LOOP

8/9

2. JAHRGANG

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter 13.4.86 DM 6,–

Uwe Koch
Frankenstraße 25
5880 LÜDENSCHEID
Tel. (02351) 26192

NEU!

dBasell, WordStar, Multiplan

für den NDR-Computer und mc-CP/M-Computer

je DM 199,–

Preissenkung bei CP/M2.2-Betriebssystem: Nur noch DM 199,–

Professionelle Software zum extrem günstigen Preis!

Endlich ist es uns gelungen, diese drei Programmpakete auch für unsere Produkte preiswert vom Markt + Technik Software Verlag zu erhalten! Ab 15. April kann die Auslieferung beginnen – zu einem Preis, bei dem sich kopieren nicht mehr lohnt!

Um allen Irrtümern vorzubeugen – es handelt sich um die offiziellen, voll lauffähigen und uneingeschränkten Versionen! Natürlich gehört die komplette Dokumentation mit dazu (im Preis enthalten)!

Die Programme laufen unter dem Betriebssystem CP/M2.2 und erfordern einen Arbeitsspeicher von nur 64 KByte.

Eine RAM-Floppy – beim mc-CP/M-Computer mit der RFLO-Baugruppe, beim NDR-Computer durch einfachen Speicherausbau auf 256 KByte oder mehr – beschleunigt das Arbeiten, ist aber nicht unbedingt erforderlich.

Nun ein kurzer Steckbrief der Programme:

dBasell

dBasell ist ein relationales Datenbanksystem.

Mit diesem Programm können Sie zunächst einmal sehr leicht Datenbanken, also Sammlungen von Datensätzen aufbauen und verwalten.

Dies können Adressen sein, Schallplatten, Literaturhinweise, Lagerinhalte, Buchhaltungsdateien und vieles mehr.

dBasell verfügt nun über mächtige Kommandos, um für eine solche Sammlung von Daten:

- Platz auf der Diskette (oder Platte) anzulegen
- Daten einzutragen
- Daten zu sortieren
- Daten zu ändern
- kurz, Daten zu verwalten.

Das Sortieren geht extrem schnell, da dBase bei einem Sortierlauf nicht die Datensätze, sondern nur sogenannte „Indexdateien" bewegt. Genau so schnell geht auch der Zugriff auf Datensätze, egal ob nur z.B. 5 oder 5000 Adressen gespeichert sind.

Die Kommandos von dBasell lassen sich auch zu Programmen zusammenfassen: dBasell hat eine eigene Programmiersprache, die sehr strukturiert aufgebaut ist.

Mit dBasell lassen sich besonders leicht kommerzielle Programme, wie z.B. Fakturierung (Rechnungen schreiben) oder Lagerverwaltung programmieren.

GES wird bald für den praktischen Einsatz der mc- und NDR-Computer ein komplettes Fakturierprogramm mit Lager- und offener Posten-Verwaltung im dBasell-Quelltext anbieten.

Wordstar

Wordstar läßt sich sehr einfach beschreiben: Es ist das meist verkaufte Textverarbeitungsprogramm der Welt.

Wordstar erlaubt eine kommerzielle Textverarbeitung, wie man sie vor wenigen Jahren nur von sehr teuren Systemen kannte. Texte können am Bildschirm erstellt und editiert werden, auf Diskette oder Platte gespeichert und wieder, auch in Bausteinen, eingeladen werden. Das Programm kann einen automatischen

In eigener Sache

LOOP diesmal als Doppelnummer!

Wir hatten so viele interessante Artikel vorliegen, daß wir vor der Entscheidung standen: Die Hälfte der Artikel verschieben oder eine Doppelnummer herauszugeben.

Wir haben uns für die Doppelnummer entschieden und hoffen, damit in Ihrem Interesse gehandelt zu haben. Teilen Sie uns doch bitte Ihre Meinung dazu mit!

Ein herzlicher Dank geht an alle Autoren, die uns helfen, die *LOOP* zu einer wirklich guten Zeitschrift zu machen!

Bis zur nächsten *LOOP*.

Rolf-Dieter Klein Gerd Graf

Inhaltsverzeichnis:

dBasell, Wordstar, Multiplan je 199,– DM Seite 1

Die 10MByte Festplatte ist lieferbar Seite 2

MODULA2 für den NDR-Computer Seite 3

RL-BASIC – Endlich ein BASIC für die 680xx-Benutzer Seite 5

WIR STELLEN AUS

Hobby-tronic Dortmund

23. – 27. April 1986 · Halle 5 Stand 4001

Dort können Sie kaufen!

Umbruch machen, d.h. nach dem Einfügen stimmen die Zeilenlängen wieder – es kann Blocksatz, Seitennummern, Adressen einfügen und vieles mehr.

Multiplan

Multiplan ist der Vater der „Tabellen-Kalkulationsprogramme" und das meistverkaufte Programm dieser Art.

Zahlen oder Formeln werden in Felder, die in einer Matrix aufgebaut und am Bildschirm sichtbar sind, eingetragen. Wie nun diese Felder mathematisch miteinander verknüpft werden sollen, wird Multiplan ganz einfach am Bildschirm „erklärt".

Beispiel: Für eine Preisliste soll eine Tabelle aufgebaut werden, die die Verkaufspreise und Staffelpreise mit 5%, 10%, 15% und 20% Rabatt enthält.

Das Arbeitsblatt vom Multiplan wird dann wie folgt angelegt:

		1	2	3	4	5	6
1	Rabatt (%)		0	5	10	15	20
2							
3			100,–	95,–			
4							
5							

Dieses Arbeitsblatt ist nach rechts in SPALTEN und nach unten in REIHEN aufgegliedert. Multiplan kann 63 Spalten und 255 Reihen verarbeiten.

In Reihe 3, Spalte 2 wird dann z.B. der Verkaufspreis mit 0% Rabatt, etwa 100,– eingegeben.

In Reihe 3, Spalte 3 soll der um 5% (wie in Reihe 1, Spalte 3 angegeben) reduzierte Preis erscheinen.

Die Formel dazu, ausgesprochen, lautet:

Berechne: Inhalt von Reihe 3, Spalte 2 minus (Inhalt Reihe 1, Spalte 5) Prozent.

Fast genauso wird Multiplan dies mitgeteilt: Man gibt eine Formel einfach durch das Positionieren auf die entsprechende Zelle und Angabe einer Funktion, wie „plus", „mal", „%" an. Sofort nach Eingabe dieser Formel erscheint das Ergebnis.

Noch nicht so aufregend – das hätten wir mit etwas Kopfrechnen oder einem Taschenrechner gerade auch noch geschafft!

Nun – stellen wir uns diese Tabelle etwas nach unten ausgedehnt vor: Nicht nur ein Preis, sondern 100 solche Preise mit Rabatten sollen berechnet werden!

Multiplan schafft dies spielend – Formeln, die einmal erstellt wurden, lassen sich kopieren. So müssen nur noch die 100 VK-Preise eingegeben werden und Multiplan berechnet sekundenschnell die Ergebnisse.

Auch das wäre noch mit dem Taschenrechner möglich. Nun soll aber eine Spalte eingefügt werden, z.B. zwischen 5 oder 10% mit 7,5% Nachlaß – auch kein Problem. Nun wollen wir uns mal alle Preise ansehen, falls wir statt 20 – 22% Rabatt geben – einfach den Wert 20 in Reihe 1, Spalte 6 von 20 auf 22 geändert – und alle 100 Werte werden neu berechnet! Hier hätten wir mit dem Rechner doch etwas länger gebraucht!

Zu den drei Programmpaketen gibt es hervorragende Literatur. Mit dBasell, Wordstar, Multiplan wird der NDR-Computer oder der mc-CP/M-Computer zum wirklich professionellen System!

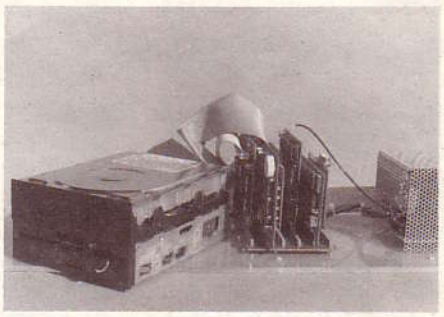
2-Jahres-Feier der Filiale Hamburg

Freitag, den 23. Mai 1986, von 10.00 bis 18.00 Uhr. Jedermann ist herzlich willkommen!

Neue Produkte – Diskussion – Rolf-Dieter Klein und Gerd Graf sind dabei!

JETZT LIEFERBAR

Die 10 MByte Festplatte ist lieferbar!



Der NDR-Computer und der mc-CP/M-Computer werden erwachsen! Beide Computer lassen sich nun mit einer 10 MByte Festplatte mit integriertem Controller ausrüsten.

Eine Festplatte ist im Prinzip nichts anderes als ein Diskettenlaufwerk, wobei die „Diskette" nicht mehr flexibel, sondern starr und fest (daher der Name) als Platte im Laufwerk eingebaut ist.

Diese Platte läuft in einem versiegelten Gehäuse, das beim Hersteller der Festplatte in einem besonders staubfreien Raum (Clean Room) geschlossen wurde.

Durch die stabile Mechanik und die Staubfreiheit, lassen sich so auf einer Platte 10 – 80 mal dichtere Aufzeichnungen herstellen, als auf einem Diskettenlaufwerk. Die Kapaziät der Festplatte beträgt 10 MByte, also etwa das 10fache einer Diskette.

Durch die Mechanik bedingt ist aber nicht nur die Kapazität, sondern auch die Zugriffsgeschwindigkeit deutlich höher. Es macht bei unserer Platte etwa den Faktor 5 aus, sie ist also etwa fünf mal schneller als die Diskette.

Der Schreib-Lesekopf einer Festplatte wird nicht, wie bei der Diskette, aufgepreßt, sondern schwebt in sehr geringem Abstand über der Platte. Dies ist allerdings auch etwas gefährlich: Eine Festplatte ist empfindlich gegen mechanische Beanspruchung, besonders dann, während sie arbeitet.

Ein kräftiger Stoß während des Zugriffs kann bereits zu ernsten Fehlern führen.

Die von GES angebotene Platte hat zwei Besonderheiten: – Einen eingebauten SASI-Controller, der dazu noch intelligenter und schneller ist als der externe. – Die gleichen Gehäuseabmessungen wie ein 5¼"-Laufwerk, Slim-Line.

Deshalb kann die Platte ohne Probleme in ein vorhandenes Gehäuse eingebaut werden. Sie benötigt lediglich die Baugruppe SASI-NDR (FESTCON) für den NDR-Computer oder FLOSASI für den mc-CP/M-Computer.

Die Platte funktioniert unter Z80 und dem Betriebssystem CP/M2.2 (dort beträgt die Kapazität jedoch nur 8 MByte) und unter der CPU 680XX und dem Betriebssystem CP/M68K mit 10 MByte.

Sie ist ab Lager lieferbar und kostet DM 1998,–.

PRODUKTBESPRECHUNG

MODULA-2 für den NDR-KLEIN-Computer mit CP/M68K

Rolf-Dieter Klein

Teil 1

Viele haben vielleicht schon vergeblich auf das Turbo-Pascal für CP/M68K gewartet. Doch leider hat sich bisher nichts von Seiten des Herstellers getan. Doch nun gibt es eine bessere Alternative: MODULA-2 von einer deutschen Firma.

MODULA ist eine Sprache, die Prof. Wirth als Weiterentwicklung der Sprache PASCAL entworfen hat und die viele interessante Eigenschaften hat. MODULA-2 war bisher unter CP/M68K nur als M-Code-Interpreter-System verfügbar und ist jetzt als echter Native-Code-Compiler da. Der MODULA-2-Compiler wurde von der Münchener Firma p1 extra für den NDR-KLEIN-Computer angepaßt und bietet daher einige interessante Fähigkeiten. *LOOP* wird von jetzt an regelmäßig MODULA-2-Programme bringen, wie auch eine Einführung in die Sprache.

Der MODULA-2-Compiler erzeugt Assembler-Quellen und erlaubt daher auch, Einblick in den erzeugten Code zu nehmen. Wer will, kann dann versuchen den Code zu optimieren, was ihm bei diesem Compiler jedoch schwer fallen wird. Der Compiler enthält den kompletten Sprachumfang, gemäß „Programmieren in MODULA-2" von Prof. Wirth, 3rd Edition Springer-Verlag und „Revisions and Amendments to MODULA-2" in Modula-2 News #0, 1984, MODULA-2 User Association.

Die mitgelieferte Bibliothek entspricht dem „MODULA-2 Standard Library", MODULA-2 News #1, 1985 mit folgenden Moduln:

Files:	für Dateiverarbeitung
Binary:	Binäre Zugriffe auf Dateien
Text:	Text-Dateien
NumberIO:	Zahlen – Lesen und Schreiben
FilePositions:	Random-Dateien
Directory:	Löschen, Umbenennen usw.
SimpleIO:	Standard-Datei-Zugriffe
RealIO:	Ein-/Ausgabe von Real-Zahlen
StandardIO:	Manipulation von Std.-Dateien
Terminal:	direkte Ein-/Ausgabe
MathLib:	Mathematische Funktionen
Program:	Programm-Aufrufe
Storage:	dynamische Speicherverwaltung
String:	Zeichenkettenverarbeitung
Convert:	Zahlkonvertierungen
ConvertLong:	Zahlkonvertierung für LONGINTEGER und LONGCARD
ConvertReal:	Zahlkonvertierung für REAL
BDOS:	Direktzugriff auf BDOS-CP/M-Funktionen

Der MODULA-2-Compiler wird im übrigen für die CPU 68020 (32bit) und den Floating Point Prozessor 68881 weiterentwickelt, so daß er direkt optimierten Code für diese Prozessoren liefert.

MODULA-2-Arbeitsweise:

Der Compiler arbeitet unter CP/M68K und wird mit einem Software-Schutz geliefert. Der Software-Schutz besteht aus einem EPROM-ähnlichen Gehäuse, das eine spezielle Schaltung beinhaltet. Diese Schaltung wird dann einfach als EPROM-Ersatz in einen freien EPROM-Platz im NDR-KLEIN-Computer eingesteckt. Bild 1 zeigt den Einbau auf der RO64-Karte. Der Software-Schutz darf irgendwo liegen, muß jedoch adressmäßig nach dem Grundprogramm liegen.

Die Prozessoren 68000 und 68020 sind wegen der anderen Aufteilung aber bereits berücksichtigt. Es ist somit möglich, von den Disketten Back-Ups zu machen, ohne daß die Disketten speziell geschützt werden müßten, wie das bei anderen Computern getan wird. Durch den Software-Schutz ist es erst möglich geworden, ein kommerzielles Produkt für

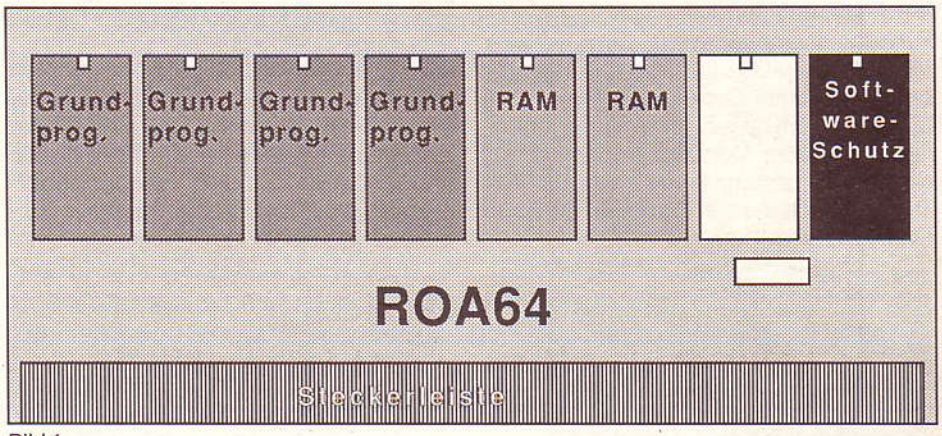


Bild 1

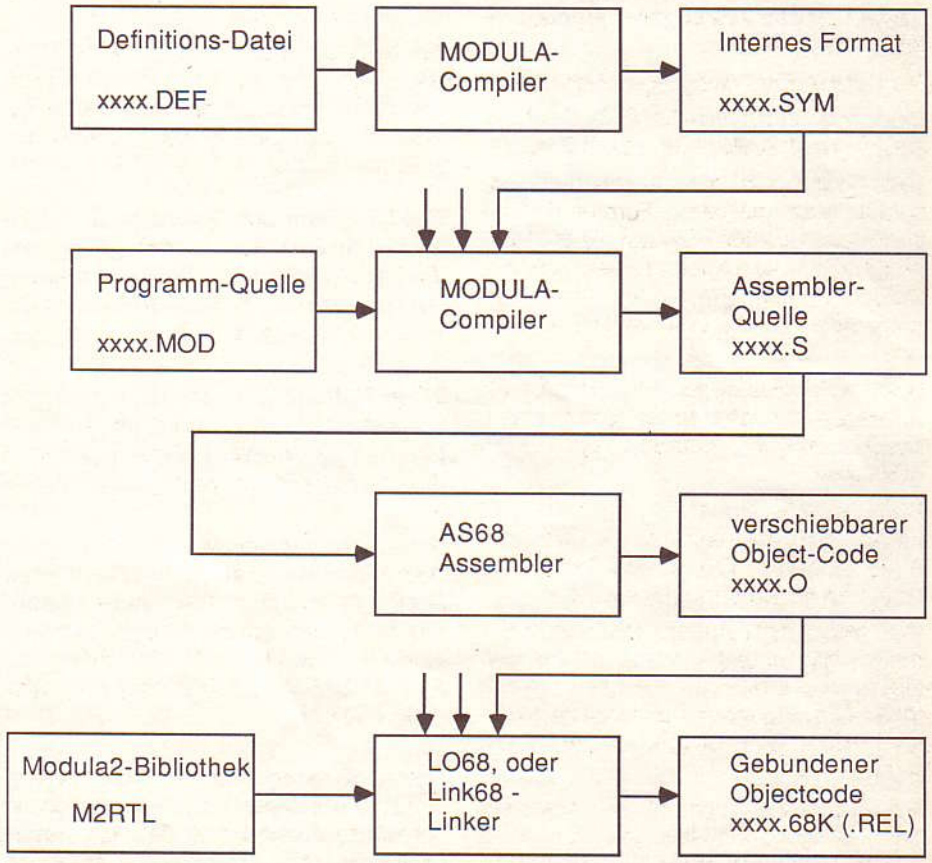


Bild 2 Übersetzungsvorgang

den NDR-KLEIN-Computer zu bekommen, denn nur ausreichend *hohe Stückzahlen erlauben eigene Entwicklungen.*

Programme, die mit dem Compiler erzeugt wurden, benötigen den Software-Schutz natürlich nicht mehr. Das MODULA-2-Projekt ist so auch als Pilot für andere Software-Produkte zu sehen. Hat es Erfolg, werden weitere Produkte folgen können.

Doch nun zur Arbeitsweise des Compilers:

MODULA ist eine Sprache, bei der man, wie schon der Name sagt, Programme aus Modulen zusammensetzen kann. Man unterscheidet drei verschiedene Modularten:

Das DEFINITION MODUL, das IMPLEMENTATION MODUL und das einfache MODUL, das am ehesten dem bekannten Programm entspricht. Im DEFINITION MODUL werden alle nach außen sichtbaren Eigenschaften von Prozeduren und Funktionen festgehalten. Das IMPLEMENTATION MODUL enthält den eigentlichen Programm-Code dazu. Wenn man nun von einem beliebigen anderen Modul aus auf andere Module zurückgreift, so wird die Konsistenz anhand des DEFINITION MODULs vom Compiler automatisch überprüft. Damit ist es möglich, daß der Compiler bereits frühzeitig Fehler meldet, die z.B. bei C für immer unentdeckt bleiben könnten. Auch die Bibliotheken werden über diesen Mechanismus eingebunden. Es lassen sich auch Maschinen- und sogar C-Programme elegant einbinden.

Der MODULA-Compiler unterscheidet selbst zwei Formen von Programmen: Die DEFINITION-Module, aus denen er eine .SYM-Datei erzeugt und alle anderen aus denen er ein Assembler-Quell-Programm erzeugt. Bild 2 zeigt ein Schema.

Der Assembler-Quelltext kann übrigens ganz normal editiert und ausgedruckt werden, falls man darauf zurückgreifen möchte.

Der MODULA-Compiler verwendet beim Übersetzen automatisch die benötigten .SYM-Dateien, um die Konsistenz der eingebauten Module zu überprüfen. So erkennt er z.B. falsche Anzahl von Parametern oder Typen-Konflikte.

Mit dem normalen AS68-Assembler wird dann die .S-Datei übersetzt und die Objekt-Datei .O erzeugt. Diese wird dann zusammen mit allen verwendeten Modulen (ebenfalls .O) und dem Laufzeitsystem (M2RTL) zu dem fertigen Programm gebunden.

Mit dem MODULA-2-Compiler kann man natürlich auch EPROM-Versionen erzeugen, da ja alle Binde-Vorgänge mit den Standard-CP/M-Programmen erfolgen.

Bild 3 zeigt ein Beispielprogramm zur Erzeugung der Mandelbrotmenge. Diese wird dann graphisch auf den Bildschirm ausgegeben, wie schon mal in einer *LOOP* gezeigt. Als Arithmetik wird hier von der 32-Bit-Integer-Arithmetik Gebrauch gemacht, die in MODULA-2 mit LONGINT für vorzeichenbehaftete Werte und mit LONGCARD für vorzeichenlose Werte angegeben wird.

Mit der Zeile FROM RayGdp IMPORT ... wird ein Modul angegeben, dessen Prozeduren dann im Inneren des Programms verwendet werden können.

Der Compiler erzeugt übrigens auf Wunsch auch eine sogenannte Cross-Reference-Liste, die Bild 4 zeigt. Hier sind alle vorkommenden Symbole und deren Auftreten verzeichnet. Die Fehlersuche wird dadurch erheblich erleichtert.

Bild 5 zeigt noch die Statistik des Compilers, die allerdings nur mit vorhandener Uhr-Baugruppe arbeitet. Man erhält Auskunft über genaue Compilations-Zeiten.

Bild 3

```
Modula-2 68K V00.72  Copyright (c) 1986 p1 Gesellschaft fuer Informatik mbH
FRACTEST.MOD                                                        Page   1

     1
     2
     3 (* Fraktal Erzeugung,  Mandelbrot Menge *)
     4 (* Rolf-Dieter Klein, V 1.0, 860325     *)
     5
     6 (* Alle Werte auf 1000 normiert ,also 1.5 ist 1500 *)
     7
     8 MODULE FractalHaupt;
     9
    10 FROM RayGdp IMPORT GdpInit,GdpClear,GdpSetDot;
    11
    12 VAR p,q : LONGINT;
    13     xpkt,ypkt : INTEGER;
    14     color : CARDINAL;
    15
    16 PROCEDURE Fractal(x,y,p,q : LONGINT; VAR color : CARDINAL );
    17  CONST
    18    maxcolor = 101;
    19    maxradius = 3000;
    20    maxradq = 3000 * 3000;
    21  VAR r,x1,y1,x2,y2 : LONGINT;
    22     col : CARDINAL;
    23 BEGIN
    24  col := 0;  (* Startwert , Iteration *)
    25  x1 := x; y1 := y;
    26  REPEAT
    27   x2 := (x1 * x1) DIV 1000 - (y1 * y1) DIV 1000 + p;
    28   y2 := (x1 * y1) DIV 500 + q; (* div 500, da 2/1000 *)
    29   x1 := x2;
    30   y1 := y2;
    31   INC(col);
    32   r := x1 * x1 + y1 * y1;
    33  UNTIL  (col >= maxcolor) OR (r > maxradq );
    34  color := col;
    35 END Fractal;
    36
    37 BEGIN (* Hauptprogramm *)
    38  GdpInit;
    39  GdpClear;
    40  FOR xpkt := 0 TO 511 DO
    41   FOR ypkt := 0 TO 255 DO
    42    p := -2250 + (LONG(xpkt) * ( 750 - (-2250))) DIV 511;
    43    q := -1500 + (LONG(ypkt) * ( 1500 - (-1500))) DIV 255;
    44    Fractal(0,0,p,q,color);
    45    IF (color MOD 2) = 0 THEN
    46     GdpSetDot(xpkt,ypkt);
    47    END;
    48   END;
    49  END;
    50 END FractalHaupt.
    51

NO ERRORS encountered
```

Bild 4

```
Modula-2 68K V00.72  Copyright (c) 1986 p1 Gesellschaft fuer Informatik mbH
FRACTEST.MOD                                                        Page   2

Cross Reference Table


col:            variable   < CARDINAL > in Fractal
    22*   24     31    33    34
color:          variable   < CARDINAL >
    14*   44     45
color:          parameter  (variable)  < CARDINAL > in Fractal
    16*   34

Fractal:        procedure
    16*   44

GdpClear:       procedure  from RayGdp
    10    39
GdpInit:        procedure  from RayGdp
    10    38
GdpSetDot:      procedure  from RayGdp
    10    46

maxcolor:       constant   < UNSIGNED-CONST > in Fractal
    18*   33
maxradius:      constant   < UNSIGNED-CONST > in Fractal
    19*
maxradq:        constant   < UNSIGNED-CONST > in Fractal
    20*   33

p:              variable   < LONGINT >
    12*   42     44
p:              parameter  (value)     < LONGINT > in Fractal
    16*   27

q:              variable   < LONGINT >
    12*   43     44
q:              parameter  (value)     < LONGINT > in Fractal
    16*   28

r:              variable   < LONGINT > in Fractal
    21*   32     33
RayGdp:         module, imported
    10

x:              parameter  (value)     < LONGINT > in Fractal
    16*   25
x1:             variable   < LONGINT > in Fractal
    21*   25     27    28    29    32
x2:             variable   < LONGINT > in Fractal
    21*   27     29
xpkt:           variable   < INTEGER >
    13*   40     42    46

y:              parameter  (value)     < LONGINT > in Fractal
    16*   25
y1:             variable   < LONGINT > in Fractal
    21*   25     27    28    30    32
y2:             variable   < LONGINT > in Fractal
    21*   28     30
ypkt:           variable   < INTEGER >
    13*   41     43    46
```

```
Modula-2 68K V00.72  Copyright (c) 1986 p1 Gesellschaft fuer Informatik mbH
FRACTEST.MOD                                                        Page   3


Compilation Summary

Active Options at End of Compilation:
   /CHECK    /NODEBUG
   /LISTING=FRACTEST.LST    /CROSSREFERENCE    /NOMACHINECODE
   /OBJECTFILE=FRACTEST.S
   /NOSYMBOLFILE

Performance Indicators:          CPU Time
   Syntax analysis             0: 0: 0. 0
   Declaration analysis        0: 0: 0. 0
   Body analysis               0: 0: 0. 0
   Code generation             0: 0: 0. 0
   Listing generation          0: 0: 0. 0

Total run time:   0: 0: 0. 0   (   0 lines/minute)
```

Bild 5

```
(* GDP Interface  860305 *)

%FOREIGN DEFINITION MODULE RayGdp;

EXPORT QUALIFIED GdpInit,GdpClear,GdpSetDot;

PROCEDURE GdpInit;

PROCEDURE GdpClear;

PROCEDURE GdpSetDot(x,y : INTEGER);

END  RayGdp.
```

Bild 6

```
*
* Assemblerteil  RayGdp
* fuer Modula
*
 .globl GdpInit,GdpClear,GdpSetDot;

GdpInit:
 link a6,#0
 movem.l d0-d7/a0-a6,-(a7)
 move #23,d0
 trap #3
 move.l a4,gruprog
 movem.l (a7)+,d0-d7/a0-a6
 unlk a6
 rts

GdpClear:
 link a6,#0
 movem.l d0-d7/a0-a6,-(a7)
 move #20,d7   * CLRSCREEN
 movea.l gruprog,a0
 jsr $420(a0)
 clr d0
 clr d1
 move #34,d7   * SETFLIP
 movea.l gruprog,a0
 jsr $420(a0)
 clr d0
 clr d1
 move #27,d7   * NEWPAGE
 movea.l gruprog,a0
 jsr $420(a0)
 movem.l (a7)+,d0-d7/a0-a6
 unlk a6
 rts

GdpSetDot:     * x,y
 link a6,#0
 movem.l d0-d7/a0-a6,-(a7)
 move 10(a6),d1 * x
 move 8(a6),d2  * y
 movea.l gruprog,a0
 move #8,d7     * MOVETO
 jsr $420(a0)
 move #$80,d0   * Dot
 movea.l gruprog,a0
 move #26,d7    * CMD
 jsr $420(a0)
 movem.l (a7)+,d0-d7/a0-a6
 unlk a6
 move.l (a7)+,(a7)
 rts

gruprog: dc.l 0

 end
```

Bild 7

Bild 6 schließlich zeigt das DEFINITION MODUL für die verwendeten Unterprogramme. Der Name %FOREIGN besagt, daß es sich hier um die Einbindung eines Programms handelt, das in einer anderen Sprache, z.B. Assembler geschrieben ist. Da sich die Parameterversorgung des MODULA-2-Compilers von der des C-Compilers unterscheidet, gibt es die Option %C mit der man auch in C geschriebene Prozeduren direkt einbinden kann.

Bild 7 zeigt das dazugehörige Assembler-Programm.

Eine weitere interessante Eigenschaft liegt im MODULA-2-Compiler. Wenn man z.B. ein DEFINITION-MODUL ändert, so muß man auch das dazugehörige IMPLEMENTATION MODUL neu übersetzen, sonst erhält man eine Fehlermeldung beim Linken (erst dort merkt der Linker anhand des Objekt-Codes den Fehler, der in einem fehlenden Namen besteht). Ein spezieller Mechanismus sorgt für die Erkennung. Damit wird verhindert, daß man schnell mal das DEFINITON MODUL ändert und vergißt, auch die Prozedure im IMPLEMENTATION MODUL zu ändern.

Bei %FOREIGN Moduln entfällt diese Prüfung, da man bei C und Assemblerprogrammen keine solche Prüfung automatisieren kann.

Zum Schluß des ersten Teils noch ein Ausblick:

In Vorbereitung befindet sich ein Mausgesteuerter Editor, mit dem man dann mit Windows und Scroll-Bars editieren kann, wie man es nur von GEM und MacIntosh her gewohnt ist. Dieser Editor wird später automatisch zum MODULA-2 mitgeliefert. Solange kostet der Compiler 498,– DM, später dann 100,– DM mehr. Der Software-Schutz ist natürlich im Preis inbegriffen. Alle Käufer können später zu einem geringen Unkostenbeitrag Upgrades des Compilers (und auch den Editor) beziehen. Derzeit ist die Real-Arithmetik noch nicht enthalten, sie wird aber, sobald vorhanden, kostenlos nachgeliefert. Den Compiler kann man sofort bestellen.

Der Compiler ist ab Lager lieferbar bei:
GES Postfach 1610, 8960 Kempten
Tel.: (08 31) 62 11
oder
p1 Gesellschaft für Informatik mbH
Gotthardstraße 99, 8000 München 21
Tel.: (0 89) 5 80 60 99

MODULA-2-Compiler mit Handbuch.
Preis: 498,– DM (5¼-Zoll-Diskette NDR-80-Format).

Empfohlene Literatur: siehe oben.

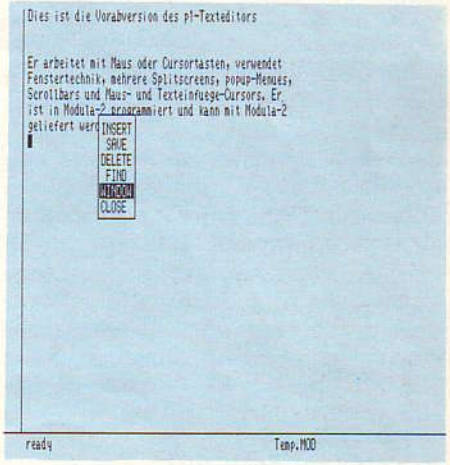


Bild 8

RL-BASIC
Endlich ein BASIC für die 680xx-Benutzer

Obwohl wir die Sprache BASIC nicht besonders bevorzugen, steht nun ein leistungsfähiger BASIC-Interpreter für den NDR-KLEIN-Computer zur Verfügung. Dieser BASIC-Interpreter kann alles, was man sich von einem BASIC erträumen kann.

Ein eigens für den Interpreter entworfener Bildschirm orientierter Editor, wie man ihn nur vom C64 her kennt, steht zur Verfügung. Man kann also den Cursor über den ganzen Bildschirm des NDR-Computers positionieren, Zeichen einfügen, löschen usw.

Ferner ist ein eigenes DOS (Betriebssystem für Disketten) im Lieferumfang enthalten, so daß man auch Programme speichern und laden kann. Das DOS arbeitet aber unabhängig vom BASIC; man kann es auch zum Speichern von 680xx Programmen und Daten verwenden.

Das BASIC enthält neben den Standard-BASIC-Befehlen Graphik-Befehle, mit denen man die GDP direkt und komfortabel ansprechen kann. Ferner ist natürlich auch die Gleitkommaarithmetik integriert.

Bild 1 zeigt die Startmeldung für das DOS, Bild 2 die Meldung des Interpreters (er kann den ganzen Adressraum ausnutzen).

Bild 3 zeigt ein kleines Beispielprogramm und Bild 4 und 5 die Ausgabe, die übrigens nur 8 Sekunden benötigt.

