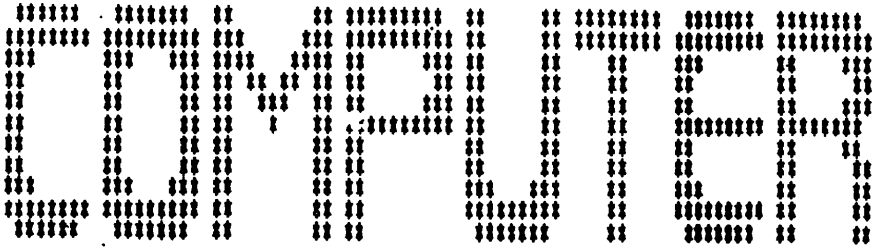
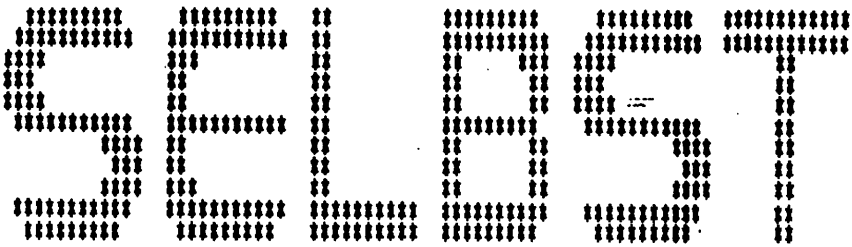
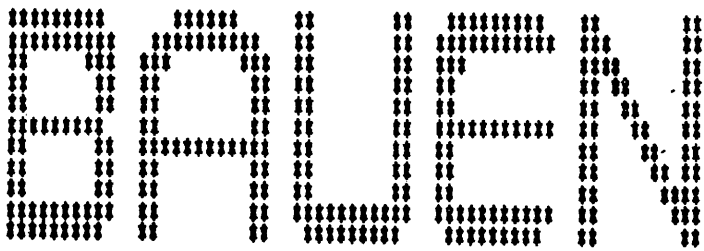






UND PROGRAMMIEREN

Heft 3

Eine Broschüre

des

- Selbstbau -
Computer-Club
Dorfstraße 4
3372 Wallmoden 1

Vorwort der Herausgeber

Nun sind es schon sehr viele Mitarbeiter geworden. In dieser Nummer haben wir einige Programme, die uns zugeschickt worden sind, abgedruckt. Aber wir konnten nicht alles unterbringen. In der nächsten wird z.B. ein Copierprogramm für zwei Laufwerke stehen. Natürlich heißt das nicht, daß Ihr Eure Hände nun in den Schoß legen sollt. Weiter so - dann bleibt diese Zeitschrift eine sehr lebendige und aktuelle Informationsquelle.

Bei der Vielzahl von Programmen liegt hier der Schwerpunkt dieser Ausgabe. Das Utility-Projekt war ein Renner, wie man den Programmen dieses Heftes entnehmen kann. Wenn es weiter so geht, dann könnten wir bald das erste Eprom oder eine Diskette zusammenstellen. Gesucht wird z.B. noch ein schnelles und sicheres Programm für den Eprom-Brenner, oder wie wäre es mit einer universellen und schnellen Sortier-Routine? Dies ist zwar kein Utility, aber als Unterprogramm sicher gut einzusetzen.

Telefonisch wurde uns schon ein Artikel über CP/M für Anfänger angekündigt. Die Sprache C wäre doch auch etwas - oder? Wo ist der nächste Redakteur? Wir meinen damit nicht ein hochgestochenes Machwerk, das auch noch den Anforderungen der kritischen Experten genügt. Es sollte ein Artikel (oder Artikelserie) über das Thema und unseren Computer sein, mit möglichst vielen Beispielen, die dann auch wirklich funktionieren.

Der Umfang und die Auflage dieser Zeitschrift hängen direkt zusammen. Zur Zeit müssen wir mit Fotokopierkosten von DM 2,40 rechnen. Bei 50 Pf Porto kann man selbst ausrechnen, was noch für den Umschlag und den Aufkleber bleibt. Etwas legen wir noch zu. Mit einer größeren Auflage können wir die Kosten senken und mehr Seiten drucken.

Die nächste Nummer kommt kurz vor Weihnachten heraus. Wie wäre es denn mit einem Weihnachtsmann? Wir können zwar keine Preise aussetzen, würden uns aber trotzdem über schöne 'Weihnatsprogramme' freuen.

Für die Redaktion

Horst Günter Eysel

.... Besondere Schwierigkeiten habe ich mit der Prommer Karte. Mit dem Oszilloskop gemessen, sind alle Signale einwandfrei, auch die Programmierspannung. Es Es werden jedoch alle Eproms sicher zerstört! Vielleicht haben Sie einen Tip für mich...

G. Wollberg, K. Schumacher Ring 39, 6200 Wiesbaden.

Wer hat ähnliche Schwierigkeiten überwunden oder weiß Abhilfe? Bei uns läuft der Prommer, von einer Ausnahme abgesehen, einwandfrei. Vielleicht reicht die Stromversorgung nicht aus? Wieviel Ampere hat diese?

Kennwort Kontakte

Nach der Lektüre Eures Heft 2, in dem Ihr uns (die Clubmitglieder) zur verstärkten Mitarbeit aufruft, habe ich mich entschlossen Euch nebst meinem Kommentar zu Eurem Heft ein Pascalprogramm zu übersenden.

Zuerst aber mein Kommentar:
Eure Zeitschrift ist an sich schon ein feines Blättchen, aber außer Euren Programmierkursen könntet Ihr mehr Programme abdrucken, die Euch von Lesern bzw. Clubmitgliedern zugesandt werden. Wie ich aus der letzten Ausgabe ersehen habe, sucht Ihr, wohl auch deshalb, aktive Mitglieder.
Ich gedenke ein solches aktives Mitglied zu werden.
Dies soll nicht etwa bedeuten, daß ich Euch nach jeder Eurer Ausgaben mit Verbesserungsvorschlägen und Kritik überhäufen will, sondern ich möchte Euch ab und zu mal ein kleines Pascal- oder Assemblerprogramm oder Hardware übersenden.
Nun zu meiner Person:
Ich bin 22 Jahre alt, und zur Zeit mache ich eine Ausbildung als Energieanlagenelektroniker. Meine Hobbies sind Elektronik und natürlich Computertechnik. Ich arbeite jetzt ca. ein Jahr mit dem NDR-Kleincomputer, so daß meine Programme eventuell noch nicht so profihaft aussehen. Ich bin aber für jeden Verbesserungsvorschlag dankbar. Soltet Ihr alles Mist finden, schmeißt meinen Schrieb einfach in den Papierkorb.
Eventuell sind meine Programme etwas fachspezifisch. Ich hoffe das stört nicht. Außer dem beiliegenden Programm hätte ich noch ein Demo-Programm zur I/O Karte, mit der ich über eine einfache Hardwareerweiterung eine LED Zeile als Lauflicht ansteuere.
Denkbar wäre auch ein Leistungsausgang zur Steuerung elektrischer Geräte im Haushalt.
Zur Zeit beschäftige ich mich mit der Sprachausgabe (im ELO 12/84 veröffentlicht). Die Platine besitze ich als Prototyp un bin nun am fehlerausmerzen. Fehler Nr. 1 : Die Adresse der Karte stimmt nicht.
Abhilfe : den 1.Jumper weglassen
Meine Hauptinteressen sind Textverarbeitung, Steuerungstechnik, Grafik, und Mathematik.

