
        LDIR            ;FUEGE AN
        LD A,E9         ;IST JP (HL)
        LD (DE),A       ;ANHAENGE AN
        LD A,BF         ;KENNUNG QUICKSAVE
        LD (10F0H),A    ;TRAGE KENNUNG EIN
        CALL 0021H      ;HEADER FUER VORPROGRAMM NORMAL AUF BAND
        JP C,ERROR
        CALL 0024H      ;SCHREIBE VORPROGRAMM NORMAL AUF BAND
        JP C,ERROR
        CALL SWAPP      ;TAUSCHE UND AENDERE MONITOR
        LD DE,1102H     ;ZIEL FUER ORIGINALPARAMETER
        LD HL,1167H     ;DORT LIEGEN DIE ORG.PARAMETER
        LD BC,6         ;6 WERTE
        LDIR            STELLE ORG.HEADER WIEDER HER
        CALL 0024H      ;SCHREIBE PGM SCHNELL AUF BAND
        JP C,ERROR
        JP KORR         ;UND ZURUECK ZUM MENUE
```

SUPERCOPY

```
> M A800
A800 3E C6 CD DC 0D 11 57 A0>+L  Wq-C2
A808 DF 31 F0 10 21 2A 00 22q1  !* "-7D
A810 A5 A1 D3 E4 CD 06 00 CDua+-L  L-9D
A818 B3 09 CD CE 0B CD 12 00m  L  L  -41
A820 FE 4C CA 5E A1 FE 53 CA L ^a S J-2E
A828 06 A2 FE 43 CA 33 A2 FE z C 3z -86
A830 4D CA 95 00 FE 51 CA 41M    A-06
A838 A0 FE 56 CA B4 A1 C3 09q V La  -DF
A840 A0 CD B3 09 CD CE 0B CDq m  L  L-9C
A848 12 00 FE 53 CA 36 A2 FE   S 6z -03
A850 43 CA 39 A2 C3 09 A0 2AC 3z  q=-7E
A858 A4 A5 9E 92 9D 9F B7 9Esupercop-0A
A860 BD 2A 20 4C A1 9C 92 B0y* Laden-D2
A868 20 43 B7 9E BD 20 51 A5 Copy Qu-88
A870 A6 9F A9 43 B7 9E BD 20ickCopy -63
A878 53 A1 AB 92 20 20 20 20Save    -81
A880 20 20 51 A5 A6 9F A9 53  QuickS-77
A888 A1 AB 92 20 4D B7 B0 A6ave Moni-58
A890 96 B7 9D 20 56 92 9D A6tor Veri-35
A898 AA BD 20 0D AB B7 B0 20fy  von -C6
A8A0 20 20 20 3A 0D 9A A6 A4   : bis-8B
A8A8 20 20 20 20 3A 0D A4 96    : st-81
A8B0 A1 9D 96 20 20 3A 0D 96art  : t-F1
A8B8 A1 9E 92 2D 92 9D 9D B7ape-erro-81
A8C0 9D 21 21 0D B0 A1 B3 92r!! name-82
A8C8 20 20 3A 0D AA A1 A4 96  : fast-0C
A8D0 0D 92 9D A6 AA BD A6 B0 erifyin-9F
A8D8 97 0D AA B7 A5 B0 9C 2Fg found/-25
A8E0 20 20 20 20 20 20 20 20       -00
A8E8 0D AB B7 B0 20 9A A6 A4 von bis-23
A8F0 20 A4 96 A1 9D 96 20 0D start  -5B
A8F8 B7 A9 21 0D CD 06 00 CDok!  L  L-2E
A900 1E 00 11 B7 A0 DF C3 09   o   -31
A908 A0 CD 06 00 11 9C A0 DFq   d   -9F
A910 CD 4B A1 22 04 11 E5 11 Ka"  e -E6
A918 A5 A0 DF CD 4B A1 D1 B7u  Ka o-65
A920 ED 52 23 22 02 11 11 AE\R0"  0-56
A928 A0 DF CD 4B A1 22 06 11  Ka" -71
A930 11 C4 A0 DF 11 A3 11 CD _  w  L-E6
A938 03 00 21 AA 11 11 F1 10  !f  0 -F1
A940 01 10 00 ED B0 3E 0D 32   \n> 2-2B
A948 01 11 C9 11 A3 11 CD 03  F w  L -78
A950 00 11 AB 11 CD 10 04 DA  v  L  J-88
A958 09 A0 CD 06 00 C9 CD 27 q  F'-39
A960 00 CD 06 00 11 DA A0 DF  L   J   -3D
A968 11 F1 10 DF CD 06 00 11 0  L  -D5
A970 E9 A0 DF 3A F0 10 FE BF  7 \-5F
A978 CA CD A1 2A 04 11 00 00 |La*  -77
A980 00 CD C7 A1 ED 5B 02 11  L \[  -98
A988 19 2B 00 00 00 CD C7 A1 +  L -79
A990 2A 06 11 00 00 00 CD C7*  L J-D5
A998 A1 CD 06 00 CD B3 09 FEa  m  F-FB
A9A0 CB CA 09 A0 CD 2A 00 DA J q  J-0F
A9A8 FC A0 CD 06 00 11 F8 A0 q  0q-18
A9B0 DF C3 09 A0 11 D1 A0 DF   q  q-AC
A9B8 CD 9B A2 CD 23 A2 21 2D xz z'--EA
A9C0 00 22 A5 A1 C3 5E A1 CD "ua ^a L-F7
A9C8 0C 00 C3 FD 05 2A 69 11  ^  0  -75

A9D0 CD C7 A1 ED 5B 67 11 19 L \[0  -0E
A9D8 23 CD C7 A1 2A 6B 11 CD# L k0q L-CB
A9E0 C7 A1 CD 06 00 11 CC A0 k L  0q-B8
A9E8 DF CD B3 09 FE CB CA 09q m  J  -04
A9F0 A0 CD 9B A2 21 67 11 11q xz'0  -54
A9F8 02 11 01 06 00 ED B0 C3      \n ^-7A
AA00 A4 A1 00 00 00 00 CD 09qa    L -1B
AA08 A1 3E 01 32 F0 10 CD 9Ba> 27 Lx-7A
AA10 A2 CD 23 A2 CD 21 00 DAz Wz L!  J-FC
AA18 FC A0 CD 24 00 DA FC A0 q W  J q-03
AA20 C3 AA A1 3E 69 32 AA 09 fa>L2f  -9A
AA28 3E 19 32 5A 07 3E 4B 32> 2Z >K2-A5
AA30 4B 0A C9 C3 09 A2 CD 09K F  z L -62
AA38 A1 2A 04 11 ED 5B 02 11a*  \[  -3B
AA40 19 23 22 08 11 11 67 11 #"   0  -08
AA48 21 02 11 01 06 00 ED B0!      \n-D8
AA50 2A 08 11 22 04 11 22 06*  "  "  -A2
AA58 11 21 50 00 22 02 11 ED !P "  \-A4
AA60 5B 08 11 21 9B A2 01 26[  !xz &-F9
AA68 00 ED B0 21 C3 A2 01 23 \n!^z #-47
AA70 00 ED B0 3E BF 32 F0 10 \n>\27 -CC
AA78 CD 21 00 DA FC A0 CD 24 L! J q W-55
AA80 00 DA FC A0 CD 9B A2 11 J q Lxz -91
AA88 02 11 21 67 11 01 06 00  '0     -B3
AA90 ED B0 CD 24 00 DA FC A0 \n W J q-04
AA98 C3 AA A1 D3 E1 11 00 D0 ^fa 0  r-A3
AAA0 21 00 00 01 01 10 C5 ED!     | \-E5
AAA8 B0 D3 E0 C1 03 EB ED B0n -  '\1-B7
AAB0 D3 E3 3E 24 32 4B 0A 3E -->02K >-DD
AAB8 0D 32 5A 07 3E 30 32 AA 2Z >02f-EA
AAC0 09 00 C9 21 67 11 11 02  F'0   -7E
AAC8 11 01 06 00 ED B0 00 CD     \n L-92
AAD0 2A 00 30 0B CD 06 00 CD* 0 L  L-05
AAD8 3E 00 3E 01 C3 07 01 2A>  > ^  *-72
AAE0 06 11 E9 FF FF FF 00 00  k   -FD
AAE8 FF FF 00 00 FF FF 00 00    -FC
```

Dieses Programm bringt einige Veraenderungen in das Basic MZ-2Z046. Mit CTRL-J oder PRINT CHR$(10) wird zwischen dem normalen und dem vom MZ-700 her bekannten zweiten Zeichensatz umgeschaltet. Das Kommando CTRL-L oder PRINT CHR$(12) schaltet in beiden Zeichensaetzen zwischen normaler und inverser Darstellung um. Die Kommandos muessen nach dem Laden des Maschinenprogrammes mit USR($558B) initialisiert werden !

```
558B  3E CD       INIT :LD A,$CD          ; x Accu = CALL-Code
558D  32 DF 05         LD ($05DF)         ; x Zum Monitor
5590  32 EC 05         LD ($05EC)         ; x
5593  21 BC 55         LD HL,ZEICH        ;   HL=Einsprungadresse
5596  22 E0 05         LD ($05E0),HL      ;   Zum Monitor
5599  21 C8 55         LD HL,INVS         ;   HL=Einsprungadresse
559C  22 ED 05         LD ($05ED),HL      ;   Zum Monitor
559F  21 AC 55         LD HL,CTRLJ        ; x CTRL-J aktivieren
55A2  22 6F 00         LD ($006F),HL      ; x
55A5  21 B4 55         LD HL,CTRLL        ; x CTRL-L aktivieren
55A8  22 73 00         LD ($0073),HL      ; x
55AB  C9               RET                ;   Zum Basic
                                          ;
55AC  3A DB 55    CTRLJ:LD A,(FLAG1)      ; x Flag fuer Zeichensatz
55AF  2F              CPL                 ; x auf 00 oder FF setzen
55B0  32 DB 55        LD (FLAG1),A        ; x
55B3  C9              RET                 ; x CTRL-J beendet
                                          ;
55B4  3A DC 55    CTRLL:LD A,(FLAG2)      ; x Flag fuer Invertieren
55B7  2F              CPL                 ; x auf 00 oder FF setzen
55B8  32 DC 55        LD (FLAG2),A        ; x
55BB  C9              RET                 ; x CTRL-L beendet
                                          ;
55BC  29          ZEICH:ADD HL,HL         ;   Displaycode * 8
55BD  CB E4           SET 4,H             ;   Offset auf 1. Zeichensatz
55BF  3A DB 55        LD A,(FLAG1)        ;   Accu = FLAG1
55C2  FE 00           CP 0                ;   Flag = 0 ?
55C4  C8              RET Z               ;   Ja - Zeichen anzeigen
55C5  CB DC           SET 3,H             ;   Offset auf 2. Zeichensatz
55C7  C9              RET                 ;   Zeichen anzeigen
                                          ;
55C8  F5          INVS :PUSH AF           ;   Accu retten
55C9  3A DC 55        LD A,(FLAG2)        ;   Accu = FLAG2
55CC  FE 00           CP 0                ;   Flag = 0 ?
55CE  28 06           JR Z,IVOFF          ;   Ja - Normale Darstellung
55D0  F1          IVON :POP AF            ;   Accu zurueck
55D1  2F              CPL                 ;   Invertieren
55D2  23              INC HL              ;   Zeiger korrigieren
55D3  D9              EXX                 ;   Register umschalten
55D4  77              LD (HL),A           ;   Accu zum Bildschirm
55D5  C9              RET                 ;   Weiter im Monitor
55D6  F1          IVOFF:POP AF            ;   Accu zurueck
55D7  23              INC HL              ;   Zeiger korrigieren
55D8  D9              EXX                 ;   Register umschalten
55D9  77              LD (HL),A           ;   Accu zum Bildschirm
55DA  C9              RET                 ;   Weiter im Monitor
                                          ;
55DB  00          FLAG1:DEFB $00          ;   Flag fuer Zeichensatz
55DC  00          FLAG2:DEFB $00          ;   Flag fuer Invertieren
```

März 1986 nr. 1

computer magazin für Pocket und MZ User

AKTUELLE SHARP NACHRICHTEN

Lieber Leser,

Dieses erste SHARP USER Magazin soll Ihnen die Bedienung und das Verstehen Ihres Computers erleichtern.

Wir haben anhand von Anrufen und Leserpost die am haufigsten anfallenden Fragen und Themen aufgegriffen und hoffen, sie ausfuhrlich geschildert zu haben

Weiterhin freuen wir uns auf weitere weitere Anregungen oder Artikel von Ihnen

Eine andere Frage, die uns interessiert. Wurden Sie ein so aufgebautes Magazin bei Ihrem Zeitschriftenhandler fur ca DM 3,- kaufen ? (In farbiger Aufmachung und ca. 20-24 Seiten stark)

Das Hauptthema dieses Magazins ist u.a der Plotter CE 516 P. Es wird anhand eines Listings gezeigt, welche Probleme wie gelost werden konnen. Außerdem sind viele USER-CLUBS mit der Frage nach Modemsoftware an uns herangetreten. Wir haben uns intensiver damit bschaftigt, und ein kleines DFU Programm entworfen Wie Sie es erweitern oder nutzen, bleibt Ihnen uberlassen

Wir freuen uns auf Ihre Anregungen und verbleiben

mit freundlichen Gruß

Detlef Korhon

** TIPS UND TRICKS **

Fehler in Zahlaufbereitung bei verschiedenen Basic-Versionen für Sharp MZ-700/800

Der Fehler tritt z. B. bei folgendem kleinen Beispiel auf:

PRINT 99999999 + .5

Das Ausgabeergebnis ist unbestimmt und dem Zufall überlassen.

Der Fehler kann durch die Eingabe von zwei Poke-Befehlen im AUTO RUN-Programm korrigiert werden:

MZ-700 Quick-Disk-Basic MZ-5Z008

POKE $ 62AD, $11 : POKE $ 62B0, $21

MZ-800 Kassetten-Basic MZ-1Z016

POKE $ 7D69, $11 : POKE $ 7D6C, $21

MZ-800 Quick-Disk-Basic MZ-5Z009

POKE $ 7DE0, $11 : POKE $ 7DE3, $21

MZ-800 Disk-Basic MZ-2Z046

POKE $ 7DE0, $ 11: POKE $ 7 DE3, $ 21

R.S.

JOYSTICKABFRAGE FUER SHARP MZ-800

```
BEZEICHNUNG CODEZAHL
TASTATUR    0
JOYSTICK 1  1
JOYSTICK 2  2
WERTETABELLE FÜR ALLE GELTEND:
LI RE OB UNT LIOB LIUNT REOB REUNT.
 7  3  1  5   8    6     3    4
BEISPIEL: 10 A=STICK(CODE)
          20 PRINT A
          30 GOTO 10
          40 REM DER BEFEHL STICK
          50 REM IST DIE EIGENTLICHE
          60 REM ABFRAGE
```

D.K.

GRAPHIK - BILDSCH. VOM MITTELPUNKT AUSMALEN

```
10 INIT"CRT:M2"
20 FOR I=160 TO 318
30 LINE (2, 1),I,0,I,199
40 LINE(2,1)319-I,0,319-I,199
50 NEXT I
```

D.K.

LIST.-BEFEHL MZ-800

DER BEFEHL LIST. ODER L.. ZEIGT ALLE EINGEGEBENEN ZEILEN AN. WURDE EIN FEHLER BEI DER PROGRAMMIERUNG GEMACHT, SO WIRD SICH DER COMPUTER MIT ERROR IN LINE XXX MELDEN. GEBEN SIE LIST. ODER L.. EIN. DIE FEHLERHAFTE ZEILE WIRD DANN AUFGELISTET.

D.K.

RUNDUNGSROUTINE FÜR POTENZEN EXPONENTEN ETC. FÜR SHARP MZ-800 SERIE

```
BEISPIEL: 10 R=16-(4↑4)
          20 R=INT(((R*1000)+5)/10)/100
          30 PRINT R
```

D.K.

IN DER INFO-AUSGABE NR.3 wurde das Programm LONGINT vorgestellt. LONGINT ist schon schön, aber hat einige Nachteile:

1) Eingabe ist nicht möglich
2) Zahlen werden "verkehrt" in die Reihungen abgelegt
3) Es ist nicnt gegen alle "Eventualitäten abgesichert:

so geht z.B. 990/31, aber nicht 990/30, die Multiplikation gent für viele Bs nicht, die auf 0 enden (z.B 990*9811 geht, 990*9810 nicht).
Meine Version(LANGZAHL) liegt bei. Es enthält noch einige überflüssige Konstante, GROESSER und GLEICH hätte man vereinen können, etc. DIe Eingabe erfolgt über eine ZK, was auf 80 Ziffern beschränkt, aber das ist für die üblicnen Zwecke wohl nicht wesentlich. Anstelle einer NORMIERUNG wird der jeweilige Übertrag gleich berücksichtigt, bei Subtrahieren habe ich absichtlich auch toleriert, daß eine negative Zahl herauskommen kann; leichter wäre es natürlich gewesen, würde man immer die kleinere von der größeren Zahl abziehen. Ich hoffe, alle nötigen Kontrollen eingebaut zu naben.

Zu MULT und DIVID muß man einen Kommentar geben, doch ich habe nur Zeit, auf die Literatur zu verweisen: diese "schnellen" Algorithmen für Multiplikation und Division, die natürlich auch unter normalen Bedingungen gehen, stammen aus WIRTH: Systematisches Programmieren (Teuber,1978).

Prof.Dr.G.Snatzke

```
)
)L
  10 PROGRAM LANGZAHL;
  15 (*$L-  ------  *)
  20 CONST
  25   NMAX 80;
  30   PL '0';MI '-';BLK ' ';CR CHR(13);
  35 TYPE
  40   ZAHL ARRAY[0..NMAX] OF INTEGER;
  45     (* Z[0] ist Laenge *)
  50 VAR
  55   X,Y,Z:ZAHL;
  60   I:INTEGER;
  65   C,CC,V:CHAR;
  70 (*  ---- -- ------- *)
  75 PROCEDURE EIN(VAR Z:ZAHL);
  80  (* liest eine lange Zahl ein, Z: werden IN 0 gewandelt  *)
  85 VAR
  90   I,J:INTEGER;
  95   C:CHAR;
 100  ZK:ARRAY[1..NMAX] OF CHAR;
 105 BEGIN
 110  WRITE(CR,'#: ');READLN;
 115 READ(ZK);
 120 I:=0;
 125 REPEAT
 130 I:=I+1;
 135 IF ZK[I]=' ' THEN ZK[I]:='0';
 140 UNTIL (ZK[I]=CHR(0)) OR (I=NMAX);
 145 IF ZK[I]=CHR(0) THEN I:=I-1;
 150 Z[0]:=I;
 155 FOR J:=I DOWNTO 1 DO Z[NMAX-I+J]:=ORD(ZK[J])-ORD('0');
 160 FOR J:=NMAX-I DOWNTO 1 DO Z[J]:=0;
 165 END;
 170 (* ---------- ---          *)
 175 PROCEDURE VORO(VAR W:ZAHL);
 180 VAR
 185   I:INTEGER;
 190 BEGIN
 195 I:=0;
 200 REPEAT
 205   I:=I+1
 210 UNTIL (W[I]<>0) OR (I=NMAX);
 215 IF I=NMAX THEN W[0]:=1 ELSE W[0]:=NMAX-I+1;
 220 END;
 225 (* ------------------  *)
 230 PROCEDURE LAENGER(A,B:INTEGER; VAR C:INTEGER);
 235  (* C wird das groessere von A,B,   oder gleich A=B   *)
 240 BEGIN
 245  IF A<B THEN C:=B ELSE C:=A;
 250 END;
 255 (* ----------- ---          *)
 260 PROCEDURE PLUS(X,Y:ZAHL; VAR Z:ZAHL);
 265 VAR
 270   UE,C,I,J:INTEGER;
 275 BEGIN
 280 UE:=0;
 285 LAENGER(X[0],Y[0],C);
 290 FOR I:=1 TO NMAX-C DO Z[I]:=0;
 295 FOR I:=NMAX DOWNTO NMAX-C+1 DO BEGIN
 300  J:=X[I]+Y[I]+UE;
 305  Z[I]:=J MOD 10; UE:=J DIV 10;
 310                        END;
 315 Z[0]:=C;
 320 IF UE<>0 THEN BEGIN
 325 IF C<NMAX THEN BEGIN Z[NMAX-C]:=UE; Z[0]:=C+1; END
 330 ELSE WRITELN('Uebertrag verloren');
 335 END;
 340 END;
 345 (*  ------------------- *)
 350 PROCEDURE INKREM(VAR W:ZAHL);
 355 (* inkrementiert W um 1  *)
 360 VAR
 365   I,J:INTEGER;
 370   Q:ZAHL;
 375 BEGIN
 380 Q[0]:=1; FOR I:=1 TO NMAX-1 DO Q[I]:=0; Q[NMAX]:=1;
 385 PLUS(Q,W,W);
 390 END;
 395 (* --------------------- *)
 400 PROCEDURE MINUS(X,Y:ZAHL; VAR Z:ZAHL);
 405  (* Z = X-Y *)
 410
 415 VAR
 420   C,I,J,UE:INTEGER;
 425   A,B:ZAHL;
 430 BEGIN
 435  LAENGER(X[0],Y[0],C);
 440 FOR I:=1 TO NMAX-C DO Z[I]:=0;
 445 UE:=0; V:=LS;
 450 FOR I:=NMAX DOWNTO NMAX-C+1 DO BEGIN
 455  J:=X[I]-Y[I]-UE;
 460   IF J<0 THEN BEGIN
 465    Z[I]:=J+10; UE:=1; END
 470    ELSE BEGIN
 475    Z[I]:=J; UE:=0; END;
 480         END;
 485 IF UE=1 THEN BEGIN
 490  V:=MI;
 495  FOR I:=NMAX DOWNTO NMAX-C+1 DO BEGIN
 500  Z[I]:=(9+UE-Z[I]) MOD 10;
 505  IF ((Z[I]=0) AND (UE=1))THEN UE:=1 ELSE UE:=0;
 510       END;
 515   END;
 520 VORO(Z);
 525 END;
 530 (* ---------------- --- *)
 535 PROCEDURE SCHREIB(V:CHAR;Z:ZAHL;C:CHAR);
 540  (* L/K:mit/ohne fuehrende 0 *)
 545 VAR
 550   I,N:INTEGER;
 555 BEGIN
 560 IF C IN ['l','k'] THEN BEGIN
```