Bild 1

```
-------------------------
Disketten-Betriebssystem
fuer NDR-Klein-Computer
--  Vers. 1.1  5/85  --
--  (c) R. Lobreyer  --
-------------------------

A>dir
BASIC - DISKETTE

 Name              Art   Status   Ladeadr.   Sektoren
 -------------------+-----+--------+----------+--------
 RL-BASIC          BIN   R/W      $010000    24K
 RL-BASIC 68000    BIN   R/W      $010000    24K
 GRAPHIK-DEMO1     BAS   R/W      $004400    1K
 GRAPHIK-DEMO2     BAS   R/W      $004400    1K
 LINES             BAS   R/W      $004400    1K
 BUBBLESORT        BAS   R/W      $004400    1K
 733  Sektoren frei
 67   Sektoren belegt
A>
```

Bild 2

```
************************
*      RL-Basic 1.0    *
* 68008/68000 Version fuer *
*   NDR-Klein Computer  *
* (c) R. Lobreyer 6/1985 *
************************
125952 Bytes free  <------ !

>files
BASIC - DISKETTE

 Name              Art   Status   Ladeadr.   Sektoren
 -------------------+-----+--------+----------+--------
 RL-BASIC          BIN   R/W      $010000    24K
 RL-BASIC 68000    BIN   R/W      $010000    24K
 GRAPHIK-DEMO1     BAS   R/W      $004400    1K
 GRAPHIK-DEMO2     BAS   R/W      $004400    1K
 LINES             BAS   R/W      $004400    1K
 BUBBLESORT        BAS   R/W      $004400    1K
 733  Sektoren frei
 67   Sektoren belegt

>
```

Bild 3

```
>list
 0   REM  GRAPHIK - DEMO 3
 1   REM  KREISE UND LINIEN
 2   REM
 3   REM  (C) R.LOBREYER 1985
 4   REM
 5   REM  DURCH JEDE TASTE ABBRECHBAR
 6   REM
 10   PAGE 3,3: CLPG
 20   FOR I = 50 TO 450 STEP 50
 30   FOR J = 450 TO 50 STEP  - 50
 40   CIRCLE (I,J),50: LINE (I,J) - (J,I)
 50   NEXT : NEXT
 70   CLPG
 80  A$ =  INKEY$ : IF A$ = "" THEN 90 ELSE  END
 90   FOR J = 1 TO 0 STEP  - 1
 95   COLOR  = J
 100   FOR I = 30 TO 180 STEP 15
 110   CIRCLE (25 + I,255),I: CIRCLE (480 - I,255),I
 120   NEXT
 150   NEXT
 170  A$ =  INKEY$ : IF A$ = "" THEN 10 ELSE  END
```

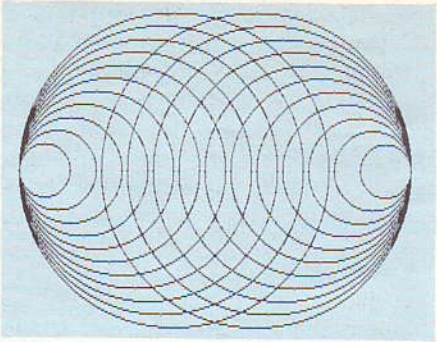


Bild 4

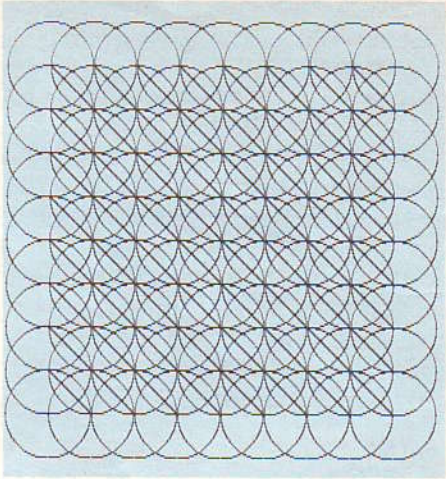


Bild 5

Der BASIC-Interpreter (mit DOS und Handbuch) kostet nur 159,– DM und ist auf Diskette (mit Beispielprogrammen und DOS) und EPROM ab Lager lieferbar.

Kurzbeschreibung:

65c02 Emulator für NDR-Computer mit CPU 68008

1. Was ist ein Emulator?

Ein Emulator ist ein Programm, das ein Stück Hardware (hier ein Prozessor) durch Software simuliert. Bei dem 65c02 Emulator wird die von Rechnern wie C64 oder Apple IC bekannte CPU 6502 emuliert (d.h. durch Software nachgebildet). Das „c" im 65c02 belegt einen gegenüber dem 6502 stark erweiterten Befehlssatz. Emulatoren sind in der Regel wesentlich langsamer als die Original-Hardware (hier ca. um den Faktor 10).

2. Warum den 8-Bit-Prozessor 6502 emulieren?

(i) Der 6502 wird heute noch in vielen kleineren Rechnern, unter anderem auch für Steuerungszwecke, eingesetzt (EMUF...).

Wie bei Großrechnern üblich, kann der Emulator zur Entwicklung und zum Austesten von Programmen für solche Rechner verwendet werden.

(ii) Des weiteren kann der Lerneffekt genannt werden: Für den NDR-KLEIN-Computer existieren bereits zwei völlig unterschiedliche CPU-Karten, die eine mit dem 8-Bit-Prozessor Z80, die andere mit dem moderneren 8-(32)-Bit-Prozessor 68008. Ein Emulator für die 65c02 CPU ermöglicht es nun, einen weiteren Prozessor kennenzulernen, ohne eine weitere CPU-Karte + Grundprogramm anschaffen zu müssen.

3. Was leistet der 65c02-Emulator?

Das Programmpaket besteht eigentlich aus 3 voneinander unabhängigen Programmteilen, nämlich

(i) a) dem eigentlichen Emulator für die Abarbeitung von 65c02 Maschinenprogrammen,

b) einem Debugger („Entwanzer"), um Programme besser austesten zu können. – Einzelschritt aus dem Grundprogramm vergleichbar –,

(ii) einem 2-Paß-Assembler für die 65c02 Assemblersprache, mit dessen Hilfe Programme im Texteditor entwickelt und dann übersetzt werden können,

(iii) einem Disassembler, der 65c02 Maschinenprogramme in verständliche Assemblerbefehle zurückübersetzt. Die Geschwindigkeit des Emulators liegt, mit 68008 CPU, bei 0,1 Megahertz, was in Anbetracht der guten Testmögichkeit aber durchaus akzeptabel ist. Eine Möglichkeit zum Aufruf von Grundprogrammroutinen ist enthalten.

Der Emulator ist ab Lager lieferbar und kostet DM 120,–.

Neu: JADOS 68008-Betriebssystem

Klaus Janßen, Krefeld

Wer *ernsthaft* mit Disketten arbeiten will (Programmentwicklung, Texterstellung), war bisher auf CP/M 68K angewiesen. Dieses sehr professionelle Betriebssystem ist leider in der Anschaffung recht kostspielig und erfordert weitere Investitionen (viel RAM, zweites Laufwerk). Die Bedienung ist insbesondere für den Neuling ziemlich kompliziert; und auch der Fortgeschrittene klagt zuweilen über Umständlichkeiten.

Mit *JADOS* gibt es nun eine preiswerte Alternative, mit der man ernsthaft arbeiten kann. Nur geringfügig teurer als Jogi-Dos bietet *JADOS* folgende *Leistungsmerkmale:*

- optimale Verwaltung der Disketten – und der Hauptspeicherkapazität

Leistungsfähige Kommandos:

- Laden und Speichern von Dateien (d.h. Texten und Programmen)

- Löschen von Dateien, wobei freigewordene Bereiche neu belegt werden können
- Ändern von Dateinamen
- Koppeln von Dateien
- Erstellen und Ändern von Textdateien (mit R.-D.-Klein-Editor)
- Volle Unterstützung des Assemblers von R.-D. Klein
- Assemblierte Programme werden als Programmdateien abgespeichert
- Laden von Programmdateien mit Autostart (auch bei Programmen mit Bibliotheksvorspann)
- Unkomplizierte Benutzerführung durch Kurzkommandos, Menüs, umfangreiche Erfolgs- und Fehlermeldungen, ständiges Anzeigen einer Hilfezeile)
- *JADOS* läuft mit und ohne Bankbootkarte
- 26 leistungsfähige Unterprogramme für den Programmierer (über TRAP aufrufbar) zur Dateiverwaltung, Stringverarbeitung, Bildschirmwiedergabe u.a.m.
- Mehrere Hilfsprogramme (u.a. komfortables Druckprogramm, Bibliotheksfunktion, Kopierprogramm, auf Wunsch auch mit Universalformatierprogramm)
- ausführliches, sehr gut verständliches Handbuch (für Einsteiger und für Profis)

Zum Betrieb von *JADOS* unbedingt erforderlich sind:
- CPU 68008
- Grafikcontroller GDP64
- Diskettencontroller FL02 oder FDC (letzteres Elektronikladen Detmold)
- 1 Minidiskettenlaufwerk, doppelseitig, 80 Spuren pro Seite, 5¼" oder 3½"

Ferner:
a) ohne Bankbootkarte:
- Grundprogramm 4.3, Adresse 0
- mindestens 40 KB RAM (besser 64)

b) mit Bankbootkarte:
- Grundprogramm 4.3, Adresse $E0000
- 8KB RAM ab Adresse $E8000
- mindestens 40 KB RAM ab Adresse 0 (besser 64)

JADOS kann mit einem oder zwei Diskettenlaufwerken (5¼" und 3½") betrieben werden und ist in zwei Varianten (mit und ohne Formatierprogramm) *ab sofort erhältlich.* Preis: DM 140,–.

TIPS + TRICKS – TIPS + TRICKS

BASIC auf dem NDR-Computer

Teil 2

von Carsten Denneburg

Wenn Sie nach wie vor zu den Anfängern gehören (ich zähle mich auch nach zwei Jahren noch dazu), dann könnte Ihnen dieser Artikel die eine oder andere Anregung bieten. Wenn nicht, schadet die Lektüre auch nicht!

Als Neuling oder jemand, der sich einfach aus Spaß an der Materie mit der Computerei beschäftigt, steht man bei einigem Nachdenken genau zwischen zwei Positionen:

Auf der einen Seite sammeln sich die Listing-Abtipper oder Spiele-Freaks (Freak heißt übrigens Monster) und auf der anderen die Turbo-Pascal ,C, strukturierten-Festplatten-Programmierer. Wir wenden uns also mit Graus nach links und rechts und wissen nicht so recht, wohin die Reise gehen soll.

BASIC scheint leicht erlernbar und zunächst für die eigenen Bedürfnisse ausreichend.

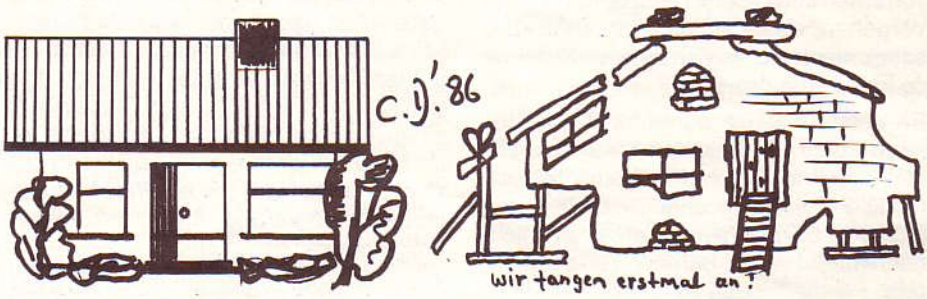
– Aber ist BASIC nicht diese Rucksackprogrammier-Sprache, die strukturierte Programme nicht zuläßt? – Und überhaupt . . . Vor lauter Ernsthaftigkeit kann einem der Spaß an der Materie vergehen.

Bleiben wir doch einfach auf dem Teppich und überlegen. Wir wollen uns mit der Hard- und Software eines Computers, seinem Innenleben vertraut machen. Wir wollen Spaß an der Beschäftigung haben und vielleicht auch unsere beruflichen Chancen verbessern. Welcher Sprache wir uns dabei „hingeben", ist letztlich völlig uninteressant!

Entscheidend ist – wie immer im Leben – etwas ganz anderes: Wir müssen planvoll, zielgerichtet vorgehen, um in die Materie einzudringen. Wenn wir uns nach diesem Motto mit BASIC beschäftigen, bleibt uns der Weg in komplexere Sprachen nicht verschlossen und wir werden auch nicht – wie oft behauptet wird – für das Programmieren verdorben.

Also der Reihe nach:

Sehen Sie sich die beiden Häuser an (und haben Sie keine Angst, wir sind hier nicht in einer Bauzeitschrift!). In welchem Haus würden Sie lieber wohnen wollen?



Ich unterstelle im linken.

Genau wie das linke Haus planvoll und zielgerichtet entworfen und gebaut worden ist, können auch Sie Ihre Programme in BASIC entwerfen und umsetzen.

Es würde sicherlich niemand auf die Idee kommen, beim konventionellen Hausbau mit dem Dach oder der Regenrinne anzufangen oder ersteinmal ein paar Steine zu kaufen, um dann irgendwo zu beginnen. Genau wie beim Hausbau sind in der EDV viele Leute der verschiedensten Aufgabenbereiche arbeitsteilig an der Entwicklung eines Projekts beteiligt. Jeder ist für seinen Bereich verantwortlich. Es gibt Organisatoren, die Betriebsuntersuchungen veranlassen, betriebliche Abläufe festhalten und versuchen, für den jeweiligen Anwendungsfall die optimale EDV-Lösung zu entwerfen. Ihre Planung des zukünftigen EDV-Projekts muß umfassend und vorausschauend alle denkbaren Anwendungsfälle berücksichtigen. Nach Übergabe aller so entwickelten Vorgaben ist es Aufgabe der Programmierung, die entwickelten Gedanken in ein ablauffähiges Programm umzusetzen.

Sie und ich – wir – sind nun in der beneidenswerten Lage, alle diese Teilbereiche in einer Person vereinen zu können. Die oben skizzierte Arbeitsteilung sollte man für sich dennoch beibehalten und sich stets bewußt machen, in welchem Arbeitsschritt man sich zur Zeit befindet.

Denn:

Einen auf Dauer (!) guten Programmentwickler erkennt man daran, daß er die

meiste Zeit der Entwicklung nicht vor dem Computer, sondern vor einem Blatt Papier mit einem spitzen Bleistift verbringt. Erst wenn alle zum geplanten Projekt notwendigen Schritte gedanklich durchgespielt und zu Papier gebracht worden sind, beginnt das eigentliche Programmieren, in diesem Sinne also nur noch das Kodieren bereits fix und fertig formulierter Gedanken.

In diesem Sinne heben wir uns also wohltuend von den nur-Abtippern ab. Bitte nicht falsch verstehen: Es ist unbestritten, daß es ziemlich unsinnig ist, bereits existierende Programme völlig neu zu erfinden. Wenn Sie also für einen bestimmten Zweck ein passendes Listing haben, dann bitte auch nutzen. Falls das fertige Programm nicht in allen Punkten Ihren Ansprüchen genügt, dann versuchen Sie das Programm zu analysieren, in seine Teile zu zerlegen und ergänzt um Ihre Wünsche in eine vernünftige Form zu bringen. Wenn Sie merken, daß es doch nur eine Stückelei wird, aufhören! Glauben Sie mir, Sie verschwenden kostbare Arbeitszeit in ein Projekt, das nie (nie!) Ihre Ansprüche erfüllen, und immer Stückwerk bleiben wird. Wenn dieser Fall eintritt, werfen Sie das Listing fort, befreien Sie sich völlig von diesem Programm und entwerfen Sie Ihr eigenes.

Sie liegen im Freibad (zur Zeit ziemlich unwahrscheinlich), in der Badewanne, im Bett oder sonstwo und Ihnen geht so einiges durch den Kopf. Man könnte doch einmal . . .

Nun gibt es zwei Möglichkeiten:

1. Sie setzen sich sofort an den Computer und legen so richtig los. Ich bin schwer von Ihnen enttäuscht. Denn genau dieses Vorgehen, das zweifellos durch BASIC sehr unterstützt wird, hat diese Sprache so in Verruf gebracht.

Es entsteht ein Programm, durch das nach kurzer Zeit niemand mehr durchsteigt – und Sie wahrscheinlich auch nicht. Es wimmelt unausweichlich von GOTO's, GOSUB's und anderen, nicht nachvollziehbaren Schleifen und Schlingen und die Freude ist, zumal bei Änderungen (und die kommen mit Sicherheit früher oder später) nur von kurzer Dauer.

Also

2. Sie bleiben mit Ihren tollen Gedanken ganz ruhig liegen und spinnen weiter, aber mit System. Lösen Sie sich von dem Gedanken, Ihre Idee mit dem NDR-Computer umsetzen zu wollen und planen Sie mit Papier und Bleistift alle Schritte mit System. Ich gebe zu, das ist mühevoll und manchmal nachgerade langweilig. Aber, der Weg zu einem guten Programm ist nun einmal steinig und mühevoll. Umso größer ist dann die Freude, wenn das Programm auch wirklich Ihren Ansprüchen genügt und Ihnen auf Dauer (!) Freude macht.

Die erste Phase (Sie erinnern sich Freibad, Badewanne pp.) möchte ich als die des geordneten Chaos bezeichnen. Alles was Ihnen jetzt einfällt, es mag noch so unwichtig sein, schnell in Stichworten notieren.

Gehen Sie in folgender Reihenfolge vor:
- was soll dargestellt werden (Heizkosten, Kontostand, Grafik pp.)
- gibt es für verschiedene Anwendungen eine gemeinsame Darstellung?
- wie soll es dargestellt werden? (Koordinatensystem, welcher Bereich (+/–), Einteilung der Achsen, Beschriftung des Systems)
- welche Daten sollen erfaßt werden, welches Datenvolumen ist zu erwarten, welche physikalische Einheit wird benutzt, Umrechnungsfaktoren . . .?
- welche Sonderfälle können auftreten, welche Bedienfehler sind denkbar, welcher Bedienkomfort soll geboten werden?
- wird der Speicherplatz reichen (SBCII), müssen ggf. Abstriche gemacht werden, wenn ja, wo?
- wie soll die Eingabe, wie die Ausgabe aussehen?
- . . . und so weiter, und so weiter . . .

Versuchen Sie nun, die skizzierten Gedanken in logische Blöcke zusammenzufassen. Beschreiben Sie (im wahrsten Sinne des Wortes die geplante Anwendung in schriftlicher Form. Also, das Programm soll . . . dann soll das und das passieren, dann soll . . .

Verfeinern Sie die Beschreibung weiter und beginnen Sie zu malen. Das heißt, nutzen Sie zeichnerische Darstellungen wie z.B. Flußdiagramme. Auch wenn Sie mit der Technik nicht vertraut sind, fangen Sie mit Diagrammen an (es genügen einige wenige Symbole). Auf diese Art und Weise werden Ihre Gedanken noch (!) geordneter und Sie erkennen problematische Bereiche Ihres zukünftigen Programmes.

An dieser Stelle kann ich Ihnen als allgemeine Übung nur empfehlen, sich Abläufe des täglichen Lebens vorzustellen und zu versuchen, diese in Flußdiagrammen oder Struktogrammen darzustellen. Sie werden sehr schnell feststellen, daß es dem Menschen ungeheuer schwer fällt, einen Lebenssachverhalt auf Computerniveau herunter zu zeichnen. Fast jede dargestellte Aktivität oder Entscheidung läßt sich in der Regel weiter verfeinern. Erst wenn wirklich nur noch ja/nein übrigbleibt, haben Sie das Niveau Ihres Computers erreicht. Sie wissen ja: Dumm wie ein Schraubenzieher ...

Nun weiter zu Ihrem Programm. Schaffen Sie sich logische Bereiche, die Sie auf getrennten Blättern festhalten. Ermitteln Sie im Programm Bereiche, in denen immer wieder die gleichen Abläufe stattfinden oder stattfinden könnten.

Ordnen Sie Ihre Aufzeichnungen wie folgt:

Erfassen der Daten = Eingabe

soll das Programm nur aktuelle Daten verarbeiten und sie dann vergesen? = input

soll das Programm aktuelle Daten erfassen und alte Daten mit darstellen können?
= Data Zeilen

soll das Programm alte Daten darstellen und nur um den jeweiligen aktuellen Stand ergänzt werden? = Da Zeilen + Input

Datenmenge = was fällt an
wenn die Datenmenge nicht vorhersehbar ist, Routinen vorsehen, die automatisch Speicherplatz zur Verfügung stellen. Nutzen Sie die Möglichkeiten der Indizierung, da Ihre Programme dann praktisch unbegrenzt offen sind. Nutzen Sie die Befehle Clear und Dim.
Berechnungsteil = was soll mit den Daten geschehen?

Sind die Werte zu berechnen oder umzurechnen?

Sind Werte zu sortieren oder zu vergleichen?

Wann und wo geschieht dies? Derartige Programmteile immer als Unterprogramme formulieren.

Ausgabe

Wie und an welcher Stelle, mit welchen Meldungen sollen Daten ausgegeben werden? Sind Kommentare notwendig?

Sind die Daten in Form einer Tabelle aufzubereiten. Könnte die Tabelle einen Bildschirm sprengen. Ist die Ausgabe anzuhalten?

Ist ein Grafikteil geplant? Diesen im Programm immer (!) an das Ende setzen.

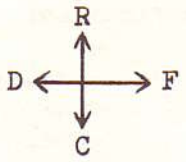
Aufbau einer Grundgrafik (z.B. eines Koordinatenkreuzes) in Form eines Unterprogramms, auf das (alle!) Ausgaben zurückgreifen können. Die Unterteilung und Beschriftung des Koordinatenkreuzes sowie die Unterteilung variabel gestalten, um für alle auftretenden Fälle gerüstet zu sein.

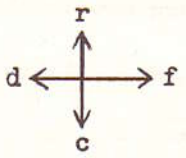
Legen Sie die Art der Darstellung fest (z.B. Kurvendiagramm, Balkendiagramm, Torte pp.)

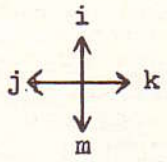
Erst wenn Sie sich über alle geschilderten Fragen im Klaren sind, können Sie damit beginnen, das Programm in BASIC oder einer anderen Sprache zu formulieren. Da Sie Ihr Programm logisch durchdacht und geordnet haben, ergibt sich die berühmte Struktur fast wie von selbst.

Mini-CAD-Programm geschrieben für/auf SBC2 in BASIC 1.5

Günter Hauke








```
Eingabe : N = Nochmal ,    E = Ende ,    L = LIST

1 CLRS
2 OUT 115,16*2+1
3 PAGE 0,0
4 POKE HEX ("87C5"),0
5 MOVETO 0,0
6 A$ = GET$
7 OUT 112,0
8 DRAWTO X,Y
10 IF A$="R" THEN 50
11 IF A$="C" THEN 55
12 IF A$="F" THEN 60
13 IF A$="D" THEN 65
14 IF A$="r" THEN 71
15 IF A$="c" THEN 76
16 IF A$="f" THEN 80
17 IF A$="d" THEN 85
18 IF A$="i" THEN 100
19 IF A$="m" THEN 150
20 IF A$="k" THEN 200
21 IF A$="j" THEN 250
39 IF A$="L" THEN CLRS
40 IF A$="L" THEN LIST
41 IF A$="N" THEN RUN
44 IF A$="E" THEN CLRS
45 IF A$="E" THEN END
49 GOTO 6
50 Y=Y+5 : DRAWTO X,Y
51 X=X+1 : DRAWTO X,Y
52 Y=Y-5 : DRAWTO X,Y
53 X=X-1 : Y=Y+5
54 DRAWTO X,Y : GOTO 6
55 Y=Y-5 : DRAWTO X,Y
56 X=X+1 : DRAWTO X,Y
57 Y=Y+5 : DRAWTO X,Y
58 X=X-1 : Y=Y-5
59 DRAWTO X,Y : GOTO 6
60 X=X+10 : GOTO 6
65 X=X-10 : GOTO 6
71 Y=Y+1 : X=X+1
72 DRAWTO X,Y : X=X-1
73 DRAWTO X,Y : GOTO 6
76 Y=Y-1 : X=X+1
77 DRAWTO X,Y : X=X-1
78 DRAWTO X,Y : GOTO 6
80 X=X+2 : GOTO 6
85 X=X-2 : GOTO 6
100 Y=Y+1 : DRAWTO X,Y
102 X=X+1 : DRAWTO X,Y
104 Y=Y-1 : MOVETO X,Y
106 Y=Y-1 : OUT 112,1
108 DRAWTO X,Y : X=X-2
110 DRAWTO X,Y : Y=Y+1
111 DRAWTO X,Y : X=X+1
112 DRAWTO X,Y : Y=Y+1
114 MOVETO X,Y : GOTO 6
150 Y=Y-1 : DRAWTO X,Y
152 X=X+1 : DRAWTO X,Y
154 Y=Y+1 : DRAWTO X,Y
156 Y=Y+1 : OUT 112,1
158 DRAWTO X,Y : X=X-2
160 DRAWTO X,Y : Y=Y-1
161 DRAWTO X,Y : X=X+1
162 DRAWTO X,Y : Y=Y-1
164 MOVETO X,Y : GOTO 6
200 X=X+2 : DRAWTO X,Y
202 X=X-4 : MOVETO X,Y
204 X=X+2 : OUT 112,1
206 DRAWTO X,Y : X=X+2
208 MOVETO X,Y : GOTO 6
250 X=X-2 : DRAWTO X,Y
252 X=X+4 : MOVETO X,Y
254 X=X-2 : OUT 112,1
256 DRAWTO X,Y : X=X-2
258 MOVETO X,Y : GOTO 6
300 STOP
```

Computer-Graphik in BASIC

von Roland Handl

Als Besitzer des NDR-KLEIN-Computers mit der Z80-Vollausbau-CPU habe ich ein Programm in BASIC geschrieben, das schöne Graphiken auf den Bildschirm projeziert. Diese sind aufgebaut aus Kreis, Ellipsen und Geraden, die alle 4° miteinander zu Dreiecken verbunden werden. Da das Programm nur ca. 1,3 KByte benötigt, läuft es auch auf der SBC2-Platine.

Doch nun zum Programmablauf. Wenn man den Computer einschaltet, im BASIC „Stift C" drückt und dann PRINT RND(1) eingibt, erscheint jedesmal die gleiche Zahl. Das bedeutet, daß das BASIC nach dem Einschalten die Zufallszahlen immer gleich initialisiert. Deshalb habe ich, damit nicht jedesmal dieselben Graphiken erzeugt werden, die Zeilen 40 und 50 in das Programm eingebaut, in der Hoffnung, daß der Mensch zufälliger denkt als der Computer.

In den nächsten beiden Zeilen legt man fest, ob der Computer hell auf dunklem (normal) oder dunkel auf hellem Hintergrund zeichnet. Wer sich nicht festlegen möchte, kann die Zeile 145 F = RV : RV = NR : NR = F eingeben. Sie bewirkt, daß der Computer abwechselnd hell und dunkel zeichnet. Dann gibt man in Zeile 60 nur noch die erste Kombination ein.

Der Bildschirm wird in Zeile 80 bereit gemacht und in den Zeilen 90 bis 110 schreibt der GDP in 5facher Breite und 3facher Höhe (Zeile 90 : HEX ("53")) die Überschrift auf den Bildschirm.

Wenn der Computer auf hellem Untergrund zeichnen möchte, muß er ihn erst hell beschreiben. Dies wird in Zeile 120 veranlaßt.

Aus 12 verschiedenen Möglichkeiten wird in Zeile 130 je eine für die Variablen A(1)–A(6) bestimmt. Sie entsprechen den 12 verschiedenen Funktionsvorschriften aus den Zeilen 300 – 410. Das ergibt insgesamt 2985984 (knapp 3 Millionen!) theoretisch verschiedene Variationsmöglichkeiten. Einige davon sind allerdings nur gespiegelte, um 90° verdrehte oder auf den Kopf gestellte Versionen einer anderen. Trotzdem werden Sie auch nach stundenlangem Zuschauen beim Zeichnen der Graphiken keine gleichen, sondern nur ähnliche Versionen wiederfinden.

Wer die Anzahl der verschiedenen Möglichkeiten erhöhen möchte, muß ab Zeile 410 neue Funktionsvorschriften finden (dabei muß B(A) immer zwischen –1 und 1 liegen!), in Zeile 130 die 12 durch die Anzahl der Vorschriften ersetzen und in Zeile 160 die neuen Zeilennummern anhängen.

In Zeile 140 wird der Schreib- (hell) bzw. Löschstift (dunkel) angesetzt.

Die 4°-Abschnitte werden in Zeile 150 festgelegt (STEP 4). Verändert man die 4 zu einer größeren Zahl, wird das Gebilde karger, verändert man sie zu einer kleineren, wird die Figur dichter und man kann die Linien weniger gut voneinander unterscheiden.

Durch die Zeile 160 wird der jeweilige Funktionswert bei einem bestimmten Winkel berechnet. Diese werden dann auf 256 x 512 Punkte vergrößert und miteinander verbunden (Zeilen 190 bis 210). Nach 270 Linien (bei 4°-Schritten) ist das Bild fertiggezeichnet.

Dieses Bild kann man sich während der Zeitdauer, die die Warteschleife in Zeile 220 benötigt, ansehen.

Dann veranlaßt Zeile 230, daß eine neue Graphik gezeichnet wird.

Je länger man die Warteschleife in Zeile 220 macht, desto länger kann man die einzelnen Bilder betrachten. Verkürzt man die Schleife, kommt man in den Genuß, mehr verschiedene Graphiken zu betrachten.

Ich wünsche Ihnen viel Spaß beim Experimentieren am Programm und beim Betrachten der Graphiken.

```
1Ø REM Computer-Graphik
2Ø REM von Roland Handl
3Ø REM fuer 8k-BASIC der SBC 2 bzw. CPU Z8Ø
4Ø INPUT"Bitte eine Zahl zum Initialisieren des Zufallsgenerators eingben";Z
5Ø FOR ZZ=1 TO ABS(Z):I=RND(1):NEXT ZZ
6Ø INPUT"Hell oder dunkel zeichnen (1/2) ";A:IF A<>1 AND A<>2 THEN 6Ø
7Ø RV=Ø:NR=1:IF A=2 THEN RV=1:NR=Ø
8Ø CLRS:POKE HEX("87C5"),Ø:PAGE Ø,Ø:GDP=HEX("7Ø"):PI=3.141593
9Ø OUT GDP+3,HEX("53"):MOVETO 2Ø,23Ø
1ØØ FOR L=1 TO 16:OUTGDP,ASC(MIDØ("Computer-Graphik",L,1)):NEXT L
11Ø OUT GDP+3,HEX("11")
12Ø IF RV=1 THEN 23Ø
13Ø FOR A=1 TO 6:A(A)=INT(RND(1)*12+1):NEXT A
14Ø OUT GDP,RV
15Ø FOR G=Ø TO 36Ø STEP 4:B=PI*G/18Ø:FOR A=1 TO 6
16Ø ON A(A) GOSUB 3ØØ,31Ø,32Ø,33Ø,34Ø,35Ø,36Ø,37Ø,38Ø,39Ø,4ØØ,41Ø
17Ø NEXT A:IF G<>Ø THEN 19Ø
18Ø MOVETO 256+B(5)*22Ø,112+B(6)*11Ø:GOTO 21Ø
19Ø DRAWTO 256+B(1)*22Ø,112+B(2)*11Ø
2ØØ DRAWTO 256+B(3)*22Ø,112+B(4)*11Ø
21Ø DRAWTO 256+B(5)*22Ø,112+B(6)*11Ø:NEXT G
22Ø FOR G=1 TO 75Ø:G=G*G/G/1:G=G:NEXT G
23Ø GOTO 13Ø
3ØØ B(A)=SIN(B):RETURN
31Ø B(A)=-SIN(B):RETURN
32Ø B(A)=COS(B):RETURN
33Ø B(A)=-COS(B):RETURN
34Ø B(A)=SIN(B+PI/4):RETURN
35Ø B(A)=-SIN(B+PI/4):RETURN
36Ø B(A)=COS(B+PI/4):RETURN
37Ø B(A)=-COS(B+PI/4):RETURN
38Ø B(A)=.5*SIN(B):RETURN
39Ø B(A)=(-.5)*SIN(B):RETURN
4ØØ B(A)=.5*COS(B):RETURN
41Ø B(A)=(-.5)*COS(B):RETURN
```

Grafik-Träumereien

von Harald Heckler und Dr. Hans Hehl

für 8K-BASIC und HEBAS.COM

Beim NDR-KLEIN-Computer mit dem 8K-BASIC im EPROM sind die Befehle MOVETO und DRAWTO vorhanden. Sehr hübsche Grafik-Demoprogramme lassen sich damit erstellen. Das folgende Beispiel wurde von Harald Heckler, Klasse 9a des Gymnasiums Markt Schwaben (8015 Markt Schwaben) entwickelt und von Dr. Hans Hehl an das 8K-BASIC (Bild 1) bzw. HEBAS.COM (Bild 2) angepaßt.

Dem Leser sind keine Grenzen gesetzt, die Parameter zu verändern. Beim DISK-BASIC-Interpreter HEBAS.COM können entweder die Befehle direkt an die Portadressen des Grafik-Prozessors gegeben werden oder man benützt die FLOMON-Einsprünge für MOVETO und DRAWTO. Am besten definiert man die ESCAPE-Sequenzen am Programmanfang.

Bei der Programmeingabe sollte man die REM-Zeilen und die Leerzeichen weglassen und möglichst viele Befehle in eine Programmzeile schreiben, um das Programm zu beschleunigen. So ist der Befehl X*X in Zeile 80 (Bild 2) viel schneller als X^2. Ebenso ist es besser, AB*0.1 anstatt AB / 10 zu schreiben. Außerdem findet bei jedem FLOMON-Zugriff die Bankumschaltung statt. Programmabbruch erfolgt wie üblich mit (CTRL-E). (Shift-A) schaltet den Grafik-Modus aus, der Cursor ist wieder da, blinkt aber nicht. Die RETURN-Taste ergibt dann einen Syntaxfehler, der ignoriert wird.

HEBAS.COM besitzt zwar keine Grafikbefehle, aber FLOMON, das Z80-Monitorprogramm, hat dafür umsomehr Möglichkeiten, die leider zu wenig bekannt und mittels ESCAPE-Sequenz von BASIC aus leicht ansprechbar sind (Sonderheft „Mikrocomputer Schritt für Schritt 2", Seite 87–91 und das CP/M-Sonderheft, Seite 58–69). So sind Fadenkreuz und Rechteck, Dreieck, Polygon und viele andere Befehle vorhanden.

```
1 REM HARALD HECKLER / DR. HEHL für 8K-BASIC
10 CLRS: PAGE 0,0
15 AX = 103: AY = 53: XV = 160 :YV = 100
100 FOR Z = -100 TO 100 STEP 10
120 FOR X = -200 TO 200 STEP 4
125 AB = SQR(Z * Z + X * X):W = AB/10 +1.57
126 Y = SIN(W)*5
130 EX = XV + X - Z/2: EY = YV + Y + Z/2
132 IF X = -200 THEN AX = EX: AY = EY
135 MOVETO AX,AY: DRAWTO EX,EY
138 AX = EX: AY = EY
140 NEXT X:NEXT Z
150 POKE HEX("87C5"),0
160 END
```

Bild 1: Beispiel für 8K-Basic (SBC2 oder SBC3)

```
10 REM HARALD HECKLER / DR. HEHL für HEBAS.COM
20 GM$ = CHR$(27) + CHR$(27) + "G"
30 GE$ = CHR$(27) + CHR$(27) + "A"
35 REM GRAPHIKMODUS EIN, Z = CLRS, P = PAGE 0,0
36 REM YO = CURSOR AUS
40 PRINT GM$;"ZP0;YO"
50 AX = 103: AY = 53: XV = 160 :YV = 100
60 FOR Z = -100 TO 100 STEP 10
70 FOR X = -200 TO 200 STEP 4
80 AB = SQR(Z * Z + X * X):W = AB * 0.1 + 1.57
90 Y = SIN(W)*5
100 EX = XV + X - Z * 0.5: EY = YV + Y + Z * 0.5
110 IF X = -200 THEN AX = EX: AY = EY
120 PRINT "M";AX,AY      :REM MOVETO AX,AY
125 PRINT "D";EX,EY      :REM DRAWTO EX,EY
130 AX = EX: AY = EY
140 NEXT X:NEXT Z
150 PRINT GE$
160 END
```

Bild 2: Beispiel für HEBAS.COM

Die Türme von Hanoi

Beispiel eines rekursiven Programms unter BASIC – Grafikdemonstration des NDR-KLEIN-COMPUTERS

von Carsten Denneburg, Wennigsen

Die Spielidee:

Auf einem Brett befinden sich drei Säulen. Auf der ersten Säule liegen n-durchbohrte Scheiben, die sich von unten nach oben verjüngen. Die Scheiben sind auf die dritte (rechte) Säule so umzuschichten, daß sie in der gleichen Reihenfolge wie auf Säule 1 abgelegt werden. Es darf nie eine größere Scheibe über einer kleineren abgelegt werden. Alle Scheiben müssen im Spiel bleiben.