Payer

Ä Suchroutine ü

von Norwin Hammes

Dieses Suchprogramm stellt eine von vielen Möglichkeiten dar, um beliebige Daten in einem definierten Speicherbereich zu suchen.
In diesen Fall werden Daten blockweise gesucht, d.h. die restlichen, zu diesem Block gehörenden Daten, werden mit ausgegeben.

Bsp:	Sie geben Meyer ein und der Block wird vollständig ausgegeben.
§	
Otto Meyer	Beispiel eines Blocks, die
Wallstr. 4	aneinander gereiht werden.
2000 Hamburg	
Tel.	
§	

Dieses '§' Zeichen trennt die Blöcke. Es ist erforderich damit das Programm den Anfang und das Ende erkennt. Man kann natürlich auch andere Zeichen verwenden, was jedoch nicht sinnvoll wäre, da dieses '§' Zeichen in Texten normalerweise nicht benutzt wird.
Die Blöcke können beliebig lang sein, d.h. sie sollten noch auf den Bildschirm passen.
Es können auch Teile eines Namens eingegeben werden, z.B. die Endung 'yer' des Namens 'Meyer'. Das Programm sucht jetzt alles mit der Endung 'yer' und gibt es aus.
Durch Veränderung des Programms kann man jedoch noch andere, sehr interessante Varianten realisieren.

Wichtig:
Nach den assemblieren des Programms muß der Textstart wieder auf den Textanfang der zu suchenden Blöcke eingestellt werden.

Programm:

```
buffer equ $8d00        ŧ z. such. Zeichen
bufl equ $8d20          ŧ Anzahl d. Zeichen
danf equ $8d30          ŧ Textstart
```

```
dadd equ $8d40          ! Blockanfang
buf2 equ $8d50

t23: dc.b '                                        ',0
t24: dc.b 'Daten nicht i. d. Datei ',0
t241: dc.b '                       ',0
t25: dc.b 'N - neue Daten   X - weitersuchen   M - Menue ',0
t26: dc.b 'Suchfunktion   ',0
ds 0

start:                  ! Programmstart
move.w  #!clr,d7
trap #1
move.w #!getstx,d7
trap #1
move.l d0,danf
bsr text6
read1:                  ;Suchzeichen einlesen
lea buffer,a0
move #$21,d0
move #130,d1
move #25,d2
move #20,d3
move.w #!read,d7
trap #1
move d4,buf1            ! Anzahl der Zeichen

lea danf,a3
movea.l (a3),a1
move buf1,d5
clr d3
clr d4
verga:
move.l a1,dadd          ! Blockanfang merken
move #0,d4
lea buffer,a2
verg:
movea.l a1,a3
vergc:                  ! für doppelt
cmp.b #'5',(a1)
```

```
beq.s vergb
cmpm.b (a1)+,(a2)+
bne.s vergw
add #1,d4
cmp d4,d5
beq.s ende1
bra.s vergc
vergw:
movea.l a3,a1
cmp.b #0,(a1)
beq text4              ! nicht gefunden
adda #1,a1
move #0,d4
lea buffer,a2
bra.s verg

vergb:
cmp.b #10,(a1)+
bne.s vergb
move.l a1,dadd
bra.s verga

ende1:
movea.l dadd,a3
lea buf2,a2
move #214,d2           ! Textposition - Y
verg1:
move.b (a3)+,(a2)+
cmp.b #'§',(a3)        ! Marke fuer Blockanfang / b.z.w. Ende
beq.s text2
cmp.b #13,(a3)
beq.s text2
bra.s verg1
text2:
move #$11,d0
move #200,d1           ! Textposition - X
lea t23(pc),a0
move.w #!write,-(sp)
trap #1
move.b #0,(a2)
```

```
lea buf2,a0
move.w #!write,d7
trap #1
cmp.b #'§',(a3)         * Marke fuer das Blockende / b.z.w. Anfang
beq.s ausg1
adda #2,a3
lea buf2,a2
sub #15,d2
bra.s verg1

ausg1:
lea t23(pc),a0          * Loeschen des alten Textes
move #$21,d0
move #160,d1
move.w  #!write,d7
trap #1
ende3:
move.w #!ci,d7          * Tastaturabfrage
trap #1
bclr #5,d0
cmp.b #'W',d0           * W - neue Daten
beq start
wende3:
cmp.b #10,(a3)+
bne.s wende3
movea.l a3,a1
cmp.b #'X',d0           * X - weitersuchen
beq verga
rts
text4:
move #6-1,d4
wdh3:
lea t24(pc),a0          * Daten nicht i. d. Datei
move #$11,d0            *
move #360,d1            * diese Rotine erzeugt das blinken
move #1,d2
move.w #!write,d7
trap #1
move #3,d0
move.w  #!delay,d7
```

```
trap #1
lea t241(pc),a9
move #$11,d0
move.w  #!write,d7
trap #1
move #1,d0
move.w  #!delay,d7
trap #1
dbra.s d4,wdh3
bra ende3
text6:
lea t25(pc),a0          ! Statuszeile
move #$11,d0
move #10,d1
move #1,d2
move.w  #!write,d7
trap #1
lea t26(pc),a0
move #$32,d0
move #130,d1
move #231,d2
move.w #!write,d7
trap #1
rts
end
```

WHAT'S NEW - WHAT'S NEW - WHAT'S NEW - WHAT'S NEW

- KEY Platine r4 vorbereitet für Einbau des Sub-D-Steckers (ähnlich wie bei SER).

- RAM64/256 Platine r4 (siehe Tips und Tricks) (übrigens: das RAM-Handbuch ist schon lange lieferbar)

- GDP64K Platine r4 (der im Franzis-Sonderheft beschriebene Umbau zur Verschnellerung der Platine ist bereits ausgeführt). Diese Karte läuft jetzt auch ohne Wartezyklen...

- D/A - Wandler. Analogausgabe für den NDR-Computer. Z.B. für Motorsteuerungen oder Helligkeitsregelungen usw.

- COL256 mit noch mehr Möglichkeiten: 256 Graustufen, TTL-RGB-Ausgänge zum Anschluß von z.B. IBM-Farbmonitoren (diese können normalerweise nur 16 Farben darstellen, mit COL256 mind. 100!) Serienfertigung frühestens Weihnachten (Beschreibung in Heft 4). Erweiterung auf Farbtabelle möglich: dann 256 aus ca. 256000 Farben! Mit Baustein 6170 (siehe c't 11/85).

- ACRTC mit dem neuen Hitachi-Controller: 6-lagige Platine, teuere Spezialbauteile: Einzelpreis mindestens 1500.--, Farbe nochmal soviel dazu. Soll für intelligente Graphik-Subsysteme verwendet werden, die auf Basis des NDR-Computers aufgebaut werden sollen. Also Hobbyisten: Sparen ist angesagt!