```
565 WRITE(V,1);
570  IF C '1' THEN R: 1 FILE U  NIL..1  TO1;
575 FOR I:  N TO NMAX DO WRITE(  (( 1 ));
580   END;
585 END;
590 (* ---                    *)
595 FUNCTION NICHTO(W:ZAHL):BOOLEAN;
600 VAR
605   I:INTEGER;
610   B:BOOLEAN;
615 BEGIN
620 B: FALSE;
625 FOR I:=1 TO NMAX DO
630  IF (NOT B) THEN IF W[I]<>0 THEN B: TRUE;
635 NICHTO:=B;
640 END;
645 (* ------ - -          *)
650 PROCEDURE DIV2(VAR W:ZAHL);
655 (* W DIV 2  *)
660 VAR
665   I,J,R,UE:INTEGER;
670 BEGIN
675  UE:=0;
680  FOR I: NMAX-W[0]+1 TO NMAX DO BEGIN
685   J:=W[I]+UE;
690   R:=J MOD 2;
695   J:=J DIV 2;
700   W[I]:=J;
705   IF R 1 THEN UE: 10 FILE UE. 0,
710    .                        END.
715 VORO(W);
720 END;
725 (* ---------             *)
730 PROCEDURE MULT(X,Y:ZAHL;VAR Z:ZAHL);
735 VAR
740   A,B:ZAHL;
745   I,J:INTEGER;
750 BEGIN
755  FOR I:=1 TO NMAX DO Z[I]:=0;
760  A:=X; B:=Y;
765 WHILE NICHTO(A) DO BEGIN
770  IF ODD(A[NMAX]) THEN PLUS(Z,B,Z);
775  DIV2(A);
780  PLUS(B,B,B);
785  END;
790 VORO(Z);
795 END;
800 (* ---------------    *)
805 FUNCTION GROESSER(A,B:ZAHL):BOOLEAN;
810  (* TRUE falls A> B  *)
815 VAR
820   Q:ZAHL;
825 BEGIN
830 MINUS(A,B,Q);
835 IF V='-' THEN GROESSER:=FALSE ELSE GROESSER:=TRUE;
840 END;
845 (* ----------- -          *)
```

```
850 FUNCTION GLEICH(A,B:ZAHL):BOOLEAN;
855 VAR
860   Q:ZAHL;
865 BEGIN
870  MINUS(A,B,Q);
875  GLEICH:= NOT NICHTO(Q);
880 END;
885 (* ---- ---------- *)
890 PROCEDURE DIVID(X,Y:ZAHL;VAR Z:ZAHL);
895 VAR
900   I,J:INTEGER;
905   Q,V:ZAHL;
910 BEGIN
915 ZIO(Z);
920 FOR I:=1 TO NMAX DO Z[I]:=0;
925 Q:=X; V:=Y;
930  WHILE GROESSER(Q,V) DO PLUS(V,V,V);
935   WHILE (NOT GLEICH(V,Y)) DO BEGIN
940    PLUS(Z,Z,Z);
945    DIV2(V);
950    IF GROESSER(Q,V) THEN BEGIN
955     MINUS(Q,V,Q); INKREM(Z); END;
960    END;
965 END;
970 (* ---------------------- *)
975 FUNCTION WAHL:CHAR;
980 VAR
985   C:CHAR;
990 BEGIN
995 WRITELN(CR,CR);
1000 REPEAT
1005  WRITELN('a (addieren)',CR,'d (dividieren)',CR,'m (multiplizieren)');
1010 WRITE('s (subtrahieren): ');READLN;READ(C); UNTIL C IN ['a','d','m',
1015 WAHL:=C;                                                    's'];
1020 END;
1025 (* ---------------------- *)
1030 (* ========== HP ========== *)
1035 BEGIN
1040 WRITE(CHR(12),'Ausgabe K(ur2)/I(ang): ');READLN;READ(CC);
1045 REPEAT
1050 V:=LS;
1055  EIN(X);
1060  EIN(Y);
1065 CASE WAHL OF
1070   'a': BEGIN PLUS(X,Y,Z);C:='+'; END;
1075   'd': BEGIN DIVID(X,Y,Z);C:=':';END;
1080   'm': BEGIN MULT(X,Y,Z);C:='*'; END;
1085   's': BEGIN MINUS(X,Y,Z);C:='-';END
1090  END;
1095 WRITE(CHR(12));
1100 SCHREIB(LS,X,'K');
1105 WRITE(LS,C,LS);
1110 SCHREIB(LS,Y,'K');
1115 WRITE(' = ');
1120 SCHREIB(V,Z,CC);
1125 WRITELN(CR,CR,CR);

1130 UNTIL FALSE;
1135 END.
```

LANGZAHL LANGZAHL LANGZAHL

APROPOS TEXTVERARBEITUNGSPROGRAMM

Im Klubinfo Nr.4 war,so meine ich ein Leckerbissen zu finden.den sich bestimmt schon viele wünschten und der durch gemeinsame Arbeit sicherlich noch schmack= hafter gemacht werden kann.Gemeint ist das Programm EDITOR von O.Steinmeyer,das einen guten Grundstock für eine Minimaltextverarbeitung darstellt und für den täglichen Hausgebrauch (Briefe,kleine Artikel e.t.c.) völlig ausreichend ist.Zumal,wenn man bedenkt,was sonst so an sogenannten Textverarbeitungspro= grammen von Händlern angeboten wird,mal abgesehen von professionellen Pro= grammen wie z.B. WORDSTAR,die aber einerseits umfangreiche Peripherie und Soft= ware erfordern (CP/M,Doppelfloppy,80-Zeichen Karte) und andererseits auch für einen ganz anderen Anwenderkreis gedacht sind.

Da ich mit einem MZ-731 und angeschlossenem ITHO 8510A Drucker arbeite,inter= essierte mich das vorgestellte Programm.Dieser Texteditor mußte doch ohne allzu großen Aufwand für jeden extern angeschlossenen Centronics-Drucker verfügbar zu machen sein,wobei selbstverständlich die Druckersteuerung (Wahl der Schriftgr., Schriftart,Zeilenvorschub e.t.c.) durch den Anwender per Dialog möglich sein sollte.

Da ich - wie wohl die meisten,die einen externen Drucker besitzen - so gut wie gar nicht mit dem eingebauten Plotter arbeite,schmiß ich zunächst mal alle Routinen zur Plotteransteuerung raus.Um das Programm universell zu halten,kann man sie natürlich auch drinlassen und dann mit ein paar zusätzlichen statements per Dialog zwischen Plotter- und Druckerausgabe umschalten.Nun,wie gesagt,ich schmiß sie raus,da ich den Plotter ohnehin nicht zur Textverarbeitung benutze. Die im folgenden vorgestellten Änderungen lassen die Möglichkeiten jedes extern angeschlossenen Druckers erst so richtig zur Geltung kommen.Als Beispiel möge dieser Beitrag dienen,der bis auf die Programmlistings ausschließlich mit dem modifizierten Textprogramm geschrieben wurde.

Zu erwähnen ist noch,daß für einen angeschlossenen Drucker mit Centronics= schnittstelle keineswegs eines der teuer auf dem Markt angebotenen Interfaces oder Verbindungskabel benötigt werden.Ein einfaches Flachbandkabel mit den ent= sprechenden Steckern versehen und freischwebend in die STROBE Leitung eingelö= tetem Inverter SN7404 ist völlig ausreichend. (siehe auch Beitrag von Bruno Volkmer in Klub-Info Nr.2)

Die Codeumwandlung wird hier vom Textprogramm erledigt und jeder,der schon ein= mal einen Lötkolben in der Hand gehabt hat,kann sich solch ein Verbindungskabel mit einem maximalen Kostenaufwand von DM 25.- bis 30.- selbst basteln. Für die weniger Erfahrenen auf dem Gebiet erscheint demnächst eine ausführliche Bauan= leitung.Es ist einfach nicht einzusehen,warum sich dank SHARP's eigenwilliger Centronics Schnittstelle und dem sonderbaren 'ASCII' Code einige Leute eine goldene Nase an derartigen Interfaces und Kabeln verdient. (Beim MZ-800 hat man da ja seitens SHARP inzwischen - Gott seis gepriesen - etwas dazugelernt.)

Doch nun zu den Modifikationen:

Es wurden die Prozeduren MODETN
MODETS
MODEGR
GPRINT
HSET
LINE
MOVE und
DINA4 gestrichen.Die Prozedur MINI wurde er= setzt durch die Prozedur DRUCKAUS,die noch dahingehend erweitert wurde,daß nun nach dreimaliger Fehleingabe der Anfangs- und/oder Endzeile des auszugebenden Textes wieder ins Hauptmenü verzweigt wird.Wie schon erwähnt kann man der Uni= versalität wegen die den Plotter betreffenden Prozeduren auch drin lassen,muß dann aber über eine zusätzliche Abfrage in der Prozedur DRUCKE entweder auf Plotter- oder Druckerausgabe umschalten.

Ein kleines Hindernis war natürlich mal wieder SHARP's dreimal bescheuerter sogenannter ASCII-Code.Das Problem wurde durch eine Umwandlungstabelle

ASCTAB:ARRAY[1..26] OF CHAR (globale Variablendeklaration)

sowie einen Puffer

ZEILE:ELEMENT (lokale Variable der Prozedur DRUCKAUS)

gelöst.

Bei Druckerausgabe wird nun zunächst jede zu druckende Zeile in den Puffer ZEILE übertragen.Anschließend wird ZEILE mittels der ORD Funktion auf das Vor= handensein von Kleinbuchstaben überprüft.(Die ORD Funktion liefert in diesem Falle Werte zwischen 96 und 123.)Ist ein Kleinbuchstabe gefunden,so wird vom Wert der ORD Funktion dieses Buchstabens 96 subtrahiert und mit dem Ergebnis i als Index in die Tabelle ASCTAB gegangen und der in ZEILE[j] stehende Code durch den in ASCTAB[i] gefundenen ersetzt.Nach Prüfung bzw. Korrektur aller Zeichen des Puffers ZEILE erfolgt die Ausgabe auf Drucker.

Für eine Anpassung der Druckparameter an andere als den ITHO Drucker sind lediglich die entsprechenden Escape-Sequenzen in der Prozedur DRAP zu ändern,die dem Hand= buch des entsprechenden Druckers zu entnehmen sind.

Wünschenswerte Änderungen bzw. Anregungen zur Entfaltung der eigenen Kreativi= tät wären noch

--- inverse Umschaltung zwischen Groß- und Kleinschrift
d.h. normal Kleinschrift,mit SHIFT Großschrift,da nur so ein flüssiges Schreiben gewährleistet ist

--- Funktion des automatischen Randausgleichs

--- Möglichkeit des Vortrennens

--- Löschen des gesamten Textbuffers

Anbei das bearbeitete Textprogramm als PASCAL Source.Außer der Ausgabe auf Centronics Drucker mit Einstellung der Druckerparameter - wie in meinem letzten Bericht beschrieben - ist jetzt noch die Funktion RANDAUSGLEICH drin,die die Zentrierung einer beliebigen Anzahl von Zeilen übernimmt.Dabei kann die Anzahl Spaces(Leerzeichen) am Ende einer Zeile,ab der kein Randausgleich mehr durchgeführt werden soll,ebenso per Dialog gewählt werden wie die Position des ersten Zeichens einer Zeile,bei der die Zentrierung für den gewählten Bereich beginnen soll.
Weiterhin habe ich die Funktion ZEILE LÖSCHEN dahingehend geändert,daß nicht nur eine Zeile,sondern ebenfalls ein freiwählbarer Bereich ge= löscht werden kann.
Ich meine,daß man mit dem Programm schon jetzt hervorragend arbeiten kann,wie auch dieses Schreiben wiederum beweisen mag.
Ich plane noch folgende weitere Modifikationen:

--- Wiederholte automatische Ausgabe eines Textes auf Drucker (z.B. für Rundbriefe,Adressaufkleber e.t.c.)
---Umschaltung Groß-/Kleinbuchstaben
(für flüssiges Schreiben ist standardmäßig inverse Umschaltung erforderlich!)
---Suchen einer bestimmten Textstelle

Vielleicht fallen dem einen oder anderen noch ein paar weitere nützliche Features ein.

PROGRAMM PROGRAMM PROGRAMM

```
>L
   20 (*$L-*)
   40
   60 PROGRAM EDITOR;
   80 (* TEXTEDITOR AUS SHF CLUB INFO NR.4
  100 O.STEINMEYER
  120
  140 VERSION 2.0      JAN 86
  160 MODIFIZIERT DURCH H.-J.JONAS
  180 IN FOLGENDEN PUNKTEN:
  200 DRUCKERAUSGABE FUER CENTRONICS DRUCKER MIT RANDAUSGLEICH,
  220 LOESCHEN EINER BELIEBIGEN ANZAHL VON ZEILEN,
  240 BEDIENERFREUNDLICHKEIT
  260 *** DIESES PROGRAMM DARF WEDER GEWERBLICH GENUTZT,NOCH OHNE WISSEN
  280 DER VERFASSER AN DRITTE - AUSSER CLUBMITGLIEDER DES SHF -
  300 WEITERGEGEBEN WERDEN! *** *)
  320
  340
  360 CONST ANZAHL 70;
  380       LAENGE 79;
  400       HOME=CHR(21);
  420         NL=CHR(13);
  440         CR CHR(29);
  460         CL CHR(20);
  480 TYPE ELEMENT ARRAY[1..LAENGE] OF CHAR;
  500       LISTE ARRAY[1..ANZAHL] OF ELEMENT;
  520 VAR,  AUSG:CHAR;
  540       ANZ,TAB: INTEGER;
  560       L:LISTE;
  580 *     ASCTAB:ARRAY[1..96] OF CHAR;
  600 *
  620 PROCEDURE LOESCHEN;
  640 VAR N,M:INTEGER;
  660 BEGIN
  680   ANZ:=0;
  700   FOR N:=1 TO ANZAHL DO
  720   FOR M:=1 TO LAENGE DO L[N,M]:=CHR(0);
  740 END;
  760
  780 PROCEDURE INIASC;
  800 BEGIN
  820   ASCTAB[1]:=CHR(#A1);
  840   ASCTAB[2]:=CHR(#9A);
  860   ASCTAB[3]:=CHR(#9E);
  880   ASCTAB[4]:=CHR(#9C);
  900   ASCTAB[5]:=CHR(#92);
  920   ASCTAB[6]:=CHR(#AA);
  940   ASCTAB[7]:=CHR(#97);
  960   ASCTAB[8]:=CHR(#98);
  980   ASCTAB[9]:=CHR(#A6);
 1000   ASCTAB[10]:=CHR(#AE);
 1020   ASCTAB[11]:=CHR(#A9);
 1040   ASCTAB[12]:=CHR(#80);
 1060   ASCTAB[13]:=CHR(#B5);
 1080   ASCTAB[14]:=CHR(#B0);
 1100   ASCTAB[15]:=CHR(#B3);
 1120   ASCTAB[16]:=CHR(#E1);
 1140   ASCTAB[17]:=CHR(#A0);
 1160   ASCTAB[18]:=CHR(#9D);
 1180   ASCTAB[19]:=CHR(#A4);
 1200   ASCTAB[20]:=CHR(#96);
 1220   ASCTAB[21]:=CHR(#A5);
 1240   ASCTAB[22]:=CHR(#A8);
 1260   ASCTAB[23]:=CHR(#A3);
 1280   ASCTAB[24]:=CHR(#9B);
 1300   ASCTAB[25]:=CHR(#BD);
 1320   ASCTAB[26]:=CHR(#A2)
 1340 END;
 1360
 1380 PROCEDURE INFO;
 1400 BEGIN
 1420   PAGE;
 1440   WRITELN('        T E X T P R O G R A M M   ');
 1460   WRITELN('------------  - ----------------------');
 1480   WRITELN;
 1500   WRITELN;WRITELN;
 1520   WRITELN('    Dieses Programm verwaltet eine');
 1540   WRITELN('      Liste von max ',ANZAHL:4,' Zeilen.');
 1560   WRITELN('     Zeilen',CHR(#BB),'nge:',LAENGE:3,' Buchstaben.');
 1580   WRITELN;
 1600   WRITELN('   Bitte d',CHR(#AD),'cken Sie die CR-Taste !');
 1620   READLN;
 1640
 1660 END;
 1680
 1700 PROCEDURE WARTEN;
 1720 VAR I:INTEGER;
 1740 BEGIN
 1760   I:=0;
 1780   REPEAT I:=I+1
 1800   UNTIL I=1000
 1820 END;
 1840
 1860 PROCEDURE RETMENU;
 1880 VAR I:INTEGER;
 1900 BEGIN
 1920   WRITELN('   Es ist noch kein Text editiert!');
 1940   WRITELN(' Ich gehe gleich wieder ins Hauptmen',CHR(#AD),'!');
 1960   FOR I:=1 TO 30 DO WARTEN
 1980 END;
 2000
 2020 PROCEDURE TASTE(VAR T:CHAR);
 2040 BEGIN
 2060   REPEAT T:=INCH;
 2080   UNTIL T<>CHR(0);
 2100   WARTEN
 2120 END;
 2140
 2160 PROCEDURE TABULATOR;
 2180 BEGIN
 2200   PAGE;
 2220   TAB:=LAENGE;
 2240   REPEAT
```

PASTEXT PASTEXT PASTEXT

```
2260        WRITELN;
2280        WRITE('     Leerschritte am Anfang : ');
2300        READ(TAB);
2320    UNTIL (TAB)-0) AND (TAB<LAENGE)
2340 END;
2360
2380 PROCEDURE WAHL(VAR WUNSCH:CHAR);
2400 BEGIN
2420    PAGE;
2440      WRITELN('        T E X T P R O G R A M M    ');
2460    WRITELN('-----------------------------------------');
2480    WRITELN;
2500    WRITELN;
2520    WRITELN('       HAUPTMEN',CHR(#B2));
2540    WRITELN('       ---------');
2560    WRITELN;
2580    WRITELN('       Tabulator setzen.....T');
2600    WRITELN('       Texteingabe..........E');
2620    WRITELN('       Text laden...........L');
2640    WRITELN('       Zeile verbessern.....C');
2660    WRITELN('       Zeile neu schreiben..N');
2680    WRITELN('       Zeilen l',CHR(#BA),'schen.....K');
2700    WRITELN('       Zeile einf',CHR(#AD),'gen.....H');
2720    WRITELN('       Liste Bildschirm.....B');
2740    WRITELN('       Drucken..............D');
2760    WRITELN('       Text speichern.......S');
2780    WRITELN('       Randausgleich........R');
2800    WRITELN('       Programmende.........P');
2820    WRITELN;
2840    WRITE('      bitte w',CHR(#BB),'hlen Sie : ');
2860    READLN;READ(WUNSCH);
2880 END;
2900
2920 PROCEDURE LISTE1;
2940 VAR N,NR,I:INTEGER;
2960         C:CHAR;
2980 BEGIN
3000    PAGE;WRITELN;WRITELN;WRITELN;WRITELN;WRITELN;
3020    IF ANZ=0 THEN RETMENU
3040    ELSE
3060    BEGIN
3080    PAGE;
3100    WRITELN('    B I L D S C H I R M A U S G A B E  ');
3120    WRITELN('-----------------------------------------');
3140    WRITELN;WRITELN;WRITELN;
3160    WRITE('Mit welcher Nr beginnen?');
3180    READLN;READ(NR);
3200    PAGE;
3220    C:='A';
3240    WHILE C<>'F' DO
3260    BEGIN
3280        IF NR<ZAHL THEN
3300        BEGIN
3320            FOR I:=NR TO (NR+23) DO
3340            BEGIN
3360                WRITELN(I:4);
3380                WRITELN(ZEILE[I])
3400            END;
3420            NR:=NR+8;
3440            TASTE(C)
3460        END
3480        ELSE
3500        BEGIN
3520            FOR I:=NR TO ANZ DO
3540            BEGIN
3560                WRITELN(I:4);
3580                WRITELN(ZEILE[I])
3600            END;
3620            TASTE(C);
3640            C:='F'
3660        END
3680    END
3700    END
3720 END;
3740
3760 PROCEDURE UMLAUT(VAR Z:CHAR);
3780 BEGIN
3800    CASE Z OF
3820        '$':Z:=CHR(#AE);
3840        '@':Z:=CHR(#BB);
3860        '#':Z:=CHR(#B9);
3880        '|':Z:=CHR(#BA);
3900        '[':Z:=CHR(#AB);
3920        '_':Z:=CHR(#B2);
3940        '~':Z:=CHR(#AD)
3960    END
3980 END;
4000
4020 PROCEDURE LESE(VAR N:ELEMENT);
4040 VAR Z:CHAR;
4060    I:INTEGER;
4080
4100 BEGIN
4120    FOR I:=1 TO LAENGE DO N[I]:=CHR(0);
4140    FOR I:=1 TO TAB DO N[I]:=' ';
4160    FOR I:=1 TO TAB DO WRITE(' ');
4180    I:=TAB+1;
4200    READLN;
4220    WHILE (I<=LAENGE) AND NOT EOLN DO
4240    BEGIN
4260        READ(Z);
4280        UMLAUT(Z);
4300        N[I]:=Z;
4320        I:=I+1
4340    END
4360 END;
4380
4400 PROCEDURE EINGABE;
4420 VAR M,N,NR:INTEGER;
4440 BEGIN
4460    PAGE;
4480    WRITELN;
4500    WRITE('Mit welcher Zeile anfangen ? ');
4520    READLN;READ(NR);
4540    IF NR<=ANZAHL THEN
4560    BEGIN
```

PROGRAMM PROGRAMM PROGRAMM

PASTEXT PASTEXT PASTEXT