Es sind zwei Geschwindigkeiten einstellbar. Die Rekursion ist dem Heft Mikrocomputer – Schritt für Schritt, Seite 105, MC Sonderheft 88, entnommen.

Um das Programm auf der SBCII ablauffähig zu halten (und damit auch dem Anfänger zugänglich zu machen) ist es bewußt extrem kurz und „platzsparend" aufgebaut. Auf Erklärungen wurde daher verzichtet.

Bei größerem Arbeitsspeicher ist der Ausbau auf ein interaktives Programm mit eigener Steuerung der Scheibenverlagerung ohne weiteres möglich.

Viel Spaß.

```
-----------------------------------------------------------
Listing des Programms - TÜRME VON HANOI -
SIEHE LOOP NR. 3 - C.DENNEBURG, WENNIGSEN
Auf der SBCII ablauffähig, daher bewusst kurz gehalten!
-----------------------------------------------------------

20 CLRS
40 SA(1)=125:SA(2)=250:SA(3)=375
50 INPUT"Wieviel Scheiben -max.7-, bis 6 Grafikausgabe";N
60 DIM X(7),Y(7),Z(7),S(7)
61 IF N>6 THEN 66
65 GOSUB 500
66 IF N>6 THEN 72
70 INPUT"Tempo 1/2";U
72 T=900
73 IF U=1 THEN T=1
75 IF N>6 THEN CLRS
80 M=0
100 X(M)=N:Y(M)=1:Z(M)=3
105 A(1)=N:A(2)=0:A(3)=0
110 GOSUB 200
115 PRINT"":PRINT"Es waren";P;"Zuege."
120 END
200 IF X(M)=0 THEN RETURN
210 M=M+1
220 X(M)=X(M-1)-1
230 Y(M)=Y(M-1)
240 Z(M)=6-Y(M-1)-Z(M-1)
250 GOSUB 200
260 M=M-1
280 P=P+1
281 IF N>6 THEN 295
282 A(Z(M))=A(Z(M))+1
283 A=A(Y(M)):E=SA(Y(M))
284 OUT 112,1
285 PAGE 0,0
286 GOSUB2000
287 FOR I=1 TO T:NEXT
288 A=A(Z(M)):E=SA(Z(M))
289 OUT 112,0
290 PAGE 0,0
291 GOSUB 2000
292 A(Y(M))=A(Y(M))-1
293 FOR I=1 TO T:NEXT
294 IF N<7 THEN 300
295 PRINT"Zug";P
296 PRINT"Scheibe";X(M);"von Sauele";Y(M);"nach Sauele";Z(M)
300 M=M+1
310 X(M)=X(M-1)-1
320 Y(M)=6-Y(M-1)-Z(M-1)
330 Z(M)=Z(M-1)
340 GOSUB 200
350 M=M-1
360 RETURN
380 END
500 PAGE 0,0
520 POKE HEX("87C5"),0
550 X1=50:X2=450:Y1=0:Y2=10
560 GOSUB 3000
570 X1=120:X2=130:Y1=10:Y2=80
580 GOSUB 3000
590 X1=245:X2=255
600 GOSUB 3000
610 X1=370:X2=380
620 GOSUB 3000
640 D=N*10
650 X1=SA(1)-D:X2=SA(1)+D:Y1=10:Y2=20
660 FOR I=1 TO N
670 GOSUB 3000
680 X1=X1+10:X2=X2-10:Y1=Y1+10:Y2=Y2+10
690 NEXT
700 RETURN
2000 H=X(M)*10
2010 X1=E-H:X2=E+H
2050 Y1=10*A:Y2=Y1+10
3000 MOVETO X2,Y1
3010 DRAWTO X2,Y2
3020 DRAWTO X1,Y2
3030 DRAWTO X1,Y1
3060 RETURN
```

HARDCOPY MIT 68008 / 68000

von Elmer Schnuit

Dieses Hardcopy-Programm ermöglicht es, eine Bildschirmkopie im „Normalformat" herzustellen, statt im Querformat, wie es im Beiheft zur Hardcopy-Baugruppe beschrieben ist.

Voraussetzung sind:
- Hardcopy-Baugruppe
- grafikfähiger Nadeldrucker
- 16 K freier Speicher (für den Bildschirminhalt)
- Speicher oder EPROM für dieses Programm
- ein wenig eigene Arbeit, um dieses Programm abzutippen und evtl. anzupassen.

Dafür ist es dann möglich, in jedem Programm, das eine Eingabe von der Tastatur zuläßt, das aktuelle Bild auf den Drukker zu geben, in den Speicher zu laden, im Speicher beliebig zu verschieben, das Bild im Speicher zu invertieren (Negativabzug), ein Bild aus dem Speicher auf den Bildschirm zu bringen und, falls der Drucker es zuläßt, Ausdrucke in verschiedenen Formaten anzufertigen.

Nach jedem Reset bzw. Einschalten des Rechners muß zunächst einmal die kleine Routine "TASTEIN" gestartet werden. Die Eingabe von "CTRL ⓪" bewirkt danach den Einsprung in das Hardcopy-Programm. Folgende Eingaben sind dann beliebig möglich:

"⓪" – Standard-Hardcopy vom Bildschirm
"A" – Aus: Bildschirm in Speicher ablegen
"E" – Ein: Bildspeicher auf den Bildschirm bringen
"V" – Verschiebt nach Abfrage 16 K von: . . . nach: . . .
"I" – Invertiert den Bildspeicher
"⓪" – Druckt den Bildschirmspeicher aus

"1" – "7" sind verschiedene Druckformate, von den Möglichkeiten des jeweiligen Druckers abhängig (hier: Taxan KP 910).

"CR" bewirkt den Rücksprung ins alte Programm.

Jetzt ist es möglich, durch Kopieren des Bildschirms in den Speicher, das Bild auf Floppy oder Kassette als Daten abzuspeichern, und z.B. später wieder einzuladen und dann wieder auf den Bildschirm zu bringen, um das Bild dann weiter mit Maus und geeignetem Programm (so man hat) zu bearbeiten . . . Der Drukker druckt zwei aufeinanderfolgende Hardcopy's nahtlos untereinander, so daß man aus drei Bildschirmen untereinander eine DIN-A-4-Seite mit einem Bild füllen kann. Bei der Kopie vom Bildspeicher auf den Bildschirm wird der Bildschirm nicht gelöscht und nur die hellen Bildpunkte gesetzt. Fürs Löschen des Bildschirms muß man selber sorgen (oder ein Mini-Unterprogramm einbauen). Dies Verfahren bietet aber die Möglichkeit, mehrere Bilder übereinander – im Overlay-Verfahren – auf den Bildschirm zu bringen: Bild herstellen, in Bildspeicher, neues Bild „malen", altes Bild wieder auf den Bildschirm . . .

Vor dem Abtippen bitte überprüfen und gegebenenfalls ändern: Bildspeicher = 16 K (wird bei jedem Hardcopy gefüllt).

Befehl: SPEICHADR EQU $70000, evtl. Speicherbereich für Buffer und Zeiger (260 Byte Platz).

Befehl: SPEICHER EQU $9A00 und die Steuerbefehle für den Drucker. Die aufgeführten Steuerbefehle sind im wesentlichen die üblichen (EPSON-) Befehle, sollten aber kontrolliert werden.

Das Programm hat einen Bibliothekseintrag, ist relokativ und berücksichtigt die Verschiebbarkeit des Grundprogramms.

Programmteile:

TASTEIN und TASTPRO sind schon vorher beschrieben worden. TASTPRO steuert die Sprünge zu den Unterprogrammen, je nach Eingabe. Bei Veränderungen hier die Stücke

TPn:
.... bis
BRA TPXXX

herausnehmen bzw. ergänzen. Dann das vorangegangene Sprungziel BRA . . . korrigieren!

Die Festlegung von RAM-Bereichen geht bei Programmen, die in EPROMS abgelegt werden können, nicht mehr nach dem Muster: BUFFER: DS.B 256! Die Übersetzung würde diesen Bereich in das (spätere) EPROM (=ROM) legen. Der Befehl SPEICHER EQU $9A00 legt den Speicherplatz für die Variable SPEICHER(.1) fest: $9A00. Mit der Länge muß man aufpassen. 'SPEICHER' als Adresspointer hat Langwortlänge. 'BUFFER' (256 Byte) darf also erst 4 Byte später beginnen, also SPEICHER+4!

HCAUS besteht einmal aus der Routine GETLINE von G. Sternberg. Die füllt den BUFFER mit 256 Byte Bildschirminhalt. Unglücklicherweise sind das (zu Beginn) die rechten (!) 8 Spalten des Bildschirms. Das erste Byte enthält im 8. Bit den letzten Bildpunkt der obersten Reihe, Bit 7 den vorletzten u.s.w. Was man als obere 8 Reihen sieht, wenn man den Bildschirm auf seine linke Seite kippt, das würde auf dem Papier erscheinen, wenn der Drukker den BUFFER jetzt ausdruckt. Also muß alles gekippt und erst im Speicher sortiert werden, bevor ein „normaler" Ausdruck erfolgen kann. Dies macht die Routine UMKIPP, gesteuert von ABLAGE. Wer will, kann ja mit Papier und Bleistift dieses Problem ergründen.

ARTANS ist eine recht nützliche Routine, die eine Zeichenserie an den Drucker weitergibt. Das erste Byte enthält die Anzahl der Zeichen (Nullbyte als Enderkennung der Liste klappt immer dann nicht, wenn man ein Nullbyte an den Drucker weitergeben will).

HCDRUCK steuert den Ausdruck vom Bildspeicher. Alle Bilder, die ausgedruckt werden sollen, müssen sich im Bildspeicher befinden.

INBYTE wird von INZEILE aufgerufen, was wieder von HCEIN gesteuert wird. Wie die Namen schon sagen, wird hier ein Byte, eine Zeile, ein Bild auf den Bildschirm gebracht.

Alle weiteren Routinen sind einfach und können (relativ) leicht nachvollzogen werden.

Wer das Programm einigermaßen verstanden hat, der sollte es gleich für seine Bedürfnisse „zurechtschneidern", z.B. den XOR-Mode einbauen, Ausschnittkopien ermöglichen, ZOOM einbauen etc. Viel Spaß.

```
ORG $0
OFFSET $1E000

HEAD:
DC.B $55,$AA,$01,$80
DC.B 'TASTEIN.'
DC.L TASTEIN-HEAD
DC.L ENDE-HEAD
DC.B 1
DC.B 0,0,0
DC.L 0,0

*************
 TASTEIN:                          ; Sprung in die Benutzerroutine
*************
   MOVE #!SETA5,D7                 ; einstellen. Nach jedem RESET
   TRAP #1                         ; neu aufrufen!
   MOVE #$4EF9,$18(A5)             * Maschinencode für JSR
   LEA TASTPRO(PC),A0              * Sprungziel (rel)
   MOVE.L A0,$1A(A5)               * in USERCI (verschiebbar)
   MOVE #$4EF9,$1E(A5)             * Sprung (s.o.)
   MOVE.L  A5,$20(A5)              * Ziel $9e0 (Vers. 4.3)
   SUB.L #$8000-$9e0,$20(a5)       * berechnet
   MOVE.B #6,$2B(A5)
  RTS

  TASTPRO:                         ; wird durch "CTRL ?" aufgerufen
   MOVE.L D7,-(A7)                 * D7 retten, wird jetzt benutzt
   MOVE #!SETA5,D7                 * RAM-Adresse des Grundprogramms
   TRAP #1                         * ermitteln, in A5
   MOVE.B #0,$2B(A5)               ; Sprung in USERCI abschalten
   MOVE #!CI,D7                    * TRAP wegen mögl. Versions-
   TRAP #1                         * änderungen
   MOVE.L (A7)+,D7                 * altes D7 wieder zurück
   CMP.B #0,D0                     ; Programmaufruf (CTRL @)? Sonst
   BNE TASTPRO1                    ; Sprung
   MOVEM.L D0-D7/A0-A6,-(A7)       ; Alle Register retten (auf Stack)
 TPXXX:
   MOVE #!CI,D7                    ; neu einlesen
   TRAP #1
   CMP.B #13,D0                    ; falls CR, dann fertig
   BEQ TPEEE
   CMP.B #'@',D0                   ; einfach Hardcopy
   BNE TP0
   BSR HCOPY
   BRA TPXXX
  TP0:
   CMP.B #'0',D0                   ; 0 = Druck einfach
   BNE TP1
   BSR HCDRUCK0
   BRA TPXXX
  TP1:
   CMP.B #'1',D0                   ; 1= Druck, doppelte Dichte
   BNE TP2
   BSR HCDRUCK1
   BRA TPXXX
  TP2:
   CMP.B #'2',D0                   ; 2= Schnelldruck, doppelte Dichte
   BNE TP3
   BSR HCDRUCK2
   BRA TPXXX
  TP3:
   CMP.B #'3',D0                   ; 3= Druck, vierfache Dichte
   BNE TP4
   BSR HCDRUCK3
   BRA TPXXX
  TP4:
   CMP.B #'4',D0                   ; 4= Druck, CRT 1
   BNE TP5
   BSR HCDRUCK4
   BRA TPXXX
  TP5:
   CMP.B #'5',D0                   ; 5= Druck, 1:1
   BNE TP6
   BSR HCDRUCK5
   BRA TPXXX
  TP6:
   CMP.B #'6',D0                   ; 6= Druck, CRT 2
   BNE TP7
   BSR HCDRUCK6
   BRA TPXXX
  TP7:
   CMP.B #'7',D0                   ; 7= Kleindruck
   BNE TPA
   BSR HCDRUCK7
   BRA TPXXX
  TPA:
   CMP.B #'A',D0                   ; A= Hcopy in Speicher
   BNE TPE
   BSR HCAUS
   BRA TPXXX
  TPE:
   CMP.B #'E',D0                   ; E= Hcopy in CRT
   BNE TPV
   BSR HCEIN
   BRA TPXXX
  TPV:
   CMP.B #'V',D0                   ; V= Hcopy im Speicher verschieben
   BNE TPI
   BSR HCSCHIEB
   BRA TPXXX
  TPI:
   CMP.B #'I',D0                   ; I= Hcopy im Speicher invertieren
   BNE TPXXX
   BSR HCINVERS
   BRA TPXXX
  TPEEE:
   MOVEM.L (A7)+,D0-D7/A0-A6       ; Alle Register zurück
 TASTPRO1:
   MOVE.B #6,$2B(A5)               ; USER
 RTS                               ; und zurück
ENDE:


LOX EQU $FFFFFF89                  ; Fkreuz X-Pos, low-Byte
HIX EQU $FFFFFF88                  ;   "      "    high-Byte
LOY EQU $FFFFFF8B                  ;   "     Y-Pos, low-Byte
HIY EQU $FFFFFF8A                  ;   "      "    high-Byte

READY EQU $FFFFFF8A                ; hcopy-port für Ready
CLEAR EQU $FFFFFF8D                ;   "          "  Clear
SPEICHADR EQU $70000
;^^^
```

```
  MOVE.B D2,HIX                  ; und .b (!) an hcopy: hix
  ROR #8,D2                      ; low-Byte (X) nach unten und
  MOVE.B D2,LOX                  ; an hcopy: lox
  MOVE.B CLEAR,D6                ; löschen: altes byte hcopy
 GETL2:
  BTST #7,READY                  ; bildpunkt  eingelesen?
  BEQ GETL2                      ; wenn nein, dann warten
  MOVE.B READY,D6                ; Punkt nach d6
  LSL.B #2,D6                    ; und in x-flag
  MOVE.B (A0),D7                 ; Bufferbyte nach d7 und um
  ROXL.B #1,D7                   ; Bit rotiert = X-Flag in Bit 7
  MOVE.B D7,(A0)+                ; zurück und Bufferzähler erhöhen
  MOVE.B CLEAR,D6                ; löschen altes Byte hcopy
 DBRA D5,GETL2                   ; Zeilenschleife: 255 mal 1 bit
 DBRA D4,GETL1                   ; 8-Spalten-Schleife, f. 255*1bit*8
 RTS
 ********************************************************

 UMKIPP:                         ;8x8 Matrix kippen. a0= buffer,
   SUBA.L #8,A1                  ; a1 = speicher / 8 byte zurück
   CLR.L (A1)+                   ; 8 byte = 2 langwort löschen
   CLR.L (A1)+
   MOVE #7,D7                    ; zähler bufferbit = speicherbyte
  UMK0:
    MOVE #7,D6                   ; zähler bufferbyte = speicherbit
    SUBQ #1,A1                   ; speicher - 1
   UMK1:
     BTST D7,(A0)+               ; buffer bit (=d7) gesetzt? Buffer erhöhen
     BEQ UMK2                    ; wenn ja, dann Bit(d6) in Speicher setzen
     BSET D6,(A1)                ; sonst nichts
    UMK2:
    DBRA D6,UMK1                 ; nächstes bufferbyte, gleiches Bit prüfen
    SUBQ #8,A0                   ; buffer auf alten stand
  DBRA D7,UMK0                   ; nächstes bit, alle 8 Bufferbyte prüfen
 RTS

 ABLAGE:                         ;legt einen buffer in den Speicher ab
   MOVEA.L SPEICHER,A1           ; für UMKIPP
   LEA BUFFER,A0                 ;      "
   MOVE #31,D1                   ; 32 Zeilen = 32 mal 8x8 Matrix kippen
  ABL1:
    BSR UMKIPP                   ; Matrix kippen
    ADDQ #8,A0                   ; nächste 8 byte in buffer
    ADDA.L #512+8,A1             ; Speicherpointer korrigieren
  DBRA D1,ABL1                   ; 32 mal
   SUBA.L #512*32+8,A1           ; neuen Wert in Speicher
   MOVE.L A1,SPEICHER
 RTS

 ATRANS:                         ; schickt eine Zeichenserie an den
   CLR D1
   MOVE.B (A1)+,D1               ; Drucker, Adresse in A1, erstes Zeichen
   SUBQ.B #1,D1                  ; = Anzahl der Zeichen in der Serie
  ATRANS1:
    MOVE.B (A1)+,D0
    MOVE #!LO,D7
    TRAP #1
  DBRA D1,ATRANS1
 RTS

 TVORSCHUB: DC.B 4,13,27,65,8
 ;^^^^^^^^^^^ Diese Liste je nach Drucker verändern: CR, 8 dot Vorschub


 TGRAFIK: DC.B 6,10,13,27,75,0,2
 ;^^^^^^^^^^^ Auch diese Liste verändern: CR,LF,Druckgrafik mit $200
 :     = 512 Zeichen
 DS 0

 **********
 HCDRUCK: *                      ; Ausdrucken eines Bildes ab SPEICHADR
 **********
   LEA TVORSCHUB(PC),A1          ; Vorschub einstellen
   BSR ATRANS
   MOVEA.L #SPEICHADR,A2         ; Speicher in A2
   MOVE #31,D2                   ; 32 Zeilen Druckumfang
  HCDR1:
    LEA TGRAFIK(PC),A1           ; Druckgrafik einstellen
    BSR ATRANS
    MOVE #511,D3                 ; 512 Byte
   HCDR2:
     MOVE.B (A2)+,D0             ; Byte aus Speicher in D0
     NOT.B D0                    ; invertieren (sonst Negativ)
     MOVE #!LO,D7
     TRAP #1                     ; an Drucker
   DBRA D3,HCDR2                 ; 512 mal
  DBRA D2,HCDR1                  ; 32 mal das Ganze
 RTS
 INBYTE:                         ; Zeichnet 1 byte senkrecht aud CRT
   MOVE #$80,D0                  ; 1 punkt (GDP) an
   MOVE #7,D7                    ; 8 Punkte
  INBY1:
    BTST.B D7,(A0)               ; Bit-Test, Bit: D7, in A0
    BNE INBY2                    ; nicht gesetzt = kein Punkt
    MOVE D7,-(A7)
    MOVE #!MOVETO,D7
    TRAP #1                      ; positionieren
    MOVE #!CMD,D7
    TRAP #1                      ; Punkt ausgeben
    MOVE (A7)+,D7
  INBY2:
    SUBQ #1,D2                   ; Bit -1
  DBRA D7,INBY1                  ; 8mal
   ADDI #8,D2                    ; alten Bitwert herstellen
 RTS


 INZEILE:                        ; Zeichnet eine Zeile auf CRT
   MOVE #0,D1                    ; A0 = Zeilenanfang im Speicher
   MOVE #511,D4                  ; d1 = X POS < 512 mal
  INZEI1:
    BSR INBYTE                   ; 1 byte (senkrecht) dasrstellen
    ADDQ #1,D1                   ; Xpos +1
    ADDQ.L #1,A0                 ; Speicher +1
  DBRA D4,INZEI1                 ; 512 mal
 RTS

 ********
 HCEIN: *                        ; Bild von Speicher auf Bildschirm
 ********
   LEA SPEICHADR,A0                ; Speicher in A0
   MOVE #255,D2                  ; Y-Pos
   MOVE #31,D3                   ; 32 Zeilen
  HCEIN1:
    BSR INZEILE                  ; Bildschirmzeile erstellen
    SUBI #8,D2                   ; eine Zeile tiefer
  DBRA D3,HCEIN1                 ; 32 mal
 RTS

 ********
 HCOPY: *                        ; Hardcopy vom akt. Bild
 ********
   BSR HCAUS
   BSR HCDRUCK
   RTS

   TSCHIEB1: DC.B 'VON:',0
   TSCHIEB2: DC.B 'NACH',0
   DS 0

   ***********
   HCSCHIEB: *                   ; Verschiebt das Bild im Speicher
   ***********
    MOVE #$11,D0                 ; Text 'von:' ausgeben
    MOVE #2,D1
    MOVE #2,D2
    LEA TSCHIEB1(PC),A0
    MOVE #!WRITE,D7
    TRAP #1

    MOVE #30,D1                  ; Startadresse einlesen
    LEA BUFFER,A0
    MOVE #10,D3
    MOVE #!READ,D7
    TRAP #1
    LEA BUFFER,A0                ; Adresse ermitteln
    MOVE #!WERT,D7
    TRAP #1
    MOVEA.L D0,A1                ; und nach A1
    MOVE #$11,D0                 ; Text 'nach' ausgeben
    MOVE #100,D1
    MOVE #2,D2
    LEA TSCHIEB2(PC),A0
    MOVE #!WRITE,D7
    TRAP #1
    MOVE #130,D1                 ; Zieladresse in Buffer
    LEA BUFFER,A0
    MOVE #10,D3
    MOVE #!READ,D7
    TRAP #1
    LEA BUFFER,A0                ; und Adresse ermitteln
    MOVE #!WERT,D7
    TRAP #1
    MOVEA.L D0,A2                ; und nach A2
    MOVE #32*512-1,D7            ; Zähler (abw), 16 K
   HCSCH1:
    MOVE.B (A1)+,(A2)+           ; Byte verschieben
   DBRA D7,HCSCH1                ; 16 K mal
   RTS

   ***********
   HCINVERS: *                   ; Bild invertieren
   ***********
     LEA SPEICHADR,A0              ; Startadresse
     MOVE #32*512-1,D7           ; Zähler
    HCINV1:
     MOVE.B (A0),D0              ; Byte in D0
     NOT.B D0                    ; invertieren
     MOVE.B D0,(A0)+             ; und zurück, A0 erhöhen
    DBRA D7,HCINV1               ; 16 K mal
   RTS

   ****************  ab hier spezielle Druckarten ****************
   *  für Taxan KP 910, der Befehlssatz für den jeweiligen
   *  Drucker muß geprüft und ggf. verändert werden

   TKGRAFIK: DC.B 7,10,13,27,94,17,0,2
   ; ^^^^^^^  CR,LF, 16 Dot Grafik mit 2*256 = 512  Bytes * 2 !

   HCDRUCK7:                     ; kleines Bild, doppelt eng
     LEA TVORSCHUB(PC),A1        ; horizontal und vertikal
     BSR ATRANS                  ; Vorschub einstellen
     LEA SPEICHADR,A2            ; Bildspeicher
     MOVE #15,D2                 ; Zeilenzähler (16 Zeilen!)
    HCKDR1:
      LEA TKGRAFIK(PC),A1        ; Grafik einstellen
      BSR ATRANS
      MOVE #511,D3               ; 512 Doppelbytes / Zeile
     HCKDR2:
       MOVE.B (A2)+,D0           ; Byte laden
       NOT.B D0                  ; invertieren
       MOVE #!LO,D7              ; an Drucker

       TRAP #1
       MOVE.B 511(A2),D0         ; Offset = 1 Bildschirmzeile * 8
       NOT.B D0                  ; laden und an
       MOVE #!LO,D7              ; Drucker
       TRAP #1
     DBRA D3,HCKDR2              ; 512 mal
     ADDA.L #512,A2              ; 2. Zeile ist schon (Offset!)
    DBRA D2,HCKDR1               ; 16 mal
   RTS

   THCD0: DC.B 4,27,63,75,0      ; Befehle zur Umsteuerung der
   THCD1: DC.B 4,27,63,75,1      ; Standardgrafik
   THCD2: DC.B 4,27,63,75,2      ; ESC "K" + n1 + n2
   THCD3: DC.B 4,27,63,75,3      ; in andere Modi
   THCD4: DC.B 4,27,63,75,4
   THCD5: DC.B 4,27,63,75,5
   THCD6: DC.B 4,27,63,75,6
   DS 0

   HCDRUCK0:                     ; einfache Dichte (Standard)
    LEA THCD0(PC),A1             ; Grafik "laden"
    BSR ATRANS
    BSR HCDRUCK                  ; Bild drucken
   RTS

   HCDRUCK1:                     ; doppelte Dichte
    LEA THCD1(PC),A1
    BSR ATRANS
    BSR HCDRUCK
   RTS

   HCDRUCK2:                     ; doppelte Dichte, doppelte
    LEA THCD2(PC),A1             ; Geschwindigkeit
    BSR ATRANS
    BSR HCDRUCK
   RTS

   HCDRUCK3:                     ; vierfache Dichte
    LEA THCD3(PC),A1
    BSR ATRANS
    BSR HCDRUCK
   RTS

   HCDRUCK4:                     ; CRT 1
    LEA THCD4(PC),A1
    BSR ATRANS
    BSR HCDRUCK
   RTS

   HCDRUCK5:                     ; Plottergrafik (1:1)
    LEA THCD5(PC),A1
```

```
 BSR ATRANS
 BSR HCDRUCK
RTS

HCDRUCK6:                          ; CRT 2
 LEA THCD6(PC),A1
 BSR ATRANS
 BSR HCDRUCK
RTS
```

```
RTS

HCDRUCK4:                          ; CRT 1
 LEA THCD4(PC),A1
 BSR ATRANS
 BSR HCDRUCK
RTS

HCDRUCK5:                          ; Plottergrafik (1:1)
 LEA THCD5(PC),A1
 BSR ATRANS
 BSR HCDRUCK
RTS

HCDRUCK6:                          ; CRT 2
 LEA THCD6(PC),A1
 BSR ATRANS
 BSR HCDRUCK
RTS

Textstart=040000 Fenster=043638 Tor=043638      amer CTRL-J=Hilfe
```

```
Textbearbeitung
   0=Text drucken
   1=Editor
   2=Text löschen
   3=neuer Textanfang
   4=Druck einstellen
   M=Menue

neuer Textanfang  $50000
```

Programmaufruf per Tastendruck

von Elmer Schnuit, Göttingen

Dieses Programm ist als Antwort auf die Frage entstanden, wie ich eine Hardcopy vom Bildschirm erstellen kann, ohne in jedem Programm einen Sprung zu „HCOPY" einzubauen. Das Ergebnis ist hier dieses kleine Programm, das auf Tastendruck (hier CTRL? oder CTRL DEL) einen Sprung in ein beliebiges (Unter-) Programm (im Beispiel: HCOPY) durchführt. Voraussetzung ist nur, daß das laufende Programm eine Eingabe zuläßt. Zur Not muß man an passender Stelle 'JSR@CI' einbauen. Jetzt ist eine Kopie eines Teils des Grundmenüs oder des „Apfelmännchens" (*LOOP* 5, S. 16) ohne Probleme möglich. Dieses Programm ähnelt in Teilen dem „Pieps-Programm". Die Lösung liegt darin, die Routine „CI" so umzusteuern, daß die normale Tastatureingabe nicht gestört wird, gleichzeitig aber geprüft werden kann, ob der Programmaufruf gewünscht wird.

Zum Programm:

Im ersten Teil wird die Umsteuerung auf die eigene Routine durch den sprungbefehl 'JSR TASTPRO' in USERCI festgelegt. Da die Grundprogrammroutine CSTS automatisch mit umgesteuert wird (was in diesem Fall garnicht erwünscht ist), wird in der Adresse $801E (USERCSTS) der Rücksprung in nach CSTS festgelegt.

Das Programm "TASTEIN" besteht nur aus einem Befehl und legt die Umlegung auf die Benutzerroutine fest ("6" in IOSTAB). Nach der Übersetzung des ganzen Programms erst einmal "TASTEIN" aufrufen, ebenso nach Betätigung des RESET-Tasters, da beim Reset die CI-Routine wieder normalisiert wird (="0" in IOSTAB).

Jeder Aufruf von CI in einem Programmteil des Grundmenues bewirkt nach diesen Vorbereitungen einen Sprung in das Programm TASTPRO. Dies geschieht, *bevor* die Routine CI durchgeführt wird. Also muß das Zeichen jetzt erst noch von der Tastatur geholt werden, damit CI, also die normale Tastatureingabe, auch weiterhin funktioniert. Warum aber dafür ein eigenes Programm schreiben, wenn es dafür den Aufruf "CI" gibt? Der Aufruf 'JSR@CI' würde jetzt aber das Programm „aufhängen", da "CI" nach TASTPRO umgeleitet wird. Also wird mit 'MOVE.B # 0, 802B' "CI" normalisiert und dann mit 'JSR @CI' das Zeichen eingelesen. Dann erfolgt die Prüfung, ob es sich um das gesuchte Zeichen "CTRL ?" handelt. Wenn nicht, wird "CI" wieder umgelenkt

Bild 1

```
ORG $8018                          ; Sprung auf eigenen Routine
JMP TASTPRO                        ; in USERCI
ORG $801E                          ; direkter Rücksprung nach
JMP $9E0                           ; CSTS in USERCSTS

ORG $9C00                          ; Speicherbereich für Programme

  TASTEIN:                         ; Sprung in die Benutzerroutine
   MOVE.B #6,$802B                 ; einstellen. Nach jedem RESET
  RTS                              ; neu aufrufen!

  TASTPRO:                         ; wird durch "CTRL ?" aufgerufen
   MOVE.B #0,$802B                 ; Sprung in USERCI abschalten
   JSR @CI                         ; Normales Einlesen von Tastatur
   CMP.B #31,D0                    ; Programmaufruf (CTRL ?)? Sonst
   BNE TASTPRO1                    ; Sprung
   MOVEM.L D0-D7/A0-A6,-(A7)       ; Alle Register retten (auf Stack)
   JSR HCOPY                       ; Sprung in Programm (hier Hardcopy)
   MOVEM.L (A7)+,D0-D7/A0-A6       ; Alle Register zurück
  TASTPRO1:
   MOVE.B #6,$802B                 ; USERCI wieder einschalten
  RTS                              ; und zurück
END
```

Bild 2

```
HEAD:
DC.B $55,$AA,$01,$80
DC.B 'TASTEIN.'
DC.L TASTEIN-HEAD
DC.L ENDE-HEAD
DC.B 1
DC.B 0,0,0
DC.L 0,0

  TASTEIN:                         ; Sprung in die Benutzerroutine
   MOVE #!SETA5,D7                 ; einstellen. Nach jedem RESET
   TRAP #1                         ; neu aufrufen!
   MOVE #$4EF9,$18(A5)             * Maschinencode für JSR
   LEA TASTPRO(PC),A0              * Sprungziel (rel)
   MOVE.L A0,$1A(A5)               * in USERCI (verschiebbar)
   MOVE #$4EF9,$1E(A5)             * Sprung (s.o.)
   MOVE.L  A5,$20(A5)              * Ziel $9e0 (Vers. 4.3)
   SUB.L #$B000-$9e0,$20(a5)       * berechnet
   MOVE.B #6,$2B(A5)
  RTS

  TASTPRO:                         ; wird durch "CTRL ?" aufgerufen
   MOVE.L D7,-(A7)                 * D7 retten, wird jetzt benutzt
   MOVE #!SETA5,D7                 * RAM-Adresse des Grundprogramms
   TRAP #1                         * ermitteln, in A5
   MOVE.B #0,$2B(A5)               ; Sprung in USERCI abschalten
   MOVE #!CI,D7                    * TRAP wegen mögl. Versions-
   TRAP #1                         * änderungen
   MOVE.L (A7)+,D7                 * altes D7 wieder zurück
   CMP.B #31,D0                    ; Programmaufruf (CTRL ?)? Sonst
   BNE TASTPRO1                    ; Sprung
   MOVEM.L D0-D7/A0-A6,-(A7)       ; Alle Register retten (auf Stack)
   JSR HCOPY(PC)                   ; Sprung in Programm (hier Hardcopy)
   MOVEM.L (A7)+,D0-D7/A0-A6       ; Alle Register zurück
 TASTPRO1:
   MOVE.B #6,$2B(A5)               ; USER
 RTS                               ; und zurück
ENDE:
END
```

(damit der „Tastendruck" beim nächsten mal auch noch funktioniert) und in das aufrufende Programm zurückgesprungen. Wenn ja, werden alle Register gerettet, damit das aufrufende Programm nach der Unterbrechung weiterarbeiten kann. Danach erfolgt der Sprung ins (Unter-) Programm HCOPY (oder ein anderes, nach eigenem Wunsch). Nach der Rückkehr werden die Register wieder hergestellt, die Umlenkung eingeschaltet und ins aufrufende Programm zurückgesprungen (Bild 1).

Das gleiche Programm hier noch einmal relokativ, wobei auch die Verschiebbarkeit des Grundprogramms berücksichtigt und das Programm mit einem Bibliotheksvorspann versehen wurde. Das aufgerufene Programm sollte in der Nähe im Speicher liegen (Sprungweite).