- REL mit 8 Relais: z.B. für Modelleisenbahnsteuerungen oder Roboter und ähnliches. (Anwendungen gesucht: bitte melden...)

- für CP/M68K auf dem NDR-Computer gibt es einen Z80 CP/M2.2-Emulator, d.h. alle CP/M-Programme laufen! Z.B. Wordstar, Disk-Utilities, DBASE usw.

Tips und Tricks - Tips und Tricks - Tips und Tricks - Tips und Tricks

---

- bei Problemen mit der Baugruppe SER, wenn der Baustein 6551 von Rockwell, Typ Mexico eingesetzt ist: unbedingt Serienresonanzquarz (Achtung: selten!) besorgen und keine Kondensatoren und keinen Widerstand am Quarz einlöten.

- FLO2: bei 8-Zoll-Laufwerken unbedingt eine dicke Masseleitung ans Gehäuse des Laufwerks legen! (Gut auch bei 5 1/4 Zoll!). Bei Betrieb mit 2 Floppylaufwerken 5 1/4 Zoll unbedingt bei einem den Abschlußwiderstand entfernen, und zwar beim in der Mitte des Kabels liegenden Laufwerk. Dies ist das flache IC-ähnliche vergossene Ding, was in der Nähe des Floppy-Steckers liegt. Ansonsten Umbauanleitung in der LOOP5 beachten!

- RAM64/256 r4 unter CPU68K: neue Jumperleiste beachten (unterhalb IC 19) bisher o-o o o-o o, besser o o-o o o-o. Dies verhindert die Erzeugung von Wartezyklen, wenn die Karte nicht angesprochen wird.

- Z80-Emulator: die EPROMs mit dem Grundprogramm müssen auf der Adresse $E0000 liegen, wird vom Emulator abgefragt (eine Art Kopierschutz).

68020-Corner - 68020-Corner - 68020-Corner - 68020-Corner

Neues vom 68020: Es soll die Serie laufen, also nicht mehr XC68020, sondern endlich MC68020. Das bedeutet endlich Fehlerfreiheit (relativ). Auch der Preis soll deutlich gesunken sein (unter 1000,-- , jedoch bitte erst anfragen...). Ich empfehle allen ernsthaft am 68020 Interessierten folgendes Buch:

MC68020 32-Bit Microprocessor User's Manual
PRENTICE-HALL, Inc., Englewood Cliffs, N.J. 07632
ISBN 0-13-541418-0

von Motorola, erhältlich im Buchhandel oder bei Distributoren von Motorola oder Motorola direkt unter der Nummer MC68020UM. Kostenpunkt um 79,-- DM. Dies ist das ausführlichste Buch über den 68020 überhaupt, unbedingt zu empfehlen.

Der 68020 ist leider ein zu umfangreiches Thema für unsere Broschüre, um es vollständig zu behandeln. Echte Interessenten haben eh das obige Buch, den anderen tun die Seiten weh, die sie nicht interessieren. Kompromiß: die 68020-Corner enthält in Zukunft wirklich nur noch aktuelle Informationen zur CPU68020-Baugruppe für den NDR-Computer.

Endlich: Getrennte Wartezyklenerzeugung für Speicher- und Peripheriebaugruppen auf der CPU-Baugruppe CPU68k.

Auf der CPU-Karte braucht man nur noch zusätzlich eine 2 ! 8-polige Stiftleiste für die Wartezykleneinstellung und ein IC 74LS153 sowie einen Widerstand mit 1K-Ohm bis 4,7KOhm. Zwei Inverter sind auf der CPU-Baugruppe vorhanden. Die Verdrahtung geht aus dem Schaltplan hervor.

Das IC 74LS153 hat die folgenden Funktionen:
Durch den Eingang -SEL wird die Freigabe des Ausgangs Y gesteuert; ist -SEL auf high, so ist auch Y immer auf high. Wenn -SEL auf low ist, hier bei einem -DS der Fall, so kann Y auch low werden, abhängig von den Eingängen. Die Eingänge sind 0, 1, 2 und 3. Welcher Eingang ausgewählt ist, wird mit den beiden Auswahlleitungen A und B bestimmt. Diese sind binär codiert. Ist B low, wird in unserer Schaltung auch Y low, da 0 und 1 mit low verbunden sind. Ist B high, entscheidet A ob 2 oder 3 als Eingang gewählt wird. Wenn A high ist (kein I/O), sind die mit Memory bezeichneten Wartezyklen gültig.

Auf der I/O-Leiste werden die benötigten Wartezyklen für alle Peripheriebaugruppen gesteckt. Die langsamste I/O-Baugruppe ist ausschlaggebend. Bei uns ist dies im allgemeinen noch die FLO2 (Jumper auf Position 3).

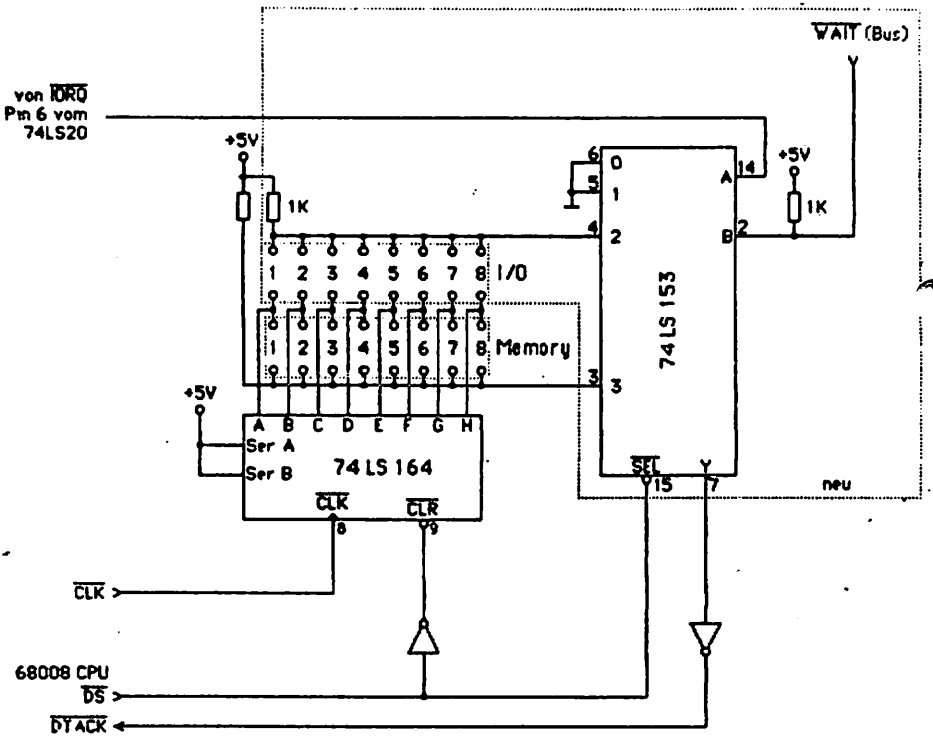
Auf der Memory-Leiste werden die für den Speicher maßgeblichen Wartezyklen eingestellt. Die langsamsten werden hier wohl EPROM's auf einer ROA44 sein. 250-ns-EPROM's benötigen jedoch in vielen Fällen noch keine Wartezyklen (keinen Jumper stecken).