```
4580        WRITELN;
4600        WRITELN('Nach der letzten Zeile 0 eingeben');
4620        WRITELN;
4640        N:=NR-1;
4660        REPEAT
4680            N:=N+1;
4700            FOR M:=1 TO 39 DO WRITE('A');
4720            WRITELN;
4740            WRITELN(N:4,' ');
4760            LESE(LINE);
4780        UNTIL (LIN,1]TAB='0') OR (N=ANZAHL);
4800        IF N=ANZAHL THEN ANZ:=N
4820        ELSE ANZ:=N-1
4840    END
4860 END;
4880
4900 PROCEDURE VERBESSERN;
4920 VAR NR,I1,I:INTEGER;
4940   WUNSCH,Z:CHAR;
4960 BEGIN
4980   PAGE;
5000   WRITELN;
5020   WRITE('     Welche Zeile verbessern? ');
5040   READ(NR);
5060   IF NR<=ANZ THEN
5080   BEGIN
5100        PAGE;
5120        WRITELN(NR:4);
5140        WRITELN(LINE);
5160        I1:=1;
5180        REPEAT
5200            TASTE(WUNSCH);
5220            CASE WUNSCH OF
5240            'J':BEGIN
5260                IF I1+4<LAENGE THEN
5280                  BEGIN
5300                    FOR I:=I1 TO I1+4 DO WRITE(LINE,I1);
5320                    I1:=I1+5;
5340                    WARTEN
5360                  END
5380                END;
5400            ' ':BEGIN
5420
5440                  IF I1<LAENGE THEN
5460                  BEGIN
5480                    WRITE(LINE,I1);
5500                    I1:=I1+1;
5520                    WARTEN
5540                  END
5560                END;
5580            'K':BEGIN
5600                  IF I1<LAENGE THEN
5620                  BEGIN
5640                    FOR I  I1 TO LAENGE-1 DO LINE,I1:=LINE,I1+1;
5660                    WRITE(' ');
5680                    WARTEN
5700                  END
5720                END;
5740            'C':BEGIN
5760                  READLN;READ(Z);
5780                  UMLAUT(Z);
5800                  LINE,I1:=Z;
5820                  WHILE NOT EOLN AND (I1<LAENGE) DO
5840                  BEGIN
5860                    READ(Z);
5880                    UMLAUT(Z);
5900                    I1:=I1+1;
5920                    LINE,I1:=Z;
5940                  END;
5960                  FOR I:=40 DOWNTO (I1 MOD 40)+1 DO WRITE(CHR(8));
5980                  I1:=I1+1
6000                  END;
6020            'I':BEGIN
6040                  READLN;READ(Z);
6060                  UMLAUT(Z);
6080                  FOR I:=LAENGE DOWNTO I1+1 DO LINE,I]:=LINE,I-1];
6100                  LINE,I1]:=Z;
6120                  WHILE NOT EOLN AND (I1<LAENGE) DO
6140                  BEGIN
6160                    READ(Z);
6180                    UMLAUT(Z);
6200                    I1:=I1+1;
6220                    FOR I:=LAENGE DOWNTO I1+1 DO LINE,I]:=LINE,I-1];
6240                    LINE,I1]:=Z
6260                  END;
6280                  FOR I:=40 DOWNTO (I1 MOD 40)+1 DO WRITE(CHR(8));
6300                  I1:=I1+1
6320                END
6340            END;
6360        UNTIL WUNSCH='F'
6380    END
6400 END;
6420
6440 PROCEDURE ZEILENEU;
6460 VAR NR:INTEGER;
6480 BEGIN
6500   PAGE;
6520   WRITELN;
6540   WRITE('     Welche Zeile soll neu ? ');
6560   READ(NR);
6580   IF NR<=ANZ THEN
6600   BEGIN
6620        WRITELN;
6640        WRITELN(LINE);
6660        WRITELN;
6680        WRITELN('Neue Zeile:   ');
6700        WRITELN;
6720        LESE(LINE)
6740    END
6760 END;
6780
6800 PROCEDURE ZAUFRUECK(NR:INTEGER);
6820 VAR I:INTEGER;
6840 BEGIN
6860   FOR I:=NR TO (ANZ-1) DO LINE: LINE+1;
6880   ANZ:=ANZ-1
6900 END;
6920
```

FASTEXT FASTEXT FASTEXT

```
6940 PROCEDURE ZEINFUEG(NR:INTEGER;Z:ELEMENT);
6960 VAR I:INTEGER;
6980 BEGIN
7000   IF ANZ+1 <= ANZAHL THEN
7020   BEGIN
7040       ANZ:=ANZ+1;
7060       FOR I:=ANZ DOWNTO NR+2 DO LI[I]:=LI[I-1];
7080       LINR+1]:=Z
7100   END
7120   ELSE
7140   BEGIN
7160       WRITELN('  Es ist nicht mehr m',CHR(#BA),'glich !');
7180       READLN
7200   END
7220 END;
7240
7260
7280 PROCEDURE ZZEINFUEG;
7300 VAR NR:INTEGER;
7320   Z:ELEMENT;
7340 BEGIN
7360   PAGE;
7380   WRITELN;
7400   WRITE('  Nach welcher Zeile beginnen ? ');
7420   READ(NR);
7440   IF NR<=ANZ THEN
7460   BEGIN
7480       PAGE;
7500       IF NR>0 THEN WRITELN(LINR[1]);
7520 *     WRITELN('x x x x x x x x x x x x x x x x');
7540 -     IF NR<ANZ THEN WRITELN(LINR+1]);
7560       WRITELN;WRITELN;WRITELN;
7580       WRITELN(NR+1:4);
7600       LESE(Z);
7620       REPEAT
7640           ZEINFUEG(NR,Z);
7660           NR:=NR+1;
7680           WRITELN(NR+1:4);
7700           LESE(Z);
7720       UNTIL ZI[TAB]='0'
7740   END
7760 END;
7780
7800 PROCEDURE DRUCKE;
7820 VAR W:CHAR;
7840     CNT,EZEI,LZEI:INTEGER;
7860
7880 PROCEDURE DRUB1;
7900 BEGIN
7920   WRITE(CHR(16))
7940 END;
7960
7980 PROCEDURE DRAP;
8000 VAR PAR:CHAR;
8020 BEGIN
8040   REPEAT
8060   PAGE;
8080   WRITELN('     EINSTELLUNG DER DRUCKERPARAMETER');
8100   WRITELN('-----------------------------------------');WRITELN;
```



```
8120   WRITELN('     Breitschrift EIN ..... 0');
8140   WRITELN('     Breitschrift AUS ..... 1');
8160   WRITELN('     Fettschrift EIN ...... 2');
8180   WRITELN('     Fettschrift AUS ...... 3');
8200   WRITELN('     Schrifttyp ELITE ..... 4');
8220   WRITELN('     Schrifttyp PICA ...... 5');
8240   WRITELN('     Schrifttyp PROPORTIONAL ..... 6');
8260   WRITELN('     Schrifttyp KOMPRIMIERT ..... 7');
8280   WRITELN('     Unterstreichen EIN ..... 8');
8300   WRITELN('     Unterstreichen AUS ..... 9');
8320   WRITELN('     Neue Zeile ..... Z');
8340 (*   WRITELN('     Neue Seite ..... S'); *)
8360   WRITELN;
8380   WRITELN('     DRUCKEREINSTELLUNG BEENDET .. E');
8400   WRITELN;
8420   WRITE('     Bitte w',CHR(#BB),'hlen Sie : ');
8440   READLN;READ(PAR);
8460   DRUB1;
8480   CASE PAR OF
8500   '0':WRITE(CHR(#0E));
8520   '1':WRITE(CHR(#0F));
8540   '2':WRITE(CHR(#1B),CHR(#21));
8560   '3':WRITE(CHR(#1B),CHR(#22));
8580   '4':WRITE(CHR(#1B),CHR(#45));
8600   '5':WRITE(CHR(#1B),CHR(#4E));
8620   '6':WRITE(CHR(#1B),CHR(#50));
8640   '7':WRITE(CHR(#1B),CHR(#51));
8660   '8':WRITE(CHR(#1B),CHR(#58));
8680   '9':WRITE(CHR(#1B),CHR(#59));
8700   'Z':WRITE(CHR(#0A))
8720 (*   'S':WRITE(CHR(#0C))  *)
8740   END;
8760   DRUB1;
8780   UNTIL PAR='E'
8800 END;
8820
8840
8860 PROCEDURE OPTION;
8880 VAR EINDR:CHAR;
8900 BEGIN
8920   PAGE;
8940   WRITELN;WRITELN;WRITELN;
8960   WRITELN('  Wollen Sie die Druckerparameter wie ');
8980   WRITELN('Schrifttyp,Zeilenabstand etc. w',CHR(#BB),'hlen ?');
9000   WRITELN;
9020   WRITE('  JA = Y , sonst irgendeine Taste ');
9040   READLN;READ(EINDR);
9060   IF EINDR='Y' THEN
9080       BEGIN
9100       DRAP;
9120       END
9140 END;
9160
9180 PROCEDURE DRUCKAUS(A,E:INTEGER);
9200 VAR I,K,OZ:INTEGER;
9220   ZEILE:ELEMENT;
9240 BEGIN
```

```
 9260   DRUBI;
 9280   FOR I:=A TO E DO
 9300       BEGIN
 9320       FOR K:=1 TO LAENGE DO
 9340         BEGIN
 9360         ZEILE[K]:=L[I,K];
 9380         END;
 9400       FOR K:=1 TO LAENGE DO
 9420         BEGIN
 9440           CASE ZEILE[K] OF
 9460           CHR(#AD):ZEILE[K]:=CHR(#7D);
 9480           CHR(#BB):ZEILE[K]:=CHR(#7B);
 9500           CHR(#BA):ZEILE[K]:=CHR(#7C);
 9520           CHR(#B2):ZEILE[K]:=CHR(#5D);
 9540           CHR(#B9):ZEILE[K]:=CHR(#5B);
 9560           CHR(#A8):ZEILE[K]:=CHR(#5C);
 9580           CHR(#AE):ZEILE[K]:=CHR(#7E)
 9600         END
 9620       END;
 9640       FOR K:=1 TO LAENGE DO
 9660         BEGIN
 9680         OZ:=ORD(ZEILE[K]);
 9700         IF (OZ>96) AND (OZ<123) THEN
 9720           BEGIN
 9740           OZ:=OZ-96;
 9760           ZEILE[K]:=ASCTAB[OZ]
 9780           END
 9800         END;
 9820       WRITELN(ZEILE)
 9840 {*      IF (I-A+1)MOD 60=0 THEN WRITE(CHR(#0C))  *}
 9860       END;
 9880   DRUBI
 9900 END;
 9920
 9940
 9960
 9980 BEGIN
10000   OPTION;
10020   CNT:=0;EZEI:=0;LZEI:=0;
10040   REPEAT
10060       PAGE;
10080       WRITELN('          AUSGABE AUF DRUCKER ');
10100       WRITELN('-----------------------------------------');
10120       WRITELN;WRITELN;
10140       CNT:=CNT+1;
10160       WRITE('   Erste zu druckende Zeile : ');READ(EZEI);
10180       WRITE('   Letzte zu druckende Zeile: ');READ(LZEI);
10200   UNTIL ((EZEI<=LZEI) AND (EZEI>0) AND (LZEI<=ANZ)) OR (CNT=3);
10220   IF CNT<>3 THEN DRUCKAUS(EZEI,LZEI);
10240 END;
10260
10280 PROCEDURE LADEN;
10300 BEGIN
10320   PAGE;
10340   TIN('TEXT    ',ADDR(L),
10360   TIN('ANZAHL  ',ADDR(ANZ))
10380 END;
```

FASTEXT FASTEXT FASTEXT