Statt: JSR @CI
jetzt: MOVE #!CI, D 7
TRAP 1

bewirkt, daß dieser Aufruf bei allen Versionen des Grundprogramms funktioniert.

Statt: MOVE.B #6, $802B
jetzt: MOVE.B #6, $2B(A5)

bewirkt, daß jetzt #6 in die Speicherzelle: Speicheranfang für Grundprogramm (A5) plus $2B abgelegt wird. Der RAM-Bereich fürs Grundprogramm kann sich verschieben, je nach Lage des Grundprogramms im Adressbereich (relokativ). In A5 befindet sich der RAM-Anfang (nach SETA5 sicher).

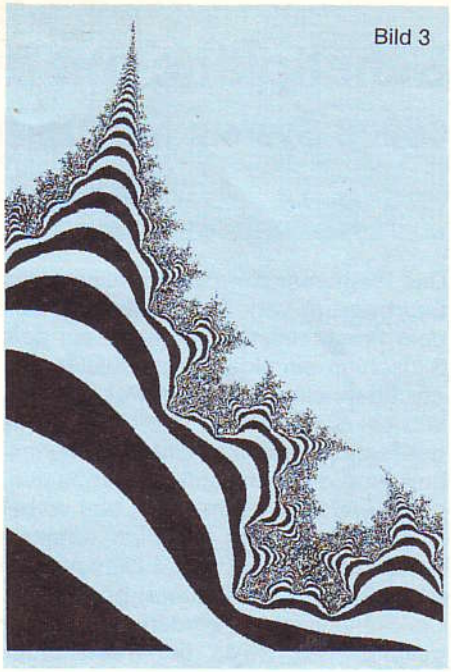


Bild 3

Statt: JSR HCOPY
jetzt: JSR HCOPY(PC)

bewirkt, daß jetzt nicht mehr die *absolute* Sprungadresse (HCOPY) im Maschinenprogramm gespeichert wird, sondern der relative Abstand der aktuellen Speicherzell (=PC, Programmzähler) zum Programm HCOPY, also die Sprung*weite*.

Für „Fortgeschrittene": Dieser eigentlich schon recht bequeme Programmaufruf arbeitet nicht mehr problemlos, wenn sowohl das laufende wie das aufgerufene Programm mit dem Bildschirm arbeiten. Dann wird natürlich der Bildschirminhalt zerstört. Eine Lösung dieses Problems ist durch ein Hardcopy-Programm möglich, das den Bildschirm in einen freien Speicherbereich kopieren und zurückschreiben kann: 68008-Register retten, GDP-Register retten, Hardcopy in Speicherbereich, Programmaufruf, Hardcopy zurück, GDP-Register zurück, 68008-Register zurück.

Die Bilder 3 und 4 zeigen eine Hardcopy.

```
ORG $8018                       ; Sprung auf eigenen Routine
JMP TASTPRO                     ; in USERCI
ORG $801E                       ; direkter Rücksprung nach
JMP $9E0                        ; CSTS in USERCSTS

ORG $9C00                       ; Speicherbereich für Programme

  TASTEIN:                      ; Sprung in die Benutzerroutine
   MOVE.B #6,$802B              ; einstellen. Nach jedem RESET
  RTS                           ; neu aufrufen!

  TASTPRO:                      ; wird durch "CTRL ?" aufgerufen
   MOVE.B #0,$802B              ; Sprung in USERCI abschalten
   JSR @CI                      ; Normales Einlesen von Tastatur
   CMP.B #31,D0                 ; Programmaufruf (CTRL ?)? Sonst
   BNE TASTPRO1                 ; Sprung
   MOVEM.L D0-D7/A0-A6,-(A7)    ; Alle Register retten (auf Stack)
   JSR HCOPY                    ; Sprung in Programm (hier Hardcopy)
   MOVEM.L (A7)+,D0-D7/A0-A6    ; Alle Register zurück
  TASTPRO1:
   MOVE.B #6,$802B              ; USERCI wieder einschalten
  RTS                           ; und zurück
END
#
Textstart=009000 Fenster=009002 Tor=009002        amer CTRL-J=Hilfe
```

Bild 4

Zeilenende-Pieps (68K)

von Elmar Schniut

Da ich den NDR-KLEIN-Computer (der mir als Techniklehrer außerordentlich gut gefällt) sehr viel zum Erstellen von Texten gebrauche, nutze ich intensiv die „Editor"-Funktion. Dabei ist das Fehlen eines Signals, das, wie bei der Schreibmaschine die Glocke, das Zeilenende anzeigt, ziemlich lästig, da man nur zu leicht über die 50 oder 60 Zeichen einer normalen Schreibmaschinenzeile hinauskommt, und dann den Rest wieder löschen und auf der nächsten Zeile neu schreiben muß.

Hier ein kleines Programm, daß den im Drucker eingebauten Piepser bei Erreichen einer beliebigen Cursor-Position ertönen läßt. Dabei ist die Editor-Funktion unberührt geblieben. Statt dessen ist hier die Möglichkeit genutzt, die Eingabe auf eine eigene (Benutzer-) Routine umzulenken. Dies ist durch den Wert 6 in der Speicherstelle IOSTATB ($802B) möglich. Der notwendige Sprungbefehl (JMP PIEPS) muß in USERCI ($8018) stehen.

Jetzt springen alle Programme beim Aufruf der CI-Routine in das Unterprogramm PIEPS, so auch EDITOR. Hier wird nun ganz einfach geprüft, ob der Spaltenzähler des Cursors CURX (= Speicherstelle $8223) den gewünschten Wert enthält. Wenn ja, wird der Piepser im Drucker aktiviert. Der Rücksprung (JMP $A30) in die Routine CI bewirkt, daß diese Routine zwar voll durchgeführt wird, die Prüfung auf Umlenken der Eingabe aber übersprungen wird. Ein Blick in's Listing des Grundprogramms (ganz am Anfang) macht's deutlich.

Wichtig: Mit der Umlenkung von CI wird auch die Routine CSTS umgelenkt. Da diese aber nicht verändert werden soll, schreibt man in den dazugehörigen Sprungvektor USERCSTS ($801E) einfach einen Rücksprung in die Routine CSTS (JMP $9E0).

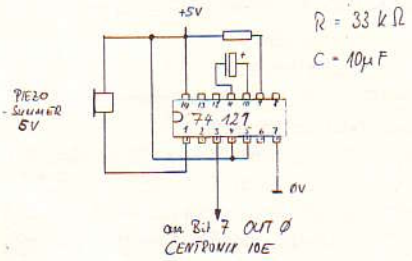
So, nun das Programm eingeben, übersetzen lassen, PIEPEIN starten, und dann piept der Drucker immer dann, wenn eine

```
ORG $801E                       ;Sprungbefehl "nach PIEP" in
  JMP PIEP                      ;USERCI; Sprungbefehl "gleich
  JMP $9E0                      ;zurück" in USERCSTS

ORG $9C00
PIEPEIN:                        ;Eingaberoutine umleiten (= 6
  MOVE.B #6,$802B               ; in IOSTATB)
RTS

PIEP:
  CMP.B #50,$8223               ; Vergleich gewünschte Zeichen-
  BNE PIEP1                     ; zahl (hier: 50) mit CURX (= akt.
    MOVE #7,D0                  ; Cursorposition).
    JSR @LO                     ; Wenn ja, Beep an Drucker
    MOVE #13,D0                 ; (mit CR!)
    JSR @LO                     ; Rücksprung in die CI-Routine
  PIEP1:                        ; ($A08), so daß CI noch ausge-
  JSR $A08                      ; führt wird.
RTS
END
```

Eingabe in die Spalte 50 erfolgt. Reset löscht, PIEPEIN startet neu.

Dieses „nackte" Programm soll nur das grundsätzliche Vorgehen verdeutlichen. Eine Spaltenabfrage macht das Programm komfortabler, eine Einbindung in die Bibliothek ist sicher auch sinnvoll. (Achtung bei den Sprungbefehlen am Anfang).

Wer keinen Drucker mit Piepser hat, oder wer seinen Drucker nicht dauernd einschalten möchte, kann die dargestellte Hardware-Lösung auf der IOE-CENT-Platine realisieren und muß dann das Programm entsprechend umschreiben.




```
Statt:  BNE PIEP1
        ...
        ...
        PIEP1:
dann:   BNE PIEP1
        BCHG.B #7, $FFFF48
        BCHG.B #7, $FFFF48
        PIEP1:
```

Schaltpläne mit dem NDR-Computer

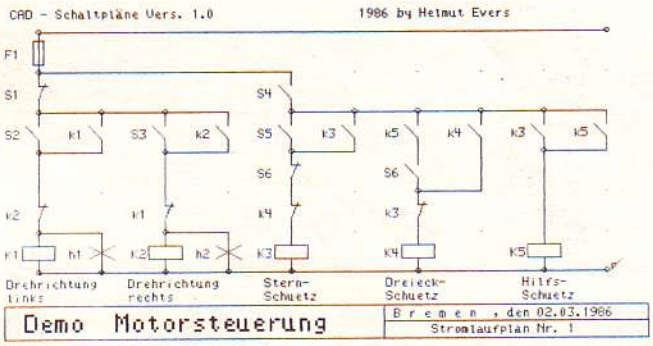
68008-System mit Diskette, Hardcopy und Jogidos

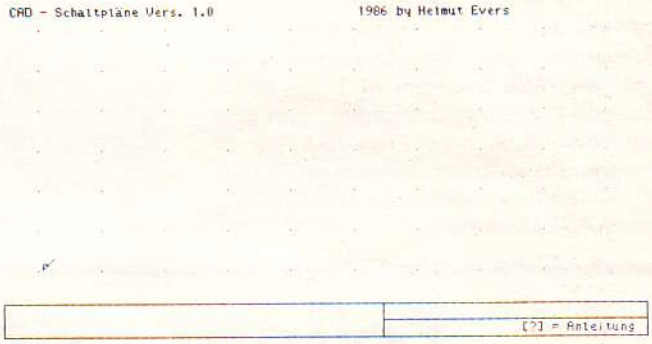
von Helmut Evers, Bremen

Das Programm dient zur Erstellung und Bearbeitung von Schaltplänen. Nach dem Programmstart erscheint auf dem Bildschirm ein Orientierungsraster zum Zeichnen, ein Zeiger zum Positionieren und ein Schriftfeld für Usereinträge. Mit der Taste [?] erscheint eine Bedienungsanleitung, die nach nochmaligem Tastendruck wieder verschwindet. Das vorher Gezeichnete bleibt natürlich erhalten. Der Zeiger wird mit den 4 Cursortasten (Cherry-Low-Cost-Tastatur) positioniert. Die Schaltsymbole werden dann an der gewünschten Stelle per Tastendruck ausgegeben. Auch die Beschriftung der Symbole erfolgt auf diese Weise. Die Grafik kann ausgedruckt und ohne Schaltungsänderung auf Diskette geschrieben und gelesen werden.

Zum Speichern der Grafik benutze ich die Hardcopy, mit der man jeden Punkt des Bildschirmes abfragen kann. Diese Informationen werden in einem 16 K RAM-Speicher abgelegt. Dann ist es kein Problem mehr, die Daten auf die Diskette zu schreiben. In diesem Programm benutze ich dafür das 'YOGIDOS', das direkt per Tastendruck aufgerufen wird. Beim Lesen werden die Daten von der Diskette in den RAM-Speicher übernommen und bei Tastendruck erscheint die Grafik auf dem Bildschirm. Die Tastenbelegung steht in der Anleitung.

Weil jeder Anwender sein JOGIDOS und Hardcopy in einen anderen Speicherbereich abgelegt hat, müssen die Startadressen nachgetragen werden.






```
*****************************************************
* CAD - Schaltplaene * Helmut Evers                 *
*                    * 28 Bremen 21                 *
* Vers. 1.0    02.86 * Kamerunstr. 59  Tel. 0421/646505 *
*****************************************************

org $10000
;-----------------------------------------------------
copy equ $xxxxx          ; Startadr. der HardcoPy eintragen !  ;
dos equ $xxxxx           ; Startadr. vom Yogidos eintragen !   ;
;-----------------------------------------------------

h88 equ $ffffff88        ; x MSB  Fadenkreuz
h89 equ $ffffff89        ; x LSB     "
h8a equ $ffffff8a        ; y MSB     "
h8b equ $ffffff8b        ; y MSB     "
h8d equ $ffffff8d        ; Speichern der Zaehlerstaende
h8e equ $ffffff8e        ; Loeschen der Zaehler
h8f equ $ffffff8f

bildsp equ $20000        ; Ram $20000-$23fff (Bildspeicher)
fxbit: dc.w 0            ; Pos. Fadenkreuz
fybit: dc.w 0            ;  "      "

buffer: ds.b 200         ; 1 Block (8 Spalten)
maske: dc.b 0            ;

Pen: dc.b 2,0,5,1,1,10
Point: dc.b 0,0,1,2,3,4,4,5,6,7,10
ds 0

x: dc.w 0                * x-Pos. Zeiger
y: dc.w 0                * y-Pos.   "
xPo: dc.w 0              * x-Pos. Merker
yPo: dc.w 0              * y-Pos.   "
en: dc.b 0               * Ende - Merker
tbuff: ds.b 100
ds 0

kreuz:                   * Raster-Punkt
 move.w xPo,d1           * x-Pos. holen
 move.w yPo,d2           * y-Pos. holen
 jsr @moveto             * dorthin
 move.b #$80,d0          * Punkt
 jsr @cmd                * ausgeben
 move.w d1,xPo           * x-Pos. merken
 move.w d2,yPo           * y-Pos. merken
rts

raster:                  * Hilfsraster
 move.w #27,xPo          * links
raloop:
 move.w #50,yPo          * unten anfangen
 move #7-1,d4            *
yloop:                   *
 move #10-1,d3           *
xloop:                   *
 jsr kreuz               *
 add.w #50,xPo           *
dbra d3,xloop            *
 add.w #30,yPo           *
 sub.w #500,xPo          *
dbra d4,yloop            *
 rts

box:
 clr d1                  * Schriftfeld
 clr d2
 jsr @moveto
 add #511,d1
 jsr @drawto
 add #25,d2
 jsr @drawto
 clr d1
 jsr @drawto
 clr d2
 jsr @drawto
 add #300,d1
 jsr @moveto
 add #25,d2
 jsr @drawto
 sub #12,d2
 jsr @moveto
 add #211,d1
 jsr @drawto
rts

start:                   * HauptProgramm
 jsr @clr                * alles loeschen
 move.b #3,d0            *
 move.b #3,d1            *
 jsr @newPage            * Seite 3
 clr.b en                * Ende-merker 0
 lea cad,a0              * Ueberschrift
 move.b #$11,d0          * Schriftgroesse
 move #5,d1              * x-Pos.
 move #240,d2            * y-Pos.
 jsr @write              * ausgeben
 move #3,d2              * y-Pos.
 move #410,d1            * x-Pos.
 lea infob,a0            * Help
 jsr @write              * ausgeben
 jsr raster              * Hilfsraster ausgeben
 jsr box                 * Schriftfeld
 move #31,x              * StartPosition fuer
```

```
 move #50,y          * Zeiger

schleife:
 bsr Pencil          * Zeiger ausgeben
 move.b en,d0        * Ende-Merker holen
 cmp.b #1,d0         * Abbruch ?
 bne schleife        * nein, dann nochmal
rts

Pos:                 * Positionierung
 jsr @ci             * Tastaturabfrage
 cmp.b #$08,d0       * links ?
 bne s1              * nein, dann weiter
 move.w x,d1         * sonst x-Pos. holen
 cmp.w #31,d1        * linker Rand erreicht ?
 beq s1              * ja, dann ignorieren
 sub #50,x           * sonst 50 Punkte n. links
s1:
 cmp.b #$09,d0       * rechts ?
 bne s2              * nein, dann weiter
 move.w x,d1         * sonst x-Pos. holen
 cmp.w #481,d1       * rechter Rand erreicht ?
 beq s2              * ja, dann ignorieren
 add #50,x           * sonst 50 Punkte nach rechts
s2:
 cmp.b #$0a,d0       * runter ?
 bne s3              * nein, dann weiter
 move.w y,d1         * sonst y-Po. holen
 cmp.w #50,d1        * unterer Rand erreicht ?
 beq s3              * ja, dann ignorieren
 sub #30,y           * sonst 30 Punkte nach unten
s3:
 cmp.b #$0b,d0       * rauf ?
 bne s4              * nein, dann weiter
 move.w y,d1         * y-Pos. holen
 cmp.w #230,d1       * oberer Rand erreicht ?
 beq s4              * ja, dann ignorieren
 add #30,y           * sonst 30 Punkte nach oben
s4:
 cmp.b #'e',d0       * Ende ?
 bne s5              * nein, dann weiter
 move.b #1,en        * sonst endemarke setzen

s5:                  * Linie von / bis
 cmp.b #'v',d0       * sorce neu ?
 bne s6              * nein, dann weiter
 move.w x,d1         * sonst x-Pos. holen
 move.w y,d2         * y-Pos. holen
 sub.w #4,d1         * nach links korrigieren
 move.w d1,xPo       * x-Pos. merken
 move.w d2,yPo       * y-Pos. merken
s6:
 cmp.b #'b',d0       * destination
 bne s7              * nein, dann weiter
 move.w xPo,d1       * x-Merker holen
 move.w yPo,d2       * y-Merker holen
 jsr @moveto         * Anfangspos.
 move.w x,d1         * x-Pos holen
 sub.w #4,d1         * korrigieren
 move.w y,d2         * y-Pos. holen
 jsr @drawto         * Linie ausgeben
s7:
 cmp.b #'c',d0       * Hardcopy ?
 bne s8              * nein, dann weiter
 jsr copy            * sonst Hardcopy ausdrucken
s8:
 cmp.b #'k',d0       * Schuetz ?
 bne s9              * nein, dann weiter
 bsr schuetz         *
s9:
 cmp.b #'s',d0       * Schliesser ?
 bne s10             * nein, dann weiter
 bsr schliesser      *
s10:
 cmp.b #':',d0       * Oeffner ?
 bne s11             * nein, dann weiter
 bsr oeffner         *
s11:
 cmp.b #'f',d0       * Sicherung ?
 bne s12             * nein, dann weiter
 bsr sicherung       *
s12:
 cmp.b #'p',d0       * Verbindungpunkt ?
 bne s13             * nein, dann weiter
 bsr Punkt           *
s13:
 cmp.b #'t',d0       * beschriften ?
 bne s14             * nein, dann weiter
 bsr write           *
s14:
 cmp.b #'?',d0       * help ?
 bne s15             * nein, dann weiter
 bsr help            *
s15:
 cmp.b #'n',d0       * Neustart ?
 bne s16             * nein, dann weiter
 bra start           *
s16:
 cmp.b #'E',d0       * Loesch-Mode ?
 bne s17             * nein, dann weiter
 jsr @eraPen         * Zeichen loeschen
 bra Pos             * Zeichen abfragen
s17:
 cmp.b #'h',d0       * Kontroll-Lampe
 bne s18             * nein, dann weiter
 bsr lampe           *
s18:                 * Schriftfeld
 cmp.b #'1',d0       * Feld 1 ?
 bne s18a            * nein, dann weiter
 bsr feld1           *
s18a:
 cmp.b #'2',d0       * Feld 2 ?
 bne s18b            * nein, dann weiter
 bsr feld2           *
s18b:
 cmp.b #'3',d0       * Feld 3 ?
 bne s19             * nein, dann weiter
 bsr feld3           *
s19:
 cmp.b #'!',d0       * Ergaenzungseintraege ?
 bne s20             * nein, dann weiter
 bsr ergaenz         *
s20:
 cmp.b #'$',d0       * Grafik speichern ?
 bne s21             * nein, dann weiter
 bsr save            * im Ram ablegen $20000-$23fff
s21:
 cmp.b #'%',d0       * Grafik abilden ?
 bne s22             * nein, dann weiter
 bsr setPict         * Grafik wiedergeben
s22:
 cmp.b #'^',d0       * Floppy ?
 bne s23             * nein, dann weiter
 jsr @getstx         * Textstart sichern
 movem.l d0,-(a7)    *
 jsr dos             * Jogidos aufrufen
 movem.l (a7)+,d0    * alter Textstart
 jsr @putstx         * geht sonst beim Datenlesen verloren
 jsr @clr            * alle Bildseiten loeschen
 move #3,d0          *
 move #3,d1          *
 jsr @newPage        *
 bsr box             *
 bsr raster          *
 bsr setPict         * Grafik darstellen
s23:
 rts                 * Ende der Tastaturabfrage

Pencil:              * Zeiger
 lea Pen,a0          *
 move #$4,d0         * Groesse
 move x,d1           * x-Position
 move y,d2           * y-Position
 add #1,d2           * Korrektur
 jsr @figur          * Zeiger ausgeben
 jsr time            * kein Flimmern
 jsr Pos             * neue Position bestimmen
rts

time:
 move #2-1,d4
tim:
 jsr @sync
 beq tim
dbra d4,tim
rts

**** schaltsymbole ****
schuetz:
 move.w x,d1
 sub.w #4,d1
 move.w y,d2
 jsr @moveto
 add #10,d2
 jsr @drawto
 sub #13,d1
 jsr @drawto
 add #10,d2
 jsr @drawto
 add #26,d1
 jsr @drawto
 sub #10,d2
 jsr @drawto
 sub #13,d1
 jsr @drawto
 add #10,d2
 jsr @moveto
 add #10,d2
 jsr @drawto
rts

schliesser:
 move.w x,d1
 sub.w #4,d1
 move.w y,d2
 jsr @moveto
 add #10,d2
 jsr @drawto
 add #10,d2
 sub #10,d1
 jsr @drawto
 add #10,d1
 jsr @moveto
 add #10,d2
 jsr @drawto
rts

oeffner:
 move.w x,d1
 sub.w #4,d1
 move.w y,d2
 jsr @moveto
 add #10,d2
 jsr @drawto
 add #12,d2
 add #5,d1
 jsr @drawto
 sub #2,d2
 add #2,d1
 jsr @moveto
 sub #7,d1
 jsr @drawto
 add #10,d2
 jsr @drawto
rts

sicherung:
 move.w x,d1
 sub.w #4,d1
 move.w y,d2
 jsr @moveto
 add #30,d2
 jsr @drawto
 sub #6,d2
 jsr @drawto
 add #4,d1
 jsr @drawto
 sub #18,d2
 jsr @drawto
 sub #8,d1
 jsr @drawto
 add #18,d2
 jsr @drawto
 add #4,d1
 jsr @drawto
rts

Punkt:               * Verbindungspunkt
 lea point,a0        * Adresse setzen
 move.b #$1,d0       * Groesse
 move x,d1           * x-Pos.
 move y,d2           * y-Pos.
 sub #5,d1           * korrigieren
 sub #1,d2           * korrigieren
 jsr @figur          * ausgeben
 jsr @setfig         * dauerhaft
rts

lampe:               * Kontroll-Lampe
 move x,d1           * x-Pos.
 sub #4,d1           * korrigieren
 move y,d2           * y-Pos.
 jsr @moveto         *
 add #30,d2          *
 jsr @drawto         *
 sub #10,d2          *
 add #10,d1          *
 jsr @moveto         *
 sub #10,d2          *
 sub #20,d1          *
 jsr @drawto         *
 add #20,d1          *
 jsr @moveto         *
 add #10,d2          *
 sub #20,d1          *
 jsr @drawto         *
rts

write:               * Symbol beschriften
 move.w x,d1         * x-Position des Zeigers holen
 sub.w #30,d1        * links neben dem Symbol
 move.w y,d2         * y-Position des Zeigers holen
```

```
 add #10,d2            * etwas hoeher
 move.b #$11,d0        * Schriftgroesse
 move #2,d3            * 2 Zeichen
 lea tbuff,a0
 jsr @read
rts

ergaenz:
 move.b #$11,d0        * Schriftgroesse
 move #2,d1            * x-Pos.
 move.b #38,d2         * y-Pos.
 move #84,d3           * max. 84 Zeichen
 lea tbuff,a0          * Buffer
 jsr @read             * Eingabe
 sub #10,d2            * naechste Zeile
 jsr @read             * Eingabe
rts

feld1:
 move.b #$22,d0        * Schriftgroesse
 lea tbuff,a0          * Buffer
 move #4,d1            * x-Pos.
 move #4,d2            * y-Pos.
 move #24,d3           * 24 Zeichen
 jsr @read             * Eingabe
rts

feld2:
 move.b #$11,d0
 lea tbuff,a0
 move #302,d1
 move #15,d2
 move #38,d3
 jsr @read
 jsr box
rts

feld3:
 move.b #$11,d0
 lea tbuff,a0
 move #302,d1
 move #3,d2
 move #38,d3
 jsr @read
 jsr box
rts

help:
 move #0,d0
 clr d1
 jsr @setflip          * keine Seitenumschaltung
 jsr @wait
 clr d2
 clr d1
 jsr @setcurxy
 lea infoa,a0          * Anleitung ausgeben
he0:                   *
 move.b (a0)+,d0       *
 jsr @co2              *
 cmp.b #0,d0           *
 bne he0               *
 jsr @wait             *
 move.b #$4,$ffffff72  * Schraegschrift ein
he1:                   *
 move.b (a0)+,d0       *
 jsr @co2              *
 cmp.b #0,d0           *
 bne he1               *
he2:                   *
 jsr @wait             *
 move.b #$0,$ffffff72  * Schraegschrift aus
 move.b (a0)+,d0       *
 jsr @co2              *
 cmp.b #0,d0           *
 bne he2               *
 jsr @ci               * zurueck zur Grafik
 move.b #3,d0          *
 move.b #3,d1          *
 jsr @newPage          * Grafik-Seite
rts

save:                  * Grafik speichern
 lea bildsp,a1         * Bildspeicher holen
 move.w #$ff41,fxbit   * Fadenkreuz x-Pos.
 sub.w #512,fxbit
 move #64-1,d7         * 64 Bloecke
sav:
bsr getpict            * Lichtpunkte abtasten
 lea buffer,a0         * Zwischenspeicher
 move #256-1,d3        * 256 Punkte/Spalte

sa:
 move.b (a0)+,(a1)+    * in Bildspeicher laden
dbra d3,sa

dbra d7,sav
move.w #$00,h88        * Fadenkreuz abschalten
move.w #$00,h8a        *      "          "
rts

getpict:               * Lichtpunkte abtasten
 lea buffer,a0         * Zwischenspeicher
 move.w #$fffe,fybit   * Fadenkreuz y-Pos.
move #8-1,d4           * 8 Spalten
```

```
getpic:
 lea buffer,a0
 move.w fybit,h8a      * Zeilennr. einstellen
 move.w fxbit,h88      * Spaltennr. einstellen
 add.w #1,fxbit
 move.b h8d,d0         * flag des 74173 loeschen

 move #256-1,d5        * 256 Punkte abtasten
getpi:
 move.b h8a,d0         * flag loeschen
 btst.b #7,d0          * Punkt gefunden ?
 beq getpi             * nein, dann nochmal

 move.b h8a,d0         * flag loeschen
 and.b #%01000000,d0   *
 cmp.b #$40,d0         * Punkt erleuchtet ?
 beq ja

nein:
 move.b (a0),d1        * Byte holen
 bset.w #10,d1         * Punkt setzen
 ror.w #1,d1           * naechster Punkt in der Spalte
 move.b d1,(a0)        * Byte ablegen
 bra next

ja:
 move.b (a0),d1        * Byte holen
 bclr.w #10,d1         * Punkt loeschen
 ror.w #1,d1           * naechster Punkt in der Spalte
 move.b d1,(a0)        * Byte ablegen

next:
 adda #1,a0            * naechste Ramzelle
 move.b h8d,d0         * flag loeschen

dbra d4,getpic
 clr.b (a0)            * Ende-Zeichen
rts

setpict:               * Grafik aus Speicher darstellen
 move #511,xpo         * Anfangsposition
 move #256,ypo         *      "
 lea bildsp,a0         * Bildspeicher holen

 move #64-1,d5         * 64 Bloecke zu 8 Spalten
setpic:
 move.b #$1,maske      * zuerst Bit 0 ausgeben

 move #8-1,d4          * 8 Bits
setpi:
 move #256-1,d3        * 256 Bytes
setp:
 move.b (a0)+,d0       *
 and.b maske,d0        * Bit gesetzt ?
 beq set               * sonst weiter
 jsr @wait
 move xpo,d1           * neue Position
 move ypo,d2           *   "     "
 jsr @moveto           * einstellen
 move.b #$80,d0        * Punkt
 jsr @cmd              * ausgeben
set:
 subq #1,ypo           * naechste Zeile
dbra d3,setp
 subq #1,xpo           * naechste Spalte
 move.b maske,d0       * Maske
 muls #2,d0            * um 1 Bit
 move.b d0,maske       * verschieben
 move #257,ypo         * wieder oben anfangen
 suba.l #256,a0        * Speicher zurueckholen
dbra d4,setpi
 adda.l #256,a0        * naechster Block
dbra d5,setpic
rts

cad: dc.b 'CAD - Schaltpläne Vers. 1.0                    1986 by Helmut Evers',0

infoa: dc.b '          ä     A  N  L  E  I  T  U  N  G',$d,$a
       dc.b '[k] = Schtz        [s] = Schlieer      [ö] = Öffner ',$d,$a
       dc.b '[f] = Sicherung    [p] = Punkt         [h] = Lampe  ',$d,$a
       dc.b '[v] = Position merken  (Linie von)',$d,$a
       dc.b '[b] = Ziel        ü     (       bis)',$d,$a
       dc.b '[n] = Bildschirm löschen',$d,$a
       dc.b '[E] = Eraser  (folgendes Symbol oder Linie wird gelöscht)',$d,$a,$a

       dc.b '[t] = Schaltsymbole Beschriften (mit Zeiger Positionieren)',$d,$a
       dc.b '[!] = Ergänzungseintrage ',$d,$a
       dc.b '[1] = Feld 1 ( Schriftfeld )',$d,$a
       dc.b '[2] = Feld 2        "       ',$d,$a
       dc.b '[3] = Feld 3        "       ',$d,$a,$a

       dc.b '[c] = Hardcopy auf Drucker ',$d,$a
       dc.b '[$] = Bild speichern       ',$d,$a
       dc.b '[%] = Bild ausgeben        ',$d,$a
       dc.b '[^] = Floppy',$d,$a,$a,0
       dc.b 'Auf Diskette schreiben: ueber [ds "Name",1,0,$20000,$23fff]',$d,$a
       dc.b 'Von Diskette lesen: wie Text im Inhaltsverzeichnis anwaehlen',$d,$a
       dc.b $a,0
       dc.b '     [e] = Programmende',0
infob: dc.b '[?] = Anleitung',0

end
```

Ein Softwareskop mit dem 68008

von Rolf-Dieter Klein

Mit dem 68008 Grundprogramm kann man ein recht brauchbares Skop für Niederfrequenzsignale herstellen. Dazu wird eine A/D-Umsetzer-Baugruppe (AD 10/1) benötigt und das neue Grundprogramm 4.3. Im Bild ist das komplette Listing abgedruckt. Das Skop arbeitet dabei mit zwei Bildebenen, so daß ein flimmerfreies Bild entsteht. Es werden dazu zunächst die A/D-Werte in einem Buffer zwischengespeichert. Dann wird eine unsichtbare Bildseite gelöscht, indem die dort dargestellte Skop-Kurve durch Überschreiben gelöscht wird. Dort wird dann auch die neue Kurve eingetragen. Dann werden die Bildseiten gewechselt und das Ganze beginnt von vorne. Man benötigt für diese Darstellung vier Buffer, denn man benötigt zum Löschen die alte Information. Hier wurde das Unterprogramm GETAD10 verwendet, das für die Baugruppe mit dem 10-Bit-AD-Umsetzer gedacht ist. Man kann aber genauso den 8-Bit-AD-Umsetzer verwenden, wenn man das Unterprogramm GETAD8 verwendet. Dann muß man im Register DS0, zuvor noch die Kanalnummer angeben, denn die Baugruppe besitzt 16 Eingangskanäle. Der Befehl ASR.W#2,D0 entfällt beim 8-Bit-Wandler natürlich. Wenn man eine Taste drückt, so wird das letzte Bild eingefroren und auf dem Bildschirm angezeigt.

```
0E9C00                  ***********************
0E9C00                  * A/D UMSETZER SCOP 1.0 *
0E9C00                  ***********************
00A800                  ORG $A800
00A800
00A800                  START:
00A800  4EB9 000E3FEC   JSR $CLRSCREEN   * BILDSCHIRM LOESCHEN
00A806  41F9 0000A90A   LEA BUFFER1,A0   * UND ALLE BUFFER
00A80C  6100 00EC       BSR CLEAR
00A810  41F9 0000AB0A   LEA BUFFER2,A0
00A816  6100 00E2       BSR CLEAR
00A81A  41F9 0000AD0A   LEA BUFFER3,A0
00A820  6100 00D8       BSR CLEAR
00A824  41F9 0000AF0A   LEA BUFFER4,A0
00A82A  6100 00CE       BSR CLEAR
00A82E  4240            CLR D0
00A830  4241            CLR D1
00A832  4EB9 000E1F06   JSR $SETFLIP
00A838  43F9 0000A90A   LEA BUFFER1,A1
00A83E  45F9 0000AB0A   LEA BUFFER2,A2
```

```
00A844  47F9 0000A00A   LEA BUFFER3,A3
00A84A  49F9 0000AF0A   LEA BUFFER4,A4
00A850                  *
00A850  33FC 0000       MOVE #0,VIEW    * MIT ZWEI BILDSEITEN ARBEITEN
00A854  0000A906
00A85B  33FC 0001       MOVE #1,WRT
00A85C  0000A90B
00A860                  *
00A860                  LOOP:
00A860  3239 0000A906   MOVE VIEW,D1
00A866  3039 0000A90B   MOVE WRT,D0
00A86C  4EB9 000E0E0C   JSR $NEWPAGE    * BILDSEITE ANWAEHLEN
00A872  2049            MOVEA.L A1,A0
00A874  6100 0066       BSR FILL        * WERTE EINLESEN
00A878  204B            MOVEA.L A3,A0
00A87A  4EB9 000E0D6E   JSR $ERAPEN
00A880  6100 00D0       BSR AUSGABE     * ALTE INFORMATION LOESCHEN
00A884  2049            MOVEA.L A1,A0
00A886  4EB9 000E0D7C   JSR $SETPEN
00A88C  6100 0024       BSR AUSGABE     * NEUE AUSGEBEN
00A890  C949            EXG A4,A1
00A892  C94A            EXG A4,A2
00A894  C94B            EXG A4,A3
00A896  0A79 0001       EOR #1,VIEW     * DANN NEUE BILDSEITE
00A89A  0000A906
00A89E  0A79 0001       EOR #1,WRT      * ANWAEHLEN
00A8A2  0000A90B
00A8A6  4EB9 000E09DB   JSR $CSTS       * BIS TASTE GEDRUECKT WIRD
00A8AC  6700 FFB2       BEQ LOOP
00A8B0  4E75            RTS
00A8B2
```

```
00A8B2
00A8B2                  AUSGABE:        * WERTE AUF DEN BILDSCHIRM AUSGEBEN
00A8B2  48E7 0080       MOVEM.L A0,-(A7)
00A8B6  4242            CLR D2
00A8B8  1410            MOVE.B (A0),D2
00A8BA  4241            CLR D1
00A8BC  4EB9 000E0EE2   JSR $MOVETO
00A8C2  363C 007F       MOVE #128-1,D3
00A8C6                  LPA:
00A8C6  1418            MOVE.B (A0)+,D2
00A8C8  4EB9 000E0F4C   JSR $DRAWTO
00A8CE  0641 0004       ADD #4,D1

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

00A8D2  51CB FFF2       DBRA D3,LPA
00A8D6  4CDF 0100       MOVEM.L (A7)+,A0
00A8DA  4E75            RTS
00A8DC
00A8DC
00A8DC                  FILL:           * AD-WERTE EINLESEN
00A8DC  48E7 0080       MOVEM.L A0,-(A7)
00A8E0  363C 007F       MOVE #128-1,D3
00A8E4                  LPF:
00A8E4  4240            CLR D0
00A8E6  4EB9 000E4C6B   JSR $GETAD10
00A8EC  E440            ASR.W #2,D0
00A8EE  10C0            MOVE.B D0,(A0)+
00A8F0  51CB FFF2       DBRA D3,LPF
00A8F4  4CDF 0100       MOVEM.L (A7)+,A0
00A8F8  4E75            RTS
```

```
00A8FA
00A8FA                  CLEAR:          * LOESCHEN DER BUFFER
00A8FA  363C 01FF       MOVE #512-1,D3
00A8FE                  LPB:
00A8FE  4218            CLR.B (A0)+
00A900  51CB FFFC       DBRA D3,LPB
00A904  4E75            RTS
00A906
00A906  0000            VIEW: DC.W 0
00A908  0000            WRT: DC.W 0
00A90A
00A90A
00A90A                  BUFFER1: DS.B 512
00A80A                  BUFFER2: DS.B 512
00AD0A                  BUFFER3: DS.B 512
00AF0A                  BUFFER4: DS.B 512
00B10A
00B10A                  END

0E8B4A  Ende-Symboltabelle
```

Das Programm läßt sich natürlich noch beliebig erweitern, wünschenswert wäre die Ausgabe eines Gitters sowie die Möglichkeit, die Zeitbasis zu bestimmen. Wir warten auf Ihre Lösung, die Sie an die *LOOP*-Redaktion senden mögen.