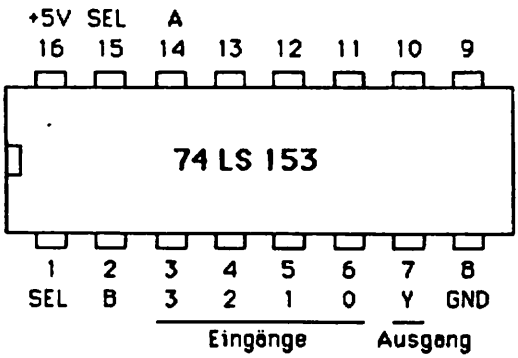
Für dynamische RAM-Karten ist vorgesorgt - die Leitung -WAIT des BUS ist auch an unsere Schaltung angeschlossen.

Achtung: Bei CPU68k r4 ist der Aufbau schwieriger - dort wurde schon eine andere Lösung für das -WAIT auf dem BUS eingebaut, die mit unserer nicht sehr verträglich ist.

Beim Aufbau dieser Schaltung sollte man unbedingt einen Schaltplan zum Vergleich dabeihaben, um nicht einige Schaltungsteile doppelt aufzubauen, denn viele Verbindungen sind schon normal vorhanden (z.B. 74LS164 und der Inverter vor Pin 9 und der Inverter, der auf -DTACK geht usw.).

Axel Granel


WAIT (Bus)
von IORQ
Pin 6 vom
74LS20
+5V
1K
1 2 3 4 5 6 7 8 I/O
1 2 3 4 5 6 7 8 Memory
+5V
A B C D E F G H
Ser A
Ser B
74 LS 164
CLK
CLR
74 LS 153
+5V
1K
SEL
neu
CLK
68008 CPU
DS
DTACK


+5V SEL A
16 15 14 13 12 11 10 9
74 LS 153
1 2 3 4 5 6 7 8
SEL B 3 2 1 0 Y GND
Eingänge
Ausgang

```
/*-------------------------------------------------------------------
 |  ###  KEULE ### > Otto Jusko 9/85                                 |
 |  Ursprünglich VAX 11/780 FORTRAN 77)                              |
 | Dieses Programm berechnet Kugelfunktionen Y - l,m                 |
 |  ( Zustände des Wasserstoff - Atoms )                             |
 |  Die gewuenschte Fkt. muß in Function CalcFkt() eingegeben werden |
 |-------------------------------------------------------------------|

#include tdio.h

#define lim 30
#define Pi 3.1415926
#define xfak 120.0
#define yfak 70.0
#define Xoffset 255
#define Yoffset 128

extern float pow(),sin(),cos();

static float Z[lim][lim],X[lim][lim],Y[lim][lim];
static short X2[lim][lim],Y2[lim][lim],Z2[lim][lim];

main ()
{
 short i,hp,p1,p2;
 float dalpha, alpha;

 puts ("\32  Keule : ein Programm zum Plotten von Kugelfunktionen Y -l,m ");
 puts (" ------------------------------------------------------------");
 puts ("\n\n\n  Dies ist Y-3,1 ");
 puts ("\n Einen Moment bitte ... ");
 CalcFkt();
 p1 = 2; p2 = 3;
 newpage (p1,p2);
 dalpha = Pi / 10.0;
 alpha = dalpha;
 for ( i = 0; i 10; ++i)
 {
   Drehe(alpha);
   hp = p1; p1 = p2 ; p2 = hp;
```

```
    newpage(p1,p2);
    clpg();
    Zeichne();
    alpha += dalpha;
  }
 }



VOID CalcFkt ()
{
 static float dphi,dtheta,F,st,ct,theta,phi;
 register short i,j;

 dphi = 2.0 * Pi /lim;
 dtheta = Pi / lim;
 theta =.0;
 for (i=0; i lim; ++i)
 {
  phi = dphi;
  ct = cos(theta);
  st = sin(theta);
  F =  st * (5.0 * ct*ct - 1 ); /* hier Kugelfkt. eingeben */
  F = abs(F);
  for (j=1; j lim; ++j)
  {
   X[i][j] = F * cos(phi) * st;
   Y[i][j] = F * st * sin(phi);
   Z[i][j] = F * ct;
   phi += dphi;
  }
  theta += dtheta;
 }
}

VOID Drehe(winkel)
float winkel;
{
 float a,b;
 register short i,j;
```

```
 a = cos (winkel);
 b = sin (winkel);

 for (i=0; i lim; ++i )
 {
  for (j=1; j lim; ++j )
  {
   Y2[i][j] = Xoffset + (short) ( xfak * Y[i][j] );
   Z2[i][j] = Yoffset + (short) ( yfak * ( b * X[i][j] + a * Z[i][j] ));
  }
 }
}

VOID Zeichne()
{
 register short i,j;

 for ( i=0; i lim; ++i)
 {
  moveto (Y2[i][1],Z2[i][1]);
  for (j=2; j lim; ++j) drawto(Y2[i][j],Z2[i][j]);
  drawto (Y2[i][1],Z2[i][1]);
 }
 for (j=1; j lim; ++j)
 {
  moveto (Y2[0][j],Z2[0][j]);
  for (i=1; i lim; ++i ) drawto (Y2[i][j],Z2[i][j]);
 }
}
```

Karl Heinz Weiland 3. Folge

Programmieren in Assembler

Irgendwann werden die meisten Programme relokativ sein müssen. Deshalb werde ich bei allen Programmen nur noch mit den 'Trap' - Befehlen arbeiten. Versuchen Sie doch mal selbst die bisher gezeigten Beispiele relokativ zu machen. Schauen Sie auch mal in das Buch vom Franzis-Software-Service 'Der NDR-Klein-Computer, 68008 Grundprogramm, Rev. 4.3'. Eigentlich finden Sie dort alles, was Sie brauchen, um sich mit den Befehlen vertraut zu machen.