```
10400
10420 PROCEDURE SPEICHERN;
10440 BEGIN
10460   PAGE;
10480   TOUT('TEXT    ',ADDR(L),SIZE(L));
10500   TOUT('ANZAHL  ',ADDR(ANZ),SIZE(ANZ))
10520 END;
10540
10560 PROCEDURE CLEIN(VAR ZL,SP:INTEGER);
10580 (* POSITIONIERT DEN CURSOR NACH ZEILE ZL UND SPALTE SP *)
10600 VAR I:INTEGER;
10620 BEGIN
10640  WRITE(HOME);
10660  FOR I:=1 TO ZL DO WRITE(NL);
10680  FOR I:=1 TO SP DO WRITE(CR);
10700  WRITE('     ');
10720  FOR I:=1 TO 5 DO WRITE(CL);
10740  READLN
10760 END;
10780
10800
10820 PROCEDURE ZEILWEG;
10840 VAR I,K,M,NRA,NRE:INTEGER;
10860 BEGIN
10880   PAGE;
10900   WRITELN('    Z E I L E N   L ',CHR(#A8),' S C H E N');
10920   WRITELN('-----------------------------------------');
10940   WRITELN;WRITELN;WRITELN;
10960   IF ANZ=0 THEN RETMENU
10980            ELSE
11000   BEGIN
11020
11040   WRITE('    Erste zu l',CHR(#BA),'schende Zeile : ');I:=5;K:=32;
11060   REPEAT
11080       CLEIN(I,K);READ(NRA);
11100   UNTIL (NRA>0)AND(NRA<=ANZ);
11120   WRITELN;WRITELN;
11140   WRITE('   Letzte zu l',CHR(#BA),'schende Zeile : ');I:=8;K:=32;
11160   REPEAT
11180       CLEIN(I,K);READ(NRE);
11200   UNTIL (NRE>=NRA)AND(NRE<=ANZ);
11220   K:=ANZ;
11240   ANZ:=ANZ-(NRE-NRA)-1;
11260   IF ANZ>0 THEN BEGIN
11280       FOR I:=NRA TO ANZ DO
11300                          BEGIN
11320                          L[I]:=L[NRE+1];NRE:=NRE+1
11340                          END
11360                END
11380            ELSE BEGIN
11400                 FOR I:=1 TO K DO
11420                 FOR M:=1 TO LAENGE DO L[I,M]:=CHR(0)
11440                END
11460   END
11480 END;
11500
11520
```

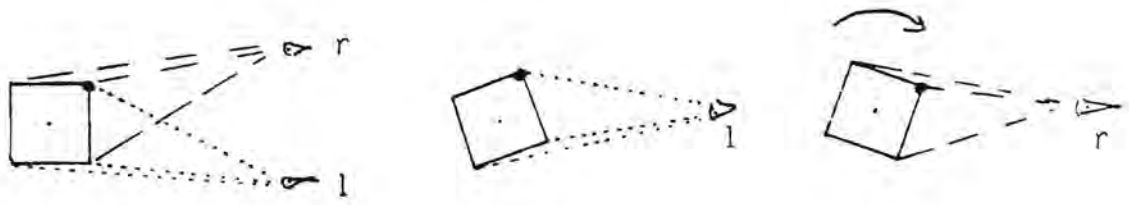
```
11540 (* RANDAUSGLEICH *)
11560 (*   JAN 86      *)
11580 PROCEDURE RANDAUSGL;
11600 VAR   FIRSTLET,ENDZEI:INTEGER;
11620       AKTPOS,MAXSP:INTEGER;
11640      I,K, ERSTE,LETZTE:INTEGER;
11660       ZEICH:CHAR;
11680 BEGIN
11700  PAGE;
11720  WRITELN('       T E X T Z E N T R I E R U N G ');
11740  WRITELN('-------------------------------------');
11760  WRITELN;WRITELN;WRITELN;
11780  IF ANZ=0 THEN RETMENU
11800           ELSE
11820 BEGIN
11840    WRITE('   Erste zu formatierende Zeile : ');I:=5;K:=34;
11860    REPEAT
11880      CLEIN(I,K);
11900      READ(ERSTE);
11920    UNTIL (ERSTE)0)AND(ERSTE(=ANZ);
11940    WRITELN;WRITELN;
11960    WRITE('  Letzte zu formatierende Zeile : ');I:=8;K:=34;
11980    REPEAT
12000      CLEIN(I,K);
12020      READ(LETZTE)
12040    UNTIL (LETZTE)=ERSTE)AND(LETZTE(=ANZ);
12060    WRITELN;WRITELN;
12080    WRITELN(' Leerzeichen am Ende einer Zeile: ');
12100 .  WRITELN('     Zul',CHR(#BB),'ssige Werte % : 2 .. 25');
12120 .  WRITELN('  Bei mehr als  % Leerzeichen am Ende');
12140    WRITELN(' einer Zeile wird diese nicht zentr.!');I:=11;K:=34;
12160    REPEAT
12180      CLEIN(I,K);
12200      READ(MAXSP)
12220    UNTIL (MAXSP)2)AND(MAXSP(25);
12240    FOR I:=1 TO 6 DO WRITE(NL);
12260    WRITELN('Beginn der Zentrierung jeder Zeile');
12280    WRITE('                      Zeichen : ');I:=19;K:=34;
12300    REPEAT
12320      CLEIN(I,K);
12340      READ(FIRSTLET)
12360    UNTIL (FIRSTLET)0)AND(FIRSTLET(20);
12380  FOR I:=ERSTE TO LETZTE DO
12400   BEGIN
12420     K:=80;
12440     REPEAT
12460       K:=K-1;
12480       ZEICH:=LIT,K];
12500     UNTIL (ZEICH)CHR(0)) OR (K=FIRSTLET);
12520     ENDZEI:=K;
12540     AKTPOS:=FIRSTLET;
12560     IF (79-ENDZEI)(MAXSP THEN
12580       BEGIN
12600         WHILE ENDZEI(79 DO
12620           BEGIN
12640             AKTPOS:=FIRSTLET+1;
12660             WHILE (AKTPOS(ENDZEI)AND(ENDZEI(79) DO
12680               BEGIN
12700                 WHILE (LIT,AKTPOS]=' ') AND (AKTPOS(ENDZEI) DO
12720                  BEGIN
12740                   AKTPOS:=AKTPOS+1
12760                  END;
12780                 IF AKTPOS(ENDZEI THEN
12800                  BEGIN
12820                    FOR K:=ENDZEI DOWNTO AKTPOS+1 DO
12840                      LIT,K+1]:=LIT,K];
12860                    LIT,AKTPOS+1]:=' ';
12880                    ENDZEI:=ENDZEI+1;
12900                    AKTPOS:=AKTPOS+2
12920                  END
12940               END
12960           END
12980       END
13000     END
13020   END
13040 END;
13060
13080 BEGIN
13100 (******** HAUPTPROGRAMM ********)
13120   LOESCHEN;
13140   INIASC;
13160   TAB:=0;
13180   INFO;
13200   REPEAT
13220       WAHL(AUFG);
13240       CASE AUFG OF
13260         'T':TABULATOR;
13280         'E':EINGABE;
13300         'C':VERBESSERN;
13320         'R':RANDAUSGL;
13340         'N':ZEILENEU;
13360         'K':ZEILWEG;
13380         'H':ZZEINFUEG;
13400         'B':LISTEB1;
13420         'D':DRUCKE;
13440         'S':SPEICHERN
13460       END;
13480   UNTIL AUFG='P';
13500   PAGE
13520 END.
```

PASTEXT PASTEXT PASTEXT

PROGRAMM PROGRAMM PROGRAMM

Stereoskopische Projektionen.

Vorbemerkung: Beim Sehvorgang wird ein Gegenstand durch die Linse des Auges auf die Netzhaut abgebildet, dort entsteht also eine "zweidimensionale" Projektion. Da die beiden Augen des Menschen aber einen Abstand von etwa 10 cm voneinander haben, so entstehen auf beiden Netzhäuten leicht verschiedene Bilder. Das Gehirn setzt diese beiden zusammen und erst dadurch entsteht der räumliche, "dreidmensionale" Eindruck. Betrachtet man etwa einen kleinen Würfel, der eine Seite genau dem Gesicht zugewandt hat, dann sieht das linke Auge außer der Vorderseite auch noch etwas von der linken Fläche, das rechte Auge etwas von der rechten, wie folgende Skizze andeutet:



Ein Einäugiger könnte diese 2 Bilder nur sehen, wenn er sich um das Objekt herumbewegt, oder wenn (was auf das gleiche herauskommt) sein Auge fix bleibt, aber das Objekt gedreht wird. Nehmen wir an, er habe nur ein linkes Auge, dann muß das Objekt für die zweite Betrachtung im Uhrzeigersinn, also "mathematisch negaiv", gedreht werden. Eine normale Kamera ist nun so ein "Einäugiger"; nimmt man zwei Bilder in der geschilderten Weise auf und blickt dann mit dem linken Auge aufs erste, getrennt davon mit dem rechten aufs zweite, dann werden diese beiden Bilder im Gehirn in gewohnter Weise zusammengesetzt und man erhält einen gleichen Eindruck, als ob man das Objekt tatsächlich räumlich vor sich hätte. Das Problem liegt eigentlich nur in der sauberen Trennung der beiden Bilder, und man kann das z.B. dadurch erreichen, daß man ein Bild grün, das andere rot zeichnet (übereinandergelegt) und dann durch eine Brille betrachtet, das ein grünes und ein rotes "Glas" hat. Dies dürfte jeder schon einmal gesehen haben, weniger bekannt (und technisch aufwendiger) ist, daß man gleiches auch mit sogenanntem polarisierten Licht erreichen kann. Ganz einfach aber geht es auf folgende Weise:

Man legt die beiden Bilder in geringem Abstand vor sich hin und hält ein Pappendeckelblatt so zwischen Nase und diese Bilder, daß das linke Auge wirklich nur das linke Bild sehen kann, das rechte nur das rechte. Verbessern kann man das noch, wenn man kleine Linsen vor den Augen anbringt, damit man schärfer sehen kann. Erfahrungsgemäß gelingt es dennoch nicht allen Menschen, dabei einen räumlichen Eindruck zu erhalten, andererseits gibt es viele, die bei etwas Übung auf den Pappendeckel verzichten können und die Augen einfach auf "unendlich" einstellen.

Die Erfahrung zeigt, daß man den besten räumlichen Eindruck bekommt, wenn man aus etwa 20 cm Entfernung betrachtet und der oben genannte Drehwinkel etwa -7 Grad beträgt, die Entfernung der beiden Einzelbilder etwa 5.5 cm. Es ist tatsächlich ein spannender Moment, wenn man zum erstenmal die beiden Bilder langsam aufeinander zuwandern sieht, und plötzlich springen sie zu einem zusammen, das jetzt "räumlich" ist!

Projektionsarten: von den vielen Möglichkeiten seien nur 2 erwähnt - die Perspektive und die Parallelprojektion. Bei letzterer nehmen wir an, daß alle Lichtstrahlen parallel einfallen, und das läßt sich einfacher programmieren. Bei der Perspektive hingegen treffen sich alle Strahlen in einem einzigen Punkt, und das hat z.B. zur Folge, daß die parallelen Schienen eines Bahngeleises auf Bildern immer enger zueinander kommen, wenn die Entfernung wächst, oder daß Linien, die parallel zueinander horizontal liegen (z.B. Fensterreihe im 1. und im 2. Geschoß eines Hauses, dessen vordere Kante uns zugewandt ist), sich im Horizont schneiden.

Prinzip des Programms: Das Programm wurde geschrieben, um Moleküle "sehen" zu können. Man kann mit geeigneten Streuungsmethoden die Koordinaten der einzelnen Atome erhalten, doch sind diese normalerweise nicht in dem üblichen Cartesischen Koordinatensystem angegeben werden (drei Achsen, X,Y Z, die alle aufeinander senkrecht stehen, und auf denen die Längeneinheit gleich groß ist), sondern in Form von "fraktionellen" Koordinaten: die drei Achsen können dabei auch andere Winkel miteinander einschließen, und die Längeneinheiten a,b,c mögen auch unterschiedlich sein. Diese Programmteile sind nicht herausgenommen worden, können aber dann, wenn man nur cartesische Koordinaten zur Konstruktion des geometrischen Gebildes, das man dreidimensional betrachten möchte, benutzt, ohne Schaden entfernt werden. Falls nicht, setzt man einfach a = b = c = 1 ein.

Programmaufbau: im HP (Hauptprogramm) wird erst der Inhalt von #48E auf 1 gesetzt, was bewirkt, daß das Ausschreiben auf Band nur einmal (und nicht wie sonst bei SHARP zweimal) erfolgt. Mit der REPEAT...UNTIL FALSE - Schleife wird das "Menü" der Abfrage immer wieder durchlaufen.

WAHL ist als Funktion aufgebaut und dient der Abfrage; Zeile 908 sieht ungewohnt aus: ich benutze das "Volkmer-pascal" (bei dem CONTROL-K nur für die Buchstaben hin- und herschaltet, nicht für die übrigen Zeichen) mit dem eingebauten Zusatz, daß viele (nicht alle) der vordefinierten Worte in Kleinschrift eingegeben werden können, dann aber beim Auslisten in Großbuchstaben erscheinen; im übrigen kann man - außer natürlich in Zeichenketten - Groß und Kleinbuchstaben beliebig mischen. Das "in" aus E (in) wird also automatisch in Großbuchstaben verwandelt, was mir aus rein ästhetischen Gründen nicht paßt, und so muß ich diese beiden z.B. über ihre Ordnungszahlen eingeben ('i','n' hätte auch funktioniert!).

Bei der ersten Erstellung einer Koordinatendatei wird man von der Tastatur eingeben müssen; dies erledigt das UP "LIESXYZ". Wird dort N eingegeben,wird "geputzt" (alle Koordinaten werden genullt). Um aber bei irrtümlichem Abbruch des Programms Wiederstart ohne Datenverlust zu ermöglichen, wird dieses Nullen bei Eingabe K übersprungen. Alle Zahleneingaben erfolgen über die Funktion ZAHL, die ich vor einiger Zeit in der CP beschrieben hatte. Um Korrekturen zu ermöglichen wird eine Liste aller Werte am KO ("Kathodenstrahloszillograph"=Bildschirm) ausgegeben. Diese Ausgabe "läuft" nur, wenn die Leertaste gedrückt wird (Zeile 392). Zur Korrektur wird erst die Zahl des Punktes eingegeben,dann X oder Y oder Z. Eingabe von "0" beendet diese Korrekturabfrage. Die "Zelldimensionen" a,b,c wird man normalerweise jeweils mit 1 belegen, die Frage ob cartesisch oder nicht mit "1" beantworten. DOBRA (= gut, polnisch) ist die Funktion, die auf richtige Eingabe abfragt. Sollte ein nicht-cartesisches System vorliegen - die Zeilen von 462 bis 472 rechnen auf rechtwinkliges System um (die entsprechenden Formeln mag man einer mathematischen Formelsammlung entnehmen, bei Interesse kann ich sie auch ableiten).

Das Objekt wird als Strichmodell gezeichnet, als nächstes muß man also die nötige Verbindungslinie dadurch aufbauen, daß man die Punkte der Reihe nach eingibt, die konsekutiv durchlaufen werden sollen. Dies geschieht im UP "BIND". Um nicht manche "Bindungen" mehrmals zeichnen zu müssen und um auch springen zu können gilt die Konvention: die Zeichenfeder ist am Papier aufgesetzt, wenn die Nummer des Punktes positiv eingegeben wird, dagegen wird die Feder abgehoben, so die Nummer des Punktes negativ eingegeben wird. Für jeden Verbindungsstrich hat man einen neuen Endpunkt zu definieren, mit dem ersten Anfang gibt es also um einen Punkt mehr, als Linien vorhanden sind, daher die Beschränkung in Zeile 716 und BM+1 in Zeile 720. Eingabe und Korrektur erfolgen wie bei LIESXYZ. Damit bei der späteren Rotation des Objekts nicht auch eine seitliche Verschiebung erfolgt wird dann der Punkt, bei dem die Zeichnung begonnen wird, zum Nullpunkt gemacht, indem seine Koordinaten von allen subtrahiert werden.

Das Zeichnen selbst besorgt das UP ZEICH. Es fragt einen Faktor ab, mit dem alle Koordinaten multipliziert werden, schaltet den Streifenplotter ein mit GRAPH, wählt die Federfarbe (PLFOAB), "hupft" auf die Mitte des Streifens (HUPF) und setzt dort den neuen Ursprung für die Feder (NEU00). Man springt auf die Projektion des ersten Punkts im Linienzug und zeichnet dann mit der Feder am Papier (FOA) oder nicht (HUPF) die (X,Y)-Projektion des Objekts (Zl.784). Danach hupft die Feder um 5.5 cm (Zl.788) in negativer Y-Richtung weiter und mit GRAPHAUS ist

man wieder aus dem Graphik-Modus ausgestiegen. Eine Einheit des Plotters bei Koordinatenangaben entspricht etwa 0.02 cm, daher der Wert für Y in Zl 288.

Rotation des Objekts: Da die beiden Projektionen in die (X,Y)-Ebene mit dem Plotter nur untereinander gezeichnet werden können muß zur Erstellung der 2. Projektion um -7 Grad um die X-Achse gedreht werden. Nun genügt es aber nicht, nur um diese Achse zu drehen, weil man das Objekt vielleicht aus einer anderen Richtung besser betrachten kann. Man muß also Rotation um alle 3 Achsen zulassen und dazu benutzt man die EULERschen Winkel: man dreht zuerst um den Winkel PHI um die X-Achse, dann um den Winkel PSI um die neue Y-Achse (auch die Y-Achse ist ja mitgedreht worden!), schließlich um den Winkel CHI um die neue Z-Achse. Dies erledigt das UP EULERWI, das aus den (immer unverändert gelassenen!) Koordinaten XYZ jene in der Reihung XYZR macht. Die Transformationsformeln sind wieder einer mathematischen Formelsammlung entnommen.

In praxi geht man dann so vor: zunächst errechnet man sich die Koordinaten in möglichst einfacher Weise, dann dreht man das Objekt so lange um die drei EULER-Winkel, bis man eine ansprechende Projektion gefunden hat, und dann zeichnet man eine zweite Projektion dazu, bei der der Rotationswinkel PHI um 7 Grad negativer gewählt wurde (PSI und CHI bleiben gleich). Der Faktor zum Zeichnen ist dabei so zu wählen, daß man einen guten räumlichen Eindruck bekommt, also z.B. so, daß eine "Normallänge" des Objekts etwa 7 mm lang wird (dies kann man natürlich auch durch die Wahl von a,b,c erreichen), doch ist das etwas abhängig vom Betrachter.

Möchte man seine Daten aufheben, dann muß man auf Band ablegen, und da schreibt man sich einen Text dazu mit GENTEXT. Ich habe es so programmiert, daß man bis zu 80 Zeichen eingeben kann, und diese auch eventuell in mehreren Zeilen zur besseren Formatierung. Ablegen und Einlesen wird mit EINAUS erledigt, wobei ich die Form DATElxxx gewählt habe (8 Zeichen!), xxx ist eine 3-stellige Nummer, die aber in ASCII-Format und nicht als Ganzzahl eingelesen wird.Wie legt man nun 3 völlig verschiedene Dateien (Text in ASCII, Bindungszug als Reihung von Ganzzahlen, und Koordinaten als Reihung von Realzahlen) ab, ohne jeweils einen neuen Vorspann schreiben zu müssen? Nun - im Speicher stehen nur Hexadezimalzahlen,was sie bedeuten, das "weiß" das Programm aus den Vereinbarungen am Programmkopf,nur beim Ausschreiben oder Einlesen von Dateien ist diese Bedeutung nicht gefragt!So wurden in Zeile 28 die drei Variablen in bestimmter Reihenfolge definiert. Erinnern wir uns daran: bei der Vereinbarung eines Typs wird noch kein Platz reserviert, das passiert erst, wenn man die Variablen definiert. Dabei beginnt Pascal "von hinten", die Reihung XYZ belegt also die nötige Zahl von Plätzen bis hinauf zu #FFFF, davor schließt sich BINDUNG an, davor liegt noch TEXT. Beginnen wir so beim Ausschreiben ab der Adresse von TEXT, und bemessen nur die Länge richtig, dann werden hintereinander TEXT, BINDUNG und XYZ ausgeschrieben in eine einzige Datei! Bei TEXT und BINDUNG lohnt es sich nicht, nur den wirklich benutzten Teil abzulegen (Um das zu erreichen, könnte man diese Daten in XYZ einschreiben, was zwar gegen das Gesetz der Typenvereinbarung verstoßt, durch einige einfache UPs aber umgangen werden kann; darüber ein andermal mehr), wohl hingegen bei XYZ, da man pro Punkt 3 Realzahlen hat, von denen jede wieder 4 Speicherplätze belegt.So kommt man auf die in Zeile 870 berechnete Länge im Befehl TOUT. Um die Zahl der Punkte auch übergeben zu können wird diese (N) als B(0) (Zl. 866) mitgeführt.Zum Einlesen braucht man nur die Anfangsadresse in TIN anzugeben, die Länge steht ja im Vorspann. Danach fischt man sich wieder N und BM heraus (Zl. 876, 878) und schreibt den text am KO aus. Schließlich muß man noch von der Reihung XYZ in die XYZR übertragen und kann zeichnen.

Da man immer in der Papierstreifenmitte mit dem Zeichnen beginnt kann es passieren, daß eine Zeichnung nicht voll aufs Papier passt. Mit ANFANG kann man einen anderen Punkt als Ursprung definieren; hierbei kann allerdings bei der Rotation eine seitliche Verschiebung eintreten, so daß man letztlich besser einen neuen Bindungszug eingibt. Da ANFANG nur auf XYZR wirkt, muß man an jede Rotation immer wieder ANFANG anschließen, bevor man zeichnet!

Über die Änderungen, die nötig sind, um anstelle einer solchen Parallelprojektion eine Perspektive zu erhalten, wird später berichtet.

VIEL SPASS!

Das Programm faßt NMAX Atome (für jedes 3 Koordinaten x,y und z), und BMAX Bindungen (also Verbindungslinien zwischen je 2 gewählten Atomen. Sie können mit dem Programm die Koordinaten einlesen, den Anfang an ein beliebiges Atom geben, den Bindungszug einschreiben (wird ein Zielatom negativ eingegeben, dann wird beim entsprechenden Strich die Mine abgehoben, also ein Sprung gemacht, und keine Linie geschrieben) Dann können Sie das Objekt in allen drei Raumrichtungen drehen (das sind die "Eulerschen Winkel", über die ich einmal schrieb), und Sie können sich einen Text dazu schreiben, wenn Sie ein solches Programm ablegen wollen, um es einmal später wieder von der Kassette einzulesen. Wenn Sie anfangen, dann müssen Sie also erst einmal Daten erzeugen, dann müssen Sie einen Bindungszug machen. Dann erst hat es einen Sinn, eine Zeichnung zu versuchen, deren Maßstab einzugeben ist. Wenn Sie die Abstände so lange machen, wie ich es etwa angegeben habe, und wie Sie es auch auf den beiliegenden Beispielen sehen, dann wird der Streifen im Zeichner gerade soweit nach unten geschoben, daß der richtige Abstand für die Betrachtung mit einer Stereobrille erzeugt wird. Bedenken Sie, daß die Originaldaten nie verändert werden, also jede Drehung wieder von der ursprünglichen Figur ausgeht. Wenn Sie um die Winkel A,B,C gedreht haben, dann müssen Sie danach um A-7, B,C drehen, um die zweite nötige Zeichnung zu kriegen. Paßt Ihnen schon die Ausgangslage, dann brauchen Sie natürlich nur um -7° zu drehen (Winkel PHI). Wenn Sie eine Rotation eingeben, dann müssen Sie dort, wo keine solche Drehung gewünscht wird, dann 0 eintippen.
Und jetzt los: Sie wollen einen Würfel richtig dreidimensional sehen. Er hat 8 Ecken, und sie numerieren durch von 1..8. Legen Sie der Einfachheit halber die Eckpunkte so, daß Sie als Koordinaten haben:

1. 0/0/0	5. 0/0/1
2. 1/0/0	6. 1/0/1
3. 1/1/0	7. 1/1/1
4. 0/1/0	8. 0/1/1

Damit Sie dann prüfen können, was Sie drin haben, kommt eine Tabelle, die aber nur "läuft", wenn Sie die Leertaste drücken (und solange Sie diese gedrückt halten). Sie tippen dann 0, wenn alles richtig ist, sonst können Sie über die Nummer des "falschen" Atoms korrigieren. Die Zelldimensionen beantworten Sie einfach mit 1,1,1. Auf die Frage Cartesisch oder nicht antworten Sie mit 1, und dann kommt der Bindungszug: Zeichnen Sie sich den Würfel auf und fahren Sie alle Kanten ab; das kann dann etwa so aussehen (nicht vergessen, auch wieder auf den Ausgangspunkt zurückzufahren!): 1-2-3-4-1-5-6-2-6-7-3-7-8-4-8-5, also haben Sie 15 Bindungen! Kontrolle und Korrektur wie gehabt. Wenn Sie jetzt zeichnen, kommt nur ein Quadrat heraus, da Sie aus der "negativen Z-Richtung" projizieren.
Nun drehen Sie drauflos, ändern eventuell den Maßstab, oder auch den Punkt, bei dem Sie zu zeichnen anfangen. Und bald werden Sie ein "Gefühl" dafür bekommen, wie man den Würfel schön plastisch herausbekommt. Und dann konstruieren Sie sich ein anderes Gebilde, z.B. eine Wendeltreppe mit Stufen, oder was auch sonst immer. Die allerschönsten legen Sie sich dann auf Kassette ab, für Demonstrationen für später und wenn Besuch kommt! Vielleicht "packt" es auch Ihren Instruktur, und Ihre Mitstreiter beim Programmieren. (Dazu noch ein Hinweis: die Funktion DOBRA wiederum wurde nicht steirisch, aber polnisch benannt! DOBRA heißt GUT!).

ZKEIN kennen Sie schon, und ich habe schon erklärt, warum man damit arbeiten soll, damit man nicht durch einen Fehler in die nächste Koordinate hineinschreibt (vgl. meine Anmerung zu Ihrem J/N).

ZAHL ist Ihnen auch geläufig, LIESXYZ ist ein allgemein brauchbares UP, das auch eine Koordinatentransformation macht, falls nötig. Die Mathematik dazu erkläre ich dann, wenn Sie Fragen dazu haben.

EULERWI ist das UP, das die Drehungen der drei Koordinaten besorgt.

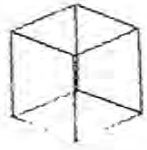
PLPAKET haben Sie auch schon, und GENTEXT generiert einen Text bis zu 80 Zeichen, die "wild" hintereinander geschrieben werden können.

BIND macht den Bindungszug, Zeich die Zeichnungen, ANFANG verschiebt eventuell alle Koordinaten auf einen neuen Ursprung. Wieder handelt es sich dabei nur um den zweiten Satz von Koordinaten, die den "rotierten" Atomen zugehören, da der Originalsatz nie verändert wird.

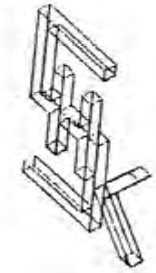
EINAUS regelt das Einlesen und Ausschreiben; die Funktionen TIN und TOUT sind Ihnen bestens vertraut, und Sie werden sehen (Zeile 942), daß nicht der gesamte Reihungsbereich ausgeschrieben wird, sondern nur soviel, wie gerade benötigt wird. Das gleiche gilt für den Bindungszug.

```
>L
  10 (*$L---------- OKTNEUV8    ---- *)
  20 PROGRAM OKTNEUV8;
  30   (*    V8, 10.06.85    *)
  40 CONST
  50    NMAX=200; (* max. Atomzahl *)
  60    BMAX=250;(* max. Bindungszahl*)
  70 TYPE
  80    ZK=ARRAY[1..40]OF CHAR;
  90    TEXT=ARRAY[1..80]OF CHAR;
 100    XYZ=ARRAY[0..NMAX,1..3]OF REAL;
 110    XYZR=ARRAY[1..NMAX,1..3]OF REAL;
 120    BINDUNG=ARRAY[0..BMAX] OF INTEGER;
 130 VAR
 140    P:XYZ;B:BINDUNG;TE:TEXT;
 150 (**** diese Reihenfolge nicht veraendern!!!! ****)
 160    PR: XYZR;
 170    FR,X,Y,Z,XO,YO,ZO,R: REAL;
 180    BM,NU,I,J,K,L,N: INTEGER;
 190    W:CHAR;
 200 (* ------------------*)
 210 FUNCTION DOBRA:BOOLEAN;
 220 VAR
 230    C.CHAR;
 240 BEGIN
 250  WRITE('Gut=g: ');READLN;READ(C);
 260  IF C='g' THEN DOBRA:=TRUE ELSE DOBRA:=FALSE;
 270 END;
 280 (* ---------------------- *)
 290 PROCEDURE ZKEIN(VAR Q:ZK;            295            VAR K:INTEGER);
 300 (* Q ist die eingelesene ZK, K die Anzahl der Zeichen darin  *)
 310 VAR
 320    I:INTEGER;
 330 BEGIN
 340 WRITE('#: ');READLN;READ(Q);
 350 K:=0;
 360 FOR I:=1 TO 40 DO
 370  IF ORD(Q[I])>0 THEN K:=I;
 380 END;
 390 (* -------- Fkt. ZAHL --------- *)
 400 FUNCTION ZAHL:REAL;
 410 CONST
 420    NMAX=20;
 430 TYPE
 440    ZK=ARRAY[1..NMAX]OF CHAR;
 450 VAR
 460    VO,(*VORZEICHEN DER ZAHL*)
 470    VK,(*VORKOMMASTELLEN*)
 480    NK,(*NACHKOMMASTELLEN*)
 490    ZL,(*ZK-LAENGE*)
 500    I,J,K    :INTEGER;
 510    X,P      :REAL;
 520    PK,(*  . *)
 530    EX,(* E *)
 540    VE,(* neg.Vorz.v.E *)
 550    GUT      :BOOLEAN;
 560    Z        :ZK;
 570 BEGIN
 580 REPEAT
 590                              (* R1 *)
 600 REPEAT
 610                              (* R2 *)
 620 READLN;GUT:=TRUE;READ(Z);ZL:=0;
 630 FOR I:=1 TO NMAX DO BEGIN
 640                              (* 1 *)
 650   IF (GUT AND (ORD(Z[I])>0))THEN ZL:=ZL+1
 660     ELSE GUT:=FALSE; END;
 670                              (*-1 *)
 680   GUT:=TRUE;
 690 FOR I:=1 TO ZL DO BEGIN
 700                              (* 2 *)
 710  IF GUT THEN BEGIN
 720                              (* 3 *)
 730   GUT:=Z[I] IN ['+','-','.','E','0'..'9'];
 740    IF NOT GUT THEN
 750   WRITELN(CHR(13),CHR(7),CHR(7),'Nochmals!');
 760                       END;    (*-3 *)
 770                   END;        (*-2 *)
 780 UNTIL GUT;
 790                              (*-R2 *)
 800 (* ------------------------- *)
 810 PK:=FALSE; EX:=FALSE; VK:=0;X:=0;J:=0;
 820  IF Z[1]='-' THEN VO:=-1 ELSE VO:=1;
 830  IF Z[1] IN ['+','-'] THEN BEGIN
 840                              (* 4 *)
 850     ZL:=ZL-1;
 860  FOR I:=1 TO ZL DO
 870   Z[I]:=Z[I+1];   END;
 880                              (*-4 *)
 890  FOR I:=1 TO ZL DO BEGIN
 900                              (* 5 *)
 910   IF (NOT EX AND (Z[I]='E'))THEN BEGIN
 920                              (* 6 *)
 930    EX:=TRUE; GUT:=EX=PK; NK:= I-VK-2;        END;
 940                              (*-6 *)
 950  IF NOT PK THEN BEGIN
 960                              (* 7 *)
 970   PK:=Z[I]='.'; IF NOT PK THEN VK:=VK+1;
 980        END; END;
 990                            (*-7/-5 *)
1000 IF PK THEN BEGIN
1010                              (* 8 *)
1020  ZL:=ZL-1;
1030   FOR I:=VK+1 TO ZL DO
1040    Z[I]:=Z[I+1]; END;
1050                              (*-8 *)
1060 IF EX THEN BEGIN
1070                              (* 9 *)
1080   ZL:=ZL-1;
1090    FOR I:=VK+NK+1 TO ZL DO Z[I]:=Z[I+1];
1100 K:=VK+NK+1;
1110 IF Z[K]='-' THEN VE:=FALSE ELSE VE:=TRUE;
1120 IF Z[K] IN ['+','-'] THEN BEGIN                    (* 10 *)
1130   ZL:=ZL-1;
1140    FOR I:=K TO K+1 DO
1150     Z[I]:=Z[I+1];  END;
```

```
                              (*-10 *)
  FOR I:=1 TO ZL DO BEGIN
                              (* 11 *)
   IF GUT THEN GUT:=(Z[I] IN ['0'..'9']);
            END;
                              (*-11 *)
   IF GUT THEN BEGIN
                              (* 12 *)
    J:=ORD(Z[K])-ORD('0');
     IF ZL-K>1 THEN GUT:=FALSE;
     IF ZL-K=1 THEN
    J:=10*J+ORD(Z[K+1])-ORD('0');
      END;  END
                          (*-12/-9 *)
      ELSE NK:=ZL-VK;
FOR I:=1 TO ZL DO
 IF GUT THEN GUT:=Z[I] IN ['0'..'9'];
 IF GUT THEN BEGIN
                              (* 13 *)
  P:=1;
   FOR I:=VK+NK DOWNTO 1 DO BEGIN          (* 14 *)
   X:=X+P*(ORD(Z[I])-ORD('0'));
   P:=P*10;
                         END;
                              (*-14 *)
  FOR I:=1 TO NK DO
   X:=X/10;
IF EX THEN BEGIN
                              (* 15 *)
IF ((X)3.4) AND (J)38)) THEN GUT:=FALSE
  ELSE  BEGIN
                              (* 16 *)
    FOR I:=1 TO J DO BEGIN
                              (* 17 *)
     IF VE THEN X:=X*10  ELSE X:=X/10;
       END;  END;
                         (*-17,-16 *)
       END;  END;
                         (*-15/-13 *)
IF (NOT GUT) THEN WRITELN('Nochmals!');
UNTIL GUT;
ZAHL:=VO*X;
END;
(* ------- LIESXYZ    -
LIEST KOORDINATEN EIN, DIE AUCH
FRAKTIONELL SEIN KOENNEN, UND
TRANSFORMIERT, FALLS NOETIG, VON
SCHIEFWINKLIGEM AUF EIN CARTESI-
SCHES KOORDINATENSYSTEM. DAS
ARRAY XYZ [0..NMAX,1..3] OF REAL
MUSS IM HP DEFINIERT SEIN. ---- *)
(* ----                    ---- *)
PROCEDURE LIESXYZ(VAR N:INTEGER;    VAR P:XYZ);
CONST
   Q=1./4533E-2;
VAR
   A2,A3,A4,A5,A6,A,B,C,X,Y,Z:REAL;
   I,J:INTEGER;
```