Einführung in C – Teil 5

Rolf-Dieter Klein

Diesmal wollen wir uns mit Dateien beschäftigen.

In C gibt es zwei verschiedene Arten auf Dateien zuzugreifen. Wir wollen uns hier mit den sogenannten Stream-Dateien beschäftigen.

Damit das Ganze nicht zu langweilig wird, soll alles an einem großen Beispiel gezeigt werden. Dazu folgende Aufgabe.

Ein Bild auf der COL256-Baugruppe soll als Datei abgespeichert und wieder geladen werden. Da ein Bild normalerweise 64 KByte braucht, wird zusätzlich ein Kompressionsverfahren verwendet, um mit weniger Platz auf der Diskette auszukommen.

Bild 1 zeigt das Speicher-Programm und Bild 2 das Lade-Programm.

```
 *  Speichern von Bildern auf der Diskette.
    Die Bilder werden dabei in einem kompakten
    Format abgelegt, so dass nur wenig Platz
    benoetigt wird.
    Rolf-Dieter Klein     850729                  */

#include <stdio.h>
EXTERN init();
EXTERN clear();
EXTERN clearcol();
EXTERN draw();
EXTERN setfdot();
EXTERN setdot();
EXTERN xordot();
EXTERN flaeche();

EXTERN getcol();

main(argc,argv)
int argc;
char **argv;
{
 WORD outFile;
 FILE *fdopen();
 FILE *data;
 int x,y,zaehler,alt,wert;
 if (argc != 2) { fprintf(stderr,"Dateiname angeben "); exit(1); }
 if (( outFile = creatb(argv[1],0666)) == NULL  ) {
  fprintf(stderr,"Error: cannot create %s ",argv[1]);
  exit(2);
 }
 init();
 data =  fdopen(outFile,"w");
 printf(" ok wird abgespeichert ");
 zaehler=1; wert = -1; alt = -1;
 for (y=0;y<256;y++) {
  for (x=0;x<256;x++) {
   alt=wert;
   wert=getcol(x,y);    /* Farbwert einlesen */
   if ((wert != alt)||(zaehler >= 255)) {  /* dann alt ausgeben */
    if (alt != -1 ) {  /* falls nicht erster Wert */
     if (zaehler==1) {
      if (alt!=0) {
       putc(alt,data);
      } else {
       putc(0,data); putc(0,data);
      }
     } else if (zaehler==2) {
      if (alt!=0) {
       putc(alt,data); putc(alt,data);
      } else {
       putc(0,data); putc(2,data); putc(0,data);
      }
     } else if (zaehler>2) {
      putc(0,data); putc(zaehler,data); putc(alt,data);
     }
    }
    zaehler = 1;
   } else {
    zaehler++;
   }
  }
 }
 if (alt != -1 ) {  /* falls nicht erster Wert */
  if (zaehler==1) {
   if (alt!=0) {
    putc(alt,data);
   } else {
    putc(0,data); putc(0,data);
   }
  } else if (zaehler==2) {
   if (alt!=0) {

    putc(alt,data); putc(alt,data);
   } else {
    putc(0,data); putc(0,2); putc(0,data);
   }
  } else if (zaehler>2) {
   putc(0,data); putc(zaehler,data); putc(alt,data);
  }
 }
 fclose(data);
 printf(" ok, Ende ");
}
```

Bild 1

Wir wollen uns zunächst das Lade-Programm ansehen, da es einfacher zu verstehen ist.

Die EXTERN-Unterprogramme dienen dem Zugriff auf die COL256, und sind ähnlich denen, die schon in früheren *LOOP*-Ausgaben abgedruckt waren.

Das C-Hauptprogramm beginnt wie üblich mit Variablen-Deklarationen.

Dabei müssen in C auch Funktionen deklariert werden, sofern sie andere Werte als INT als Ergebnis liefern. Die Funktion GETC liest ein Zeichen von einer Datei und liefert dieses als CHAR-Größe ab. Dann brauchen wir eine Variable fi, die als Pointer auf den Datentyp FILE deklariert wird und die Funktion FOPENB, die einen solchen Pointer als Ergebnis liefert.

Der Datentyp FILE ist durch "STDIO.H" automatisch definiert.

FOPENB hat die Aufgabe, eine Datei zu öffnen, das heißt, sie wird auf der Diskette gesucht und, falls vorhanden, eine interne Datenstruktur für den weiteren Zugriff angelegt. In C gibt es *verschiedene solcher* Öffnungs-Funktionen. FOPENB öffnet eine Datei als Binaer-Datei. Das heißt

```
00A844  47F9 0000AD0A    LEA BUFFER3,A3
00A84A  49F9 0000AF0A    LEA BUFFER4,A4
00A850                   *
00A850  33FC 0000        MOVE #0,VIEW    * MIT ZWEI BILDSEITEN ARBEITEN
00A854  0000A906
00A858  33FC 0001        MOVE #1,WRT
00A85C  0000A908
00A860                   *
00A860                  LOOP:
00A860  3239 0000A906    MOVE VIEW,D1
00A866  3039 0000A908    MOVE WRT,D0
00A86C  4EB9 000E0E0C    JSR $NEWPAGE    * BILDSEITE ANWAEHLEN
00A872  2049             MOVEA.L A1,A0
00A874  6100 0066        BSR FILL        * WERTE EINLESEN
00A878  204B             MOVEA.L A3,A0
00A87A  4EB9 000E0D6E    JSR $ERAPEN
00A880  6100 0030        BSR AUSGABE     * ALTE INFORMATION LOESCHEN
00A884  2049             MOVEA.L A1,A0
00A886  4EB9 000E0D7C    JSR $SETPEN
00A88C  6100 0024        BSR AUSGABE     * NEUE AUSGEBEN
00A890  C949             EXG A4,A1
00A892  C94A             EXG A4,A2
00A894  C94B             EXG A4,A3
00A896  0A79 0001        EOR #1,VIEW     * DANN NEUE BILDSEITE
00A89A  0000A906
00A89E  0A79 0001        EOR #1,WRT      * ANWAEHLEN
00A8A2  0000A908
00A8A6  4EB9 000E09DB    JSR $CSTS       * BIS TASTE GEDRUECKT WIRD
00A8AC  6700 FFB2        BEQ LOOP
00A8B0  4E75             RTS
00A8B2
```

```
00A8B2
00A8B2                  AUSGABE:        * WERTE AUF DEN BILDSCHIRM AUSGEBEN
00A8B2  48E7 0080        MOVEM.L A0,-(A7)
00A8B6  4242             CLR D2
00A8B8  1410             MOVE.B (A0),D2
00A8BA  4241             CLR D1
00A8BC  4EB9 000E0EE2    JSR $MOVETO
00A8C2  363C 007F        MOVE #128-1,D3
00A8C6                  LPA:
00A8C6  1418             MOVE.B (A0)+,D2
00A8C8  4EB9 000E0F4C    JSR $DRAWTO
00A8CE  0641 0004        ADD #4,D1

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

00A8D2  51CB FFF2        DBRA D3,LPA
00A8D6  4CDF 0100        MOVEM.L (A7)+,A0
00A8DA  4E75             RTS
00A8DC
00A8DC
00A8DC                  FILL:                   * AD-WERTE EINLESEN
00A8DC  48E7 0080        MOVEM.L A0,-(A7)
00A8E0  363C 007F        MOVE #128-1,D3
00A8E4                  LPP:
00A8E4  4240             CLR D0
00A8E6  4EB9 000E4C68    JSR $GETAD10
00A8EC  E440             ASR.W #2,D0
00A8EE  10C0             MOVE.B D0,(A0)+
00A8F0  51CB FFF2        DBRA D3,LPP
00A8F4  4CDF 0100        MOVEM.L (A7)+,A0
00A8F8  4E75             RTS
```

```
00A8FA
00A8FA                  CLEAR:          * LOESCHEN DER BUFFER
00A8FA  363C 01FF        MOVE #512-1,D3
00A8FE                  LPB:
00A8FE  4218             CLR.B (A0)+
00A900  51CB FFFC        DBRA D3,LPB
00A904  4E75             RTS
00A906
00A906  0000            VIEW: DC.W 0
00A908  0000            WRT: DC.W 0
00A90A
00A90A
00A90A                  BUFFER1: DS.B 512
00A80A                  BUFFER2: DS.B 512
00AD0A                  BUFFER3: DS.B 512
00AF0A                  BUFFER4: DS.B 512
00B10A
00B10A                  END

0EBB4A  Ende-Symboltabelle
```

Das Programm läßt sich natürlich noch beliebig erweitern, wünschenswert wäre die Ausgabe eines Gitters sowie die Möglichkeit, die Zeitbasis zu bestimmen. Wir warten auf Ihre Lösung, die Sie an die *LOOP*-Redaktion senden mögen.

Einführung in C – Teil 5

Rolf-Dieter Klein

Diesmal wollen wir uns mit Dateien beschäftigen.

In C gibt es zwei verschiedene Arten auf Dateien zuzugreifen. Wir wollen uns hier mit den sogenannten Stream-Dateien beschäftigen.

Damit das Ganze nicht zu langweilig wird, soll alles an einem großen Beispiel gezeigt werden. Dazu folgende Aufgabe.

Ein Bild auf der COL256-Baugruppe soll als Datei abgespeichert und wieder geladen werden. Da ein Bild normalerweise 64 KByte braucht, wird zusätzlich ein Kompressionsverfahren verwendet, um mit weniger Platz auf der Diskette auszukommen.

Bild 1 zeigt das Speicher-Programm und Bild 2 das Lade-Programm.

```
 *  Speichern von Bildern auf der Diskette.
    Die Bilder werden dabei in einem kompakten
    Format abgelegt, so dass nur wenig Platz
    benoetigt wird.
    Rolf-Dieter Klein     850729                  */

#include <stdio.h>
EXTERN init();
EXTERN clear();
EXTERN clearcol();
EXTERN draw();
EXTERN setfdot();
EXTERN setdot();
EXTERN xordot();
EXTERN flaeche();

EXTERN getcol();

main(argc,argv)
int argc;
char **argv;
{
 WORD outFile;
 FILE *fdopen();
 FILE *data;
 int x,y,zaehler,alt,wert;
 if (argc != 2) { fprintf(stderr,"Dateiname angeben "); exit(1); }
 if (( outFile = creatb(argv[1],0666)) == NULL  ) {
  fprintf(stderr,"Error: cannot create %s ",argv[1]);
  exit(2);
 }
 init();
 data =  fdopen(outFile,"w");
 printf(" ok wird abgespeichert ");
 zaehler=1; wert = -1; alt = -1;
 for (y=0;y<256;y++) {
  for (x=0;x<256;x++) {
   alt=wert;
   wert=getcol(x,y);    /* Farbwert einlesen */
   if ((wert != alt)||(zaehler >= 255)) {  /* dann alt ausgeben */
    if (alt != -1 ) {  /* falls nicht erster Wert */
     if (zaehler==1) {
      if (alt!=0) {
       putc(alt,data);
      } else {
       putc(0,data); putc(0,data);
      }
     } else if (zaehler==2) {
      if (alt!=0) {
       putc(alt,data); putc(alt,data);
      } else {
       putc(0,data); putc(2,data); putc(0,data);
      }
     } else if (zaehler>2) {
      putc(0,data); putc(zaehler,data); putc(alt,data);
     }
    }
    zaehler = 1;
   } else {
    zaehler++;
   }
  }
 }
 if (alt != -1 ) {  /* falls nicht erster Wert */
  if (zaehler==1) {
   if (alt!=0) {
    putc(alt,data);
   } else {
    putc(0,data); putc(0,data);
   }
  } else if (zaehler==2) {
   if (alt!=0) {
    putc(alt,data); putc(alt,data);
   } else {
    putc(0,data); putc(0,2); putc(0,data);
   }
  } else if (zaehler>2) {
   putc(0,data); putc(zaehler,data); putc(alt,data);
  }
 }
 fclose(data);
 printf(" ok, Ende ");
}
```

Bild 1

Wir wollen uns zunächst das Lade-Programm ansehen, da es einfacher zu verstehen ist.

Die EXTERN-Unterprogramme dienen dem Zugriff auf die COL256, und sind ähnlich denen, die schon in früheren *LOOP*-Ausgaben abgedruckt waren.

Das C-Hauptprogramm beginnt wie üblich mit Variablen-Deklarationen.

Dabei müssen in C auch Funktionen deklariert werden, sofern sie andere Werte als INT als Ergebnis liefern. Die Funktion GETC liest ein Zeichen von einer Datei und liefert dieses als CHAR-Größe ab. Dann brauchen wir eine Variable fi, die als Pointer auf den Datentyp FILE deklariert wird und die Funktion FOPENB, die einen solchen Pointer als Ergebnis liefert.

Der Datentyp FILE ist durch "STDIO.H" automatisch definiert.

FOPENB hat die Aufgabe, eine Datei zu öffnen, das heißt, sie wird auf der Diskette gesucht und, falls vorhanden, eine interne Datenstruktur für den weiteren Zugriff angelegt. In C gibt es *verschiedene solcher* Öffnungs-Funktionen. FOPENB öffnet eine Datei als Binaer-Datei. Das heißt

```
/* Laden von abgespeicherten Bildern.
   Die Bilder werden dabei in einem
   kompakten Format abgelegt, so dass
   nur wenig Platz benoetigt wird.
   Rolf-Dieter Klein 850729          */

#include <stdio.h>
EXTERN init();
EXTERN clear();
EXTERN clearcol();
EXTERN draw();
EXTERN setfdot();
EXTERN setdot();
EXTERN xordot();
EXTERN flaeche();

EXTERN getcol();

main(argc,argv)
int argc;
char **argv;
{
 char getc();
 FILE *fi,*fopenb();
 int i,x,y,zaehler,wert;
 if (argc != 2) { fprintf(stderr,"Dateiname angeben "); exit(1); }
 if (( fi = fopenb(argv[1],"r")) == NULL  ) {
  fprintf(stderr,"Error: cannot open %s ",argv[1]);
  exit(2);
 }
 init();
 clearcol(0x00);
 printf(" ok wird geladen ");
 y=0; x=0;
 do {
  wert=getc(fi);
  if (wert !=0 ) {
   setfdot(x,y,wert);
   x++;
   if (x==256) { x=0; y++; }
  } else {
   zaehler=getc(fi);
   if (zaehler==0) {
    setfdot(x,y,0);
    x++;
    if (x==256) { x=0; y++; }
   } else {
    wert=getc(fi);
    for (i=1;i<=zaehler;i++) {
     setfdot(x,y,wert);
     x++;
     if (x==256) { x=0; y++; }
    }
   }
  }
 } while (y<256);
 fclose(fi);
 printf(" ok, Ende ");
}
```

Bild 2

alle Zeichen werden transparent erreichbar. Die Funktion FOPEN z.B. öffnet Text-Dateien; einige Steuerzeichen z.B. EOF haben eine besondere Bedeutung.

Wir brauchen für dieses Beispiel aber die Binaerdatei, da die Bilddaten direkt binaer abgespeichert bzw. geladen werden sollen.

Eine weitere Besonderheit sind die Angaben ARGC und ARGV. Damit ist es möglich, z.B. Dateinamen direkt aus der Befehlszeile, vom Aufruf des Programms einzulesen. Man kann hier z.B. angeben: *PICLOAD name.pic.* Nach einer Abfrage, ob auch ein solcher Name eingegeben wurde, ist in ARGV[1] der Name als String vorhanden.

Anschließend wird die Datei zum Lesen geöffnet. Dazu wird in FOPENB neben dem Dateinamen "ARGV[1]" auch der String "r" übergeben. Als Ergebnis erhält man den sogenannten File-Descriptor, dessen Adresse in die Variable FI gespeichert wird. Ist dieser Wert = 0, so liegt ein Fehler vor und der *Ladevorgang* wird *abgebrochen* "cannot open . . ."

Nun gehts in die Programmschleife. Zuvor werden noch die Graphik-Unterprogramme mit init() vorbereitet und mit clearcol(0x00) wird der Bildschirm gelöscht.

In der Programmschleife wird nun mit GETC() Zeichen für Zeichen gelesen.

Die Kompression des Bildes arbeitet dabei wie folgt. Wenn das Zeichen mit dem Wert 0 gelesen wird, so enthält das nächste Zeichen einen Zähler, der angibt wievielmal das nachfolgende Zeichen ausgegeben werden soll. Ist der Zähler 0, so wird das Zeichen 0 ausgegeben.

Nach der Hauptschleife wird noch mit FCLOSE(fi) die Datei geschlossen, was aber beim Lesen nicht unbedingt nötig ist.

Nun zum Schreiben.

Zunächst wird vorm Öffnen die Datei mit CREATB erzeugt. Erst danach kann sie mit FDOPEN geöffnet werden. FDOPEN wird verwendet, da dem Programm die Datei durch das Anlegen schon bekannt ist. FDOPEN bezieht sich daher nicht auf den Namen der Datei, sondern auf eine Kennung, die in der Variablen outFile gespeichert wurde. Durch das "w" wird angegeben, daß die Datei zum Schreiben geöffnet wird.

Mit PUTC werden die Daten dann auf Diskette geschrieben. Das Programm ist umfangreicher als das Leseprogramm, da hier die Kompression noch berechnet werden muß.

Am Schluß des Schreibvorgangs wird durch FCLOSE das Dateiende angezeigt und erst dann wird das Inhaltsverzeichnis auf der Diskette auf aktuellen Stand gebracht.

(Fortsetzung folgt)

GES erfolgreich auf der Hannover Messe CEBIT

32 Bit System überzeugt

Eine äußerst erfolgreiche Hannover Messe konnte GES verbuchen. Der Stand in Halle 13 (neben Apple) war immer voll belegt, und viele interessante Gespräche konnten geführt werden.

Besonders angesprochen haben die neuen "Profi"-Baugruppen, wie der 32 Bit Computer 68020 und das ACRTC-Graphiksystem.

Diese Baugruppen werden in der nächsten *LOOP* ausführlich vorgestellt.

Hilfsprogramme für den 68008

Teil 3

von Rüdiger Bäcker

Das dritte Hilfsprogramm für den 68008 dient dem Vergleichen von Speicherbereichen. Damit können beliebige Speicherbereiche auf identische Speicherinhalte geprüft werden. Auch diese Routine ist natürlich wieder relokativ und mit Bibliothekskopf versehen. Es werden die Startadresse, die Endadresse und die Adresse, ab der verglichen werden soll, in HEX eingegeben (Bild 1). Sollte eine Speicherstelle einen vom Ursprungswert abweichenden Inhalt aufweisen, so erscheint eine Fehlermeldung mit Angabe der Adresse, in der der falsche Wert gefunden wurde. Ist alles o.k., so erscheint eine entsprechende Meldung und man gelangt mit 'm' wieder ins Grundprogramm zurück. Zu erwähnen ist noch, daß man auch in der ersten Eingabezeile durch 'm' wieder in das Grundprogramm springen kann. Dies ist auch bei RUBATRANS und RUBADUMP so!

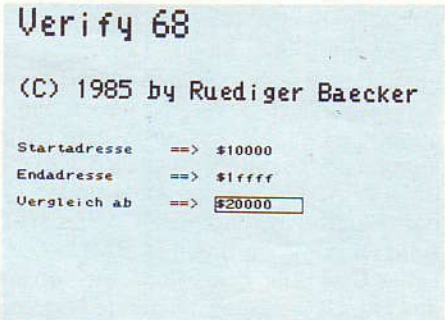


Bild 1 : Die Eingabe der Datenbereiche erfolgt in HEX

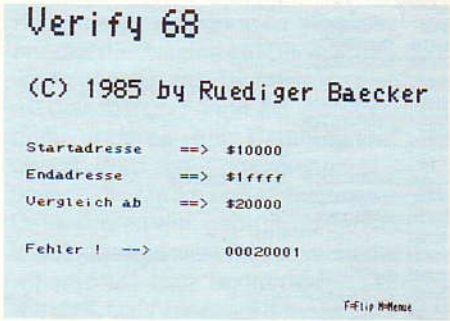


Bild 2 : Vergleichsfehler werden mit Adresse gemeldet !

```
000400                  ORG $400
000400                  *       V E R I F Y  6 8
000400                  *
000400                  *       ROUTINE ZUM VERGLEICHEN VON SPEICHERBEREICHEN
000400                  *
000400                  *       COPYRIGHT (C) 1985 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
000400                  *
000400
000400                  KOPF:
000400  55 AA 01 80     DC.B $55,$AA,$01,$80            * EINTRAG IN BIBLIOTHEK
000404  5645524946592O  DC.B 'VERIFY  '
00040B  20
00040C  00000020        DC.L START-KOPF
000410  000001DB        DC.L ENDEA-KOPF
000414  01              DC.B 1
000415  00 00 00        DC.B 0,0,0
000418  00000000        DC.L 0,0
00041C  00000000
000420
000420
= 000000F4              ZIELADR         EQU     $F4             * ADRESSE EINGABEBUFFER
= FFFFFF6B              TAST            EQU     $FFFFFF68
000420
000420                  START:
000420  3E3C 0011       MOVE #!CLPG,D7                  * BILDSCHIRM LOESCHEN
000424  4E41            TRAP #1
000426
000426  4239 FFFFFF69   CLR.B TAST+1                    * TASTATURPORT ZURUECKSETZEN
00042C  4280            CLR.L D0                        * REGISTER LOESCHEN
00042E  4281            CLR.L D1
000430  4282            CLR.L D2
000432  4283            CLR.L D3
000434
000434  41FA 013F       LEA TEXT3(PC),A0                * TEXTE AUSGEBEN
00043B  103C 0043       MOVE.B #$43,D0
00043C  123C 0000       MOVE.B #$0,D1
000440  143C 00C8       MOVE.B #200,D2
000444  3E3C 000A       MOVE #!WRITE,D7
00044B  4E41            TRAP #1
00044A  0442 002D       SUB #45,D2
00044E  41FA 016B       LEA TEXT7(PC),A0
000452  103C 0032       MOVE.B #$32,D0
000456  3E3C 000A       MOVE #!WRITE,D7
00045A  4E41            TRAP #1
00045C  0442 0028       SUB #40,D2
000460  103C 0021       MOVE.B #$21,D0
000464  41FA 0119       LEA TEXT4(PC),A0
00046B  3E3C 000A       MOVE #!WRITE,D7
00046C  4E41            TRAP #1
00046E
00046E  0442 0014       SUB #20,D2
000472  41FA 011F       LEA TEXT5(PC),A0
000476  3E3C 000A       MOVE #!WRITE,D7
00047A  4E41            TRAP #1
00047C  0442 0014       SUB #20,D2
000480  41FA 0125       LEA TEXT6(PC),A0
000484  3E3C 000A       MOVE #!WRITE,D7
00048B  4E41            TRAP #1
00048A
00048A                  GETW1:                          * DANN DATEN HOLEN (VON, BIS, MIT)
00048A  103C 0021       MOVE.B #$21,D0
00048E  143C 0073       MOVE.B #115,D2
000492  123C 00FA       MOVE.B #250,D1
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

```
000496  163C 0009       MOVE.B #9,D3
00049A  41ED 00F4       LEA ZIELADR(A5),A0
00049E  3E3C 000B       MOVE #!READ,D7
0004A2  4E41            TRAP #1
0004A4  41ED 00F4       LEA ZIELADR(A5),A0
0004A8  1810            MOVE.B (A0),D4
0004AA  0C04 006D       CMP.B #'m',D4                   * 'm' GEDRUECKT ?
0004AE  6700 005E       BEQ VERIEEND                    * JA, DANN ENDE
0004B2  3E3C 001D       MOVE #!WERT,D7
0004B6  4E41            TRAP #1
0004BB  2640            MOVEA.L D0,A3
0004BA
0004BA                  GETW2:
0004BA  143C 005F       MOVE.B #95,D2
0004BE  103C 0021       MOVE.B #$21,D0
0004C2  123C 00FA       MOVE.B #250,D1
0004C6  41ED 00F4       LEA ZIELADR(A5),A0
0004CA  3E3C 000B       MOVE #!READ,D7
0004CE  4E41            TRAP #1
0004D0  41ED 00F4       LEA ZIELADR(A5),A0
0004D4  3E3C 001D       MOVE #!WERT,D7
0004DB  4E41            TRAP #1
0004DA  2840            MOVEA.L D0,A4
0004DC
0004DC                  GETW3:
0004DC  143C 004B       MOVE.B #75,D2
0004E0  103C 0021       MOVE.B #$21,D0
0004E4  123C 00FA       MOVE.B #250,D1
0004EB  41ED 00F4       LEA ZIELADR(A5),A0
0004EC  3E3C 000B       MOVE #!READ,D7
0004F0  4E41            TRAP #1
0004F2  41ED 00F4       LEA ZIELADR(A5),A0
0004F6  3E3C 001D       MOVE #!WERT,D7
0004FA  4E41            TRAP #1
0004FC  2440            MOVEA.L D0,A2
0004FE
0004FE                  VERIFYLP:                       * NUN DATEN VERGLEICHEN
0004FE  B50B            CMPM.B (A3)+,(A2)+              * GLEICH ?
000500  6600 000E       BNE ERROR                       * NEIN DANN FEHLER
000504  B9CB            CMPA.L A3,A4                    * ALLES VERGLICHEN ?
000506  6700 0042       BEQ OK                          * JA, DANN OK-MELDUNG
00050A  6000 FFF2       BRA VERIFYLP                    * SONST WEITER
00050E                  VERIEEND:
00050E  4E75            RTS
000510
000510                  ERROR:
000510  41FA 0050       LEA TEXT1(PC),A0                * FEHLERMELDUNG
000514  103C 0021       MOVE.B #$21,D0
000518  123C 0000       MOVE.B #$0,D1
00051C  143C 0028       MOVE.B #40,D2
000520  3E3C 000A       MOVE #!WRITE,D7
000524  4E41            TRAP #1
000526  200A            MOVE.L A2,D0
000528  41ED 00F4       LEA ZIELADR(A5),A0
00052C  3E3C 002C       MOVE #!PRINTBX,D7
000530  4E41            TRAP #1
000532  41ED 00F4       LEA ZIELADR(A5),A0
000536  103C 0021       MOVE.B #$21,D0
00053A  343C 0028       MOVE #40,D2
00053E  323C 00FA       MOVE #250,D1
000542  3E3C 000A       MOVE #!WRITE,D7                 * ADRESSE DES FALSCHEN WERTES
000546  4E41            TRAP #1
000548  4E75            RTS
00054A
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 3

```
00054A                  OK:
00054A  41FA 0024       LEA TEXT2(PC),A0                * OK-MELDUNG
00054E  103C 0021       MOVE.B #$21,D0
000552  123C 0000       MOVE.B #$0,D1
000556  143C 0028       MOVE.B #40,D2
00055A  3E3C 000A       MOVE #!WRITE,D7
```

```
000556  4E41            TRAP #1
000560  4E75            RTS
000562
000562                  TEXT1:
000562  46656866657220  DC.B 'Fehler ! -->',0
000569  212020202D3E00
000570                  TEXT2:
000570  4F4B202100      DC.B 'OK !',0
000575                  TEXT3:
000575  566572696679 20 DC.B 'Verify 68',0
00057C  363800
00057F                  TEXT4:
00057F  53746172746164  DC.B 'Startadresse    ==>',0
0005B6  72657373652020
0005BD  20203D3D3E00
000593                  TEXT5:
000593  456E64616472 65 DC.B 'Endadresse      ==>',0
00059A  737365202020 20
0005A1  20203D3D3E00
0005A7                  TEXT6:
0005A7  566572676C6569  DC.B 'Vergleich ab    ==>',0
0005AE  636B206162202 0
0005B5  20203D3D3E00
0005BB                  TEXT7:
0005BB  28432920313938  DC.B '(C) 1985 by Ruediger Baecker',0
0005C2  35206279205275
0005C9  65646967657220
0005D0  426165636B6572
0005D7  00
0005D8
0005D8                  ENDEA:
0005DB                  END.

0E8E3E  Ende-Symboltabelle
```

UP-DATE-SERVICE für HEBAS.COM (BASIC-Interpreter CP/M2.2)

Nach Absprache mit der Firma Graf Elektronik GmbH bietet der Autor des Handbuches vom HEBAS-BASIC-Interpreter einen Up-date-Service an. In jeder *LOOP* werden in der Rubrik „TIPS und TRICKS mit HEBAS" Ergänzungen und Verbesserungen gebracht. Wem die notwendigen Änderungen mit einem Debugger zu umständlich oder zu schwierig sind, kann nun seine HEBAS-Version gegen einen geringen Unkostenbetrag in eine neue Version umtauschen. Zusätzlich wird die Datei READ.ME mit den Änderungen für das Handbuch mitgeliefert. Voraussetzung ist der Erwerb von HEBAS bei der Firma Graf oder beim Software-Service des Franzis-Verlages. Folgende Formate stehen z.Z. zur Verfügung: 5,25-Zoll ECMA 70 (40 Spuren), 8-Zoll IBM (einfache und doppelte Dichte), 5,25-Zoll NDR-Computer (80-Spuren). Das Format bitte angeben und den Absender nicht vergessen!

Die alte HEBAS-Version muß an folgende Adresse versandt werden:

Dr. H. Hehl,
Lindenstraße 20, 8059 Wartenberg.

Der Unkostenbetrag beträgt inkl. Mehrwertsteuer 10,– DM. Es gibt folgende Zahlungsmöglichkeiten:

a) Beilage eines 10,– DM Scheines
b) Beilage eines Verrechnungsschecks
c) Einzugsverfahren bei Angabe von Konto-Nr., Bankname und Bankleitzahl

Nach Gutschrift des Betrages erfolgt die Rücksendung der Diskette mit der neuesten HEBAS-Version und den Handbuch-Ergänzungen. So enthält z.B. das neue Handbuch V 1.2 eine Schlüsseltabelle mit den Einsprungstellen aller BASIC-Befehle, so daß der erfahrene Maschinensprache-Programmierer den Interpreter abändern kann.

Utilitys für den 68008 – Teil 4

Umlenkung der CO2-Routine ins RAM mit dem Programm RAM-USR

Beim 68008 Grundprogramm erfolgt die Ausgabe von Zeichen über die Routine CO2. Diese Routine weist als Besonderheit auf, daß man programmieren kann, ob die Ausgabe auf den Bildschirm (CRT) oder den Drucker (LST) geht oder unterdrückt wird (NIL). Desweiteren kann man die Routine auf eine Benutzerroutine umlenken (USR). In dieser Benutzerroutine kann man dann die im Register D0 stehenden Zeichen frei weiter bearbeiten. Eine mögliche Anwendung ist zum Beispiel der Anschluß eines seriellen Druckers. Hier könnte man die Ausgabe auf USR umschalten und in einer eigenen Routine die Grundprogrammroutine SO anspringen, die dann ihrerseits das Zeichen auf die serielle Schnittstelle gibt. Konkreter Anlaß, diese Routine zu entwickeln war bei mir der Wunsch, die Ausgabe eines recht verbreiteten Disassemblers ins RAM zu verlegen, um anschließend mit dem Editor Änderungen am disassemblierten Programm durchführen zu können bzw. um Kommentare einzufügen. Zunächst jedoch noch etwas zur grundsätzlichen Funktion der CO2-Routine.