Richten wir uns gleich nach diesem Buch. Der nächste Befehl ist hier der Befehl 'MOVETO'. Mit diesem Befehl werden die Koordinaten einer Schreibstelle bestimmt. Wenn wir das abgedruckte Programm von Rolf - Dieter Klein relokativ schreiben, dann sieht dies so aus:

```
start:
        move #10,d1             ! X-Achse = 10
        move #30,d2             ! Y-Achse = 30
        move #!moveto,d7        ! Position setzen
        trap #1
        move #$80,d0            ! Befehl für ASCII-Zeichen $80
        move #!cmd,d7           ! Befehl an den GDP schicken
        rts
```

Nehmen wir $48 an Stelle von $80, erscheint unten links an Stelle eines Punktes ein 'H'.
Mit dem Befehl 'CMD' kann man Befehle direkt an den GDP ausgeben. Es gibt eine ganze Reihe von Befehlen, die man bei der Beschreibung der GDP EF9366 findet. Bevor wir uns näher damit befassen, wollen wir uns erst noch die Befehle 'WAIT' und 'SYNC' betrachten.
Will man Daten direkt auf die GDP - Ports ausgeben, muß man warten, bis der GDP den letzten Befehl ausgeführt hat. Herzu bnutzen wir den Befehl 'WAIT'

```
start:
        move #!wait,d7
        trap #1
        rts
```

Auch mit dem Befehl 'SYNC' kann man den GDP direkt ansprechen. Man kann mit diesem Befehl auch quarzgenaue Messungen vornehmen. Bei einer Dauerabfrage werden alle 20ms der Wert $FFFF geliefert. Hier das Beispiel aus dem Grundprogrammbuch:

```
uhr:        clr d4              ! Zeitzähler
schleife:   move #50-1,d3       ! 50 ! 20ms
warte:      move #!sync,d7      ! messen
            trap #1
            beq warte           !    bis 20ms
            dbra d3,warte       !    bis 1 sec
            add #1,d4
            lea buffer,a0       ! ausgeben
            move d4,d0          ! muß klein 20ms sein
            ext.l d0            ! erst Minuten
            divs #60,d0         ! ausgeben
            move.l d0,-(a7)     ! merken
            move #!print4d,d7   ! dezimal ausgeben
            trap #1
            move.b #':',(a0)+   ! trennen
            move.l (a7)+,d0     ! dann Sekunden
            swap d0             ! ist Modulo-Rest
            move #!print4d,d7   !    bei DIVS
            trap #1
            move.b #' ',(a0)+   ! zum Löschen der
            clr.b (a0)          !    1er Stelle
            lea buffer,a0       ! und auf Bildschirm
            move #$22,d0        ! Schriftgröße
            move #2,d1          ! X-Achse = 2
            move #128,d2        ! Y-Achse = 128
            move #!write,d7     ! und ausgeben
            trap #1
            bra schleife
buffer:     ds.b 100            ! Textbuffer
```

Durch Verändern der Schriftgröße und der Achsen kann man die Ausgabegröße und Lage neu bestimmen. Versuchen Sie doch mal dieses Programm in andere Programme als Unterprogramm einzubinden. Sie haben dann ständig die Betriebsdauer-Anzeige. Auch für flimmerfreie bewegte Grafiken ist der Befehl 'SYNC' wichtig.

Der Befehl 'DRAWTO' zeichnet eine Linie von einem mit 'MOVETO' festgelegten Ausgangspunkt zu einem 2. Koordinaten-Schnittpunkt. Die Start- und Endpunkte können auch außerhalb des Bildschirms liegen.
Zum zeichnen einer Linie geben wir den Startpunkt mit z.B.

```
move #20,d1
```

```
        move #30,d2
        move #!moveto,d7
        trap #1
und den Endpunkt mit    move #500,d1
                        move #250,d2        ein.
Mit dem Befehl          move #!drawto,d7
                        trap #1             wird die Verbindung gezeichnet.
```

Die Befehle 'MOVETO' und 'DRAWTO' arbeiten mit den Registern 'd1' und 'd2'.
Die Befehle 'CMD' und 'SYNC' Benutzen das Register 'd0.b', also move.b ....
Der Befehl 'WAIT' steht für sich allein.

Der Befehl 'CI' liest Zeichen von der Tastatur ein. Auch hier ein kleines abgewandeltes Programm von Rolf-Dieter Klein als Beispiel:

```
start:  move #!ci,d7            ! Zeichen einlesen
        trap #1
        cmp.b #'f',d0           ! war es die Taste 'f'
        bne.s st1               ! wenn nicht, weiter
        move #10,d0             ! wenn ja, 10 Schritte
        move #!schreite,d7      ! mit der Schildkröte
        trap #1
        bra.s start             ! dann wieder zurück
st1:    cmp.b #'d',d0           ! war es die Taste 'd'
        bne.s st2               ! wenn nicht, weiter
        move #45,d0             ! wenn ja, 45 Grad
        move #!drehe,d7         ! Schildkröte drehen
        trap #1
        bra.s start             ! dann wieder zurück
st2:    cmp.b #'h',d0           ! war es die Taste 'h'
        bne.s st3               ! wenn nicht, weiter
        move #!hebe,d7          ! wenn ja, heben
        trap #1
        bra.s start             ! dann wieder zurück
st3:    cmp.b #'s',d0           ! war es die Taste 's'
        bne.s ende              ! wenn nein, dann weiter
        move #!senke,d7         ! wenn ja, senken
        trap #1
        bra.s start             ! dann wieder zurück
ende:   cmp.b #'e',d0           ! war es die Taste 'e'
        bne.s start             ! wenn nicht, zurück
        rts                     ! wenn ja, Ende
```


Fortsetzung folgt

```
* Programm zum Verschieben von zusammenhaengenden Speicherbereichen
* von E. Payerl, Hedderichstr. 96, 6000 Frankfurt/M. 70.
* Die einzugebende Laenge des zu verschiebenden Bereichs muss
* eine ganze Zahl sein, also $890 statt $88f.

        org $0
        offset $89c00
        dc.l $55aa0180
        dc.b 'Schieben'
        dc.l start
        dc.l ende-start
        dc.b 1
        dc.b 0,0,0
        dc.l 0,0
        dc.l 0
        ds 0
start:
        move.b #$22,d0
        move #2,d1
        move #230,d2
        lea text1(pc),a0
        move #!write,d7
        trap #1
        move.b #$22,d0          * Textgroesse
        move #2,d1              * X-Koordinate
        move #210,d2            * Y-Koordinate
        move #7,d3              * maximale Anzahl der einzulesenden Zeichen
        lea buffer(pc),a0       * Adresse des Textbuffers in a0
        move #!read,d7          * Einleseprozedur des Grundprogramms
        trap #1
        lea buffer(pc),a0
        move #!wert,d7          * Text wird in arithmetische Groesse uebersetzt
        trap #1                 * Ergebnis wird in d0 abgelegt
        movea.l d0,a1           * Anfang des zu verschiebenden Speicherbereiches
                                * wird in a1 gespeichert
        move.b #$22,d0
        move #2,d1
        move #190,d2
        lea text2(pc),a0
```

```
        move #!write,d7
        trap #1
        move.b #$22,d0          ! Wie oben, es wird die Laenge des zu ver-
        move #2,d1              ! schiebenden Speicherbereichs eingegeben.
        move #170,d2
        move #7,d3
        lea buffer(pc),a0
        move #!read,d7
        trap #1
        lea buffer(pc),a0
        move #!wert,d7
        trap #1
        movea.l d0,a2           ! Laenge wird in a2 zwischengespeichert
        move.b #$22,d0
        move #2,d1
        move #150,d2
        lea text3(pc),a0
        move #!write,d7
        trap #1
        move.b #$22,d0          ! Eingabe der Zieladresse
        move #2,d1
        move #130,d2
        move #7,d3
        lea buffer(pc),a0
        move #!read,d7
        trap #1
        lea buffer(pc),a0
        move #!wert,d7
        trap #1
        movea.l d0,a3           ! Zieladresse kommt nach a3
        subi.l #1,(a2)          ! Zaehlervorbereitung
        move a2,d7              ! Zaehler nach d7

loop:
        move.b (a1)+,(a3)+      ! Uebertragen der Inhalte. Hochsetzen um 1.
        dbra d7,loop
        rts

buffer:
        ds.b 100
```

```
text1:
        dc.b 'Startadresse:',0
text2:
        dc.b 'Laenge:',0
text3:
        dc.b 'Zieladresse:',0

ende:
end
```