```
   S,WAHL:CHAR;
BEGIN
REPEAT
WRITE('Neu=n, Korr.=k: ');READLN;READ(S);
UNTIL (S='n') OR (S='k');
 REPEAT
  WRITE('Atomanzahl:  ');
  N:=ROUND(ZAHL);
 UNTIL N<=NMAX;
  IF S='n' THEN BEGIN   (* Neu *)
  FOR I:=1 TO N DO BEGIN
      WRITE('X(',I,'): ');P[I,1]:=  ZAHL;
      WRITE('Y:      ');P[I,2]:=   ZAHL;
      WRITE('Z:      ');P[I,3]:=  ZAHL;
  END;   END;          (*Neu*)
REPEAT
  WRITELN(CHR(13),CHR(13));
WRITELN('LS drueckenn!':26);
     FOR I:=1 TO N DO BEGIN
      WRITE(I:3,':');
   FOR J:=1 TO 3 DO
       WRITE(P[I,J]:11:4);
     REPEAT UNTIL INCH=' ';
     WRITELN;
   END;
  WRITELN('Korrektur? Tippe Atom-#,');
  WRITELN('0 beendet Abfrage.');
  I:=ROUND(ZAHL);
 IF I >0 THEN BEGIN
  FOR J:=1 TO 3 DO
      WRITE(P[I,J]:9:4);
      WRITELN;
  REPEAT
    WRITE('x,y oder z: ');
    READLN;READ(WAHL);
  UNTIL (WAHL='x')OR (WAHL='y') OR   (WAHL='z');
    WRITE('  : ');
  CASE WAHL OF
      'x': P[I,1]:=ZAHL;
      'y': P[I,2]:=ZAHL;
      'z': P[I,3]:=ZAHL
  END;            END;
UNTIL I=0;
 REPEAT   WRITE('Zelldimensionen: a= ');A:=ZAHL;
    WRITE('   b:  ');B:=ZAHL;
      WRITE('   c:  ');C:=ZAHL;
UNTIL DOBRA;
  FOR I:=1 TO N DO BEGIN
      P[I,1]:=A*P[I,1]; P[I,2]:=B*P[I,2]; P[I,3]:=C*P[I,3];
   END;
  WRITE('Cartesisch = 1, nicht = 2 :'); J:=ROUND(ZAHL);
  IF J<>1 THEN BEGIN
  REPEAT
WRITE('Alpha =  ');A:=ZAHL;
WRITE('Beta  =  ');B:=ZAHL;
WRITE('Gamma =  ');C:=ZAHL;
 UNTIL DOBRA;
  A2:=COS(Q*C);A3:=COS(Q*B);A4:=SIN(Q*C);A5:=(COS(Q*A)-A2*A3)/A4;
```



```
2320   A6:=SQRT(1-SQR(COS(Q*A))-SQR(A3)-SQR(A2)+2*COS(Q*A)*A3*A2)/A4;
2330   FOR I:=1 TO N DO BEGIN
2340    P[I,1]:=P[I,1]+A2*P[I,2]+A3*P[I,3];
2350    P[I,2]:=P[I,2]*A4+P[I,3]*A5; P[I,3]:=P[I,3]*A6;
2360   END;  END;
2370   FOR I:=1 TO N DO
2380     FOR J:=1 TO 3 DO
2390      PR[I,J]:=P[I,J];
2400   END;
2410    (* ------------ *)
2420 (* ***** EULERWI ******
2430 TRANSFORMIERT DIE KOORDINTEN DES
2440 ARRAYS XYZ [0..NMAX,1..3] AUF
2450 SOLCHE DES BEREICHS XYZR [1..NMAX,1..3], BEIDE OF REAL, UND IM HP
2460 VORHER ZU DEFINIEREN. ******* *)
2470    (* ------------ *)
2480 PROCEDURE EULERWI(N:INTEGER;
2490                    P:XYZ;
2500                VAR PR:XYZR);
2510    CONST   Q=1.745329E-2;
2520 TYPE
2530   TRANSF=ARRAY[1..3,1..3]OF REAL;
2540    SINCOS=ARRAY[1..6] OF REAL;
2550    EULWI=ARRAY[1..3] OF REAL;
2560   HILF=ARRAY[1..3] OF REAL;
2570 VAR
2580   I,J,K:INTEGER;
2590   X,Y,Z:REAL;
2600   A:TRANSF;R:HILF;SC:SINCOS;
2610    WI:EULWI;
2620 BEGIN
2630  REPEAT
2640   WRITELN( 'KOORDINTENROTATION');
2650   WRITE('PHI: ');WI[1]:=ZAHL;
2660   WRITE('PSI:  ');WI[2]:=ZAHL;
2670   WRITE('CHI: ');WI[3]:=ZAHL;
2680   FOR I:=1 TO 3 DO
2690      WRITE(WI[I]:8:3);
2700      WRITELN;
2710  UNTIL DOBRA;
2720 FOR I:=1 TO 3 DO WI[I]:=-WI[I];
2730  (*       Leider !!!!      *)
2740   FOR I:=1 TO 3 DO BEGIN
2750       WI[I]:=WI[I]*Q;
2760       SC[I]:=SIN(WI[I]);
2770       SC[I+3]:=COS(WI[I]);
2780    END;
2790 A[1,1]:=SC[5]*SC[6];A[2,1]:=-SC[5]*SC[3];A[3,1]:=SC[2];
2800 A[1,2]:=SC[4]*SC[3]+SC[1]*SC[2]*SC[6]; A[2,2]:=SC[4]*SC[6]-SC[1]*SC[2]
3];
2810 A[3,2]:=-SC[1]*SC[5];
2820 A[1,3]:=SC[1]*SC[3]-SC[2]*SC[4]*SC[6];
2830 A[2,3]:=SC[1]*SC[6]+SC[4]*SC[2]*SC[3]; A[3,3]:=SC[4]*SC[5];
2840   FOR I:=1 TO N DO BEGIN
2850    FOR J:=1 TO 3 DO BEGIN
2860     R[J]:=P[I,J]; PR[I,J]:=0;
2870    END;
```

PROGRAM

PROGRAM

```
2880    FOR J:=1 TO 3 DO BEGIN
2890     FOR K:=1 TO 3 DO
2900      PR[I,J]:=PR[I,J]+A[J,K]*R[K];
2910    END;
2920   END;
2930 END;
2940 (* ------------------- *)
2950 PROCEDURE GRAPH;
2960  BEGIN
2970   WRITE(CHR(16),CHR(2),CHR(13));
2980 END;
2990 (* ------------------- *)
3000 PROCEDURE GRAPHAUS;
3010  BEGIN
3020   WRITE('A',CHR(16),CHR(13));
3030 END;
3040 (* ------------------- *)
3050 PROCEDURE NEU00;
3060  BEGIN
3070   WRITE('I',CHR(13));
3080 END;
3090 (* ------------------- *)
3100 PROCEDURE PLFOAB(J:INTEGER);
3110  BEGIN
3120 WRITE('C',(J MOD 4),CHR(13));
3130 END;
3140 (* ------------------- *)
3150 PROCEDURE HUPF(X,Y:REAL);
3160 VAR IX,IY:INTEGER;
3170  BEGIN
3180   IX:=ROUND(X);IY:=ROUND(Y);
3190   IF(ABS(IX)<481)AND(ABS(IY)<1000)  THEN WRITE('M',IX,',',IY,CHR(13)
3200 END;
3210 (* ------------------- *)
3220 PROCEDURE FOA(X,Y:REAL);
3230 VAR IX,IY:INTEGER;
3240 BEGIN IX:=ROUND(X);IY:=ROUND(Y);
3250 IF(ABS(IX)<481)AND (ABS(IY)<1000)   THEN WRITE('D',IX,',',IY,CHR(13))
3260 END;
3270 (* ------------------- *)
3280 PROCEDURE GENTEXT;
3290 VAR
3300    I,K,KO:INTEGER;
3310    Q:ZK;
3320 BEGIN
3330  WRITELN(CHR(13),'Text',CHR(13),'-bis zu 80 Zeichen,auch mehrere Zei
;
3340 WRITELN('@ beendet Eingabe':28);
3350 REPEAT                  (* R 1 *)
3360 FOR I:=1 TO 80 DO TE[I]:=CHR(0);
3370   KO:=0;
3380  REPEAT                 (* R 2 *)
3390  ZKEIN(Q,K);
3400   FOR I:=1 TO K DO IF (KO+I)<81 THEN TE[KO+I]:=Q[I];
3410   KO:=KO+K+1;
3420   IF Q[K]='@' THEN TE[KO-1]:=CHR(0   ) ELSE TE[KO]:=CHR(13);
3430   IF KO>=40 THEN WRITELN('   ******** noch',80-KO:3,' Z. *****');
3440  UNTIL ((Q[K]='@') OR (KO>80) );                          (* -R2 *)
```

```
------------------------------

X(1): 0 (CR)
Y   :  0 (CR)
Z   :   0 (CR)

X(2): 1 (CR)
Y   :  0 (CR)
Z   :   0 (CR)

X(3): 1 (CR)
Y   :   1 (CR)
Z   :    0 (CR)

X(4): 0 (CR)
Y   :   1 (CR)
Z   :    0 (CR)

X(5): 0 (CR)
Y   :   0 (CR)
Z   :    1 (CR)

X(6): 1 (CR)
Y   :   0 (CR)
Z   :    1 (CR)

*X(7): 1 (CR)
 Y   :   1 (CR)
 Z   :    1 (CR)

X(8): 0 (CR)
Y   :   1 (CR)
Z   :    1 (CR)

---------------------------
```

Nach dem diese Eingaben erfolgt sind, erscheint 'LS druekken!' d.h. die Leertaste(Space). Es werden alle bis dahin eingegebenen Werte aufgelistet und falls Korrekturen notwendig sind, können sie noch durch geführt werden. Ist alles OK, gibt man eine '0' ein.

Bei der Frage Zelldimension: wird jeweils für a,b,c der Wert 1 eingegeben.

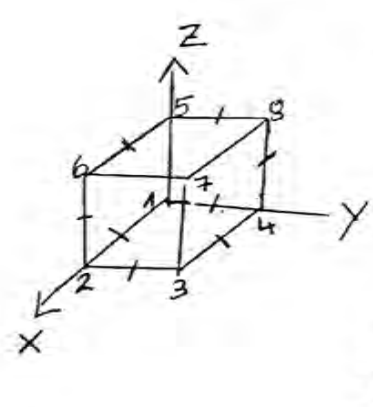
Alles bis dahin richtig ?
Dann kann man GUT mit 'g' beantworten, sonst wird die Abfrage wiederholt.

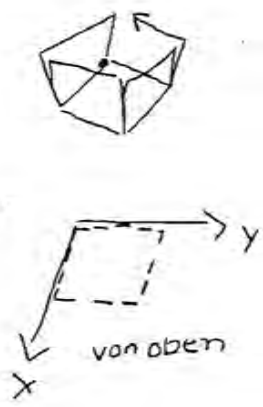
Bei Cartesisch = 1, nicht = 2 : 1 (CR) eingeben.

Da nach erscheint wieder das Menü.
Jetzt geben wir 'b' ein, denn jetzt muß mitgeteilt werden welche Punkte mit einander verbunden werden sollen.

Hier noch mal am Beispiel des Würfels der Eingabeablauf:

Wir zeichnen den Würfel auf und nummerieren die Eckpunkte (ATOME) von 1 bis 8 durch.
Dann zählen wir die Strecken (Bindungen), die die Eckpunkte verbinden, für den Würfel 15 Linien und schreiben die Koordinaten für X,Y u. Z auf. (Siehe Beispiel)





	X	Y	Z
1	0	0	0
2	1	0	0
3	1	1	0
4	0	1	0
5	0	0	1
6	1	0	1
7	1	1	1
8	0	1	1

OKTNEU meldet sich mit einem Menü:

```
------------------------------

a(nfangspunkt verschieben)
b(indungszug neu)
d(aten neu od. korr)
e(in/aus)
r(otation)
t(ext schreiben)
z(eichnen)

          #:
------------------------------
```

WIR GEBEN 'd' (CR) EIN.

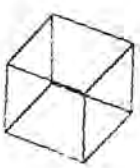
UND 'n' (CR) BEI DER FRAGE NACH 'NEU' oder 'KORR.'.

'Atomanzahl: 8 (CR)

Dann werden die Daten eingeben:

```
3450  WRITELN;
3460  FOR I:=1 TO K0-1 DO WRITE(TE[I]);
3470  WRITELN;
3480 UNTIL DOBRA;          (* -R1 *)
3490 END;
3500 (* ---------------------- *)
3510 PROCEDURE BIND;
3520 (* setzt IN XYZ den Punkt auf 0/0/0, der 1. im Bindungszug   *)
3530 VAR
3540   I,J:INTEGER;
3550 BEGIN
3560 REPEAT
3570 WRITE('Bindungszahl:  ');BM:=ROUND(ZAHL);
3580 UNTIL (DOBRA AND (BM<BMAX));
3590 WRITELN('Atomreihenfolge, neg. fuer Federabheben:  ');
3600 FOR I:=1 TO BM+1 DO BEGIN
3610   WRITE('#-',I:3,': ');B[I]:=ROUND   (ZAHL);  END;
3620 REPEAT
3630 WRITELN(CHR(13),'Bindungszug, Leertaste druecken!');
3640 FOR I:=1 TO BM+1 DO BEGIN
3650   WRITELN(I:3,B[I]:5);
3660    IF ABS(B[I])>N THEN WRITELN(        'TEPP!';20);
3670   REPEAT UNTIL INCH=' ';
3680                  END;
3690 WRITE('Korrig.? Tippe #, 0 beendet:  ');READ(J);
3700 IF J<>0 THEN B[J]:=ROUND(ZAHL);
3710 UNTIL J=0;
3720 FOR J:=1 TO 3 DO P[0,J]:=P[B[1],J];
3730 FOR I:=1 TO N DO FOR J:=1 TO 3 DO       P[I,J]:=P[I,J]-P[0,J];
3740 END;
3750 (* ------------------------ *)
3760 PROCEDURE ZEICH;
3770   (* aus XYZR-Koordinaten ! *)
3780 VAR
3790   I,J,BO:INTEGER;
3800   X,Y,XO,YO,FR:REAL;
3810 BEGIN
3820 WRITE(CHR(13),'Faktor zum Zeichnen:  ');FR:=ZAHL;
3830 GRAPH;
3840 PLFOAB(1);
3850 HUPF(240,0);NEUOO;
3860  XO:=FR*PR[B[1],1];YO:=FR*PR[B[1],2];
3870  HUPF(XO,YO);
3880 FOR I:=2 TO BM+1 DO BEGIN
3890   BO:=ABS(B[I]);
3900   X:=FR*PR[BO,1];
3910   Y:=FR*PR[BO,2];
3920    IF B[I]>0 THEN FOA(X,Y) ELSE HUPF(X,Y);
3930           END;
3940 HUPF(-240,-274);
3950  GRAPHAUS;
3960 END;
3970 (* ------------------- *)
3980 PROCEDURE ANFANG;
3990 (* verschiebt nur "rotierte" Koordinaten XYZR  *)
4000 TYPE
4010   NULLR=ARRAY[1..3]OF REAL;
4020 VAR
4030   POR:NULLR;
4040   NU,I,J:INTEGER;
4050 BEGIN
4060  REPEAT
4070  WRITE(CHR(13),'Ursprung bei Atom:  ');NU:=ROUND(ZAHL);
4080  UNTIL NU<=N;
4090 FOR I:=1 TO 3 DO
4100   POR[I]:=PR[NU,I];
4110 FOR I:=1 TO N DO
4120  FOR J:=1 TO 3 DO
4130   PR[I,J]:=PR[I,J]-POR[J];
4140 END;
4150 (* ---------------------- *)
4160 PROCEDURE EINAUS;
4170 TYPE
4180   NAME=ARRAY[1..8]OF CHAR;
4190 VAR
4200   I,J,K:INTEGER;
4210   S: CHAR;
4220   Z:NAME;
4230 BEGIN
4240 REPEAT
4250  WRITE('Ein=E, Aus=A: ');READLN;    READ(S);
4260 UNTIL (S='A') OR (S='E');
4270 REPEAT
4280  WRITE('Kennzahl fuer Daten (3-stellig!): ');READLN;
4290   FOR I:=6 TO 8 DO READ(Z[I]); WRITELN;
4300   FOR I:=6 TO 8 DO WRITE(Z[I]); WRITELN;
4310 UNTIL DOBRA;
4320  Z[1]:='D';Z[2]:='A';Z[3]:='T';Z[4]:='E';Z[5]:='I';
4330  B[0]:=N;
4340 IF S='A' THEN
4350  TOUT(Z,ADDR(TE[1]),80+2*(1+BMAX)+(N+1)*12)
4360 ELSE BEGIN
4370 FOR I:=0 TO BMAX DO B[I]:=0;
4380   TIN(Z,ADDR(TE[1]));  N:=B[0]; I:=0; WHILE B[I]<>0 DO BEGIN
4390 BM:=I-1;I:=I+1; END;
4400 WRITELN(CHR(13),CHR(13));
4410  FOR I:=1 TO 80 DO                    IF ORD(TE[I])<>0 THEN K:=I;
4420  FOR I:=1 TO K DO WRITE(TE[I]);
4430         END;
4440  WRITELN(CHR(13),CHR(13));
4450 FOR I:=1 TO N DO FOR J:=1 TO 3 DO PR[I,J]:=P[I,J];
4460 END;
4470 (* ---------------------- *)
4480 FUNCTION WAHL:CHAR;
4490 VAR
4500   C:CHAR;
4510 BEGIN
4520 REPEAT
4530  WRITELN(CHR(13),'a (nfangspunkt verschieben)',CHR(13),'b (indungszug
');
4540  WRITELN('d (aten neu od. korr.)',CHR(13),'e (',CHR(#69),CHR(#6E),
4550  '/aus)',CHR(13),'r (otation)');
4560  WRITELN('t (ext schreiben)',CHR(13),'z (eichnen)');
4570 WRITE(CHR(13),'#:':12);
4580  READLN; READ(C);
4590 UNTIL (C='a') OR (C='b') OR (C='d') OR (C='e') OR (C='r') OR (C='t')
='z');
```

```
 4600 WAHL:=C;
 4610 END;
 4620 (* ======== HP ========== *)
 4630 BEGIN
 4640 POKE(#48E,CHR(1));
 4650 REPEAT
 4660  CASE WAHL OF                    'a': ANFANG;
 4670   'b': BIND;
 4680   'd': LIESXYZ(N,P);
 4690   'e': EINAUS;
 4700   'r': EULERWI(N,P,PR);
 4710   't': GENTEXT;
 4720   'z': ZEICH
 4730  END;
 4740 UNTIL FALSE;
 4750 END.
10000 PROGRAM T;
10005 BEGIN
10010 POKE(#13FC,#1400);
10020 POKE(#13FB,CHR(#C3));
10030 END.
>
>
```



Die folgenden Eingaben werden immer mit <CR> abgeschlossen.

Bindungszahl: 15

GUT=g g

Atomreihenfolge, neg. für Federabheben:

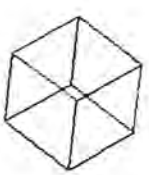
(d.h., daß der Plotterstift abhebt und zum nächsten Eingabepunkt fährt ohne einen Strich zu ziehen.)