Beim Aufruf der Routine wird die Speicherstelle IOSTAT im Arbeitsspeicher des Grundprogrammes abgefragt. In dieser Speicherstelle steht für die Ausgabearten ein Code. Entsprechend dieses Codes wird dann der jeweilige Ausgabekanal angewählt. Die Umschaltung auf einen anderen Ausgabekanal geschieht also durch ändern des Codes in IOSTAT. Hierzu gibt es im Grundprogramm entsprechende Routinen. Zum Umschalten auf den Benutzerkanal dient die Routine USR. Die Umschaltung muß vor dem Aufruf des Programmes geschehen, dessen Ausgabe umgelenkt werden soll. Die Umschaltung erledigt in unserem Beispiel die Routine USERINIT. Die CO2-Routine springt im USR-Modus immer die Speicherstelle $8024 an. Will man nun eine eigene Routine durchlaufen, so muß

```
Programm starten

Adr:userinit
```

Bild 1 : USERINIT wird initialisiert

```
Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

---- 0 1 2 3 4 5 6 7 8 9 A B C D E F
00030000 >52 6F 6C 66 20 4C 6F 62 72 65 79 65 72 20 20 36
00030000  R  o  l  f    L  o  b  r  e  y  e  r     6
00030010 38 30 30 30 2F 36 38 30 30 38 20 20 44 69 73 61
00030010  8  0  0  0  /  6  8  0  0  8     D  i  s  a
00030020 73 73 65 6D 62 6C 65 72 20 20 32 2E 31 20 20 20
00030020  s  s  e  m  b  l  e  r     2  .  1
00030030 28 63 29 20 31 39 38 34 0A 0D 0A 0D 0A 0D 30 30
00030030  (  c  )    1  9  8  4                    0  0
00030040 30 34 30 30 20 20 33 42 37 43 20 34 45 46 39 20
00030040  0  4  0  0    3  B  7  C    4  E  F  9
00030050 20 20 20 20 20 4D 4F 56 45 2E 57 20 20 20 20 23
00030050                 M  O  V  E  .  W                 #
00030060 24 34 45 46 39 2C 24 30 30 32 34 28 41 35 29 0A
00030060  $  4  E  F  9  ,  $  0  0  2  4  (  A  5  )
00030070 0D 30 30 30 34 30 34 20 20 30 30 32 34 20 20 20
00030070    0  0  0  4  0  4     0  0  2  4
```

Bild 2 : So sieht der Speicher nach dem Aufruf der umzulenkenden Routine aus ($30038 = $0A $0D)

```
Programm starten

Adr:modify
```

Bild 3 : Nun modifizieren

```
Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

---- 0 1 2 3 4 5 6 7 8 9 A B C D E F
00030000 >52 6F 6C 66 20 4C 6F 62 72 65 79 65 72 20 20 36
00030000  R  o  l  f    L  o  b  r  e  y  e  r     6
00030010 38 30 30 30 2F 36 38 30 30 38 20 20 44 69 73 61
00030010  8  0  0  0  /  6  8  0  0  8     D  i  s  a
00030020 73 73 65 6D 62 6C 65 72 20 20 32 2E 31 20 20 20
00030020  s  s  e  m  b  l  e  r     2  .  1
00030030 28 63 29 20 31 39 38 34 0D 0A 0D 0A 0D 0A 30 30
00030030  (  c  )    1  9  8  4                    0  0
00030040 30 34 30 30 20 20 33 42 37 43 20 34 45 46 39 20
00030040  0  4  0  0    3  B  7  C    4  E  F  9
00030050 20 20 20 20 20 4D 4F 56 45 2E 57 20 20 20 20 23
00030050                 M  O  V  E  .  W                 #
00030060 24 34 45 46 39 2C 24 30 30 32 34 28 41 35 29 0D
00030060  $  4  E  F  9  ,  $  0  0  2  4  (  A  5  )
00030070 0A 30 30 30 34 30 34 20 20 30 30 32 34 20 20 20
00030070    0  0  0  4  0  4     0  0  2  4
```

Bild 4 : Und siehe da ....

```
Rolf Lobreyer  68000/68008  Disassembler  2.1   (c) 1984

000400  3B7C 4EF9      MOVE.W   #$4EF9,$0024(A5)
000404  0024
000406  2B7C 00000420  MOVE.L   #$00000420,$0026(A5)
00040A  0026
00040E  3E3C 0033      MOVE.W   #$0033,D7
000412  4E41           TRAP     #1
000414  23FC 00030000  MOVE.L   #$00030000,$0000048C
00041A  0000048C
00041E  4E75           RTS
000420  48E7 0004      MOVEM.L  A5,-(A7)
000424  4CF9 2000      MOVEM.L  $0000048C,D2-D3/D5/A2
000428  0000048C
00042C  1AC0           MOVE.B   D0,(A5)+
00042E  1ABC 0000      MOVE.B   #$0000,(A5)
000432  48F9 2000      MOVEM.L  A5,$0000048C
000436  0000048C
00043A  4CDF 2000      MOVEM.L  (A7)+,A5
00043E  4E75           RTS
000440  41F9 00030000  LEA      $00030000,A0
000446  1018           MOVE.B   (A0)+,D0
000448  0C00 0000      CMPI.B   #$00,D0
Textstart=030000 Fenster=030000 Tor=0308ED      amer CTRL-J=Hilfe
```

Bild 5 : Die Ausgabe im Editor

in die Speicherstelle $8024 ein Sprung auf diese Routine eingetragen werden. Dies geschieht in unserem Beipiel am Anfang der Routine USERINIT. $4EF9 ist der sedezimale Code für einen Sprung, User ist die symbolische Adresse der Benutzerroutine. Nun muß noch die Routine USR zum Umschalten aufgerufen werden und die Ablageadresse für den Text irgendwo zwischengespeichert werden. Da man nicht vorher sagen kann, welche Register in der umzulenkenden Routine benutzt werden, kann man die Adresse nicht in einem Register ablegen, sondern muß sie im Ram zwischenspeichern. Wir tun dies in der Speicherstelle MERKER. Damit ist die Initialisierung abgeschlossen.

Die Routine USER lädt die Ablageadresse dann in das Adressregister A5, dessen Ursprungswert vorher auf den Stack gelegt wurde. Nun wird das in D0.B befindliche Zeichen in die Speicherstelle eingeschrieben, auf die A5 zeigt und das Register A5 erhöht, so daß es auf die nächste Speicherstelle zeigt. Da wir die Ausgabe anschließend mit dem Editor bearbeiten wollen, müssen wir diesem mitteilen, wo unser Text endet. Der Editor erkennt das Textende daran, daß in der ersten Speicherstelle hinter dem Text der Wert $0 steht. Daher schreiben wir in die Stelle, auf die A5 nach der Erhöhung zeigt, eine 0. Der aktuelle Wert von A5 wird dann wieder im 'Merker' abgelegt und der Ursprungswert vom Stack zurückgeholt. Damit ist dann auch unsere „Umlenkroutine" beendet.

Damit steht der Text also im Ram und könnte im Grunde genommen mit dem Editor bearbeitet werden. Also rufen wir den Editor auf. Wer nun meint, alles wäre o.k., der sieht sich unter Umständen an dieser Stelle getäuscht. Der Editor erwartet am Ende einer Zeile die Zeichenfolge $D $A. Es kann vorkommen, daß ein Programm am Ende einer Zeile für den Zeilensprung die Kombination $A $D verwendet. Dies hat zwar im Grunde genommen die gleiche Wirkung, bringt den Editor jedoch durcheinander. Um die richtige Reihenfolge zu erhalten muß noch eine Routine her, die prüft, ob die Reihenfolge stimmt und ggf. bei falscher Reihenfolge korrigierend eingreift. Dies erledigt die Routine MODIFY für uns. Hier wird zunächst der Textstart ins Register A0 geladen und dann ein Zeichen nach dem anderen ins Datenregister D0 geholt. Wird das Endezeichen erkannt, so wird die Marke Ende1 angesprungen. Desweiteren wird geprüft, ob die Reihenfolge schon stimmt. Ist das nicht der Fall, wo wird bis zum Textende alles umgewandelt. Die Ablageadresse für den Text ist hier $20000, es kann natürlich durch ändern der Variable TEXT eine andere Adresse gewählt werden.

Zum Schluß nochmal eine Zusammenfassung des Bedienungsablaufes:

- Programm laden
- assemblieren
- Routine USERINIT starten
- Programm, dessen Ausgabe umgelenkt werden soll ausführen
- Routine MODIFY aufrufen

Danach kann der Text mit dem Editor bearbeitet werden.

```
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

000400                 ORG $400
000400
000400                 *      R A M - U S R
000400                 *
000400                 *      ROUTINE ZUM UMLENKEN DER CO2-ROUTINE INS RAM
000400                 *      MIT ANPASSUNG DER ZEICHENFOLGE FUER EDITOR
000400                 *
000400                 *      COPYRIGHT (C) 1985 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
000400
= 00030000             TEXT   EQU $30000              * ABLAGEADRESSE FUER TEXT AUS CO2-ROUTINE
000400
000400                 USERINIT:                      * RELEGT IOSTAT MIT STARTADRESSE DER ROUTINE
000400  3B7C 4EF9 0024 MOVE.W #$4EF9,$24(A5)           * SEDDEZIMALER CODE FUER SPRUNGBEFEHL
000406  2B7C 00000420  MOVE.L #USER,$24+2(A5)          * ADRESSE DER USER-ROUTINE
00040C  0026
00040E  3E3C 0033      MOVE #!USR,D7                  * UMSCHALTEN AUF USR-AUSGABE
000412  4E41           TRAP #1
000414  23FC 00030000  MOVE.L #TEXT,MERKER            * ABLAGEADRESSE IN MERKER
00041A  0000048C
00041E  4E75           RTS
000420
000420                 USER:                          * HIER EIGENTLICHE USR-ROUTINE
000420  48E7 0004      MOVEM.L A5,-(A7)               * A5 RETTEN, DA VOM GRPRG. BENUTZT
000424  4CF9 2000      MOVEM.L MERKER,A5              * DANN ABLAGEADRESSE HOLEN
00042B  0000048C
00042C  1AC0           MOVE.B D0,(A5)+                * WERT AUS D0 DORTHIN
00042E  1ABC 0000      MOVE.B #$0,(A5)                * ENDEZEICHEN DAHINTER
000432  48F9 2000      MOVEM.L A5,MERKER              * ABLAGEADRESSE MERKEN
000436  0000048C
00043A  4CDF 2000      MOVEM.L (A7)+,A5               * UND A5 WIEDER VOM STACK HOLEN
00043E  4E75           RTS
000440
000440                 MODIFY:                        * HIER ANPASSUNG VON ZEILENSCHALTUNG
000440  41F9 00030000  LEA TEXT,A0                    * ABLAGEADRESSE HOLEN
000446
000446                 MODIFLP:                       * WANDLEROUTINE
000446  1018           MOVE.B (A0)+,D0                * ZEICHEN HOLEN
00044B  0C00 0000      CMP.B #$0,D0                   * ENDEZEICHEN ?
00044C  6700 000E      BEQ ENDE1                      * JA, DANN ENDE
000450  0C00 000A      CMP.B #$A,D0                   * LF ?
000454  6700 001A      BEQ WANDCR                     * JA, DANN WANDEL
000458  6000 FFEC      BRA MODIFLP                    * SONST WEITER
00045C
00045C                 ENDE1:                         * BEENDET WANDLEROUTINE
00045C  203C 00030000  MOVE.L #TEXT,D0                * TEXTSTART FUER EDITOR AUF ABLAGEADRESSE
000462  3E3C 0043      MOVE #!PUTSTX,D7               * SETZEN
000466  4E41           TRAP #1
00046B  3E3C 0034      MOVE #!NIL,D7                  * UND AUF 'Nur Fehler'-AUSGABE UMSCHALTEN
00046C  4E41           TRAP #1
00046E  4E75           RTS
000470
000470                 WANDCR:                        * ZEICHENFOLGE $A,$D IN $D,$A WANDELN
000470  1010           MOVE.B (A0),D0                 * ZEICHEN HOLEN
000472  0C00 000D      CMP.B #$D,D0                   * MUSS $D SEIN
000476  6600 FFE4      BNE ENDE1                      * SONST ENDE
00047A  91FC 00000001  SUBA.L #1,A0                   * AUF VORHERIGE SPEICHERSTELLE ZEIGEN
0004B0  10FC 000D      MOVE.B #$D,(A0)+               * DANN $D,$A SETZEN
0004B4  10FC 000A      MOVE.B #$A,(A0)+
0004B8  6000 FFBC      BRA MODIFLP                    * UND WEITER
0004BC
0004BC                 MERKER: DS.L 1
000490
000490                 END.

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

0E8ADE  Ende-Symboltabelle
0109E6  Ende-Debug-Tabelle
```

Rubafill 68

von Rüdiger Baecker

Die Reihe der Hilfsprogramme für den 68008 wird heute mit einer Routine zum Füllen von Speicherbereichen fortgesetzt. Diese Routine mit dem Namen RUBAFILL ermöglicht das Beschreiben beliebiger RAM-Bereiche mit einem konstanten Wert. Auch diese Routine kann wieder aus der Bibliothek aufgerufen werden und ist natürlich relokativ.

Wie alle bisherigen Programme aus dieser Reihe werden die einzugebenden Daten vom Programm in HEX erwartet. In der ersten Zeile gelangt man durch drükken von 'm' wieder ins Grundprogramm zurück.

Die Leser, die schon die anderen Listings abgetippt haben, werden sicher schon festgestellt haben, daß einige Teile der Programme stets identisch sind, man kann sich hier also Tipparbeit sparen, indem man nur die veränderten Teile neu eingibt.

```
Rubafill 68

(C) 1985 by Ruediger Baecker

Startadresse    ==>  $20000
Endadresse      ==>  $25000
Fuellwert       ==>  $A0
```

```
Rubadump (C) 1985 by Ruediger Baecker Do 26.12.85 - 18:30:30
00020000 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020010 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020020 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020030 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020040 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020050 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020060 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020070 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020080 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020090 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
000200A0 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
000200B0 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
000200C0 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
000200D0 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
000200E0 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
000200F0 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020100 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020110 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020120 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020130 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020140 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020150 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ................
00020160 --> AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD  ...........
```

Bild 2 - Der beschriebene Speicherbereich ...

```
Do 26.12.85 - 18:24:36
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

000400                  ORG $400
000400
000400                  *       R U B A F I L L  6 8
000400                  *
000400                  *       ROUTINE ZUM FUELLEN VON SPEICHERBEREICHEN
000400                  *
000400                  *       COPYRIGHT (C) 1986 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
000400
000400                  KOPF:
000400  55AA01B0        DC.L $55AA01B0                  * EINTRAG IN BIBLIOTHEK
000404  5275626166696C  DC.B 'Rubafill'
00040B  6C
00040C  00000020        DC.L RUBAFILL-KOPF
000410  0000010A        DC.L FILLEND-KOPF
000414  01              DC.B 1
000415  00 00 00        DC.B 0,0,0
00041B  00000000        DC.L 0,0
00041C  00000000
000420
= 000009DA              ZIELADR         EQU     $9DA            * ADRESSE EINGABEBUFFER
= FFFFFF6B              TAST            EQU     $FFFFFF6B
000420
000420                  RUBAFILL:
000420  3E3C 0011       MOVE #!CLPG,D7                  * BILDSCHIRM LOESCHEN
000424  4E41            TRAP #1
000426
000426  4239 FFFFFF69   CLR.B TAST+1                    * TATATURPORT ZURUECKSETZEN
00042C  4280            CLR.L D0                        * REGISTER LOESCHEN
00042E  4281            CLR.L D1
000430  4282            CLR.L D2
000432  4283            CLR.L D3
000434
000434  41FA 00D4       LEA TEXT9 (PC),A0               * TEXTE AUSGEBEN
00043B  103C 0043       MOVE.B #$43,D0
00043C  123C 0000       MOVE.B #$0,D1
000440  143C 00CB       MOVE.B #200,D2
000444  3E3C 000A       MOVE #!WRITE,D7
000448  4E41            TRAP #1
00044A  0442 002D       SUB #45,D2
00044E  41FA 0102       LEA OT2(PC),A0
000452  103C 0032       MOVE.B #$32,D0
000456  3E3C 000A       MOVE #!WRITE,D7
00045A  4E41            TRAP #1
00045C  0442 0028       SUB #40,D2
000460  103C 0021       MOVE.B #$21,D0
000464  41FA 00B0       LEA TEXT10(PC),A0
00046B  3E3C 000A       MOVE #!WRITE,D7
00046C  4E41            TRAP #1
00046E
00046E  0442 0014       SUB #20,D2
000472  41FA 00B6       LEA TEXT11(PC),A0
000476  3E3C 000A       MOVE #!WRITE,D7
00047A  4E41            TRAP #1
00047C  0442 0014       SUB #20,D2
000480  41FA 00BC       LEA TEXT12(PC),A0
000484  3E3C 000A       MOVE #!WRITE,D7
00048B  4E41            TRAP #1
00048A
00048A                  GETW1:                          * DANN DATEN HOLEN (VON, BIS, MIT)
00048A  103C 0021       MOVE.B #$21,D0
00048E  143C 0073       MOVE.B #115,D2
000492  123C 00FA       MOVE.B #250,D1
000496  163C 0009       MOVE.B #9,D3


Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

00049A  41ED 09DA       LEA ZIELADR(A5),A0
00049E  3E3C 000B       MOVE #!READ,D7
0004A2  4E41            TRAP #1
0004A4  41ED 09DA       LEA ZIELADR(A5),A0
0004A8  1810            MOVE.B (A0),D4
0004AA  0C04 006D       CMP.B #'m',D4                   * 'm' GEDRUECKT ?
0004AE  6700 005B       BEQ ENDE                        * JA, DANN ENDE
0004B2  3E3C 001D       MOVE #!WERT,D7
0004B6  4E41            TRAP #1
0004B8  2640            MOVEA.L D0,A3                   * VON IN A3
0004BA
0004BA                  GETW2:
0004BA  143C 005F       MOVE.B #95,D2
0004BE  103C 0021       MOVE.B #$21,D0
0004C2  123C 00FA       MOVE.B #250,D1
0004C6  41ED 09DA       LEA ZIELADR(A5),A0
0004CA  3E3C 000B       MOVE #!READ,D7
0004CE  4E41            TRAP #1
0004D0  41ED 09DA       LEA ZIELADR(A5),A0
0004D4  3E3C 001D       MOVE #!WERT,D7
0004DB  4E41            TRAP #1
0004DA  2840            MOVEA.L D0,A4                   * BIS IN A4
0004DC
0004DC                  GETW3:
0004DC  143C 004B       MOVE.B #75,D2
0004E0  103C 0021       MOVE.B #$21,D0
0004E4  123C 00FA       MOVE.B #250,D1
0004E8  41ED 09DA       LEA ZIELADR(A5),A0
0004EC  3E3C 000B       MOVE #!READ,D7
0004F0  4E41            TRAP #1
0004F2  41ED 09DA       LEA ZIELADR(A5),A0
0004F6  3E3C 001D       MOVE #!WERT,D7
0004FA  4E41            TRAP #1                         * FILLWERT IN D0
0004FC
0004FC                  FILL:                           * NUN SPEICHER FUELLEN
0004FC  16C0            MOVE.B D0,(A3)+
0004FE  B9CB            CMPA.L A3,A4
000500  6700 0006       BEQ ENDE
000504  6000 FFF6       BRA FILL
00050B
00050B                  ENDE:
00050B  4E75            RTS
00050A                  FILLEND:                        * MERKER FUER BIBLIOTHEKSEINTRAG
00050A
00050A
00050A                  TEXT9:
00050A  5275626166696C  DC.B 'Rubafill 68',0
000511  6C20363800
000516                  TEXT10:
000516  53746172746164  DC.B 'Startadresse    ==>',0
00051D  726573736520202
000524  20203D3D3E00
00052A                  TEXT11:
00052A  456E6461647265  DC.B 'Endadresse      ==>',0
000531  73736520202020
00053B  20203D3D3E00
00053E                  TEXT12:
00053E  4675656C6C7765  DC.B 'Fuellwert       ==>',0
000545  72742020202020
00054C  20203D3D3E00
000552                  OT2:
000552  2843292031393B  DC.B '(C) 1985 by Ruediger Baecker',0
000559  35206279205275
000560  6564696765722C

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 3

000567  426165636B6572
00056E  00
00056F

0E8B5C  Ende-Symboltabelle
```

TIPS + TRICKS — TIPS + TRICKS

FLOBOOT 68

von Axel Granel

Es gibt viele NDR-Computer-Benutzer, die nur noch mit CP/M68K arbeiten. Für diese ist es sehr lästig, nach dem Einschalten des Rechners und nach jedem RESET im Grundprogramm bis zum Menue 'Floppy Start' zu gehen, um dann endlich CP/M68K booten zu können.

Mit dem unten abgebildeten Programm wird nun nach dem Einschalten und nach jedem RESET automatisch der Floppy-Start aufgerufen. Die Initialisierungen sind nötig, da CP/M68K auf viele Grundprogramm-Routinen zugreift.

Das Programm ist auf eine ganz spezielle Hardware-Umgebung zugeschnitten, wenn Ihr Computer anders aufgerüstet ist, müssen Sie die entsprechenden Stellen ändern. Achtung: es wird keine Fehlerbehandlung durchgeführt, der Rechner sollte also vorher schon einwandfrei funktionieren, wie z.B. mit EBOOT68

```
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

0E9C00                  * FLOBOOT68
0E9C00                  * FUNKTIONIERT NUR MIT EASSO-3 V4.3 !!!
0E9C00                  * 02.02.86   V1.2        AUTOR: AXEL GRANEL, KEMPTEN
0E9C00
= 00000001              CPU     EQU     1                       * 1=68008, 2=68000
= 000E0000              GRUND   EQU     $E0000                  * EASSO-3 STECKEN AUF
0E9C00                                                          *  BANK E

= 00000004              RAM     EQU     4                       * 4 8KBYTE-RAMS
0E9C00                                                          * STECKEN HINTER
0E9C00                                                          * DEM GRUNDPROGRAMM
0E9C00
0E9C00                          OFFSET  $C000                   * PROGRAMM HIER
0E9C00                                                          * ABLEGEN (RAM !???)
000000                          ORG     0                       * ABER CODE FUER $0000

000000                                                          * $0000 ERZEUGEN
000000
000000  00008FFE                DC.L    $8FFE*CPU               * DUMMY STACK IM RAM
000004  00000008                DC.L    START
000008
000008                  START:
000008  363C 003F               MOVE    #ENDE-ANF-1,D3
00000C  41F9 00008000           LEA     $8000*CPU,A0
000012  43FA 000E               LEA     ANF(PC),A1
000016                  TRANSPORT:
000016  10D9                    MOVE.B  (A1)+,(A0)+
00001A  51CB FFFC               DBRA    D3,TRANSPORT
00001C  4EF9 00008000           JMP     $8000*CPU
000022
000022                  ANF:
000022                  BOOT:
000022  13FC 0080               MOVE.B #$80,$FFFFFFCB*CPU       * BANKBOOT ABSCHALTEN
000026  FFFFFFCB
00002A  4A39 FFFFFF69           TST.B $FFFFFF69*CPU             * KEY RESET
000030  4BF9 000E8000           LEA GRUND+$8000,A5
000036  4FF9 000EFFFE           LEA GRUND+$8000+$2000*RAM-2,A7  * RICHTIGEN STACK
00003C                                                          *  FESTLEGEN
00003C  4EB9 000E2FE6           JSR GRUND+$2FE6                 * INIT
000042  13FC 0001               MOVE.B #1,$FFFFFF49*CPU         * LOINIT
000046  FFFFFF49
00004A  4EB9 000E0A68           JSR GRUND+$0A68                 * CIINIT2
000050  4EB9 000E0E14           JSR GRUND+$0E14                 * AKTPAGE
000056  4EB9 000E0DA0           JSR GRUND+$0DA0                 * CLRALL
00005C
00005C                  *       HIER EVTL. WEITEREN INITIALISIERUNGS-CODE EINFUEGEN !
00005C
00005C  4EF9 000E374E           JMP GRUND+$374E                 * FLOPPY-BOOT
000062
000062                  DS 0
000062                  ENDE:
000062                          END

0E8ADE  Ende-Symboltabelle
```

nachgeprüft werden kann. Bitte schreiben Sie uns, wenn das Programm auf Ihrem Rechner *nicht* funktionieren sollte, damit wir es verbessern können.

Dieses Programm ersetzt das EPROM EBOOT68, man benötigt also einen PROMMER, um das übersetzte Programm in ein EPROM zu programmieren.

Graphik mit FLOMON — Teil 1

Rolf-Dieter Klein

Vielfach wurde schon der Wunsch an mich herangetragen, einmal etwas mehr über Graphik mit dem FLOMON zu berichten.

Hier soll nun dieser Wunsch erfüllt werden.

FLOMON besitzt die Möglichkeit, komplexe Graphik-Befehle von jeder Programmiersprache aus anzusprechen. Damit dies möglich wird, bedient man sich eines Tricks. Alle Programmiersprachen können Zeichen auf den Bildschirm ausgeben. Damit ist also eine gemeinsame Grundlage geschaffen. Nun muß man nur nach einem Zeichen suchen, das für die normale Textausgabe nicht gebraucht wird – mit diesem Zeichen schaltet man dann auf Graphik um. Alle nachfolgenden Zeichen werden dann nicht mehr als Text auf den Bildschirm ausgegeben, sondern als Befehle interpretiert. Dies geschieht so lange, bis ein weiteres Zeichen das Ganze wieder auf den Normalstand zurückschaltet.

Bei FLOMON besteht „das Zeichen" aus mehreren einzelnen Zeichen, nämlich:

ESC ESC G

ESC ist dabei die Abkürzung für ein Sonderzeichen mit dem Namen Escape. Dieses Sonderzeichen läßt sich nicht als Text darstellen, und besitzt die Codierung Dezimal 27 oder Hex: 1B. Es wurde eine solch komplizierte Sequenz gewählt, damit eine andere Möglichkeit, die FLOMON bietet, nicht verbaut wird. FLOMON versteht nämlich die Sonderzeichensequenz des Terminals TELEVIDEO 950. Diese Sonderzeichen kann man z.B. verwenden, um den Cursor zu steuern, oder den Bildschirm zu löschen, doch

```
Fertig
LIST

10 PRINT CHR$(27);CHR$(27);"GZY0;P0"
20 FOR X = 0 TO 510 STEP 20
30 FOR Y = 0 TO 250 STEP 10
40 PRINT "M";X;Y;";";"D";X;Y
50 NEXT Y
60 NEXT X
65 PRINT "M";0;128
70 FOR T=0 TO 511
80 F = INT(SIN(T/511*3.141592653*10) * 120) + 128
90 PRINT "D";T;F
100 NEXT T
110 PRINT"A"
120 GOTO 120  : REM Warten auf CTRL-E

Fertig
```

Bild 1

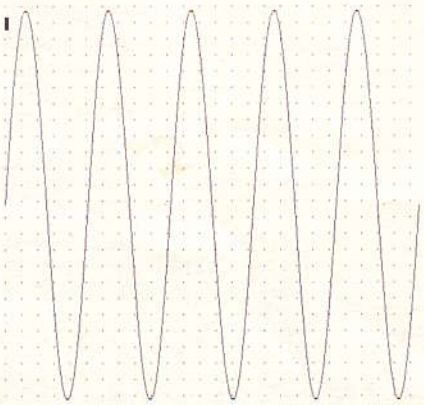


Bild 2

das ist wieder ein anderes Thema. Übrigens enthalten die meisten CP/M- Programme, wie TURBO-PASCAL, DBASEII, WORDSTAR usw. das Terminal 950 (oder 912, 925) beim Installationsmenue, und man kann es dann angeben.

Nachdem die Sequenz ESC ESC G gegeben wurde, versteht FLOMON nur noch Graphik-Befehle. Um den Mode wieder zu verlassen, gibt man einfach das Zeichen A oder die Sequenz ESC ESC A aus.

Das erste einfache Programm soll ein Punktraster auf den Bildschirm malen und anschließend eine SIN-Funktion darstellen. Dazu benötigt man nur zwei einfache, aber sehr universelle Graphik-Befehle: Moveto und Drawto. Moveto bedeutet bewegen und wird von FLOMON durch M abgekürzt verstanden. Drawto heißt „zeichne dorthin" und wird mit D abgekürzt. Beide Befehle benötigen noch eine Koordinatenangabe, nämlich die x- und y-Koordinate. Die x-Koordinate besitzt einen Wertebereich von 0 . . . 511 und die x-Koordinate reicht von 0 . . . 255. Der Punkt 0,0 liegt links unten. Jeder Befehl kann durch einen Strichpunkt ";" oder durch das Zeilenende abgeschlossen werden. Bild 1 zeigt das Programmbeispiel in BASIC.

In Zeile 10 erfolgt die Umschaltung in den Graphik-Mode. Hinter dem "G" stehen noch weitere Befehle. "Z" bedeutet Bildschirm löschen, "Y0" bedeutet Bildseitenumschaltung ausschalten, denn wir brauchen nur eine Bildseite. Mit "P0" wird die Bildseite 0 eingestellt. Die Befehle "G" und "Z" müssen nicht durch Strichpunkte getrennt werden, da sie keine Parameter besitzen, wohl aber der nachfolgende Befehl Y0 von P0, da Y0 den Wert 0 als Parameter besitzt.

Die Ausgabe der Gitterpunkte erfolgt in Zeile 40. Mit dem "M"-Befehl wird positioniert und der "D"-Befehl zeichnet eine Verbindlungslinie. Hier ist allerdings die Koordinate von Drawto mit Moveto iden-

tisch und daher wird auch nur ein Punkt gezeichnet.

Die Funktion wird in Zeile 80 berechnet und in Zeile 90 ausgegeben.

In Zeile 65 wird zunächst der Startpunkt festgelegt und in Zeile 90 jeweils die Verbindungslinie neuen Koordinate gezeichnet.

In Zeile 110 wird der Graphik-Mode wieder ausgeschaltet (Achtung, dann erscheint der Cursor im Bild). Damit nun keine Fertig-Meldung auf den Bildschirm ausgegeben wird, ist in Zeile 120 eine Endlos-Schleife programmiert. Sie kann man durch Eingabe CTRL-E beenden. Bild 2 zeigt das entstandene Bild, bevor man abbricht. Wen der Cursor stört, der kann auch mit P5 auf der Bildseite 1 ausgeben, da der Cursor nur auf Seite 0 erscheint. Allerdings ist dann auch nach Abbruch kein Cursor mehr sichtbar. Ihn erhält man wieder, wenn man den Befehl PRINT CHR$(26) eingibt und damit den Bildschirm löscht und alle Modi wieder auf Normal stellt.

Die genannten Befehle, die FLOMON im Graphik-Modul versteht, sind im Sonderheft „Mikrocomputer Schritt für Schritt" abgedruckt. (Fortsetzung folgt)

EXD-Schreibprinter am NDR-Computer

von Werner Erhard

Auf der Suche nach einer preiswerten Kombination zwischen Schreibmaschine und Drucker für den NDR-Computer, bin ich auf den EXD-10 Schreibprinter von der Fa. Görlitz GmbH, Postfach 852, 5400 Koblenz, Telefon (0261) 2044, gestoßen (DM 398,– inkl. MWSt.). Dieser Drucker hat einen Anschluß für eine serielle Schnittstelle.

Um auch die serielle Schnittstelle einzusparen, paßte ich über beiliegende Schaltung die IOE-Karte an den Drucker an. Das bietet dann den großen Vorteil, daß man z.B. das fehlende ä usw. per Software leicht anpassen kann. Auch Sondertasten können so elegant programmiert werden.

Noch ein Hinweis für diejenigen, die das Programm erst ausprobieren und dann die Bedienungsanleitung des Schreibprinters durchlesen: Verwendet man im Programm die TAB-Taste, so muß man den Tabulator am Schreibprinter vorher setzen!!!

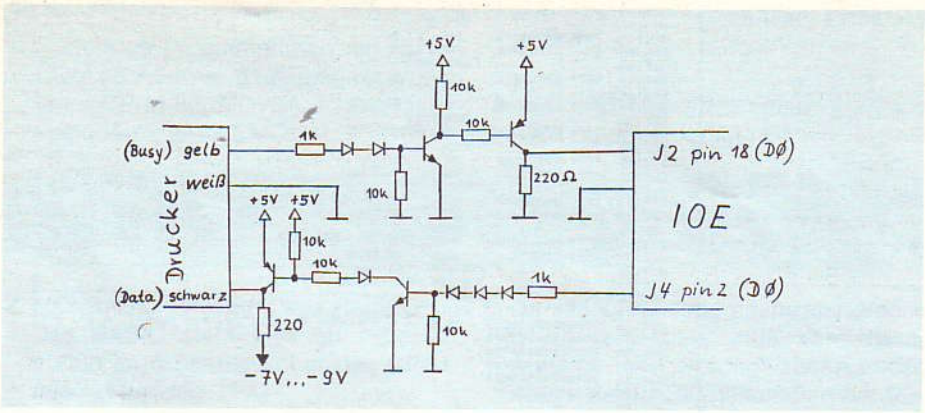
Hinweis der Redaktion:

Der preisgünstige Vorschlag ist sicher auch für WordStar-Anwendungen interessant!

Suche Erfahrungsaustausch mit NDR-Benützern im Raum Nürnberg, möglichst mit Akustikkoppler und CP/M.