* Dieses Programm nach dem Assemblieren ueber die Bibliothek aufrufen. Es wird
* dann nach der Startadresse gefragt. Nach der Eingabe der Startadresse fragt
* das Programm nach der Laenge, die in '$xxx' eingegeben werden muß. Danach wird
* noch nach der Zieladresse gefragt.
* Z.B. ein Programm von 2048 Bytes beginnt bei Adresse $10000 und soll nach Ad-
* resse $24000 gelegt werden. Dies geschieht wie folgt:
*   Startadresse: $10000
*   Laenge: $800
*   Zieladresse: $24000
* Hiernach liegt der Programmanfang auf Adresse $24000 und $10000, da das Pro-
* gramm an der alten Adresse nicht geloescht wurde, sondern zur neuen Adresse
* kopiert wurde.
* Beim Grundprogramm Version 4.3 kann man auch Programme mit den Marken 'Ctrl
* kb, kk und kw' verschieben, solange diese im Editor sind. Sollen jedoch Spei-
* cherbereiche verschoben werden, d.h. Daten, dann ist dieses Programm sehr von
* Nutzen. Man muß aber beachten, daß das Programm relokativ ist.

3. Folge

Nachdem wir die Figur nun um die x-Achse gedreht haben, fehlen nur noch die y- und die z-Achse. Da das Programm so übersichtlich aufgebaut ist, ist es überflüssig, das gesamte Programm noch einmal abzudrucken, es werden nur die Unterprogramme für die anderen Achsen eingefügt. Sie sind unten abgedruckt.

Die Formeln sind den Anmerkungen zu entnehmen. In der nächsten Ausgabe wird es darum gehen, bei Körpern die verdeckten Linien zu löschen, damit nur die sichtbaren Seiten gezeichnet werden. Dies Programm kann dann mit der Drehung, die ja auch für beliebige Figuren eingesetzt werden kann, kombiniert werden.

```
*****************
alpha: dc.w 000 * Drehwinkel um z
beta:  dc.w 000 * Drehwinkel um y
gamma: dc.w 000 * Drehwinkel um x
*****************

.....

drehalpha: * Drehung um die z-Achse mit Datenveraenderung an x und y
lea dehndata,a0 * Berechnungsformel : x'=x*cos(alpha)+y*sin(alpha)
move #anz1-1,d5                       y'=-x*sin(alpha)+y*cos(alpha)
wdh2:
move (a0),d1  * x nach d1
move 2(a0),d2 * y nach d2
move d1,d3
move d2,d4
move alpha,d0 * Drehwinkel in d0
jsr §cos
muls.l d0,d1 * x mit cos alpha multiplizieren
move alpha,d0
jsr §sin
muls.l d0,d2 * y mit sin alpha multiplizieren
add.l d2,d1
divs.l #256,d1
move d1,(a0) * x -> x'
*****************
move d3,d1  * x nach d1
move d4,d2  * y nach d2
move alpha,d0
```

```
jsr §sin
neg d1
muls.l d0,d1
move alpha,d0
jsr §cos
muls.l d0,d2
add.l d1,d2
divs.l #256,d2
move d2,2(a0) * -y -> y'
*****************
adda #6,a0
dbra d5,wdh2
rts
drehbeta: * Drehung um die y-Achse mit Datenveraenderung an x und z
lea dehndata,a0 * Berechnungsformel : x'=x*cos(beta)+z*sin(beta)
move.#anz1-1,d5                      z'=-x*sin(beta)+z*cos(beta)
wdh3:
move (a0),d1  * x nach d1
move 4(a0),d2 * z nach d2
move d1,d3
move d2,d4
move beta,d0 * Drehwinkel in d0
jsr §cos
muls.l d0,d1 * x mit cos beta multiplizieren
move beta,d0
jsr §sin
muls.l d0,d2 * z mit sin beta multiplizieren
add.l d2,d1
divs.l #256,d1
add mx,d1    * x + Mittelpunkt x
move d1,(a0) * x -> x'
*****************
move d3,d1  * x nach d1
move d4,d2  * z nach d2
move beta,d0
jsr §sin
neg d1
muls.l d0,d1
move beta,d0
jsr §cos
```

```
muls.l d0,d2
add.l d1,d2
divs.l #256,d2
move d2,4(a0) ! z -> z'
!!!!!!!!!!!!!!!!
adda #6,a0
dbra d5,wdh3
rts
drehgamma: ! Drehung um die x-Achse mit Datenveraenderung an y und z
lea dehndata,a0 ! Berechnungsformel : y'=y!cos(gamma)+z!sin(gamma)
move #anz1-1,d5                      z'=-y!sin(gamma)+z!cos(gamma)
.....          (siehe Nr. 2)

zweidtrans: ! Umrechnung in 2 D Koordinaten
lea dehndata,a0 ! Berechnungsformel : x' = (yo!x)/(yo-z)
lea zweidk,a1   !                     y' = (yo!y)/(yo-z)
move #anz1-1,d5
wdh5:
.....          (siehe Nr. 2)

start:
move #0,gamma
move #0,beta
move #0,alpha
wdhstart:
jsr dehnung
jsr drehalpha
jsr drehbeta
jsr drehgamma
jsr zweidtrans
jsr ausgabe
add #3,gamma          ;nur als Beispiel
add #3,beta           ;eigene Unterprogramme mit
add #3,alpha          ;Tatstaturabfrage möglich
jsr wechsel           ;Bildschirmumschaltung
bra.s wdhstart
```