```
#- 1: 1
#- 2: 2
#- 3: 3
#- 4: 4
#- 5: 1      (das untere Quadrat wird geschlossen)
#- 6: 5
#- 7: 6
#- 8: 2
#- 9: 6
#-10: 7
#-11: 3
#-12: 7
#-13: 8
#-14: 4
#-15: 8
#-16: 5
```

Bei Bindezug Leertaste druecken! (Werden noch einmal die Eingabewerte zur Kontrolle und Korrektur aufgelistet.)

Danach landen wir wieder im Menü.
Geben jetzt 'a' und 1 ein. Damit bestimmen wir, daß die Zeichnung bei 1 beginnt.
Wir bleiben im Menü und geben 'z' ein.
Bitte darauf achten, daß Papier im Plotter ist, daß Farbmine (blau) funktioniert und auf "Plotter" geschaltet ist, (besonders wichtig bei externem Drucker).Falls Pokebefehle für den externen Drucker eingeben worden sind, müssen diese unbedingt rückgängig gemacht werden.

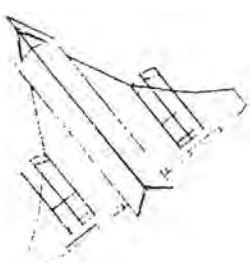
.Faktor zum Zeichnen: 50
Sollte jetzt alles richtig eingegeben sein, bekommen wir die erste Zeichung.
Allerdings werden wir erstmal nur ein Quadrat zusehen bekommen, denn wir schauen bei der Voreinstellung genau von oben auf den Gegenstand.
Um jetzt den Würfel dreidimensional sichtbar zu machen, müssen wir die Zeichnung rotieren lassen.
Also geben wir vom Menü aus 'r' ein.



PHI : 40
PSI : 30
CHI : -40

GUT=g, g

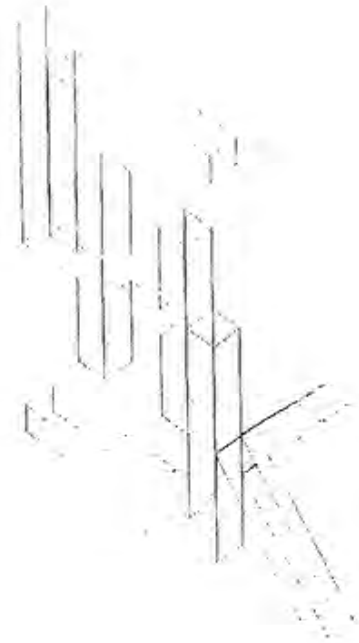
Von Menü aus wieder 'z' aufrufen, Zeichenfaktor eingeben.... und ab geht es.

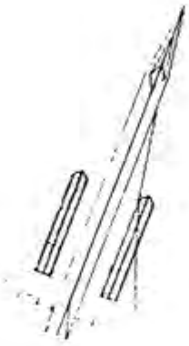


EINGABE-DATEN FÜR DAS FLUGZEUG:

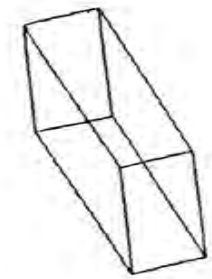
ATOMANZAHL: 77

Nr.	X	Y	Z
1	10.5	1	0
2	12	11	0.3
3	12	48	0.3
4	9	48	0.3
5	9	11	0.3
6	10.5	5	0.5
7	11.5	11	0.5
8	10.5	13	1.5
9	9.5	11	0.5
10	10.5	43	0
11	10.5	48	1.5
12	10.5	9	1.5
13	12	15	0.3
14	14	27	0.3
15	17	36	0.3
16	19	41	0.6
17	19	45	0.6
18	17	45	0.3
19	12	45	0.3
20	9	45	0.3
21	4	45	0.3
22	2	45	0.6
23	2	41	0.6
24	4	36	0.3
25	7	27	0.3
26	9	15	0.3
27	10.5	38	1.5
28	10.5	44	4
29	10.5	45	5
30	10.5	47.5	5
31	10.5	47.5	1.5
32	10.5	47.5	4.8
33	10.5	46.8	4.8
34	10.5	46.8	1.7
35	10.5	47.5	1.7
36	16.8	45	0.3
37	16.8	44	0.3
38	14.3	44	0.3
39	12.3	44	0.3
40	12.3	45	0.3
41	14.3	45	0.3
42	8.7	45	0.3
43	8.7	44	0.3
44	6.7	44	0.3
45	4.2	44	0.3
46	4.2	45	0.3
47	6.7	45	0.3





48	15	26	0.3
49	16	27	0.3
50	16	41	0.1
51	14	41	0.1
52	14	27	0.3
53	14	27	-0.3
54	15	26	-0.3
55	16	27	-0.3
56	16	41	-0.3
57	14	41	-0.3
58	14	37	0.3
59	16	37	0.3
60	15	41	0.1
61	15	41	-0.3
62	6	26	0.3
63	7	27	0.3
64	7	41	0.1
65	5	41	0.1
66	5	27	0.3
67	5	27	-0.3
68	6	26	-0.3
69	7	27	-0.3
70	7	41	-0.3
71	5	41	-0.3
72	5	37	0.3
73	7	37	0.3
74	6	41	0.1
75	6	41	-0.3
76	15	37	0.3
77	6	37	0.3



======
======

BINDUNGEN: 117

1/2/3/10/4/5/1/6/9/8/7/6/12/9/-12/7/
-12/8/11/3/-11/4/-10/1/-13/14/15/16/17/
18/19/-20/21/22/23/24/25/26/-27/28/29/30/31/-35/
34/33/32/-36/37/38/39/40/-41/38/-42/43/44/45/46/-47/
44/45/-48/49/59/50/51/58/52/48/54/55/56/57/53/
54/61/60/76/48/-58/59/-52/53/-49/55/-50/56/-51/57/-62/
63/73/64/65/72/66/62/68/69/70/71/67/68/75/74/77/
62/-72/73/-66/67/-63/69/-64/70/-65/71

BILDSCHIRMVERWALTUNG

MZ 80

Bildschirmverwaltung des Sharp-MZ-800

Das Videosystem des MZ 800 unterscheidet sich voellig von dem des MZ 700 und dessen Vorgaengern MZ 80 A und MZ 80 K. Aus diesem Grund soll an dieser Stelle eine Beschreibung erfolgen , die ich in spaeteren Briefen fortsetzen werde.

Machen wir uns kurz deutlich, was beim MZ 700 geschah, wenn ein Zeichen auf dem Bildschirm erscheinen sollte:

1. Die Cursorkoordinaten sind gegeben fuer x=Spalte und y = Zeile.
2. Aus diesen Werten muss eine Adresse errechnet werden. Da der 700er 25 Zeilen a 40 Zeichen besitzt, also 1000 Plaetze auf dem Bildschirm frei hat, muss die Y-Koordinate mit 40 multipliziert werden und dann die X-Koordinate addiert werden. Das ergibt dann eine relative Adresse zu Null. Addiert man zum Ergebnis den Wert D0000, so erhaelt man die absolute Bildschirmadresse. Bildschirmadresse ist eigentlich das falsche Wort, denn die CPU schreibt gar nicht auf den Bildschirm, sondern in ein Video-RAM.
3. Es ist nun Aufgabe einer Electronic, die unabhaengig von der CPU arbeitet, folgendes zu tun:

   Interpretiere das Ascii-Zeichen im Video-Ram als eine Startadresse fuer ein ROM, in dem startend ab dieser Adresse 8 Bitmuster liegen, die untereinander auf den Bildschirm projiziert ein Muster ergeben, das z.B. dem Buchstaben "A" entspricht.

   Der Zeitpunkt der Projektion, also der Ort auf dem Bildschirm ergibt sich aus der Lage des Zeichen im VideoRAM.

Es ist also ersichtlich, dass der Programmierer auf diesen Vorgang keinen Einfluss hat.

Arbeitet man beim MZ 800 im 700er Modus, so finden wir ein aehnliches Verfahren. Nur gibt es hier einen entscheidenden Vorteil: die "Electronic" holt sich die 8 Bitmuster fuer ein Zeichen nicht aus einem ROM, sondern aus einem RAM. Natuerlich muss beim Start des Computers dieser Mustervorrat dort hineingeschrieben worden sein. Dafuer sorgt aber das Bootprogramm, so dass sich der Nutzer darum nicht zu kuemmern braucht. Die Zeichen liegen also im Character-RAM, wo sie dann (grosser Vorteil!!) veraendert werden koennen.

Soweit der kurze Ausflug zum 700er. Nun zum 800 mit seiner hervorragenden Grafikmoeglichkeiten.

Adressen: Video-RAM 8000H bis BFFFH. Character-ROM 1000 - 2000H.

Das sind Adressen, die mitten im Anwenderbereich zu liegen scheinen. Tun sie auch, aber auf einer anderen Bank. Mit einem Umschaltbefehl kann erwirkt werden, dass die CPU nicht mehr das normale RAM mit dem eventuellen Programm, sondern das Video-RAM bedient. Dieses ist 16000 Bytes gross, fuer 2000 Zeichen (25*80) stehen je 8 Bytes zur Verfuegung. Und an diese 16000 Bytes kommt

BILDSCHIRMVERWALTUNG

MZ 80

die CPU, und damit auch der Programmierer mit seinem Programm heran. Endlich ist die Moeglichkeit da, Bit fuer Bit, also Punkt fuer Punkt selbst zu setzen.

Doch vorher: wie kommt nun das Zeichen auf den Bildschirm ?

1. Schritt
   Errechnung der absoluten Grafikadresse aus den Cursorkoordinaten y und x:
   8 Grafikzeilen pro Textzeile, jede Textzeile zu 80 Bytes, macht 640 Bytes pro Textzeile. Die Y-Koordinate ist also mit 640 zu multiplizieren und dann einfach x hinzuzaehlen. Zu dieser relativen Adresse ist dann noch 8000H als Anfang des Videospeichers hinzuzuaddieren.
2. Schritt
   Errechnung der Lage der 8 Zeichenmuster im Charact.-ROM
3. Schritt
   Banken, also Umschalten, dass fuer die CPU das Video RAM und das CharacterROM zugaeglich sind.
   (Hier vielleicht schon der Hinweis: Das Programmm, das dieses Banken veranlasst, darf um Himmels Willen nicht im Bankbereich liegen, sondern immer ausserhalb, andernfalls findet die CPU nach Ausfuehrung des Schaltbefehls keine Z-80 Befehle vor , sondern zufaelligen Inhalt des Video-RAMs. Und damit ist der Tod des Programmes schon vorprogrammiert)
4. Schritt
   Aus dem Ch.ROM werden nun schoen brav 8 Bytes als Grafikmuster in das VideoRAM uebertragen, und zwar immer 80 Bytes weiter (eine Grafikzeile ist 80 Bytes lang, soll also unter ein Zeichen ein weiteres geschrieben werden, so ist 80 zur Adresse zuzuaddieren)
5. Schritt
   Ist das erledigt, muss das Video-RAM der CPU wieder entzogen werden. Dies geschieht wieder mit einem Schaltbefehl.
6. Schritt
   Rueckkehr zum aufrufenden Programm

Eine Schritt muss noch vor dem 3. eingefuegt werden. Es muss dem Video-Kontroller gesagt werden, was er eigenlich mit den Bytes machen soll, die jetzt ins RAM geschrieben werden sollen. Dieser Controller nimmt dem Programmierer viel Arbeit ab. Warum muss hier eine Info an einen Controller gehen, wo man doch nur ein Byte "einpoken" will ?

Neue Bytes koennen mit bereits an gleicher Stelle verknuepft werden. Was soll das? Nun, stellen Sie sich vor, Sie wollen genau an Stelle 10 von links einen Punkt setzen. Einzelne Bits koennen nicht ins RAM geschrieben werden, nur ganze Bytes. Also den Wert 0000 0000 an die erste Stelle (braucht nicht, steht schon da, nur zum Verstaendnis der Lokalitaet) und dann 01000000 als zweites Byte. Damit waere der 10. Punkt gesetzt. OK ! Und nun wollen wir den 12. auch noch setzen. Das zweite Byte saehe dann so aus: 00010000. Schreiben wir das ins RAM, dann ist der Punkt an 10. Stelle weg. Ich muss also den neuen Wert mit dem bereits an gleicher Adresse vorliegenden oder-verknuepfen. Dazu muss

eder

der alte Wert erst ausgelesen, dann mit dem neuen oderiert und dann

BILDSCHIRMVERWALTUNG

MZ 80

e rgeschrieben werden.Doppelte Arbeit fuer die CPU! Man kann abert dem Controller sagen , dass alles , was bis zur naechsten Anweisung in den Speicher geschrieben wird, grundsaetzlich mit dem bereits vor handenden oder.ert werden soll. Und man kann sagen, dass nur ein, und auch wirklich nur ein Bit gesetzt wird, und dass auch eine exclusiv-oder-verknuepfung erfolgen soll und .. und. kurz gesagt, eine grosse Sache.

Betreiben wir nun ein kleines Programm, das all das tut, was oben geschrieben wurde. In Maschinensprache selbstverstaendlich, denn nur so kann man den Weg verstehen. Poken vom Basic aus ist kein Programmieren sondern blindes Herumstochern im Sysytem, derlebenfalls noch gefaehrlich fuer das Systemprogramm!

In diesem Programm werden die 16-Bit-Register HL, DE und BC der Z80 CPU sowie der Zweitregistersatz HL', DE', BC' benutzt. Auf diese kann man nur geschlossen mit dem Befehl EXX zugreifen.

```
        ;Programm Print e.r Zeichen auf den 80 Zeichen-Schirm
        ;
PRINT:      LD  HL,(CURSOR)     ;H:=Y, L:=X
            CALL  GRAFADR       ;errechne die Grafikadresse
            CALL  ASCIDISP      ;wandle Ascii in Displaycode
                                ;um, da die Zeichen im Char. Rom
                                ;nach den Videocode abgelegt
                                ;sind.
            CALL ZEICHENAUS     ;ab Grafikadresse eintragen.
            RET
;
        ;aus x und y wird die Grafikadresse errechnet
;
GRAFADR:    PUSH  AF            ;A enthaelt das auszugebende Z.
            LD  D,H             ;in H liegt die Y-Koordinate, sie
                                ;wird fuer spaetere Addition in die
                                ;10er Stelle D gelegt
            LD  A,L             ;X-Koordinate
            LD  H,L             ;auch nach H
            LD  E,0
            LD  H,0             ;H und E auf Null
            ADD HL,HL           ;y*2
            ADD HL,HL           ;*4
            ADD HL,HL           ;*8
            ADD HL,HL           ;*16
            ADD HL,HL           ;*32
            ADD HL,HL           ;*64
            ADD HL,DE           ;DE enthaelt das 256 fache von
                                ;Beispiel: D:=00, E:=05, DE hat
                                ;demnach 0005H. Wird nun die 5
                                ;einfach nach D geschrieben, so
                                ;steht in DE 0500H, also das 100
                                ;fache. Da wir aber hexadezimal
                                ;arbeiten, das 10*16, also 256-
                                ;fache.
                                ;Stand also jetzt 64+256=320
            ADD  HL,HL          ;und * 2 = 640 !!!
            LD   DE,8000h       ;Videoanfang
            ADD  HL,DE          ;zaehle zusammen
            POP  AF
            RET                 ;HL bringt Grafikadresse mit !!!
;
```

```
        ;Umwandlung von Ascii in Display-code ist eine
        ;umfangreiche Tabellenroutine. Es kann die Routine 03BFH
        ;aus dem Monitorprogramm mit den dazugehoerigen Tabellen
        ;genommen werden.
ASCIIBSP:   .............       ;ab 03BFH abschreiben:
        ;
;
        ;Zeichenaus : Zeichen ins Grafik-VRAM
        ;
ZEICHENAUS: DI                  ;sperren moeglicher Interrupts
            EXX                 ;Zweitreg.Satz benutzen
            LD  H,0             ;'100er' auf 0
            LD  L,A             ;L hat nun den Displaycode des
                                ;auszugebenden Zeichens.
            ADD HL,HL           ;*2
            ADD HL,HL           ;*4
            ADD HL,HL           ;*8, da immer 8 Bytes/Zeichen
            LD  A,81H           ;80H ist Anweisung an den V.Gerae-
                                ;Controller, dass er alles, was
                                ;jetzt folgt, ueberschreibend ins
                                ;VideoRAM schreiben soll (Code
                                ;'REPLACE'). Die 1 am Ende sagt
                                ;aus, dass nur mit einer
                                ;Farbebene gearbeitet wird, was
                                ;bei 80 Zeichen und nicht
                                ;aufgeruestetem Video-Ram auch nur
                                ;moeglich ist.
            OUT (0CCH),A        ;und gib Wert an das Write-Format
                                ;Register des Videokontrollers
            LD  DE,.000H        ;Basisadresse des Char.ROMs
                                ;wuerde hier 1800H stehen, waere
                                ;der Zeichensatz aktiv
                                ;elegantere Loesung: SET 4,H
            ADD HL,DE           ;Zeiger auf Zeichen
            EXX                 ;1. CPU Satz mit Graf.adr
            LD  B,8             ;8 Bitmuster pro Zeichen
            LD  DE,80           ;immer 80 hinzuzaehlen
            IN  A,(E0H)         ;danken, jetzt VRAM aktiv
                                ;und Char-ROM
LOOP:       EXX                 ;wieder CharROM Adresse
            LD  A,(HL)          ;Byte vom Char.ROM
            INC HL              ;Zeiger gleich weiter
            EXX                 ;wieder 1.Satz
            LD  (HL),A          ;trage in Video ein
            ADD HL,DE           ;Videoadresse + 80
            DJNZ  LOOP          ;decrementiere den Zaehler B und
                                ;wiederhole das ganze ab loop bis
                                ;b 0 geworden ist.
            IN  A,(E1H)         ;schalte wieder auf User-RAM
            EI                  ;Interrupts wieder ermoeglichen
            RET                 ;und fertig

;
```

BASIC-TIPS

DIR-BYTE von W.Stadlbauer

```
65000 LABEL"DIR":CLS:'21.2.1986
65010 PRINT 5,2 "      DIR-BYTE       W.Stadlbauer          ":DIR:CURSOR23,2:PRINT 1
,7 " Bytes     ":SU=0:SM=0
65020 FORX=$2000TO$27EF STEP32:LB$=HEX$(PEEK(X+20)):GOSUB"ZERO":HB$=HEX$(PEEK(X+
21)):BY$="$"+HB$+LB$:IFPEEK(X)<>0:GOSUB"BYTE":SU=SU+VAL(BY$):SM=SM+VAL(BY$)+70:P
RINTTAB(23)USING"######";VAL(BY$)
65040 NEXT:PRINT:PRINT"----------------------------------------":PRINT"Gesamt: ":
:PRINTTAB(23)USING"######";SU,:PRINTTAB(30)"Bytes":PRINTTAB(22)"(";SM;" + Dir.)"
:MUSIC"-C7":PRINT:PRINT:END
65050 LABEL "BYTE":IFVAL(BY$)<0:BY=VAL(BY$):BY=65536+BY:BY$=STR$(BY)
65060 RETURN
65070 LABEL"ZERO":IFLEN(LB$)=1:LB$="0"+LB$
65080 RETURN
```

Hardcopy eines Programm-Laufes:

Starten mit < GOTO "DIR" >

```
     DIR-BYTE       W.Stadlbauer

DIRECTORY OF 00:          Bytes
 OBJ   'BASIC M2-52008'   34303
 BTX   'MUSIC'             2304
 BTX   'ANALYSE'           12389
 BTX   'BASICALC'          4550
 BTX   'AUTO RUN'          3772
 BTX   'DIR-BYTE'           259
 BTX   'A-R 20.2'          3774
 BTX   'A-R/2'             3774

----------------------------------------
Gesamt:                   65125 Bytes
                        ( 65585 + Dir.)
```

Ein Miniprogramm mit dem in Basic die Daten aus dem Directory gelesen werden können (vor allem der Speicherbedarf der Programme auf der Diskette). Da vielleicht ein größeres Interesse an diesem Programm besteht, habe ich es etwas verfeinert. (Da es ursprünglich für eine Zeile konzipiert war, hat es leider das typische wirre Basic-Aussehen). Mit Hardcopy habe ich auch gleich einen Lauf nach dem Listing vorgeführt.

BASIC-TIPS

Dr.W.Stadlbauer: Basic Tips für QD-Basic 5Z008

Basic-Befehle:

Zusätzlich zu den schon im Kassetten-Basic nicht erwähnten Befehlen (BOOT, HEX$,TRON, TROFF, CLS) existieren noch folgende Basicwörter:

1) CSRH....gibt die horizontale Cursorposition an.
   Format: PRINT CSRH, IF CSRH = ...
2) CSRV....vertikale Cursorposition; Anwendung wie 1)
3) EDIT....ähnlich wie List, Cursor steht aber nach der Zeilennummer.
3) BEEP....ersetzt USR(62)
4) OR und AND müssen durch Blanks von den Argumenten getrennt sein

Benchmarktest: S-Basic - QD-Basic 5Z008 - BBG-Compiler

1) 30.000 mal summieren:	83 - 87 - 19 sek
2) 30.000 mal 'IF':	67 - 70 - 27 sek
3) 30.000 mal 'PEEK':	79 - 88 - 30 sek
4) 100 mal Bildschirm mit POKE füllen:	263 - 333 - 104 sek
5) 1000 mal Wert PRINTen	76 - 77 - 74 sek
6) 1000 mal PRINTUSING	80 - 83 - xx sek
7) 30.000 mal LEN(X$)-2	107 - 116 -xx sek
8) 30.000 mal STR$-Zuweisung	162 - 173 - xx sek
	========== xx : nicht gemessen

Noch ein weiterer Punkt aus der letzten Pascal-Zeitung:
Es wurde dort hingewiesen, daß der 2. Zeichensatz im Handbuch nicht erwähnt wird. Ich habe mir inzwischen einen zweiten MZ 731 gekauft. Das neue Handbuch ist um die Hinweise auf den 2. Zeichensatz und um die Seitennummern im Inhaltsverzeichnis erweitert.

BASIC-TIPS

Kurzbeschreibung des Disk-Basic-Interpreter der Firma sds computer Service, Mainzer Straße 47, 5568 Daun-Eifel

Version 2,5 A.

Es gibt drei verschiedene Versionen:
1) x.x A beinhalten den vollen Befehlsumfang
20kb
2) xx P wie A, aber CMT, QD und RSx werden nicht mehr angesprochen (freier Speicher ca. 24 KB)
3) x,x B stark abgemagerte P-Version. Die meisten Grafik-und erweiterten mathematischen Befehle fehlen hier.(Vorteil: ca. 30 KB freier Speicher)
In Version A sind alle Befehle der Cassettenversion implementiert, man kann auch weiterhin Programme und Daten auf die Datassette schreiben oder von ihr lesen. Zusätzliche Befehle:
Lock - schützt softwaremäßig Dateien vor dem überschreiben
Unlock - hebt den Softwareschutzt wieder auf
SWP - ruft auf der Diskette gespeicherte Programme als Subroutinen auf und kehrt nach deren Abarbeitung wieder zum Hauptprogramm zurück (Hauptprogramm wird währenddessen auf der Diskette zwischengespeichert, Variablenwerte werden erhalten)
xOPEN - eröffnet Random-Dateien (WOPEN und ROPEN weiterhin möglich)
EOF - Abfrage des Dateiendes
DIR - gibt Disketteninhaltsverzeichnis mit Bytessize der einzelnen Dateien aus
BOARDER - stellt die Randfarbe des Bildschirms ein
TCOPY - kopiert den Textbildschirm auf angeschlossenen Drukker
VARLEN - legt fest, wieviele Zeichen des Variablennamens maximal auf Gleichheit überprüft werden
MAXDIM - legt fest wieviele Dimensionen eine Matrix maximal haben darf
DOKE- 'poked' einen Wert von 0-65535 gleichzeitig in zwei hintereinanderliegende Speicherzellen
DEEK - Umkehrung von DOKE
UPDATE - ersetzt Befehlsfolge DELETE Prog, dient zum überschreiben von geänderten Programmen
CTRL D - ändert Tastatur auf Belegung mit deutschen Umlauten
CTRL A - stellt Orginal-Tasaturbelgung wieder her
CTRL H - verändert Hintergrundfarbe
CTRL Z - verändert Zeichenfarbe
Wie man sieht handelt es sich hier um ein sehr komfortables Basic.
Dies ist im Moment auch noch der Hauptgrund, warum ich noch gerne in Basic programmiere. Aber ich merke, daß ich langsam an die Grenze stoße, wo ich mich auch dem PASCAL zuwenden werde. Vor allem die Möglichkeit Prozeduren zu programmieren mit globalen und lokalen Variablen fehlt mir.

Bernd Bast

Bruno Volkmer
-

2940 Wilhelmshaven, den 30.1.1985
Bernauer Weg 8