W. Erhard
Ostlandstraße 16, 8501 Cadolzburg
Tel. (09103) 1809

```
D r u c k e r s t e u e r u n g  für  EXD - 10  Schreibprinter
;*************************************************************
;Werner Erhard  8501 Cadolzburg  (C) 12.85
;Geschrieben für Z80 - Assembler mit ZEAT - Betriebssystem

WSTART  EQU     00000H
SYSTEM  EQU     00005H
TEXTan  EQU     247
TEXTen  EQU     248

START1: LD      C,TEXTan        ;Anfangsadresse des Editors in
        CALL    SYSTEM          ;das HL - Reg. laden
BEGINN: PUSH    HL
        EX      DE,HL           ;Anfangsadresse in DE - Reg. laden
        LD      C,TEXTen        ;letzte Adresse aus dem Editor in
        CALL    SYSTEM          ;das HL-Reg. laden
        LD      A,D
        CP      H               ;Pruefung, ob TEXTen erreicht ist
        JP      NZ,START2
        LD      A,E
        CP      L
        JP      NZ,START2
        LD      C,WSTART
        CALL    SYSTEM
START2: POP     HL
        LD      A,01
        OUT     (30H),A         ;DATA-Ltg. auf "1" stellen
START3: IN      A,(30H)
        RRCA                    ;Pruefung, ob Busy-Ltg. auf "1"
        JP      NC,START3
START4: IN      A,(30H)         ;warte, bis Busy-Ltg. auf "0"
        RRCA
        JP      C,START4
        LD      A,00
        OUT     (30H),A         ;Startbit senden
        CALL    T2
        LD      A,(HL)          ;das zu sendende Byte in den Akku laden
        XOR     'ö'
        JP      Z,OE
        LD      A,(HL)
        XOR     'ä'
        JP      Z,AE
        LD      A,(HL)
        XOR     'ü'             ; ä,ö,ü,Ä,Ö,Ü,ß  umprogrammieren
        JP      Z,UE
        LD      A,(HL)
        XOR     'Ä'
        JP      Z,GAE
        LD      A,(HL)
        XOR     'Ö'
        JP      Z,GOE
        LD      A,(HL)
        XOR     'Ü'
        JP      Z,GUE
        LD      A,(HL)
        XOR     'ß'
        JP      Z,SS
        LD      A,(HL)

START5: INC     HL              ;HL-Reg. auf naechstes Byte stellen
        LD      B,A             ;B-Reg. als Zwischenspeicher fuer Akku verw.
        AND     01              ;maskieren von D0
        OUT     (30H),A         ;1. bit ausgeben
        NOP
        CALL    T1
        LD      D,07            ;noch 7x ein bit ausgeben
SEND:   LD      A,B             ;Byte aus dem Zwischenspeicher in Akku
        RRCA
        LD      B,A             ;Byte zurueck in Zwischenspeicher
        AND     01              ;D0 maskieren
        OUT     (30H),A         ;2. bis 8. bit ausgeben
        CALL    T1
        DEC     D               ;naechstes bit
        JP      NZ,SEND
        LD      A,01
        OUT     (30H),A
        JP      BEGINN

T1:     LD      A,0DEH
S1:     DEC     A
        JP      NZ,S1           ;Zeitglied 1,66 ms minus Programmlaufzeit
S2:     DEC     A
        JP      NZ,S2
        RET
T2:     LD      A,088H
S3:     DEC     A
        JP      NZ,S3
S4:     DEC     A               ;Zeitglied 1,66 ms minus Programmierzeit
        JP      NZ,S4           ;des Startbits
        RET

AE:     LD      A,0D1H
        JP      START5
OE:     LD      A,0D5H
        JP      START5
UE:     LD      A,0D3H
        JP      START5
GAE:    LD      A,0BDH
        JP      START5          ;ä,ö,ü,Ä,Ö,Ü,ß  umprogrammieren
GOE:    LD      A,0C1H
        JP      START5
GUE:    LD      A,0BFH
        JP      START5
SS:     LD      A,0AEH
        JP      START5
```

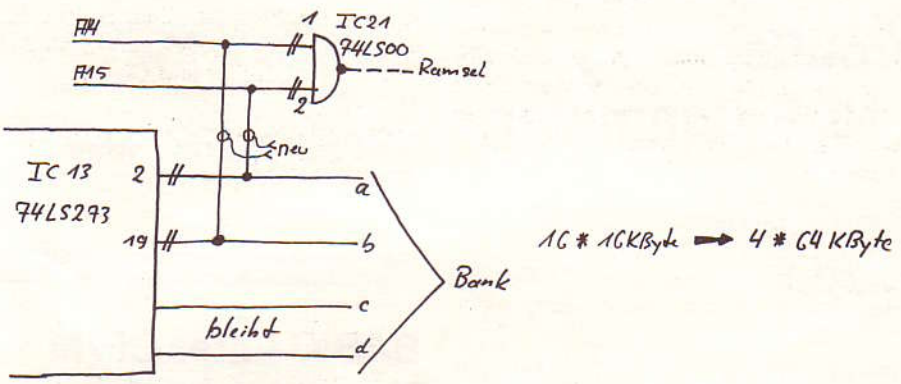


Z80 Systemen betrieben werden soll. Beim Betrieb in einem 68008- bis 68020-System ist es jedoch sinnvoll, mit einer kleinen Änderung die Seitenumschaltung durch eine normale Adressierung zu ersetzen, so daß der Bildspeicher dann als linearer 64 KByte Speicher adressierbar ist. Ein wesentlich schnellerer Bildaufbau ist die Folge. Besonders Anwendern, die sich mit der Bildverarbeitung beschäftigen wollen, empfehle ich diese Änderung, da man beim Umkopieren, Addieren, Subtrahieren, Filtern und so weiter mit einer einfachen inkrementellen Adressierung auskommt.

Änderung an der COL256

von Willi Sicking

Über ein 16 KByte großes Speicherfenster und ein 2 Bit Pageregister hat man in der Normalausführung der Baugruppe Zugriff auf den Bildspeicher. Diese Unterteilung in verschiedene Segmente war nötig, da die COL256 auch in NDR-Klein



Die Adressleitungen A14 und A15 werden zu diesem Zweck von der Dekodierschaltung IC 21 (74LS00) Pin 1 und 2 abgetrennt und mit den Bankleitungen a und b verbunden, die natürlich ebenfalls vom Pageregister IC 13 (74LS273) Pin 2 und 19 isoliert werden müssen. Diese Änderung geht gänzlich ohne Kratzen vonstatten, wenn man die IC-Füßchen in bekannter Weise über die Sockel biegt. Die Bankleitungen c und d bleiben am Pageregister angeschlossen. Damit kann man bei einem späteren Ausbau des Bildspeichers auf 256 KByte vier Pages à 64 KByte anwählen.

Über Steckbrücken wählt man nun noch die Lage im Gesamtadressraum, bei mir z.B. D0000. Der Bildspeicher darf nicht am Ende des ersten Speicherbereiches liegen, da der Rechner sich hier den Stack einrichtet.

```
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

0EE000                  ORG $EE000                          ;JUBS (C); 22.3.1986
0EE000                  *-------------------------------------------------------*
0EE000                  *                                                       *
0EE000                  *                    N  A  C  H  A  S  S                *
0EE000                  *                 ------------------------------        *
0EE000                  *        DRUCKERANPASSUNG FUER AUSDRUCK LISTING         *
0EE000                  *              MENUE-WAHL NACH AUSGABE LISTING          *
0EE000                  *-------------------------------------------------------*
0EE000                  * DIE MENUES DES RDK-68000-GRUNDPROGRAMMS SIND ZWAR     *
0EE000                  * WEGEN DER GROESSE IHRER TEXTE FUER DEMONSTRATIONS-    *
0EE000                  * ZWECKE GUT GEEIGNET. FUER DEN EINZELANWENDER SIND     *
0EE000                  * SIE SIE JEDOCH BEI DER ENTWICKLUNG UND DEM AUSTES-    *
0EE000                  * TEN VON PROGRAMMEN EHER UMSTAENDLICH. DAS NACHFOL-    *
0EE000                  * GENDE PROGRAMM, DAS IN DAS RDK-68000-GRUNDPROGRAMM    *
0EE000                  * LEICHT ZU INTEGRIEREN IST, BEHEBT DIESEN MANGEL. -    *
0EE000                  * DAS PROGRAMM FRAGT ZUNAECHST DIE VARIABLE "IOSTAT"    *
0EE000                  * DES RDK-68000-GRUNDPROGRAMMS AB UND VERZWEIGT JE      *
0EE000                  * NACH DESSEN INHALT ZUR AUSGABE DES LISTINGS AUF CRT   *
0EE000                  * ODER LST, WOBEI EINE DRUCKERANPASSUNG ERFOLGEN KANN,  *
0EE000                  * WENN ERFORDERLICH. ANDERNFALLS MUSS DAS PROGRAMM      *
0EE000                  * BEI "5. ASSEMBLIEREN ..." BEGINNEN, WEITER MIT        *
0EE000                  * "7. WAHL ...". - NACH AUSGABE DES LISTUNGS KOENNEN    *
0EE000                  * VERSCHIEDENE ROUTINEN AUS DEM RDK-68000-GRUNDPRO-     *
0EE000                  * PROGRAMM DIREKT AUFGERUFEN WERDEN. - CPM/68K:         *
0EE000                  * ACHTUNG, CURSOR ERSCHEINT NACH DEM ERSTEN ASSEM-       *
0EE000                  * BLIEREN NICHT, CTRL-Z EINGEBEN.                       *
0EE000                  *-======================================================*
0EE000                  * 1. LISTING RDK-68000-GRUNDPROGRAMM, S. 211,           *
0EE000                  *    AENDERUNG:                                         *
0EE000                  *    ASSTEXT:                                           *
0EE000                  *    JMP $EE000          * EASS4                        *
0EE000                  *    NOP                                                *
0EE000                  *    FINMENUE:                                          *
0EE000                  *    ...                                                *
0EE000                  *-------------------------------------------------------*
0EE000                  * 2. NEUES EPROM EASS1 PROGRAMMIEREN:                   *
0EE000                  *    VON EASS1 (ALT) $E2000 - $E2819 NACH               *
0EE000                  *        EASS1 (NEU)     $0 - $819                      *
0EE000                  *    NACH $81A AUF EASS1 (NEU):                         *
0EE000                  *        JMP $EE000                                     *
0EE000                  *        NOP                                            *
0EE000                  *    VON EASS1 (ALT) $E2822 - $E3FFF NACH               *
0EE000                  *        EASS1 (NEU)    $822 - $1FFF                    *
0EE000                  *-======================================================*
0EE000                  * 3. LISTING AUF LST?                                   *
0EE000                  *-======================================================*
0EE000  0C39 0003        CMPI.B  #3, $E002A      * IOSTAT: LISTING AUF DRUCKER?
0EE004  000E002A
0EE008  6658             BNE.S   ASSCRT
0EE0A                   *-======================================================*
0EE0A                   * 4. DRUCKERANPASSUNG ("COMPUTE MATE-100 CPA 80 [80]")*
0EE0A                   *-======================================================*
0EE0A   103C 001B        MOVE.B  #$1B, D0        * DRUCKER AUF NORMALMODUS
0EE00E  4EB9 000C0D28    JSR     @LO
0EE014  103C 0040        MOVE.B  #'@', D0
0EE018  4EB9 000C0D28    JSR     @LO
0EE01E  103C 001B        MOVE.B  #$1B, D0        * 72 ZEILEN
0EE022  4EB9 000C0D28    JSR     @LO
0EE028  103C 0043        MOVE.B  #'C', D0
0EE02C  4EB9 000C0D28    JSR     @LO
0EE032  103C 0048        MOVE.B  #72, D0
0EE036  4EB9 000C0D28    JSR     @LO
0EE03C  103C 001B        MOVE.B  #$1B, D0

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

0EE040  4EB9 000C0D28    JSR     @LO
0EE046  103C 006C        MOVE.B  #'l', D0        * LINKER RAND
0EE04A  4EB9 000C0D28    JSR     @LO
0EE050  103C 0002        MOVE.B  #2, D0
0EE054  4EB9 000C0D28    JSR     @LO
0EE05A                  *-======================================================*
0EE05A                  * 5. ASSEMBLIEREN / LISTING = LST                       *
0EE05A                  *-======================================================*
0EE05A  4EB9 000C6504    JSR     @ASSEMBLE
0EE060  6106             BSR.S   PROGWAHL
0EE062                  *-======================================================*
0EE062                  * 6. ASSEMBLIEREN / LISTING = CRT                       *
0EE062                  *-======================================================*
0EE062                  ASSCRT:
0EE062  4EB9 000C6504    JSR     @ASSEMBLE
0EE068                  *-======================================================*
0EE068                  * 7. WAHL NEUES PROGRAMM                                *
0EE068                  *-======================================================*
0EE068                  * KLINGELZEICHEN NACH AUSGABE LISTING                    *
0EE068                  *-------------------------------------------------------*
0EE068                  PROGWAHL:
0EE068  303C 0007        MOVE    #7, D0          * KLINGEL
0EE06C  4EB9 000C0D28    JSR     @LO
0EE072                  *-------------------------------------------------------*
0EE072                  * VERZWEIGUNGSMENUE / ANWAHL BILDSEITEN                  *
0EE072                  *-------------------------------------------------------*
0EE072  4239 000E8217    CLR.B   $E8217          * VIEWPAGE
0EE078  13FC 0001        MOVE.B  #1, $E8218      * WRTPAGE
0EE07C  000E8218
0EE080  4EB9 000E0E14    JSR     $E0E14          * AKTPAGE
0EE086  6112             BSR.S   VZWMEN1
0EE088  4239 000E8218    CLR.B   $E8218
0EE08E  4EB9 000E0E14    JSR     $E0E14
0EE094  6104             BSR.S   VZWMEN1
0EE096  6100 0060        BSR     VZWMEN3
0EE09A                  *-------------------------------------------------------*
0EE09A                  VZWMEN1: * VERZWEIGUNGSMENUE / MENUE-KASTEN             *
0EE09A                  *-------------------------------------------------------*
0EE09A  4EB9 000C0D7C    JSR     @SETPEN
0EE0A0  4241             CLR     D1
0EE0A2  4242             CLR     D2
0EE0A4  4EB9 000C0EE2    JSR     @MOVETO
0EE0AA  323C 01FF        MOVE    #511, D1
```

```
ØEEØAE  4EB9 ØØØCØF4C    JSR     @DRAWTO
ØEEØB4  343C ØØØA        MOVE    #1Ø, D2
ØEEØB8  4EB9 ØØØCØF4C    JSR     @DRAWTO
ØEEØBE  4241             CLR     D1
ØEEØCØ  4EB9 ØØØCØF4C    JSR     @DRAWTO
ØEEØC6  4242             CLR     D2
ØEEØC9  4EB9 ØØØCØF4C    JSR     @DRAWTO
ØEEØCE                  *-----------------------------------------------*
ØEEØCE                  * VERZWEIGUNGSMENUE / AUSGABE TEXT              *
ØEEØCE                  *-----------------------------------------------*
ØEEØCE                  VZWMEN2:
ØEEØCE  41F9 ØØØEE17A    LEA     ZEILE1, AØ
ØEEØD4  303C ØØ11        MOVE    #$11, DØ
ØEEØD8  323C ØØØ4        MOVE    #4, D1
ØEEØDC  343C ØØØ1        MOVE    #1, D2
ØEEØEØ  4EB9 ØØØC1386    JSR     @WRITE
ØEEØE6  41F9 ØØØEE1A3    LEA     ZEILE11, AØ
ØEEØEC  323C ØØFB        MOVE    #251, D1
ØEEØFØ  4EB9 ØØØC1386    JSR     @WRITE
ØEEØF6  4E75             RTS
ØEEØFB                  *-----------------------------------------------*

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 3

ØEEØF8                  VZWMEN3: * VERZWEIGUNGSMENUE / MENUEWAHL        *
ØEEØF8                  *-----------------------------------------------*
ØEEØF8  4EB9 ØØØCØAØØ    JSR     @CI
ØEEØFE  13FC ØØØØ        MOVE.B  #Ø, $EB02E      * AUTOFLIP AUS
ØEE1Ø2  ØØØEB02E
ØEE1Ø6  4EB9 ØØØE1B2E    JSR     $E1B2E          * "BUCHECK"
ØEE1ØC  ØC00 0041        CMPI.B  #'A', DØ
ØEE11Ø  6726             BEQ.S   ANSEHEN
ØEE112  ØC00 0045        CMPI.B  #'E', DØ
ØEE116  672C             BEQ.S   EDITOR
ØEE11B  ØC00 0046        CMPI.B  #'F', DØ
ØEE11C  672C             BEQ.S   FLOPPY
ØEE11E  ØC00 004D        CMPI.B  #'M', DØ
ØEE122  672C             BEQ.S   MENRDK
ØEE124  ØC00 004E        CMPI.B  #'N', DØ
ØEE12B  672C             BEQ.S   NEUTXT
ØEE12A  ØC00 0053        CMPI.B  #'S', DØ
ØEE12E  6732             BEQ.S   START
ØEE13Ø  ØC00 0054        CMPI.B  #'T', DØ
ØEE134  673B             BEQ.S   TRACE
ØEE136  6ØCØ             BRA.S   VZWMEN3
ØEE138                  ANSEHEN:
ØEE138  4EB9 ØØØCØDAØ    JSR     @CLR
ØEE13E  4EF9 ØØØE3494    JMP     $E3494
ØEE144                  EDITOR:
ØEE144  4EF9 ØØØE34C2    JMP     $E34C2
ØEE14A                  FLOPPY:
ØEE14A  4EF9 ØØØE374E    JMP     $E374E
ØEE15Ø                  MENRDK:
ØEE15Ø  4EF9 ØØØE343E    JMP     $E343E
ØEE156                  NEUTXT:
ØEE156  4EB9 ØØØCØDAØ    JSR     @CLR
ØEE15C  4EF9 ØØØE35BØ    JMP     $E35BØ
ØEE162                  START:
ØEE162  4EB9 ØØØCØDAØ    JSR     @CLR
ØEE168  4EF9 ØØØE3486    JMP     $E3486
ØEE16E                  TRACE:
ØEE16E  4EB9 ØØØCØDAØ    JSR     @CLR
ØEE174  4EF9 ØØØE3B1C    JMP     $E3B1C
ØEE17A                  *-----------------------------------------------*
ØEE17A                  * VERZWEIGUNGSMENUE / TEXT                      *
ØEE17A                  *-----------------------------------------------*
ØEE17A                  ZEILE1:
ØEE17A  41203D20414E53   DC.B 'A = ANSEHEN, E = EDITOR, F = FLOPPY, M =', Ø
ØEE181  4548454E2C2045
ØEE188  203D2045444954
ØEE18F  4F522C2046203D
ØEE196  20464C4F505059
ØEE19D  2C204D203DØØ
ØEE1A3                  ZEILE11:
ØEE1A3  4D454E55452C20   DC.B 'MENUE, N = NEUER TEXT, S = START, T = TRACE', Ø
ØEE1AA  4E203D204E4555
ØEE1B1  455220544554 54
ØEE1B8  2C205320 3D2053
ØEE1BF  544152542C2054
ØEE1C6  203D2054524143
ØEE1CD  4500
ØEE1CF                  *-=============================================-*
ØEE1CF                  * 8. EPROM EASS4 PROGRAMMIEREN                  *
ØEE1CF                  *    VON $EEØØØ - $EE1CF NACH $Ø - $1CF          *
ØEE1CF                  *-=============================================-*
ØEE1CF                  END

ØE8B4A  Ende-Symboltabelle
```

TIPS + TRICKS — TIPS + TRICKS

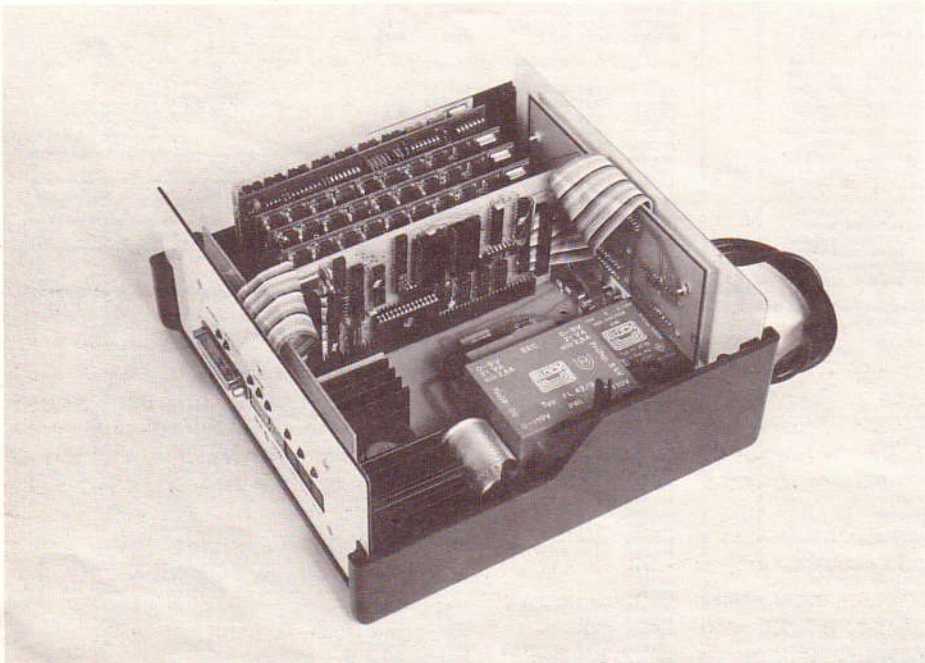
1 Megabyte Druckerpuffer – aufgebaut mit dem NDR-Computer

Einen „Druckerpuffer" mit 1 Megabyte (=1.048.576 Zeichen) Speicherkapaziät stellte uns die Firma Jürgens Elektronik aus Wilhelmshaven vor.

Ein Druckerpuffer ist ein Zwischenspeicher, der die Signale, die zu einem Drukker gesendet werden, speichert (puffert). Der Computer sendet also mit voller Geschwindigkeit Zeichen für Zeichen zum Puffer, der diese dann wieder auf den (langsameren) Drucker weitergibt.

Bisherige Druckerpuffer speichern zwischen 2 KByte und 64 KByte – ein Puffer mit der enormen Kapazität von 1 MByte ist der Redaktion bis heute noch nicht bekannt!

Wie viel Papier ist denn 1 MByte? Nun, auf eine DIN-A-4-Seite passen 72 Zeilen à 83 Zeichen ($^1/_{10}$" Zeichenabstand vorausgesetzt), das sind also maximal 5976 Zeichen. Da aber in der Praxis eine Seite nie ganz vollgeschrieben ist, rechnet man mit 3 – 4 KByte pro Seite.



Der Druckerpuffer speichert also zwischen 250 und 350 Seiten Ausdruck!

Das Gerät besteht aus der CPU68K, vier (oder weniger) RAM64/256 und einer selbst entwickelten Schnittstelle.

Näheres, auch über die Preise, bei GES.

BASIC unter CP/M – Verschieben des EPROM-BASIC in den Speicherbereich des CP/M

von C. Denneburg, Wennigsen

In der *LOOP* Nr. 6, Seiten 9 + 12, wurden kurze Verschieberoutinen vorgestellt, die den mächtigen Blockverschiebebefehl des Z80-Prozessors nutzen. Hier nun ein weiteres Beispiel:

Unterstellt:

- Sie haben den NDR-Rechner auf CP/M ausgebaut,
- Sie haben noch keinen BASIC-Interpreter für CP/M,
- Sie besitzen das NDR-BASIC ab Adresse 4000h als EPROM,

dann verschieben Sie es in den Speicherbereich der Floppy! Das ist zwar nur ein Behelf, aber immer noch wesentlich schneller, als das Arbeiten mit dem Kassettenrekorder. Das Beispiel geht von der RAM 64/256K Karte als Speicher aus. Wenn Sie eine ROA verwenden, müßte die Zieladresse entsprechend verändert werden!

Also:

1. Folgendes Prögrämmchen mit DDT.-COM ab Adresse 0100 eingeben und unter CP/M als MOVBASIC.COM abspeichern ('SAVE 5 MOVBASIC.COM')

```
d
0100 21 00 40 11 00 40 0E 0E 06 00 CD 5B F0 01 80 00 !.§..§.....Ä....
0110 EB 09 EB 09 01 00 60 78 AC C2 06 01 C3 00 00 00 ......`x........
```

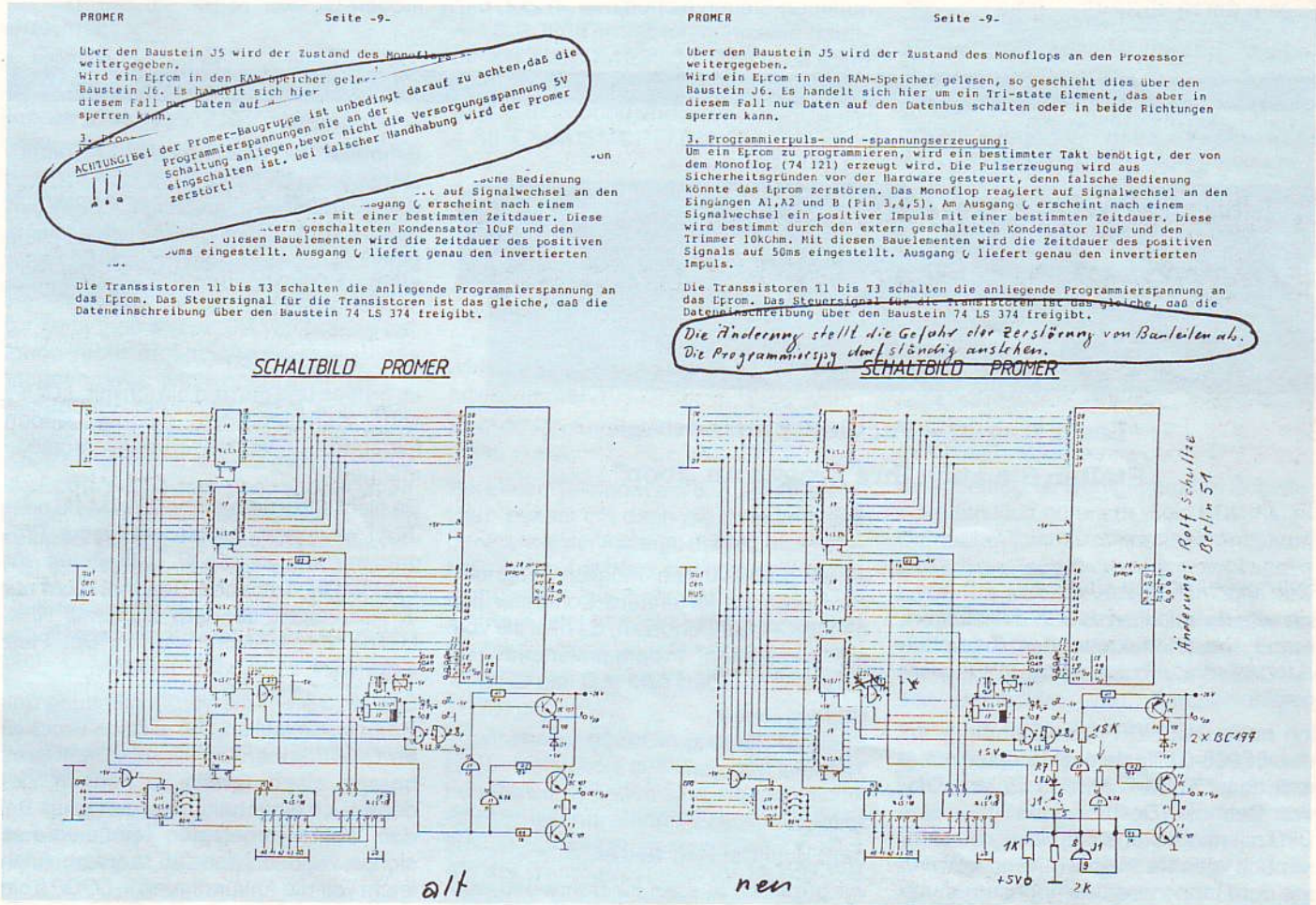
2. MOVBASIC von CP/M aus starten
3. Das BASIC steht jetzt ab Adresse 4000h im Arbeitsspeicher des CP/M (also in der sog. TPA)
4. Mit 'SAVE 180 NDRBASIC. COM' auf Diskette abspeichern
5. Mit 'DDT NDRBASIC.COM' zurück in den Speicher holen
6. Da 'alle' CP/M-Programme ab Adresse 0100h anfangen, muß von dieser Adresse ein Sprung zur Adresse 4000h des BASIC geschaffen werden. Dies geht am einfachsten wieder mit DDT.
7. Mit 'S0100' in den Speicherbereich ab Adresse 0100 gehen und den Sprung eingeben. 'C3' –CR– '00' –CR– '40' –CR– und dann '.' –CR–, Control C. Sie müssen jetzt wieder mit dem >A auf dem Bildschirm sein?!
8. Jetzt den Interpreter endgültig unter 'SAVE 180 NDRBASIC.COM' abspeichern.

Jetzt können Sie mit dem BASIC-Interpreter „ganz normal" arbeiten, bis Ihnen etwas Feineres zur Verfügung steht.

Änderung PROMER-Schaltung

Rolf Schultze, Berlin

Die Änderung der PROMER-Schaltung verhindert mit Sicherheit die Beschädigung von Bauteilen einschließlich des zu programmierenden PROMS. Der Hinweis in der PROMER-Beschreibung, die Programmierspannung darf nur an oder abgeschaltet werden, wenn der Rechner betriebsbereit ist, braucht nicht mehr



beachtet zu werden. Funktion der Änderung: Das dem J1 an Anschluß 11 und 13 zugeführte Signal wird invertiert. Die Invertierung wird durch die „neuen" Transistoren BC 177 (T1 + 2) wieder aufgehoben. Somit ist die alte Funktion erhalten geblieben. Der wesentliche Unterschied besteht aber im Verhalten der neuen Schaltung bei Ausfall der + 5 V Spannung. Die Transistoren T1 + T2 sind bei fehlender + 5 V Spannung gesperrt. Die „alte" Schaltung ist „LOW AKTIV" und schaltet bei fehlender + 5 V Spannung T1 + T2 ungewollt durch. Das führt dann zur Zerstörung des PROMS.

Wer auf eine korrekte Anzeige durch die LED Wert legt, kann auch noch die gestrichelt eingezeichnete Änderung durchführen. Die LED leuchtet dann nur noch während der Programmierung und nicht schon beim Einschalten des Rechners.

Erforderliche Bauteile:
2 Stck. Transistoren BC 177
2 Stck. Widerstände 18 K Ohm
1 Stck. Widerstand 1 K Ohm
1 Stck. Widerstand 2 K Ohm

Viel Spaß beim Umbau und Freude an der nun sicher arbeitenden Schaltung.

PS: „Anfänger sollten die Änderung nicht selbst durchführen. Das vorliegende Layout macht die Änderung schwierig.

Druckfehler: LOOP 7 Balkengraphikprogramm

Leider haben sich beim Abdruck des Listings Verwechslungen ergeben. Der Autor hat uns ein neues, relokatives Listing zugesandt. Wir können es aus Platzgründen nicht noch einmal abdrukken, bieten jedoch jedem *LOOP*-Leser an, bei Zusendung eines frankierten (DM 1,40) Rückumschlags, ihm dieses Listing zuzusenden.

Stichwort:
Listing Balkengraphik, *LOOP* 7.

FLOH MARKT

Verkaufe günstig für Aufsteiger: ROA64, ORIGINAL-EASSO-3, V 4.3, Buch dazu (neu), BUS2, alles betriebsbereit.

Tyko Strassen
Trichtenhausenstr. 40, 8053 Zürich
Telefon (Schweiz) 01 55 52 46

Verkaufe:
GEH1-Gehäuse (Schroff-19") neuwertig, DM 160,–.

Karl Kauper
Stephanstraße 28, 85 Nürnberg 30
Telefon (09 11) 46 32 49

Verkauf:
SBC2 und POW5 voll aufgebaut mit EGRUND und ESKOP2 für insgesamt 220,– DM.

Heinz Bischoff
Sommerau 18, 8800 Ansbach-Eyb
Telefon (09 81) 9 47 36 ab 17.00 Uhr

Verkaufe:
ROA16-Speicherkarte für 16 KByte, ohne RAM, 40,– DM + NN und Porto.

Hubert Kettler
Horasser Weg 79, 6400 Fulda

TEXT 68K System 68008 Bibl. Prof. Textverarbeitung u. Drucksteuerung für Epson RX80 u. komp.-13 Menues, relok., alle Druckparam., Block- u. Flattersatz, Satzumbruch, Textopt., Headline, Seitennummer, Steuerzeichen usw. In 2x2764/250 + Anleitung für DM 85,– p. NN. nur bei H.-J. Bessel, Lange Straße 38a, 3304 Wendeburg, Tel. (0 53 02) 35 85

HEXEY: Komfortable HEX-Dump Ein- und Ausgabe mit zus. Funktionen: Datenverschieben, Summen, Start, Editor-Druck zus. Bibl.-funkt., Schnittstellen zu TEXT-, DAT-, DOSEY, Jogi-DOS, DOS-I. Auf EPROM 40,– DM per NN oder Vorkasse. Info 1,60 in Briefm. an H. G. Eysel, Klostergut Riechenberg 7, 3380 Goslar.

Selbstbau-Computer-Club, eine Vereinigung der NDR-68K-Benutzer. Club-Broschüre (32 S.) alle 2 Monate mit Programmen, Tips und Tricks 3,– DM. Jahresbeitrag 18,– DM inkl. Brosch. Info und Bestellung: SCC, Dorfstraße 4, 3372 Wallmoden 1.

IN & OUT

Leser fragen – Fachleute antworten

Stellen Sie auch Ihre Fragen an „loop"

Sehr geehrter Herr Graf,

ich wende mich an Sie in Ihrer Eigenschaft als Redakteur der Zeitschrift *„LOOP"*, die mir im übrigen exzellent gefällt.

Ich habe den NDR-Klein-Computer mit der 68008-CPU, dem Grundprogramm und dem DRAM-128K des Elektronikladen Detmold. Deshalb muß ich meine CPU mit mindestens 5 (!) Waitzyklen fahren. Ich möchte meinen Computer nun mit der Floppy ausrüsten, dazu nun folgende Fragen:

a) Ist es möglich, auch mit 5 Waitzyklen das TEAC-Laufwerk FD 55F und die FLO2-Baugruppe fehlerfrei zu betreiben?

b) Wenn nein, beschleunige ich den Rechner, wenn ich die DRAM 256K Ihrer Firma allein benutze (eine Erweiterung war so oder so vorgesehen)?

c) Ist es möglich CP/M 68K und vor allem "C" (!) auch mit einem Laufwerk zu fahren?

d) Gibt es bereits für den NDR-Klein-Computer angepaßte weitere Programmiersprachen? Wichtig wären für mich vor allem FORTRAN und COBOL! – Wenn ja, würde ich gern den Preis erfahren.

Ist es grundsätzlich möglich Programmiersprachen für andere Computer mit 68008 CPU zu benutzen, da hier die von mir genannten Programmiersprachen bereits vorhanden sind (z.B. mc CP/M)?

Michael Reiner,
Stephan-Blattmann-Str. 29,
7743 Furtwangen

Antwort *LOOP:*

Sehr geehrter Herr Reiner,

wir beziehen uns auf Ihr Schreiben vom 17. Dezember, das wir wie folgt beantworten:

a) ein sicherer Betrieb ist nur bei zwei bis vier Waitzyklen gewährleistet.

b) zur DRAM128K ist eine CPU68KR4 erforderlich, dann ist der Rechner wesentlich schneller (Wait über Busleitung). Bei unserer RAM64/256 sind drei Waitzyklen erforderlich.

c) Ist möglich, wenn RAM-Floppy vorhanden (RAM256 ab Adresse $8000, aber *sehr* umständlich).

d) Wir sind in Kontakt mit einigen Software-Häusern, um das Software-Angebot für den 68000-Rechner erheblich zu verstärken.

Grundsätzlich ist der mc-CP/M-Computer mit dem Betriebssystem CP/M2.2 und der 68000er Computer mit CP/M68K ausgerüstet – hier gibt es leider keine gemeinsamen Programme. (Z80 Emulator hilft!)

Sehr geehrter Herr Graf,

im großen und ganzen bin ich mit *„LOOP"* sehr zufrieden. Ich möchte jedoch folgende Verbesserungsvorschläge machen:

Da nicht alle Kunden über den M-80 oder ZEAT verfügen, sollten so kleine Programme wie in *LOOP 6* (Kaltstart für EFLOMON, Transport-Routine) nicht nur in Assembler, sondern auch als HEX-Listing veröffentlicht werden. Der Platz dafür wäre vorhanden gewesen.

Seit Nr. 3 steht die Veröffentlichung des Gomoku-Spiels an. Der winzige Brocken in *LOOP 6* ist mehr als ein Witz. Dann ist es besser, einen Termin zu nennen, der dann auch eingehalten werden kann. Bei dem bisher vorgelegten Tempo wird es sicher Weihnachten '86 werden. (Vielleicht war die Ankündigung in *LOOP 5* ein Druckfehler?)