Bei diesem Programm wird erreicht, daß mit dem Menüpunkt 4 'Floppystart' die ersten 50 K ab der ersten Spur der Diskette gelesen werden. Die Werte und Adressen können angepaßt (s.u.) werden. Beim ersten Programmstart wird das vorher assemblierte Programm auf die Spur 0, Sektor 1 geladen. Das Directory der Diskette wird dadurch nicht berührt, da es erst bei Sektor 2 beginnt. Das assemblierte Programm muß auf Adresse $9c00 (bzw. $e9c00) liegen. Gestartet wird dann mit 'begin'.

```
!!!!!!!!!!!!!!!!!!
! Autoload 68K   !
! u. Autostart   !
! V. 1.0         !
! H.G. Eysel     !
! 850907         !
!!!!!!!!!!!!!!!!!!
```

```
!!!!! die ersten 50 K des Speichers (ab $400) werden
!!!!! beim Floppystart in den Speicher geladen
!!!!! das erste Bibliotheksmenü wird gestartet
```

```
nop                         ;Kennung für Betriebssystem
st:                         ;Start
move #1,d4
move #!getflop,d7           ;Dichtecode
trap #1
move #50,d6                 ;50 K                  !:1:!
move #1,d1                  ;lesen
move #0,d3                  ;später 1. Spur
movea.l #$400,a0            ;Speicherstart         !:2:!
wdh1:
add #1,d3                   ;nächste Spur
move #1,d2                  ;Sektor 1
wdh2:
move #!floppy,d7
trap #1
adda.l #$400,a0             ;erhöhen
add #1,d2                   ;nächster Sektor
sub #1,d6
cmp #0,d6                   ;127 K geladen?
```

```
beq.s autostart
cmp #6,d2                    ;Sektor voll?
beq.s wdh1
bra.s wdh2
autostart:
move #!grapoff,d7            ;sicherheitshalber
trap #1
move.l #$4000,d0             ;Spur4, Sektor 1        #:3:#   #
move #!putstx,d7             ;Textstart setzen               #
trap #1                                                      #
lea $400,a0                  ;1. Bibliothekspr.      #:4:#
cmp.l #$55aa0180,(a0)        ;Bibl.?
bne.s ende
movea.l 12(a0),a1            ;Start
cmp.b #1,20(a0)              ;reloc ?
bne.s st2
adda.l a0,a1
st2:
jsr (a1)                     ;Programm starten
ende:
rts

begin:                       ;Erststart
lea $1c00(a5),a0             ;wegen CPM
                             ;hier assembliertes Programm
move #1,d4
move #!getflop,d7            ;Dichtecode
trap #1
move #2,d1                   ;schreiben
move #1,d2                   ;1. Sektor
move #0,d3                   ;Spur 0
move #!floppy,d7
trap #1
rts
end
```

Zu den Anmerkungen #:n:#

1) Dieser Wert gibt die Größe des Speicherbereiches an. Hier werden 50K von der Diskette ab $400 (siehe 2) geladen.

2) Der Speicherstart kann ebenfalls geändert werden. Davon wird aber auch
   $:3:$ betroffen. Für die ursprüngliche Version müßte hier $9000 stehen.
3) Der Textstart wird auf $4000 gelegt. Bei entsprechender Software (in 4)
   sollte hier (d.h. auf der Diskette Spur 4, Sektor 1) ein Text stehen.
   Dies gilt für Computer mit Boot-Eprom, sonst lautet die Adresse $9000.
4) Am definierten Speicherstart (s.a. 2) kann ein Programm mit Bibliotheks-
   vorspann liegen. Dies wird dann automatisch gestartet. Mit $:3:$ wird
   der Textstart für dieses Programm festgelegt.
   Steht hier aber kein Text, so sucht das Programm vergeblich nach der 0
   des Endes. Dieser Teil sollte dann nicht mit eingegeben werden (s.o.
   Zeilen mit den '#' Zeichen im Kommentar).


```
$$$    Disketten-Prüfprogram    $$$
$ von Ernst Payerl,  Frankfurt/M. $

org $a000

$ Variable
spur: dc.b 0
sektor: dc.b 0

start:
clr d4
move #$21,d4            $ Dichtecode
start1:
move.b #1,spur
start2:
move.b #1,sektor
clr d1
clr d2
clr d3
move.b spur,d3          $ Spur
sektorwechsel:
lea buffer,a0           $ Zieladresse
move #2,d1              $ schreiben
move.b sektor,d2        $ Sektor
```

```
move #!floppy,d7
trap #1
lea buffer,a0
move #1,d1              * lesen
move #!floppy,d7
trap #1
lea buffer,a0
move #512-1,d1          * Zähler: Es wird wortweise überprüft,
                        * deswegen nicht 1023.
zaehler:
move (a0)+,d6           * ist Inhalt der Adresse gleich 0 ?
bne fehler              * wenn nein, dann Ausgabe Fehlertext.
dbra d1,zaehler
add.b #1,sektor         * der nächste Sektor
cmp.b #5,d2             * bei 5 ist Schluß
bne sektorwechsel
add.b #1,spur           * die nächste Spur
cmp #79,d3
bne start2
add #$80,d4             * jetzt Rückseite ansprechen
cmp #$121,d4            * es gibt nur eine Rückseite
bne start1
rts

fehler:
movem.l d0-d4/d6/a0,-(a7)       * auf Stack retten
move.b #$32,d0                  * Schriftgröße
move #121,d1                    * X-Koordinate
move #128,d2                    * Y-Koordinate
lea fehlertext,a0
move #!write,d7
trap #1
movem.l (a7)+,d0-d4/d6/a0       * Daten von Stack zurück
rts

buffer: ds.b 1024
fehlertext:
dc.b 'FEHLER ENTDECKT',0
ende: end
```

Kapazitive Leistungsverminderung von ohmschen Verbrauchern.

Das nachfolgende Programm ermöglicht die Berechnung eines Vorschaltkondensators zur Leistungsverminderung. - Was versteht man darunter? Nehmen wir ein einfaches Beispiel: Jeder kennt einen Lichtdimmer. Ein Dimmer ermöglicht das stufenlose Einstellen der Helligkeit einer Glühlampe. Die gewünschte Helligkeit wird meist auf dem Einstellknopf markiert um sie jederzeit wiederzufinden.
Mit diesem Programm kann man sich nun einen Kondensator berechnen, der die Glühlampe von z.B. 60 Watt auf 30 Watt reduziert.
Und das geht folgendermaßen:
Nach dem Programmstart wählt man den Programmpunkt 1. Jetzt wird man nach der Leistung des Geräts gefragt. Wir tippen 60 ein. Nun wird nach der reduzierten gefragt, also 30 eintippen. Nach 'CR' erhalten wir die Ergebnisse:

Leistung 1	:	60 Watt
Verminderte Leistung	:	30 Watt
Kondensatorwert	:	3.945 Mikrofarad
Spannung am Kondensator	:	155.56 Volt
Aufgenommener Strom	:	0.192 Ampere

Natürlich hat niemand einen Kondensator von 3.945 Mikrofarad. Deshalb nehmen wir einen ähnlichen normierten Wert (hier 4.2 Mikrofarad). Nun wählen wir Programmpunkt 2 an, da wir ja wissen wollen, auf wieviel Watt wir nun mit diesem Kondensator reduziert haben. Zuerst geben wir wieder die ursprüngliche Leistung der Lampe, also 60 Watt, ein. Nun geben wir den Kondensatorwert, also 4.2 Mikrofarad ein. Nach 'CR' erhalten wir die Ergebnisse:

Leistung 1	:	60 Watt
Verminderte Leistung	:	31.869 Watt
Kondensatorwert	:	4.2 Mikrofarad
Spannung am Kondensator	:	150.63 Volt
Aufgenommener Strom	:	0.198 Ampere

Wir wählen also einen Kondensator mit den Werten 4.2 Mikrofarad/250 Volt. Dieser wird mit der Lampe in Reihe geschaltet.
Der Vorteil eines Kondensators gegenüber einem Vorwiderstand liegt in der nahezu verlustleistungslosen Art der Spannungsreduzierung. Dazu kommt, daß durch eine Verminderung der Spannung am Glühfaden um nur 10 % sich die Lebnsdauer der Lampe um das 10-fache verlängert. Mit etwas nachdenken wird man noch andere Verwendung finden für deise Art der Leistungsverminderung. (z.B.: Lötkolben in Lötpause auf 1/2 Leistung; Stufenschaltung für Ätztemperatur;.....)