```
?
? D
  167E
SO SIEHT DER DRUCKERTREIBER IM S-BASIC AUS, AENDERUNG SIEHE BLATT 2 01

167E C5                 PUSH BC
167F D5                 PUSH DE
1680 E5                 PUSH HL
1681 F5                 PUSH AF
1682 AF                 XOR A
1683 CD 9B 16           CALL 169BH
1686 F1                 POP AF
1687 F5                 PUSH AF
1688 D3 FF              OUT (FFH),A
168A 3E 80              LD A,80H
168C D3 FE              OUT (FEH),A
168E 3E 01              LD A,01H
1690 CD 9B 16           CALL 169BH
1693 AF                 XOR A
1694 D3 FE              OUT (FEH),A
1696 F1                 POP AF
1697 E1                 POP HL
1698 D1                 POP DE
1699 C1                 POP BC
169A C9                 RET
169B 57                 LD D,A                  W
169C 01 00 00           LD BC,0000H
169F 1E 20              LD E,20H
16A1 DB FE              IN A,(FEH)
16A3 E6 0D              AND 0DH
16A5 BA                 CP D
16A6 C8                 RET Z
16A7 1D                 DEC E
16A8 20 F7              JR NZ,-7;16A1
16AA 0B                 DEC BC
16AB 78                 LD A,B
16AC B1                 OR C
16AD 20 F0              JR NZ,-14;167F
16AF AF                 XOR A
16B0 32 4D 00           LD (004DH),A
16B3 3E 41              LD A,41H
16B5 C3 1F 18           JP 181FH
16B8 3E 20              LD A,20H
16BA CD 78 16           CALL 1678H
```

```
? D
  167E
GEAENDERTER DRUCKERTREIBER IM S-BASIC FUER CENTRONICS-DRUCKER 02

167E C5                 PUSH BC
167F D5                 PUSH DE
1680 E5                 PUSH HL
1681 F5                 PUSH AF
1682 CD A2 16           CALL 16A2H
1685 F1                 POP AF
1686 F5                 PUSH AF
1687 FE 60              CP 60H
1689 38 09              JR C,+11;1694
168B D3 E2              OUT (E2H),A
168D CD B9 95           CALL 0B89H
1690 D3 E0              OUT (E0H),A
1692 D6 20              SUB 20H
1694 D3 FF              OUT (FFH),A
1696 3E 80              LD A,80H
1698 D3 FE              OUT (FEH),A
169A AF                 XOR A
169B D3 FE              OUT (FEH),A
169D F1                 POP AF
169E E1                 POP HL
169F D1                 POP DE
16A0 C1                 POP BC
16A1 C9                 RET
16A2 DB FE              IN A,(FEH)
16A4 E6 01              AND 01H
16A6 C8                 RET Z
16A7 CD A0 04           CALL 04A0H
16AA 20 F6              JR NZ,-8;16A2
16AC 00                 NOP
16AD 00                 NOP
16AE 00                 NOP
16AF AF                 XOR A
16B0 32 4D 00           LD (004DH),A
16B3 3E 41              LD A,41H
16B5 C3 1F 18           JP 181FH
16B8 3E 20              LD A,20H
```

ggfs diese beiden Befehle verlauschen
Da Ref. a 1698 steht dann natürlich a 1697

Um aber doch eine, wie ich meine, gute Loesung (so habe ich immer meine Sharp-Programme an meinen Centronic-Drucker angepasst) zu liefern, habe ich den Druckertreiber ab Adresse 167E geaendert. Die kleinen Buchstaben kommen jetzt richtig, die Umlaute allerdings nicht. Bei Basic konnte ich damit bisher sehr gut leben. In diesem Text, den Du jetzt liest, sind auch keine Umlaute.

Es bleibt dem Basic-Programmierer ueberlassen, das kurze neue Maschinenprogramm in einen Basic-Text mit DATAs umzuwandeln, um damit dann den Treiber zu aendern. Bessere Loesung: S-Basic nach der Aenderung vom Monitor aus neu abspeichern.

Mit freundlichen Gruessen

Bruno Volkmer

ANALOG-UHR

```
 10 REM --------------------------------
15 REM SCHWARZ/WEISS OHNE VIDEO RAM
16 REM ERWEITERUNG  INIT M3
20 REM ****** ANALOG - UHR ************
30 REM * ERNST KUNZE   5630 REMSCHEID *
40 REM --------------------------------
50 GOTO"INITIALISIEREN"
60 LABEL"HPG"
70 REM --- ZEIGER ZEICHNEN -----
80 REM      MINUTEN
90 LINE U1(0),V1(0),U1(1),V1(1),U1(2),V1(2),U1(3),V1(3),U1(0),V1(0)
100 REM      STUNDEN
110 LINE G1(0),H1(0),G1(1),H1(1),G1(2),H1(2),G1(3),H1(3),G1(0),H1(0)
120 REM      S)%00
130 LINE E1(0),F1(0),E1(1),F1(1),E1(2),F1(2),E1(0),F1(0)
140 REM --- ZEIGER STEUERUNG -------
150 T$=TI$
160 IF T$=TI$ THEN 160
170 IF RIGHT$(TI$,2)="00"THEN GOSUB "KOORD.MIN"
180 W$=RIGHT$(TI$,4)
190 IF(W$="0001")OR(W$="1201")OR(W$="2401")OR(W$="3601")OR(W$="4801")THEN G
"KOORD.STD"
200 GOSUB "KOORD.SEK"
210 GOTO"HPG"
220 REM ----------------------------
230 :
240 LABEL "KOORD.SEK"
250 BLINE E1(0),F1(0),E1(1),F1(1),E1(2),F1(2),E1(0),F1(0)
260 FOR I = 0 TO 2
270 TMP=E(I)*K - F(I)*S
280 F(I)=E(I)*S + F(I)*K
290 E(I)=TMP
300 E1(I)=E(I)*2.8+319
310 F1(I)=99-(F(I)*1.12)
320 NEXTI
330 RETURN
340 :
350 LABEL "KOORD.MIN"
360 BLINE G1(0),H1(0),G1(1),H1(1),G1(2),H1(2),G1(3),H1(3),G1(0),H1(0)
370 FOR I = 0 TO 3
380 TMP=G(I)*K - H(I)*S
390 H(I)=G(I)*S + H(I)*K
400 G(I)=TMP
410 G1(I)=G(I)*2.8+319
420 H1(I)=99-(H(I)*1.12)
430 NEXTI
440 RETURN
450 :
460 LABEL"KOORD.STD"
470 BLINE U1(0),V1(0),U1(1),V1(1),U1(2),V1(2),U1(3),V1(3),U1(0),V1(0)
480 FOR I = 0 TO 3
490 TMP=U(I)*K - V(I)*S
500 V(I)=U(I)*S + V(I)*K
510 U(I)=TMP
520 U1(I)=U(I)*2.8+319
530 V1(I)=99-(V(I)*1.12)
```

OHNE VIDEO-RAM

```
540 NEXTI
550 RETURN
560 :
570 :
580 REM ******************************
590 LABEL"INITIALISIEREN"
600 REM ******************************
610 :
620 INIT "CRT:M3"
630 REM START-ZEITPUNKT IN DIE ERSTEN
640 REM 20 SEKUNDEN DER MINUTE LEGEN
650 Z$=RIGHT$(TI$,2)
660 Z=VAL(Z$)
670 CURSOR 9,3
680 IF Z > 20 THEN 690 ELSE 720
690 PRINT"UHR STARTET IN";120-Z;" SEKUNDEN !"
700 WAIT (65-Z)*1000:CLS:GOTO 720
710 :
720 K=COS(-(/30))
730 S=SIN(-(/30))
740 DIM E(2),F(2),E1(2),F1(2)
750 DIM G(3),H(3),G1(3),H1(3)
760 DIM U(3),V(3),U1(3),V1(3)
770 R=76:P=/6 :REM FUER DIE 12 MARKEN !
790 FOR I = 220 TO 229
800 CIRCLE 319,100,I,.4,0,2*
810 NEXTI
820 REM *** AUSSENKREIS WEISS ***
830 CIRCLE 319,100,230,.4,0,2*
860 REM *** MITTEKPUNKT KREIS ***
870 CIRCLE 319,98,10,.4,0,2*
900 REM ***** 12 MARKEN SETZEN ******
910 FOR I = 0 TO 2* STEP P
920 X = R*COS(I)*2.8+316
930 Y = 102-R*SIN(I)*1.12
940 POSITION X,Y
950 PATTERN 4,CHR$($FF,$FF,$FF,$FF)
960 NEXTI
970 :
980 REM --- ZEIGER 980 REM --- ZEIGER 1 ---
990 DATA -1,5,0,70,1,5
1000 FOR I = 0 TO 2
1010 READ E(I),F(I)
1020 E1(I)=E(I)*2.8+319
1030 F1(I)=99-(F(I)*1.12)
1040 NEXTI
1050 :
1060 REM --- ZEIGER 2 ---
1070 DATA -2,5,-1,70,1,70,2,5
1080 FOR I = 0 TO 3
1090 READ G(I),H(I)
1100 G1(I)=G(I)*2.8+319
1110 H1(I)=99-(H(I)*1.12)
1120 NEXTI
1130 :
1140 REM --- ZEIGER 3 ---
1150 DATA 0,5,-4,30,0,60,4,30,0,5
1160 FOR I = 0 TO 3
1170 READ U(I),V(I)
1180 U1(I)=U(I)*2.8+319
1190 V1(I)=99-(V(I)*1.12)
1200 NEXTI
1210 :
1220 LABEL "UHR SETZEN"
1230 XS = VAL(LEFT$(TI$,2))
1240 IF XS > 12 THEN LET  XS=XS-12
1250 XM = VAL(MID$(TI$,3,2))
1260 FOR J = 1 TO XS*5+INT(XM/12)
1270 GOSUB"KOORD.STD"
1280 NEXTJ
1290 FOR J = 1 TO XM+1
1300 GOSUB"KOORD.MIN"
1310 NEXTJ
1320 IF RIGHT$(TI$,2)<>"00" THEN 1320
1330 BEEP:GOTO"HPG"
```

ANALOG-UHR

```
5 REM  **********16K VIDEORAM**********
10 REM --------------------------------
20 REM ****** ANALOG - UHR ************
30 REM * ERNST KUNZE   5630 REMSCHEID *
40 REM --------------------------------
50 GOTO"INITIALISIEREN"
60 LABEL"HPG"
70 REM --- ZEIGER ZEICHNEN -----
80 REM       MINUTEN
90 LINE U1(0),V1(0),U1(1),V1(1),U1(2),V1(2),U1(3),V1(3),U1(0),V1(0)
100 REM       STUNDEN
110 LINE G1(0),H1(0),G1(1),H1(1),G1(2),H1(2),G1(3),H1(3),G1(0),H1(0)
120 REM       SEKUNDEN
130 LINE E1(0),F1(0),E1(1),F1(1),E1(2),F1(2),E1(0),F1(0)
140 REM --- ZEIGER STEUERUNG -------
150 T$=TI$
160 IF T$=TI$ THEN 160
170 IF RIGHT$(TI$,2)="00"THEN GOSUB "KOORD.MIN"
180 W$=RIGHT$(TI$,4)
190 IF(W$="0001")OR(W$="1201")OR(W$="2401")OR(W$="3601")OR(W$="4801")
195 THEN GOSUB "KOORD.STD"
200 GOSUB "KOORD.SEK"
210 GOTO"HPG"
220 REM ----------------------------
230 :
240 LABEL "KOORD.SEK"
250 BLINE E1(0),F1(0),E1(1),F1(1),E1(2),F1(2),E1(0),F1(0)
260 FOR I = 0 TO 2
270 TMP=E(I)*K - F(I)*S
280 F(I)=E(I)*S + F(I)*K
290 E(I)=TMP
300 E1(I)=E(I)*2.8+319
310 F1(I)=99-(F(I)*1.12)
320 NEXTI
330 RETURN
340 :
350 LABEL "KOORD.MIN"
360 BLINE G1(0),H1(0),G1(1),H1(1),G1(2),H1(2),G1(3),H1(3),G1(0),H1(0)
370 FOR I = 0 TO 3
380 TMP=G(I)*K - H(I)*S
390 H(I)=G(I)*S + H(I)*K
400 G(I)=TMP
410 G1(I)=G(I)*2.8+319
420 H1(I)=99-(H(I)*1.12)
430 NEXTI
440 RETURN
450 :
460 LABEL"KOORD.STD"
470 BLINE U1(0),V1(0),U1(1),V1(1),U1(2),V1(2),U1(3),V1(3),U1(0),V1(0)
480 FOR I = 0 TO 3
490 TMP=U(I)*K - V(I)*S
500 V(I)=U(I)*S + V(I)*K
510 U(I)=TMP
520 U1(I)=U(I)*2.8+319
530 V1(I)=99-(V(I)*1.12)
540 NEXTI
```

MIT VIDEO-RAM

ANALOG-UHR

```
550 RETURN
560 :
570 :
580 REM ******************************
590 LABEL"INITIALISIEREN"
600 REM ******************************
610 :
620 INIT "CRT:M4"
630 REM START-ZEITPUNKT IN DIE ERSTEN
640 REM 20 SEKUNDEN DER MINUTE LEGEN
650 Z$=RIGHT$(TI$,2)
660 Z=VAL(Z$)
670 CURSOR 9,3
680 IF Z > 20 THEN 690 ELSE 720
690 PRINT"UHR STARTET IN";120-Z;" SEKUNDEN !"
700 WAIT (65-Z)*1000:CLS:GOTO 720
710 :
720 K=COS(-(/30))
730 S=SIN(-(/30))
740 DIM E(2),F(2),E1(2),F1(2)
750 DIM G(3),H(3),G1(3),H1(3)
760 DIM U(3),V(3),U1(3),V1(3)
770 R=76:P=/6 :REM FUER DIE 12 MARKEN !
780 REM **** ROTE KREISE *****
790 FOR I = 220 TO 229
800 CIRCLE [2,0]319,100,I,.4,0,2*
810 NEXTI
820 REM *** AUSSENKREIS WEISS ***
830 CIRCLE [3,0]319,100,230,.4,0,2*
840 REM *** UMFELD BLAU ***
850 PAINT[1]10,10,3
860 REM *** MITTELPUNKT KREIS ***
870 CIRCLE [3,0]319,98,10,.4,0,2*
880 REM *** MITTELPUNKT FUELLEN ***
890 PAINT[2]319,100,3
900 REM ***** 12 MARKEN SETZEN ******

910 FOR I = 0 TO 2* STEP P
920 X = R*COS(I)*2.8+316
930 Y = 102-R*SIN(I)*1.12
940 POSITION X,Y
950 PATTERN[3,0]4,CHR$($FF,$FF,$FF,$FF)
960 NEXTI
970 :
980 REM --- ZEIGER 1 ---
990 DATA -1,5,0,70,1,5
1000 FOR I = 0 TO 2
1010 READ E(I),F(I)
1020 E1(I)=E(I)*2.8+319
1030 F1(I)=99-(F(I)*1.12)
1040 NEXTI
1050 :
1060 REM --- ZEIGER 2 ---
1070 DATA -2,5,-1,70,1,70,2,5
1080 FOR I = 0 TO 3
1090 READ G(I),H(I)
1100 G1(I)=G(I)*2.8+319
1110 H1(I)=99-(H(I)*1.12)
1120 NEXTI
1130 :
1140 REM --- ZEIGER 3 ---
1150 DATA 0,5,-4,30,0,60,4,30,0,5
1160 FOR I = 0 TO 3
1170 READ U(I),V(I)
1180 U1(I)=U(I)*2.8+319
1190 V1(I)=99-(V(I)*1.12)
1200 NEXTI
1210 :
1220 LABEL "UHR SETZEN"
1230 XS = VAL(LEFT$(TI$,2))
1240 IF XS > 12 THEN LET  XS=XS-12
1250 XM = VAL(MID$(TI$,3,2))
1260 FOR J = 1 TO XS*5+INT(XM/12)
1270 GOSUB"KOORD.STD"
1280 NEXTJ
1290 FOR J = 1 TO XM+1
1300 GOSUB"KOORD.MIN"
1310 NEXTJ
1320 IF RIGHT$(TI$,2)<>"00" THEN 1320
1330 BEEP:GOTO"HPG"
```

ANALOG-UHR

Ernst Kunze, Baisieper Str. 34, 5630 Remscheid

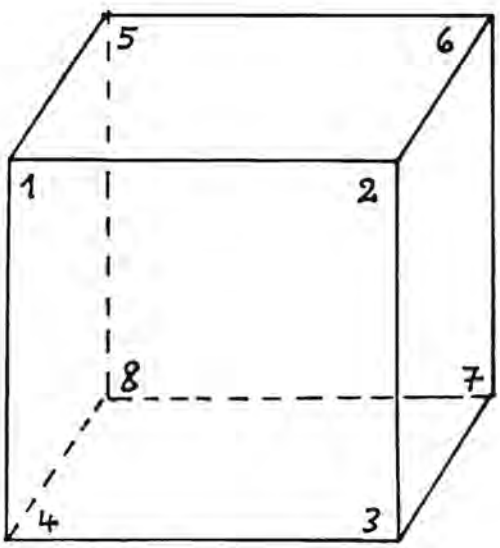
Das Programm ANALOG UHR zeichnet eine Uhr, ähnlich einer Fernseh-Uhr auf den Bildschirm. Da die Ausrechnung der Zeigerstellung beim Start einige Zeit in Anspruch nimmt, soll der Start in den ersten 20 Sekunden einer neuen Minute erfolgen. Der richtige Startzeitpunkt wird im Programm bestimmt. Man kann also zu jeder Zeit mit RUN das Programm starten. Voraussetzung ist natürlich, daß TI$ auf Normalzeit gesetzt worden ist !

70-140 -Routinen zum Zeichnen der Zeiger.
160 Warteschleife bis zum Beginn einer neuen Sekunde !
170 Wenn die letzten Stellen des Zeitstrings "00" sind, wird die nächste Stellung des Minutenzeigers berechnet.
180-190 Der Stundenzeiger springt 5 mal in einer Stunde, deshalb wird hier geprüft, ob 12 Minuten voll sind. Die eine Sekunde, welche der Stundenzeiger zu spät springt, ist ein Kompromiß. Der Computer schafft es nicht, alle 3 Zeiger in einer Sekunde umzurechnen. Es würde 1 Sekunde verloren gehen.
200 Der Sekundenzeiger wird in jeder Sekunde neu berechnet.
240-550 Hier sind die Routinen, in denen die Koordinaten der Zeiger neu berechnet werden. Beim nächsten LINE-Befehl steht dann der Zeiger um 6 Grad bzw. ¶/30 weiter. Die Werte für Sinus u. Cosinus sind negativ, damit die Zeiger auch im Uhrzeigersinn laufen. Vor der Umrechnung wird der momentane Zeiger in der BLINE-Anweisung gelöscht. Die Routinen enthalten auch jeweils die Umrechnung für die Bildschirmausgabe !
650-700 Der Computer braucht zur Berechnung der Zeigerstellung eine gewisse Zeit. Es wird deshalb geprüft, ob die Computeruhr noch innerhalb der ersten 20 Sekunden der neuen Minute ist. Sonst wird durch eine WAIT-Anweisung die Startzeit verzögert bis zum Anfang der nächsten Minute.
770-960 Routinen zum Aufbau des Zifferblattes.
980-1200 In diesen 3 Routinen werden die Zeiger-Daten eingelesen und auf Bildschirmausgabe umgeformt. Wenn der Zeiger auf Null startet, durchläuft er nicht erst die Umformroutine (240-550).
1220-1330 Der Zeitstring wird auf die Anzahl der Stunden und Minuten untersucht und entsprchend oft zur Koordinatenumformung geschickt, um beim Start die richtige Stellung einzunehmen. Dann wird in einer Schleife der Startpunkt "00" Sekunden abgewartet und mit einem BEEP erscheinen die Zeiger. Die Bildschirmuhr läuft jetzt synchron mit der Computeruhr !

WUERFEL WUERFEL

Das Programm zeichnet einen Würfel in Parallelprojektion auf den Bildschirm. Gegenüberliegende Flächen haben die gleiche Farbe. Welche Achse gedreht wird und um wieviel Grad, wird mit der RND-Funktion dem Zufall überlassen. Die gewünschte Größe des Würfels in m/m muß eingegeben werden. Eine bewährte Größe ist 100 m/m.

Zeile 110-170
Die DATA-Werte werden eingelesen und mit dem Vergrößerungsfaktor multipliziert und gerundet. Die Kantenlänge entspricht jetzt dem Eingabewert.
Zeile 190 Sprung zum Hauptprogramm am Ende des Programms.
Zeile 210-260
Der Winkel, um den der Würfel gedreht werden soll (0°-45°), wird als Zufallszahl ermittelt.
Zeile 280-530
Hier sind die drei Formeln zum Drehen um die einzelnen Achsen X Y oder Z.
Zeile 650-730
Die errechneten X und Y - Werte zum Zeichnen des Würfels werde mit einem Faktor multipliziert, um auf dem Bildschirm in X u. Y-Richtung gleiche Maßstäbe zu erhalten. Gleichzeitig wird eine Verlagerung des Koordinatenursprungs in die Mitte des Bildschirms vorgenommen (X + 159) u. (99 - Y). Diese Werte werden A u. B zugewiesen.
Zeile 750-790
Durch Zufall wird die Achse ermittelt, um welche gedreht werden soll. Man kann hier das Programm abändern um eine festgelegte Folge ablaufen zu lassen. Ebenso kann man den Drehwinkel vorher mit einer INPUT-Anweisung fest eingeben.
Zeile 810-920
Hier wird der größte Wert der Z - Achse ermittelt. Dieser ist maßgebend dafür, welche drei Seiten des Würfels zu sehen sind. Die Nr. dieses Punktes wird in V abgelegt und dient dazu in der ON GOSUB-Anweisung die entsprechende Routine zu durchlaufen.
Zeile 950-2450
Routinen zum zeichnen des Würfels. Der Kreuzungspunkt der Verbindungslinien je zweier Punkte einer Fläche wird als Ausgangspunkt für die PAINT-Anweisung genommen. In seltenen Fällen kann hier der Punkt durch Rundungsfehler auch außerhalb liegen, dann wird der Bildschirm um den Würfel eingefärbt.
Zeile 248o-2550
Hauptprogramm.



Lage der 8 Punkte
im Programm.
Koordinaten-Nullpunkt ist
Mitte des Würfels!

Ernst Künze

```
10 REM---------------------------------
20 REM** ERNST KUNZE 5630 REMSCHEID ***
25 REM*****     WUERFEL       *********
30 REM---------------------------------
40 INIT"CRT:M4"
50 CLR:RESTORE 560
60 DEF FNR(X)=INT(X+.5)
70 INPUT"GROESSE IN 3/3  = ";MA
80 MA=MA/2:PZ=8
90 DIM X(PZ),Y(PZ),Z(PZ),A(PZ),B(PZ)
100 :
110 REM ** PUNKTE EINLESEN **
120 FOR I = 1 TO PZ
130 READ X(I),Y(I),Z(I)
140 X(I)=X(I)*MA
150 Y(I)=Y(I)*MA
160 Z(I)=Z(I)*MA
170 NEXTI
180 :
190 GOTO"HPG"
200 :
210 LABEL"WINKEL"
220 REM** DREHWINKEL AUS ZUFALLSZAHL **
230 WI=INT(RND(1)*45)+1
240 S=SIN(RAD(WI))
250 K=COS(RAD(WI))
260 RETURN
270 :
280 REM---------------------------------
290 REM ** KOORDINATEN TRANSFORMATION
300 REM---------------------------------
310 LABEL"X-ACHSE"
320 FOR I =*1 TO PZ
330 TMP=Y(I)*K-Z(I)*S
340 Z(I)=Y(I)*S+Z(I)*K
350 Y(I)=TMP
360 NEXTI
370 RETURN
380 :
390 LABEL"Y-ACHSE"
400 FOR I = 1 TO PZ
410 TMP=X(I)*K-Z(I)*S
420 Z(I)=X(I)*S+Z(I)*K
430 X(I)=TMP
440 NEXTI
450 RETURN
460 :
470 LABEL"Z-ACHSE"
480 FOR I = 1 TO PZ
490 TMP=X(I)*K-Y(I)*S
500 Y(I)=X(I)*S+Y(I)*K
510 X(I)=TMP
520 NEXTI
530 RETURN
540 REM---------------------------------
550 :
560 DATA 1,1,-1
570 DATA 1,-1,-1
580 DATA -1,-1,-1
590 DATA -1,1,-1
600 DATA 1,1,1
610 DATA 1,-1,1
620 DATA -1,-1,1
630 DATA -1,1,1
640 :
650 LABEL"SCHIRM"
660 REM ** KORREKTUR BILDSCHIRM **
670 FOR I = 1 TO PZ
680 A(I)=X(I)*2.8 +319
690 A(I)=FNR(A(I))
700 B(I)=99 -Y(I)*1.12
710 B(I)=FNR(B(I))
720 NEXTI
730 RETURN
740 :
750 LABEL"XYZ"
760 REM** ACHSE X Y Z DURCH ZUFALL ***
770 Q=INT(RND(1)*3)+1
780 ON Q  GOSUB "X-ACHSE","Y-ACHSE","Z-ACHSE"
790 RETURN
800 :
810 LABEL"HOECHSTER PUNKT"
820 REM ** HOECHSTER PUNKT ZUM BE - **
830 REM ** TRACHTER SUCHEN: Z-ACHSE **
840 G=Z(1)
850 FOR I = 1 TO 8
860 IF Z(I) >=G THEN 880
870 G=Z(I)
880 NEXTI
890 FOR J = 1 TO 8
900 IF G = Z(J) THEN LET V=J
910 NEXTJ
920 RETURN
930 :
940 :
950 LABEL"I"
960 LINE(3,0)A(5),B(5),A(6),B(6),A(2),B(2),A(3),B(3),A(4),B(4),A(8),B(8),A(5),B(
5),A(1),B(1),A(4)B(4),A(1),B(1),A(2),B(2)
970 E=(A(6)+A(1))/2
980 T=(B(2)+B(5))/2
990 U=POINT(E,T)
1000 IF U=3 THEN 1020
1010 PAINT(1)E,T,3
1020 E=(A(2)+A(4))/2
1020 E=(A(2)+A(4))/2
1030 T=(B(3)+B(1))/2
1040 U=POINT(E,T)
1050 IF U=3 THEN 1070
1060 PAINT(2)E,T,3
1070 E=(A(1)+A(8))/2
1080 T=(B(4)+B(5))/2
1090 U=POINT(E,T)
```

WUERFEL WUERFEL

```
1100 IF U=3 THEN 1120
1110 PAINT(3)E,T,3
1120 RETURN
1130 :
1140 LABEL"2"
1150 LINE(3,0)A(6),B(6),A(7),B(7),A(3),B(3),A(4),B(4),A(1),B(1),A(5),B(5),A(6),B
(6),A(2),B(2),A(1),B(1),A(2),B(2),A(3),B(3)
1160 E=(A(5)+A(2))/2
1170 T=(B(1)+B(6))/2
1180 U=POINT(E,T)
1190 IF U=3 THEN 1210
1200 PAINT(1)E,T,3
1210 E=(A(2)+A(7))/2
1220 T=(B(3)+B(6))/2
1230 U=POINT(E,T)
1240 IF U=3 THEN 1260
1250 PAINT(3)E,T,3
1260 E=(A(2)+A(4))/2
1270 T=(B(1)+B(3))/2
1280 U=POINT(E,T)
1290 IF U=3 THEN 1310
1300 PAINT(2)E,T,3
1310 RETURN
1320 :
1330 LABEL"3"
1340 LINE(3,0)A(7),B(7),A(8),B(8),A(4),B(4),A(1),B(1),A(2),B(2),A(6),B(6),A(7),B
(7),A(3),B(3),A(2),B(2),A(3),B(3),A(4),B(4)
1350 E=(A(2)+A(4))/2
1360 T=(B(3)+B(1))/2
1370 U=POINT(E,T)
1380 IF U=3 THEN 1400
1390 PAINT(2)E,T,3
1400 E=(A(3)+A(8))/2
1410 T=(B(7)+B(4))/2
1420 U=POINT(E,T)
1430 IF U=3 THEN 1450
1440 PAINT(1)E,T,3
1450 E=(A(2)+A(7))/2
1460 T=(B(6)+B(3))/2
1470 U=POINT(E,T)
1480 IF U=3 THEN 1500
1490 PAINT(3)E,T,3
1500 RETURN
1510 :
1521520 LABEL"4"
1530 LINE(3,0)A(8),B(8),A(5),B(5),A(1),B(1),A(2),B(2),A(3),B(3),A(7),B(7),A(8),B
(8),A(4),B(4),A(3),B(3),A(4),B(4),A(1),B(1)
1540 E=(A(4)+A(2))/2
1550 T=(B(1)+B(3))/2
1560 U=POINT(E,T)
1570 IF U=3 THEN 1590
1580 PAINT(2)E,T,3
1590 E=(A(4)+A(5))/2
1600 T=(B(1)+B(8))/2
1610 U=POINT(E,T)
1620 IF U=3 THEN 1640
1630 PAINT(3)E,T,3
1640 E=(A(4)+A(7))/2
1650 T=(B(8)+B(3))/2
1660 U=POINT(E,T)
1670 IF U=3 THEN 1690
1680 PAINT(1)E,T,3
1690 RETURN
1700 :
1710 LABEL"5"
1720 LINE(3,0)A(1),B(1),A(4),B(4),A(8),B(8),A(7),B(7),A(6),B(6),A(2),B(2),A(1),
(1),A(5),B(5),A(6),B(6),A(5),B(5),A(8),B(8)
1730 E=(A(5)+A(4))/2
1740 T=(B(1)+B(8))/2
1750 U=POINT(E,T)
1760 IF U=3 THEN 1780
1770 PAINT(3)E,T,3
1780 E=(A(5)+A(7))/2
1790 T=(B(8)+B(6))/2
1800 U=POINT(E,T)
1810 IF U=3 THEN 1830
1820 PAINT(2)E,T,3
1830 E=(A(5)+A(2))/2
1840 T=(B(1)+B(6))/2
1850 U=POINT(E,T)
1860 IF U=3 THEN 1880
1870 PAINT(1)E,T,3
1880 RETURN
1890 :
1900 LABEL"6"
1910 LINE(3,0)A(2),B(2),A(1),B(1),A(5),B(5),A(8),B(8),A(7),B(7),A(3),B(3),A(2),
(2),A(6),B(6),A(7),B(7),A(6),B(6),A(5),B(5)
1920 E=(A(6)+A(3))/2
1930 T=(B(2)+B(7))/2
1940 U=POINT(E,T)
1950 IF U=3 THEN 1970
1960 PAINT(3)E,T,3
1970 E=(A(6)+A(1))/2
1980 T=(B(2)+B(5))/2
1990 U=POINT(E,T)
2000 IF U=3 THEN 2020
2010 PAINT(1)E,T,3
2020 E=(A(6)+A(8))/2
2030 T=(B(5)+B(7))/2
2040 U=POINT(E,T)
202050 IF U=3 THEN 2070
2060 PAINT(2)E,T,3
2070 RETURN
2080 :
2090 LABEL"7"
2100 LINE(3,0)A(3),B(3),A(2),B(2),A(6),B(6),A(5),B(5),A(8),B(8),A(4),B(4),A(3),
(3),A(7),B(7),A(8),B(8),A(7),B(7),A(6),B(6)
2110 E=(A(7)+A(4))/2
2120 T=(B(8)+B(3))/2
2130 U=POINT(E,T)
2140 IF U=3 THEN 2160
2150 PAINT(1)E,T,3
2160 E=(A(7)+A(2))/2
```

```
2170 T=(B(3)+B(6))/2
2180 U=POINT(E,T)
2190 IF U=3 THEN 2210
2200 PAINT(3)E,T,3
2210 E=(A(7)+A(5))/2
2220 T=(B(8)+B(6))/2
2230 U=POINT(E,T)
2240 IF U=3 THEN 2260
2250 PAINT(2)E,T,3
2260 RETURN
2270 :
2280 LABEL"8"
2290 LINE(3,0)A(4),B(4),A(3),B(3),A(7),B(7),A(6),B(6),A(5),B(5),A(1),B(1),A(4),B
(4),A(8),B(8),A(5),B(5),A(8),B(8),A(7),B(7)
2300 E=(A(1)+A(8))/2
2310 T=(B(4)+B(5))/2
2320 U=POINT(E,T)
2330 IF U=3 THEN 2350
2340 PAINT(3)E,T,3
2350 E=(A(4)+A(7))/2
2360 T=(B(3)+B(8))/2
2370 U=POINT(E,T)
2380 IF U=3 THEN 2400
2390 PAINT(1)E,T,3
2400 E=(A(8)+A(6))/2
2410 T=(B(7)+B(5))/2
2420 U POINT(E,T)
2430 IF U=3 THEN 2450
2440 PAINT(2)E,T,3
2450 RETURN
2460 :
2470 :
2480 LABEL"HPG"
2490 GOSUB"WINKEL"
2500 GOSUB"XYZ"
2510 GOSUB"SCHIRM"
2520 GOSUB"HOECHSTER PUNKT"
2530 CLS
2540 ON V GOSUB "1","2","3","4","5","6","7","8"
2550 GOTO"HPG"
```

WUERFEL

WUERFEL

BASIC-TIPS

NEUE ERWEITERUNGSBOX FUR DEN MZ-8XX

Wie uns mitgeteilt wurde, wird eine neue Erweiterungsbox für den MZ-8XX angeboten.
Die Erweiterungsbox wird direkt auf dem MZ-8XX aufgeschraubt anstatt auf das linke obere Gehäuseteil. Die Box hat folgende Anschlüsse:

2 Slots, um Ram-Boards,OD's etc. anzuschließen

Eine Möglichkeit, um eine 3,5 Zoll Diskettenstation einzubauen.

Nähere Informationen können direkt bei:

J.Frank
Boschstr.47 II
2000 Hamburg 50
Tel: 040/8992387

abgefordert werden.

D.K.

```
10 REM----------------------------------
15 REM OHNE VIDEO RAM ERWEITERUNG
20 REM** ERNST KUNZE 5630 REMSCHEID ***
25 REM*****      WUERFEL        *********
30 REM----------------------------------
40 INIT"CRT:M1"
50 CLR:RESTORE 560
60 DEF FNR(X)=INT(X+.5)
70 INPUT"GROESSE IN M/M  = ";MA
80 MA=MA/2:PZ=8
90 DIM X(PZ),Y(PZ),Z(PZ),A(PZ),B(PZ)
100 :
110 REM ** PUNKTE EINLESEN **
120 FOR I = 1 TO PZ
130 READ X(I),Y(I),Z(I)
140 X(I)=X(I)*MA
150 Y(I)=Y(I)*MA
160 Z(I)=Z(I)*MA
170 NEXTI
180 :
190 GOTO"HPG"
200 :
210 LABEL"WINKEL"
220 REM** DREHWINKEL ALS ZUFALLSZAHL **
230 WI=INT(RND(1)*45)+1
240 S=SIN(RAD(WI))
250 K=COS(RAD(WI))
260 RETURN
270 :
280 REM*----------------------------
290 REM** KOORDINATEN TRANSFORMATION
300 REM---------------------------- ---
310 LABEL"X-ACHSE"
320 FOR I = 1 TO PZ
330 TMP=Y(I)*K-Z(I)*S
340 Z(I)=Y(I)*S+Z(I)*K
350 Y(I)=TMP
360 NEXTI
370 RETURN
380 :
390 LABEL"Y-ACHSE"
400 FOR I = 1 TO PZ
410 TMP=X(I)*K-Z(I)*S
420 Z(I)=X(I)*S+Z(I)*K
430 X(I)=TMP
440 NEXTI
450 RETURN
460 :
470 LABEL"Z-ACHSE"
480 FOR I    1 TO PZ
490 TMP=X(I)*K-Y(I)*S
500 Y(I)=X(I)*S+Y(I)*K
510 X(I)=TMP
520 NEXTI
530 RETURN
540 REM--  -- ----------------------------
550 :
560 DATA 1,1,-1
570 DATA 1,-1,-1
580 DATA -1,-1,-1
590 DATA -1,1,-1
600 DATA 1,1,1
610 DATA 1,-1,1
620 DATA -1,-1,1
630 DATA -1,1,1
640 :
650 LABEL"SCHIRM"
660 REM ** KORREKTUR BILDSCHIRM **
670 FOR I = 1 TO PZ
680 A(I)=X(I)*1.4 +159
690 A(I)=FNR(A(I))
700 B(I)=99 -Y(I)*1.12
710 B(I)=FNR(B(I))
720 NEXTI
730 RETURN
740 :
750 LABEL"XYZ"
 Z%=188 ***
770 Q=INT(RND(1)*3)+1
780 ON Q  GOSUB "X-ACHSE","Y-ACHSE","Z-ACHSE"
790 RETURN
800 :
810 LABEL"HOECHSTER PUNKT"
820 REM ** HOECHSTEN PUNKT ZUM BE- **
830 REM ** TRACHTER SUCHEN: Z-ACHSE **
840 G=Z(1)
850 FOR I = 1 TO 8
860 IF Z(I) >=G THEN 880
870 G=Z(I)
880 NEXTI
890 FOR J = 1 TO 8
900 IF G = Z(J) THEN LET V=J
910 NEXTJ
920 RETURN
930 :
940 :
950 LABEL"1"
960 LINE(3,0)A(5),B(5),A(6),B(6),A(2),B(2),A(3),B(3),A(4),B(4),A(8),B(8),A(5),B(
5),A(1),B(1),A(4)B(4),A(1),B(1),A(2),B(2)
970 E=(A(6)+A(1))/2
980 T=(B(2)+B(5))/2
990 U=POINT(E,T)
1000 IF U=3 THEN 1020
1010 PAINT(E,T,3
1020 E=(A(2)+A(4))/2
1030 T=(B(3)+B(1))/2
1040 U=POINT(E,T)
1050 IF U=3 THEN 1070
1060 PAINT(E,T,3
1070 E=(A(1)+A(8))/2
1080 T=(B(4)+B(5))/2
1090 U=POINT(E,T)
1100 IF U=3 THEN 1120
```

```
1110 PAINT(3)E,T,3
1120 RETURN
1130 :
1140 LABEL"2"
1150 LINE(3,0)A(6),B(6),A(7),B(7),A(3),B(3),A(4),B(4),A(1),B(1),A(5),B(5),A(6),B
(6),A(2),B(2),A(1),B(1),A(2),B(2),A(3),B(3)
1160 E=(A(5)+A(2))/2
1170 T=(B(1)+B(6))/2
1180 U=POINT(E,T)
1190 IF U=3 THEN 1210
1200 PAINT(1)E,T,3
1210 E=(A(2)+A(7))/2
1220 T=(B(3)+B(6))/2
1230 U=POINT(E,T)
1240 IF U=3 THEN 1260
1250 PAINT(3)E,T,3
1260 E=(A(2)+A(4))/2
1270 T=(B(1)+B(3))/2
1280 U=POINT(E,T)
1290 IF U=3 THEN 1310
1300 PAINT(2)E,T,3
1310 RETURN
1320 :
1330 LABEL"3"
1340 LINE(3,0)A(7),B(7),A(8),B(8),A(4),B(4),A(1),B(1),A(2),B(2),A(6),B(6),A(7),B
(7),A(3),B(3),A(2),B(2),A(3),B(3),A(4),B(4)
1350 E=(A(2)+A(4))/2
1360 T=(B(3)+B(1))/2
1370 U=POINT(E,T)
1380 IF U=3 THEN 1400
1390 PAINT(2)E,T,3
1400 E=(A(3)+A(8))/2
1410 T=(B(7)+B(4))/2
1420 U=POINT(E,T)
1430 IF U=3 THEN 1450
1440 PAINT(1)E,T,3
1450 E=(A(2)+A(7))/2
1460 T=(B(6)+B(3))/2
1470 U=POINT(E,T)
1480 IF U=3 THEN 1500
1490 PAINT(3)E,T,3
1500 RETURN
1510 :
1520 LABEL"4"
1530 LINE(3,0)A(8),B(8),A(5),B(5),A(1),B(1),A(2),B(2),A(3),B(3),A(7),B(7),A(8),B
(8),A(4),B(4),A(3),B(3),A(4),B(4),A(1),B(1)
1540 E=(A(4)+A(2))/2
1550 T=(B(1)+B(3))/2
1560 U=POINT(E,T)
1570 IF U=3 THEN 1590
1580 PAINT(2)E,T,3
1590 E=(A(4)+A(5))/2
1600 T=(B(1)+B(8))/2
1610 U=POINT(E,T)
1620 IF U=3 THEN 1640
1630 PAINT(3)E,T,3
1640 E=(A(1)+A(7))/2
1650 T=(B(8)+B(3))/2
1660 U=POINT(E,T)
1670 IF U=3 THEN 1690
1680 PAINT(1)E,T,3
1690 RETURN
1700 :
1710 LABEL"5"
1720 LINE(3,0)A(1),B(1),A(4),B(4),A(8),B(8),A(7),B(7),A(6),B(6),A(2),B(2),A(1),B
(1),A(5),B(5),A(6),B(6),A(5),B(5),A(8),B(8)
1730 E=(A(5)+A(4))/2
1740 T=(B(1)+B(8))/2
1750 U=POINT(E,T)
1760 IF U=3 THEN 1780
1770 PAINT(3)E,T,3
1780 E=(A(5)+A(7))/2
1790 T=(B(8)+B(6))/2
1800 U=POINT(E,T)
1810 IF U=3 THEN 1830
1820 PAINT(2)E,T,3
1830 E=(A(5)+A(2))/2
1840 T=(B(1)+B(6))/2
1850 U=POINT(E,T)
1860 IF U=3 THEN 1880
1870 PAINT(1)E,T,3
1880 RETURN
1890 :
1900 LABEL"6"
1910 LINE(3,0)A(2),B(2),A(1),B(1),A(5),B(5),A(8),B(8),A(7),B(7),A(3),B(3),A(2),B
(2),A(6),B(6),A(7),B(7),A(6),B(6),A(5),B(5)
1920 E=(A(6)+A(3))/2
1930 T=(B(2)+B(7))/2
1940 U=POINT(E,T)
1950 IF U=3 THEN 1970
1960 PAINT(3)E,T,3
1970 E=(A(6)+A(1))/2
1980 T=(B(2)+B(5))/2
1990 U=POINT(E,T)
2000 IF U=3 THEN 2020
2010 PAINT(1)E,T,3
2020 E=(A(6)+A(8))/2
2030 T=(B(5)+B(7))/2
2040 U=POINT(E,T)
2050 IF U=3 THEN 2070
2060 PAINT(2)E,T,3
2070 RETURN
2080 :
2090 LABEL"7"
2100 LINE(3,0)A(3),B(3),A(2),B(2),A(6),B(6),A(5),B(5),A(8),B(8),A(4),B(4),A(3),B
(3),A(7),B(7),A(8),B(8),A(7),B(7),A(6),B(6)
2110 E=(A(7)+A(4))/2
2120 T=(B(8)+B(3))/2
2130 U=POINT(E,T)
2140 IF U=3 THEN 2160
2150 PAINT(1)E,T,3
2160 E=(A(7)+A(2))/2
2170 T=(B(3)+B(6))/2
```

```
2180 U=POINT(E,T)
2190 IF U=3 THEN 2210
2200 PAINT(3)E,T,3
2210 E=(A(7)+A(5))/2
2220 T=(B(8)+B(6))/2
2230 U=POINT(E,T)
2240 IF U=3 THEN 2260
2250 PAINT(2)E,T,3
2260 RETURN
2270 :
2280 LABEL"8"
2290 LINE(3,0)A(4),B(4),A(3),B(3),A(7),B(7),A(6),B(6),A(5),B(5),A(1),B(1),A(4),B
(4),A(8),B(8),A(5),B(5),A(8),B(8),A(7),B(7)
2300 E=(A(1)+A(8))/2
2310 T=(B(4)+B(5))/2
2320 U=POINT(E,T)
2330 IF U=3 THEN 2350
2340 PAINT(3)E,T,3
2350 E=(A(4)+A(7))/2
2360 T=(B(3)+B(8))/2
2370 U=POINT(E,T)
2380 IF U=3 THEN 2400
2390 PAINT(1)E,T,3
2400 E=(A(8)+A(6))/2
2410 T=(B(7)+B(5))/2
2420 U=POINT(E,T)
2430 IF U=3 THEN 2450
2440 PAINT(2)E,T,3
2450 RETURN
2460 :
2470 :
2480 LABEL"HPG"
2490 GOSUB"WINKEL"
2500 GOSUB"XYZ"
2510 GOSUB"SCHIRM"
2520 GOSUB"NAECHSTER PUNKT"
2530 CLS
2540 ON V GOSUB "1","2","3","4","5","6","7","8"
2550 GOTO"HPG"
```

WUERFEL