Sollte noch jemand Fragen haben dann schreibt mir ruhig:
Hans Herbert Stärkert, Gleiwitzer Weg 10, 3380 Goslar

```
(**********************************)
(*  Kapazitive Leistungsminderung *)
(*     für ohmsche Verbraucher    *)
(**********************************)

program leistung (input,output);

const   u = 220.0;
        f = 50.0;
        pi = 3.141592654;

var     r,i,z,xc,p1,p2,c,uc : real;
        x : integer;
        ch : char;
procedure menue;
 egin
  writeln ('Sie haben zwei Wahlmöglichkeiten :');
  writeln ('==================================');
  writeln;
  writeln ('1. Berechnung eines Kondensators');
  writeln ('   zur Leistungsreduzierung');
  writeln ('   eines Verbrauchers');
  writeln;
  writeln ('2. Leistungsverminderung beim Vor-');
  writeln ('   schalten eines Kondensators');
  writeln;
  writeln;
    repeat
        write ('Geben Sie 1 oder 2 ein :        '); readln (x);
        if (x>2) or (x<1) then
     writeln ('Unzulässige Eingabe !');
  until (x<3) and (x>0) ;
end;
procedure menue2;

begin
 repeat
  writeln;
  writeln;
write ('Geben Sie bitte die Leistung des Gerätes ein :          '); readln (p1);
write ('Auf welche Leistung soll das Gerät reduziert werden :   '); readln (p2);
until (p1>p2) and (p1>0) and (p2>=0);
end;
```

```
procedure menue3;

begin
 repeat
  writeln;
  writeln;
  write ('Geben Sie bitte die Leistung des Geräts ein :        '); readln (p1);
  writeln ('Welchen Kondensator haben Sie zur');
  write ('Verfügung (in Mikrofarad) ?                          '); readln (c);
 until (p1>0) and (c>0);
end;
 procedure rech2;

 begin
   r := sqr(u) / p1;
   xc := 1.0e6 / (2 * pi * f * c);
   z := sqrt( sqr(r) + sqr(xc));
   i := u / z;
   p2 := sqr(i) * r;
   uc := xc * i;
 end;


 procedure rech1;

 begin
   r := sqr(u) / p1;
   i := sqrt ( p2 / r );
   z := u / i;
   xc := sqrt ( sqr(z) - sqr(r));
   c := 1e6 / (2 * pi * f * xc);
   uc := xc * i;
 end;

 procedure clr;

 begin
   writeln (chr(1),'E §clrscreen');
 end;
```

```
procedure ende;

begin
  writeln;
  writeln ('Für neue Werte beliebige Taste drücken');
  writeln ('Mit der Eingabe von 'M' zurück in's Menue');
  readln (ch);
end;


begin
 repeat
  menue;
  clr;
   case x of
             1 : menue2;
             2 : menue3;
   end;
  clr;

   case x of
             1 : rech1;
             2 : rech2;

   end;
  clr;
  writeln;
  writeln (' Ergebnisse :');
  writeln (' ============');
  writeln;
  writeln ('Leistung 1 :                          ',p1:9:3,' Watt');
  writeln ('Verminderte Leistung :                ',p2:9:3,' Watt');
  writeln ('Kondensatorwert :                     ',c:9:3,' Mikrofarád');
  writeln ('Spannung am Kondensator :             ',uc:9:3,' Volt');
  writeln ('Aufgenommener Strom :                 ',i:9:3,' Ampere');
  ende;
  clr;
 until ch = 'M';
end.
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! KOPIERPROGRAMM  VER 1.0    !
! von Otto Jusko             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 ORG $400

TRK: DS.W 1
SEK: DS.W 1
J: DS.W 1
K: DS.W 1
PART: DS.W 1

 ORG 0
 OFFSET $80000 ; ODER IM EPROM SONSTWO

HEAD:
DC.L $55AA0180
DC.B 'UCOPY   '
DC.L COPY-HEAD
DC.L ENDP-COPY
DC.B 1
DC.B 0,0,0
DC.L 0,0


COPY:
 JSR $CLRSCREEN
 LEA WELCOME(PC),A0
 BSR PRINT
 CLR D1
 MOVE #$83,D3
 JSR $FLOPPY
 CLR PART
PARTS:
 MOVE PART,D0
 MULU #20,D0
 MOVE D0,J
 ADD #19,D0
 MOVE D0,K
 LEA SOURCETXT(PC),A0
 BSR PRINT
 JSR $CI
 MOVE J,TRK
 LEA BUFFER,A1
SOURCETRK:
 MOVE #1,SEK
 MOVE.B #'R',D0
 JSR $CO2
SOURCESEK:
 MOVEA.L A1,A0
 MOVE #1,D1
 MOVE SEK,D2
 MOVE TRK,D3
 MOVE #$21,D4
 BSR FLOPPY
 ADDA.L #1024,A1
 ADD #1,SEK
 CMP #5,SEK
 BLS SOURCESEK
 ADD #1,TRK
 MOVE TRK,D0
 CMP K,D0
 BLS SOURCETRK
 LEA DESTINTXT(PC),A0
 BSR PRINT
 JSR $CI
 MOVE J,TRK
 LEA BUFFER,A1
DESTINTRK:
 MOVE #1,SEK
 MOVE.B #'W',D0
 JSR $CO2
DESTINSEK:
 MOVEA.L A1,A0
 MOVE #2,D1
 MOVE SEK,D2
 MOVE TRK,D3
 MOVE #$21,D4
 BSR FLOPPY
 ADDA.L #1024,A1
```

```
 ADD #1,SEK
 CMP #5,SEK
 BLS DESTINSEK
 ADD #1,TRK
 MOVE.TRK,D0
 CMP K,D0
 BLS DESTINTRK
 ADD #1,PART
 CMP #7,PART
 BLS PARTS
FIN:
 LEA ENDMESS(PC),A0
 BSR PRINT
RTS

FLOPPY:
 ROR.W #1,D3
 BCC.S FL01
 BSET #7,D4
FL01:
 AND.B #$7F,D3
 JSR §FLOPPY
RTS

PRINT:
 MOVE.B (A0)+,D0
 BEQ.S BACK
 JSR §C02
 BRA.S PRINT
BACK: RTS

WELCOME: DC.B 'Willkommen zu Otto's Single Drive Copy  V1.0 ',$D,$A
         DC.B '-----------------------------------------',$d,$A,$A,0

SOURCETXT: DC.B $D,$A,'INSERT SOURCE --------',0
DESTINTXT: DC.B $D,$A,'INSERT DESTINATION  --',0
ENDMESS:   DC.B $D,$A,' ** COPY COMPLETE **',0

BUFFER EQU $410   ; DUMMY , uses 100k Memory
ENDP:
 END
```