Sicher wäre es von Interesse, wenn Sie – wie bei der Software – die aktuelle Version der Platinen (mit Angabe der Änderungen) veröffentlichen würden.

Dr. Jürgen Gabriel,
Goethestr. 12, 4270 Dorsten 1

Antwort *LOOP:*

1. Ihr Vorschlag über das HEX-Listing wird dankend angenommen. Wir werden uns in den nächsten *LOOP's* daran halten.

2. GOMOKU ist schon etwas ein Trauerspiel. Wir wollten ursprünglich das Original „GOMOKU" veröffentlichen, wie es

der Autor geschrieben hat. Dies erschien uns jedoch nicht sinnvoll, da die Komentare sehr spärlich waren. Also mußten wir das Programm noch einmal völlig neu tippen. Der Umfang ist mittlerweile etwa acht DIN-A4-Seiten, der den Rahmen der *LOOP* sprengt.

Wir haben uns deshalb entschlossen, GOMOKU als externes Listing anzubieten und auch über unsere Datenbank abrufbar zu machen. Wir haben wieder einmal dazugelernt.

Sehr geehrte Damen und Herren,

da ich eine Teilnehmerkennung für DATEX-P beantragt habe und mir einen Akustikkoppler zulegen möchte, habe ich folgende Fragen:

1. Sind Sie (Fa. GES) über DATEX-P erreichbar?

2. Wie sieht es mit Btx aus, lohnt es sich z.B., einen entsprechenden Akustikkoppler (also auch 1200/75 bit/s.) zuzulegen?

3. Ist eine CPU 8086/8088 geplant (lt. MC ja) oder in Arbeit?

Wenn Sie mir vorab zu 1. und 2. Nachricht geben könnten, wäre ich Ihnen sehr dankbar.

Zur Zeitschrift selbst: Die Gestaltung hat schon recht ordentliche Formen angenommen. Was mich noch stört, ist die Verstreutheit der Artikel, die m. E. zusammengehören. Sicher, wenn die neue *LOOP* eingetroffen ist, liest man zunächst alles durch. Will man aber später einen Artikel bzw. eine Beschreibung einer Neuerscheinung nachschlagen, so ist man am blättern und suchen.

Mein Vorschlag: Neuveröffentlichungen und Hardwaretips und -verbesserungen nach CPU's getrennt aber zusammen, desgleichen für Software und deren Tips und Beschreibungen.

Für Ihre Aufmerksamkeit danke ich und verbleibe

Harald Kittler,
Klosterkamp 11, 2360 Bad Segeberg

Antwort *LOOP:*

Sehr geehrter Herr Kittler,

zu ihren Fragen:

1. Nein.

2. Wir haben mit Btx-Zugriff über einen schnellen Akustikkoppler keine Erfahrung, können diese Frage also nur an unsere Leser weitergeben. Wer hier etwas weiß, setzt sich bitte mit Herrn Kittler direkt in Verbindung.

3. Die CPU8086/8088 ist bereits im Fädelaufbau vorhanden, es ist jedoch noch nicht sicher, ob wir daraus ein Produkt machen werden.

Herzlichen Dank zur Kritik über die *LOOP*-Zeitung. Wir versuchen bereits seit der *LOOP 6* die für Einsteiger für Z80 und die für 68000 geeigneten Artikel einigermaßen zusammen unterzubringen. Jedoch sind uns aufgrund des noch geringen Umfanges hier Grenzen gesetzt.

Bezugnehmend auf Ihr Versprechen in *LOOP 6,* Seite 1, auch negative Leserbriefe abzudrucken, möchte ich meinem Ärger über das Geschäftsgebaren Ihrer Firma Luft machen.

Am 27. 1. 1986 bestellte ich laut Angebot im MC-Heft 2/86, Seite 94:

4 Stck. MAUSMCP je DM 39,50	
	DM 158,00
1 Stck. MAUSMCH	DM 10,00
gesamt	DM 168,00

Am 3. 2. 1986 erhielt ich dann eine Nachnahme über DM 230,35. Was mich dann glatt umhaute, denn da mußte ich folgende Rechnung vorfinden:

4 Stck. MAUSMCP je DM 43,42	
	DM 173,68
1 Stck. MAUSMCH	DM 17,54
+ MWSt.	DM 26,77
das macht gesamt	DM 217,99

Alles ohne jegliche Begründung, geschweige denn vorheriger Benachrichtigung, was normalerweise so üblich ist. – Aber was soll's, per Nachnahme klappt ja sowas immer.

Meiner Rechnung nach sind dies DM 49,99 zuviel.

Bei einem Telefonat am 6. 2. mit einem H. Herb bekam ich dann die Auskunft, daß der Preis in der Anzeige der MC eine Fehlkalkulation und aus zeitlichen Gründen nicht mehr zu berichtigen war. Man könne ja, wenns nicht paßt, alles wieder zurückschicken; was leicht zu sagen ist, da ja im Moment nur Fa. Graf diese Platte liefert.

Inzwischen muß ich in der mitgelieferten *LOOP* feststellen, daß dort genau dieselben Preise zu finden sind, ohne jegliche Richtigstellung. Benötigen Sie etwa für Ihre 24 Seiten genau soviel Zeitaufwand wie der Franzis-Verlag für die MC mit 128 Seiten?

Ich bitte um Ihre Stellungnahme!

Anbei DM 3,– in Briefmarken für die *LOOP*-Zeitung, in der dieser Leserbrief veröffentlicht wird.

Hochachtungsvoll

Raimund Hoffmann,
Im Angel 57, 7242 Dornhan 5

Antwort *LOOP:*

Sehr geehrter Herr Hoffmann,

es tut uns leid, daß Sie aufgrund der Preisdifferenzen mit uns böse sind. Das Problem in der Kalkulation der Hardcopy-Maus-Platine lag darin, daß von Seiten unserer Kalkulation angenommen wurde, daß es sich bei dieser Platine um eine identische, sowohl für mc als auch für NDR einsetzbare Platine, handelt.

Leider ist das nicht der Fall, die NDR-Computer-Platine ist kürzer, die mc-Computer-Platine ist länger, und so hat uns der Hersteller der Platinen auch entsprechend andere Preise verrechnet. Aufgrund dieser Fehlkalkulation waren wir gezwungen, den Preis kurzfristig zu ändern. Wir bitten hierfür um Ihr Verständnis. Natürlich besteht das Angebot von Herrn Herb, die Baugruppe unter voller Gutschrift zurückzugeben, immer weiter.

Solche Preisanpassungen sind uns ebenfalls sehr unangenehm, bleiben jedoch grundsätzlich nicht aus. Die Veröffentlichung der identischen Preise rührt daher, daß der Film für die mc und für die *LOOP* von der gleichen Werbeagentur hergestellt wurde und nicht mehr geändert werden konnte.

Wir glauben jedoch, daß der Preis von DM 43,42 plus Mehrwertsteuer für diese Platine sicher nicht zu hoch liegt. Dies können Sie gerne bei jedem Platinenhersteller erfragen.

Wir hoffen, Ihnen hiermit geholfen zu haben. Dieser Leserbrief und die Antwort wird voraussichtlich in *LOOP* 8 abgedruckt werden.

Sehr geehrte Herren,

nach mehr oder weniger großen Schwierigkeiten läuft nunmehr mein NDR-Klein-Computer. Ich bin immer noch Anfänger, weil ich die Fernsehsendung nicht sehen kann. Bin also auf das „Buch" und das „Heft" Schritt für Schritt 1 angewiesen. Wobei mir das „Heft" am meisten gegeben hat. Warum geht es nicht weiter? Bei der Bestellung von Bauanleitungen erfuhr ich oft: nicht lieferbar!

Mein Problem ist es, den Computer nach *LOOP* zu programmieren. Nicht immer sind die Anweisungen klar und aufschlußreich. So habe ich auch den Einsender des Briefes Nr. 8 in *LOOP* 3, Seite 13, verstanden, er spricht mir aus der Seele. Ich bin sogar noch unwissender. Warum werden die Bilder nicht nummeriert? Was ist Bild, was Tabelle, wenn sie nicht immer entsprechend gekennzeichnet sind.

Ich könnte mir eine Anweisung verständlicher vorstellen, z.B. mit der Überschrift: „Programm in Grundprogramm: Bewegedreieck. Erforderlich SBC2, CPU-Voll oder 68 mit EPROM Grundprogramm". Ich könnte mir vorstellen, daß dafür viele Anfänger dankbar wären. Es wird immer wieder Anfänger geben. Vielleicht kann man bestimmte, zusammengehörige Artikel in Sonderdrucken gezielt zusammenfassen, um sich nicht immer wieder zu wiederholen. Kann man die Pro-

gramme nicht so verdeutlichen – zumindest für die Anfänger – daß erkennbar wird, was automatisch erscheint (Grundprogramm die Adressen), was ich in die Tastatur eingeben muß (BASIC10 moveto . . .) und was hierdurch ausgelöst wird. Ich glaube, daß – wie alle Spezialisten – auch Sie bei Ihrem Gesprächspartner das gleiche Wissen voraussetzen.

Weiterer Ausbau: (Der Appetit kommt beim Essen! Andererseits will man mit dem Ding ja nicht nur Striche oder Kreise ziehen, wenn es schon so viel Geld gekostet hat, dann soll es gefälligst auch etwas sinnvolles tun)!

Ich habe bis jetzt CPU-Vollausbau, ROA 64, GDP 64, KEY, IOE und CAS. Was kann ich erreichen, wenn ich welche Bausätze mit welchen Eproms anschaffe? Ich suche so etwas wie einen roten Faden, eine Grafik, wie ich sie in einer früheren ELO gesehen habe. Aber die dürfte schon überholt sein. Natürlich würde ich gern einen Drucker oder Plotter haben wollen, für Textverarbeitung und Datei. Wann ist eine Floppy-Disc erforderlich und warum? Wenn es etwas länger dauert, stört mich das nicht. Das alles, damit ich mir ausrechnen kann, das ich mir als Ruheständler noch erlauben kann.

Ich lese manchmal Angebote von preiswerten Rekordern, Laufwerken oder Druckern, aber kann ich sie anschließen? Wenn es z.B. heißt für apple, für Commodore. Dann suche ich Zeichnungen von selbstgebauten Adaptern.

Ich wäre Ihnen dankbar, wenn Sie mir hier weiterhelfen könnten, wobei ich hoffe, daß es mir gelungen ist, die Probleme eines Anfängers zu verdeutlichen.

Mit freundlichen Grüßen

Dipl.-Ing. Erich Rohweder,
Weberstraße 7, 4005 Meerbusch 1

Antwort *LOOP:*

Sehr geehrter Herr Rohweder,

herzlichen Dank für Ihr Schreiben. Wir sind gerade dabei, einen neuen umfangreichen Farbkatalog für den NDR-Computer zu erstellen, in dem die von Ihnen erwähnten Grafiken abgedruckt werden. Als *LOOP*-Abonnent werden Sie, sobald dieser Farbkatalog – vermutlich Mitte des Jahres – verfügbar ist, diesen kostenlos erhalten. Wir haben volles Veständnis, daß es Ihnen wie vielen Kunden geht, denen die Produktvielfalt schon etwas über den Kopf gewachsen ist.

Sehr gerne nehmen wir Ihre Anregung auf und machen die Bauanleitung und die Programmlistings noch übersichtlicher, wobei wir sie mit einem Code versehen, welche Konfiguration hier benötigt wird. Mittlerweile gibt es vom Franzis-Verlag das Heft „Schritt für Schritt II", das Sie selbstverständlich bei uns bestellen können.

Grundsätzlich versuchen wir, mit dem NDR-Computer einen Industriestandardbus und Anschlußbelegungen darzustellen. Grundsätzlich können Sie Standardgeräte wie Tastaturen mit paralleler Schnittstelle, Laufwerke mit Schugart-Schnittstellen, Drucker mit Centronics-Schnittstellen unserem Computer anschließen. Im Zweifelsfalle fragen Sie lieber bei uns an.

Vielleicht gibt es in Ihrer Region einige NDR-Computer-Benutzer, mit denen Sie sich zusammentun könnten, um solche und andere Fragen zu klären.

Wann wird die Baugruppe mit der CPU 68020 lieferbar, und zu welchem Preis?

Wäre es nicht möglich, einen starken BASIC wie z.B. in Macintosh oder der 520 ST, in EPROM zu entwickeln?

Auf meinem Bildschirm treten in regelmäßigen Abständen helle Streifen auf, die von oben nach unter wandern. Ich möchte gern wissen, ob es überhaupt ein Fehler ist, und wenn ja, wie könnte man ihn beheben? Ich habe schon andere Monitore und Fernseher ausprobiert, und den Änderungsvorschlag von *LOOP* durchgeführt, nichts hat sich geändert.

Ansonsten: Ich finde, daß *LOOP* für uns NDR-Computer-Anwender, unentbehrlich und in der jetzigen Form am besten ist, also – weiter so!

Hr Sabouret Joël,
Rosenstraße 6, 8492 Furth-im-Wald

Antwort *LOOP:*

Sehr geehrter Herr Sabouret,

vielen Dank für Ihre Vorschläge, wir werden sie aufnehmen.

Zu Ihren Fragen: CPU68020 wird ab April/Mai lieferbar sein. Die Baugruppe wird DM 1.698,— kosten (nur als Fertiggerät lieferbar).

Ein starkes Basic ist derzeit für den 68000 vorgestellt worden – siehe diese LOOP.

Ihre Bildschirmstreifen könnten von einer Brummspannung herrühren. Prüfen Sie bitte Ihr Netzgerät, ob hier unzulässig hohe Brummspannungen auftreten. Verwenden Sie probeweise ein Netzgerät aus unserem Lieferprogramm.

Vielen Dank für Ihre Kritik zur *LOOP.*

Sehr geehrter Herr Graf, sehr geehrter Herr Klein,

„in eigener Sache" verkünden Sie in *LOOP* Nr. 6, 1985 sei das „Jahr der Dokumentationen" gewesen. Wie so oft, klopfen Sie sich dabei gleich fleißig auf die eigene Schulter.

Also, nur weil inzwischen für jeden Ihrer Bausätze ein kleines Heftchen mit Beschreibungen etc. vorliegt, können Sie gewiß kein uneingeschränktes Lob für sich reklamieren. Leider werden nämlich Ihre „Dokumentationen" den Erwartungen Ihrer Kunden nur zum Teil gerecht. Gelungen sind in Wahrheit lediglich die neueren Hefte im DIN A5-Format. Wenn alle „Dokumentationen" für alle Ihre Baugruppen deren Standard erreicht haben und vor allem auch laufend aktualisiert werden, wird man einmal über Ihr „Jahr der Dokumentationen" sprechen können.

Dieses Jahr soll nun das Jahr der Software werden. Bitte, ja! Aber, denken Sie dabei an das folgende:

Nach meiner Einschätzung arbeiten die meisten Anwender des NDR-Klein-Computers mit dem 68008-Prozessor, was übrigens jedem zu empfehlen ist, der lernen will, wie Computer arbeiten und zu programmieren sind.

Für den 68000er gibt es aber bisher kaum kommerzielle Software. Derjenige unter den Anwendern des NKC, der damit nicht nur experimentieren, sondern auch einmal ernsthaft arbeiten will, muß deshalb auf die Z80-Software zurückgreifen, die es reichlich gibt, preiswert und erprobt.

Damit beginnt aber das, was mit dem an sich so ausgezeichneten modularen Konzept des NKC wirklich nicht zu vereinbaren ist, nämlich das immer lästige Umrüsten der Anlage auf den Z80 oder den 68000/68008. Warum bringen Sie nicht eine Schaltung, die den alternativen Betrieb des NKC mit dem Z80 oder dem 68000/68008 erlaubt, ohne daß die CPU-Karten ausgewechselt werden müssen? Und wenn Sie den alternativen Betrieb der beiden Prozessoren eröffnen, dann bitte Software-gesteuert und nicht nur durch eine Unterbrechung der Stromversorgung der einen oder andeen CPU.

Ganz besonders interessant wäre es außerdem, wenn auch noch jede CPU mit einer – wieder softwarefähig – auszuwählenden Sprache arbeiten könnte, man also beispielsweise den gerade aktiven 68000/68008 entweder mit Assembler oder Pascal programmieren kann.

Sie sehen, in Sachen Software für den NKC gibt es viel zu tun. Hoffen wir, daß 1986 wirklich Ihr „Jahr der Software" wird.

Mit freundlichen Grüßen

Dieter Wilke,
Op de Elg 12 a, 2000 Hamburg 65

P.S.: Ein kleines Listing zur Verbesserung der Arbeit mit dem NDR 68K nach dem Assemblieren (= "NACHASS") füge ich bei.

Antwort *LOOP:*

Sehr geehrter Herr Wilke,

vielen Dank für Ihren Brief und Ihre kritischen Anmerkungen. Mit der Schulter-

klopferei haben Sie recht, wir wollen uns in Zukunft nicht mehr selbst loben, sondern Taten sprechen lassen.

Die neuen Bauanleitungen sind der Schritt in die richtige Richtung – auch die „alten" Hefte werden Zug um Zug überarbeitet.

Nun zur Umschaltung.

Gewiß ist es, besonders mit den nun für den Z80 verfügbaren preiswerten Programmen sinnvoll, das System umschalten zu können.

Wir arbeiten gerade an einer Lösung, die ein Umschalten per Software möglich macht.

Die bisher uns bekannte trivialste Lösung sieht zwei Bank-Boot-Karten vor. Nun wird einfach durch einen simplen Umschalter entweder die CPUZ80 und die Bank-Boot-Karte mit dem 68008-Grundprogramm mit Spannung versorgt – die + 5V werden umgeschaltet. Nachteil dieses Verfahrens: Zwei Bank-Boot-Karten, beim Umschalten = Reset.

Vielen Dank für Ihr Listing, das wir gerne veröffentlichen.

Weiter viel Erfolg mit dem NKC und vergessen Sie bitte die Zwei-Jahres-Feier in Hamburg am Freitag, den 23. Mai nicht!

Ende 1983 habe ich mir das Buch von Herrn Klein „Mikrocomputer selbstgebaut . . ." gekauft. Meine Platinenbestellung erfolgte im Januar 1984 (Auslieferung 30. 1.). Somit war ich praktisch einer der ersten, der die neue, geänderte Version, also SBC2 erhielt. Nun habe ich ein altes Buch, aber neue Platinen und das neue Grundprogramm. Inzwischen habe ich im Buchhandel die zweite Auflage des Buches gesehen und mußte feststellen, daß es auch um runde 100 Seiten dicker geworden ist. Meine Frage: Kann ich irgendwo eine Art Sonderdruck kaufen, der die Änderungen der 2. gegenüber der 1. Auflage enthält?

Nun kommt ein weiteres Anliegen: In der Anlage sende ich Ihnen zwei kleine Programme, die ich mit dem NDR-Klein-Computer erarbeitet habe. Es handelt sich um Drucker-Programme zum Anschluß und Betrieb des kleinen Metallpapierdruckers aus dem Christiani-Kurs „Mikroprozessor-Labor" an den NDR-Klein-Computer. Möglicherweise haben einige *„LOOP"*-Leser diesen Kurs vor Jahren gemacht und haben das Ding noch irgendwo herumstehen. Mit Hilfe dieser Programme könnte man den Drukker noch sinnvoll einsetzen. Wenn Sie den Beitrag als nicht zu geringwertig einschätzen, würde ich damit auch in Ihrem *LOOP*-Wettbewerb teilnehmen.

Hans-Georg Farr
Bergbahnstraße 9, 7500 Karlsruhe 41

Drucker Programm: „Speicherausdruck" — schreibt je Zeile acht Speicherzelleninhalte, beginnend mit der Adresse der ersten.

Hauptprogramm

PR	8F 1F	ED	5B	10	8F	LD DE, (8F10)	Drucker-Startadresse erzeugen durch Addition der 1. Adr + Zeichenzahl-1 für jede Zeile. Ergebnis im Zeichenzeigerspeicher ablegen
	23	21	07	00		LD HL, 0007	
	26	19				ADD HL, DE	
	27	22	C6	8F		LD 8FC6, HL	
	2A	DD	21	00	8F	LD IX, 8F00	1. Adr. ASCII-String im Register IX ablegen
	2E	2A	12	8F		LD HL, (8F12)	Zeilenzähler laden
	31	CD	0F	00		CD SCHL	
	34	ED	5B	C6	8F	LD DE, (8FC6)	1. Zeichen abholen
	38	0E	08			LD C, 08	Zeichenzähler laden
	3A	CD	96	8F		CD PRUEF1	übernahmebereit ?
	3D	1A				LDA, (DE)	LOB des Zeichens codieren und ausgeben
	3E	CD	A4	8F		CD COLOB	
	41	CD	96	8F		CD PRUEF1	übernahmebereit ?
	44	1A				LDA, (DE)	HOB des Zeichens codieren und ausgeben
	45	CD	B2	8F		CD COHOB	
	48	CD	14	8F		CD BLANC	1 Blanc ausgeben
	4B	ED	53	C8	8F	LD 8FC8, DE	letzten Inhalt von DE zwischenspeichern
	4F	1B				DEC DE	(DE)-1
	50	0D				DEC C	C - 1
	51	79				LDA, C	alle Zeichen einer Zeile ausgedruckt !
	52	20	E6			JRNZ	
	54	ED	53	C6	8F	LD 8FC6, DE	letzten Stand v. DE retten + Zeilenanfang
	58	CD	96	8F		CD PRUEF1	
	5B	3A	C8	8F		LDA, (8FC8)	
	5E	CD	A4	8F		CD COLOB	
	61	CD	96	8F		CD PRUEF1	
	64	3A	C8	8F		LDA, (8FC8)	Zeilenanfangsadresse aus Zwischenspeicher holen, codieren und ausgeben
	67	CD	B2	8F		CD COHOB	
	6A	CD	96	8F		CD PRUEF1	
	6D	3A	C9	8F		LDA, (8FC9)	
	70	CD	A4	8F		CD COLOB	
	73	CD	96	8F		CD PRUEF1	
	76	3A	C9	8F		LDA, (8FC9)	
	79	CD	B2	8F		CD COHOB	
	7C	CD	14	8F		CD BLANC	1 Blanc ausgeben
	7F	ED	5B	C6	8F	LD DE, (8FC6)	neue Zeilenendeadresse erzeugen durch Addition des Zeichenzeigerstandes mit 18h und abspeichern
	83	21	18	00		LD HL, 0018	
	86	19				ADD HL, DE	
	87	22	C6	8F		LD 8FC6, HL	
	8A	3E	00			LDA, 00	Druckbefehl löschen
	8C	D3	30			OUT (30), A	
	8E	CD	12	00		CD ENSCHL	
	91	C3	00	00		ENDE	

Bedienung: 1.) Startadresse in 8F10 u. 11 laden
2.) gew. Zeilenzahl in 8F12 u. 13 laden
3.) Start mit Adresse „PR"

Blatt 2 (zu Druckerprogramm „Speicherausdruck")

Unterprogramme

Label	Adr					
BLANC	8F14	CD	96	8F		CD PRUEF1 — 1 St. Leerzeichen drucken
	17	3E	E0			LDA, E0
	19	D3	30			OUT (30), A
	1B	CD	9D	8F		CD PRUEF0
	1E	C9				RET

PRUEF1	8F96	DB	30			IN A, (30) — bereit für Zeichenübernahme !
	98	E6	01			AND 01
	9A	28	FA			JRZ
	9C	C9				RET

PRUEF0	8F9D	DB	30			IN A, (30) — Zeichen übernommen ?
	9F	E6	01			AND 01
	A1	20	FA			JRNZ
	A3	C9				RET

COLOB	8FA4	E6	0F			AND, 0F — HOB löschen
	A6	32	AB	8F		LD 8FAB, A — LOB ins Displ. d. folg. Befehls laden
	A9	DD	7E	00		LDA, (IX+d) — bek. ASCII-Zeichen abholen und ausgeben
	AC	D3	30			OUT (30), A
	AE	CD	9D	8F		CD PRUEF0
	B1	C9				RET

COHOB	8FB2	CB	3F			4x SRLA — LOB löschen
	B4	CB	3F			
	B6	CB	3F			
	B8	CB	3F			
	BA	32	BF	8F		LD 8FBF, A — HOB ins Displ. d. folg. Befehls laden
	BD	DD	7E	00		LDA, (IX+d) — bek. ASCII-Zeichen abholen und ausgeben
	C0	D3	30			OUT (30), A
	C2	CD	9D	8F		CD PRUEF0
	C5	C9				RET

8FC6	xx		Spezellen f. Zeichenzeiger
C7	xx		
8FC8	xx		Zwischenspeicher für (DE)
C9	xx		

Die Speicherzellen 8F00 bis 8F09 sind mit F0 bis F9 zu laden
" " 8F0A bis 8F0F " " C1 " C6 " "

Beispiel zum Druckerprogramm: „Speicherausdruck"

Der Drucker hat den Inhalt des Speicherbereiches, in dem sein eigenes Steuerprogramm steht, wie folgt ausgedruckt:

```
8F00 F0 F1 F2 F3 F4 F5 F6 F7
8F08 F8 F9 C1 C2 C3 C4 C5 C6
8F10 00 8F 1A 00 CD 96 8F 3E
8F18 E0 D3 30 CD 90 8F C9 ED
8F20 5B 10 8F 21 07 00 19 22
8F28 C6 8F DD 21 00 8F 2A 12
8F30 8F CD 8F 00 ED 5B C6 8F
8F38 0E 08 CD 96 8F 1A CD A4
8F40 8F CD 96 8F 1A CD B2 8F
8F48 CD 14 8F ED 53 C8 8F 1B
8F50 00 79 20 E6 ED 53 C6 8F
8F58 CD 96 8F 3A C8 8F CD A4
8F60 8F CD 96 8F 3A C8 8F CD
8F68 B2 8F CD 96 8F 3A C9 8F
8F70 CD A4 8F CD 96 8F 3A C9
8F78 8F CD B2 8F CD 14 8F ED
8F80 5B C6 8F 21 18 00 19 22
8F88 C6 8F 3E 00 D3 30 CD 12
8F90 00 C3 00 00 00 00 D8 30
8F98 E6 01 28 FA C9 DB 30 E6
8FA0 01 28 FA C9 E6 0F 32 AB
8FA8 8F DD 7E 03 D3 30 CD 90
8FB0 8F C9 CB 3F CB 3F CB 3F
8FB8 CB 3F 32 BF 8F DD 7E 00
8FC0 D3 30 CD 90 8F C9 C7 8F
8FC8 C9 8F 00 00 00 00 00 00
```

Drucker-Programm: „Schriftdruck" druckt Texte aus, die in einen Speicherbereich eingeschrieben wurden

88 00	xx	z.B.	00		Speicherzellen für Startadresse
01	xx		89		
02	xx		06		„ für Zeilenzahl
03	xx		00		

Hauptprogramm:

PR	8804	ED	5B	00	88	LD DE, (8800)
	08	21	1D	00		LD HL, 001D
	0B	19				ADD HL, DE
	0C	22	41	88		LD 8841, HL
	0F	2A	02	88		LD HL, (8802)
	12	CD	0F	00		CD SCHL
	15	0E	1E			LDC, 1E
	17	ED	5B	41	88	LD DE, (8841)
	1B	CD	43	88		CD AUSDR
	1E	1B				DEC DE
	1F	0D				DEC C
	20	79				LDA, C
	21	20	F8			JR NZ
	23	ED	53	41	88	LD 8841, (DE)
	27	CD	4F	88		CD BLANC
	2A	ED	5B	41	88	LDDE, (8841)
	2E	21	3C	00		LD HL, 003C
	31	19				ADD HL, DE
	32	22	41	88		LD 8841, HL
	35	3E	00			LDA, 00
	37	D3	30			OUT(30), A
	39	CD	12	00		CD ENSCHL
	3C	C3	00	00		ENDE
	88 41	xx				Zeilenende-Speicher
	42	xx				

Startadresse für Drucker erzeugen, durch Addition mit Zeichenzahl je Zeile −1 und im Zeilenendespeicher ablegen ($1D_H = 29_D$)

Zeilenzähler laden

Zeichenzähler laden

1. Zeichen abholen, Adr. steht im ZE-Speicher

ausdrucken

nächstes Zeichen adressieren

Zeichenzähler −1

alle Zeichen einer Zeile ausgedruckt.

letzten Stand v. DE retten

1 Blanc ausgeben

letzte Zeilenendeadresse holen, durch Addition mit $3C_H$ (= 60_D) die neue Zeilenadresse bilden und abspeichern

Druckbefehl löschen

Unterprogramme:

AUSDR	8843	CD	5A	88	CD PRUEF1
	46	1A			LDA, DE
	47	CB	FF		Set Bit 7
	49	D3	30		OUT(30), A
	4B	CD	61	88	CD PRUEF0
	4E	C9			RET

DER DRUCKER SCHREIBT NUR DANN KORREKT, WENN FOLGENDE BEDINGUNGEN ERFUELLT SIND: GENAU 30 ZEICHEN PRO ZEILE, KEIN BLANC AM ZEILENANFANG, AM ENDE AUFGEFUELLT MIT LEERZEICHEN!

BLANC	884F	CD	5A	88	CD PRUEF1
	52	3E	E0		LDA, E0
	54	D3	30		OUT(30), A
	56	CD	61	88	CD PRUEF0
	59	C9			RET
PRUEF1	885A	DB	30		INA, (30)
	5C	E6	01		AND01
	5E	28	FA		JRZ
	60	C9			RET
PRUEF0	8861	DB	30		INA, (30)
	63	E6	01		AND01
	65	20	FA		JRNZ
	67	C9			RET

FOLGENDE ZEICHEN SIND MOEGLICH
@ABCDEFGHIJKLMNOPQRSTUVWXYZ
[=]()!"&%$'()*+,-./0123456789
:;(=)?

Christiani-Metallpapier-Drucker
DIL 16-Stecker

NDR-Klein-Computer
IOE-Stecker zur Peripherie

Pin Nr.	Bezeichnung	Bedeutung	
7	CALL	Handshake 1= Übernahme bereit 0= „ gesperrt	an E_0
8	Masse	Bezugspotential	an $I/O_{0,0}$ / an Masse, 0V
9	DB7	Druckbefehl 1= Start 0= Stop	an $Out_{0,7}$
11	DB5	bit Nr. 5	an $Out_{0,5}$
12	DB4	bit Nr. 4	an $Out_{0,4}$
13	DB3	bit Nr. 3	an $Out_{0,3}$
14	DB2	bit Nr. 2	an $Out_{0,2}$
15	DB1	bit Nr. 1	an $Out_{0,1}$
16	DB0	bit Nr. 0	an $Out_{0,0}$

7 × 3K3

an +5V

Struktogramm zum Druckerprogramm: „Schriftdruck"

```
Start
Startadresse für Drucker erzeugen
Zeilenzähler laden
wiederhole
    Zeichenzähler laden
    1. Zeichen abholen u. umkodieren f. Druckerstart
    wiederhole
        Übernahmebereit - Handshake
        Zeichen holen u. ausdrucken
        Übernahmeende - Handshake
        Zeichenzeiger −1
        Zeichenzähler −1
    bis alle Zeichen einer Zeile ausgedruckt
    Zeilenende speichern
    Übernahmebereit - Handshake
    1 Blanc ausgeben
    Übernahmeende - Handshake
    neue Zeilenendeadresse errechnen und abspeichern
    Druckbefehl löschen
bis alle Zeilen ausgedruckt
```

so sieht das Programm im Speicher:

```
Adr.| Inhalte v. je 8 Speicherzellen
    |
8800 00 89 06 00 ED 5B 00 89
8808 21 1D 00 19 22 41 88 2A
8810 02 88 CD 0F 00 0E 1E ED
8818 5B 41 88 CD 43 88 1B 0D
8820 79 20 F8 ED 53 41 88 CD
8828 4F 88 ED 5B 41 88 21 3C
8830 00 19 22 41 88 3E 00 D3
8838 30 CD 12 00 C3 00 00 00
8840 00 D1 89 CD 5A 88 1A C8
8848 FF D3 30 CD 61 88 C9 CD
8850 5A 88 3E E0 D3 30 CD 61
8858 88 C9 DB 30 E6 01 28 FA
8860 C9 DB 30 E6 01 20 FA C9
```

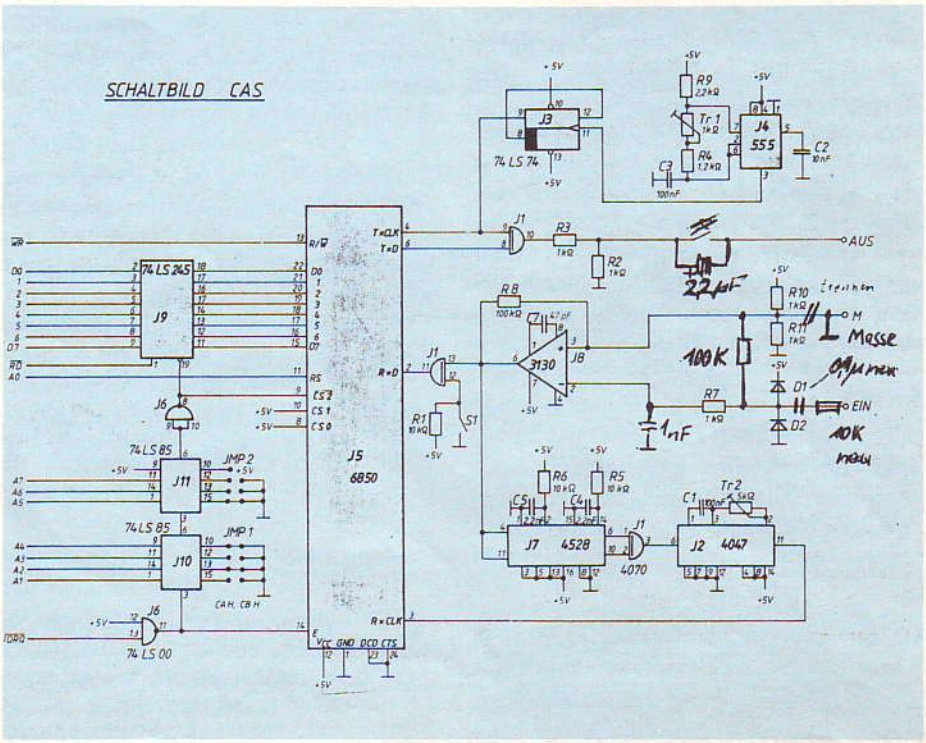

Messe in Mönchengladbach
23. 5. bis 25. 5. 1986
Computer-Tage '86
in der Kaiser-Friedrich-Halle
wir sind bei der Firma Symic vertreten – Besuchen Sie uns!

CAS-Verbesserung

Auch aus dem WORKSHOP in Berlin, der dort alle 14 Tage, donnerstags, stattfindet, erreichen die Redaktion einige Verbesserungsvorschläge für den NDR-Computer. So haben sich dort Herr Rolf Schulze und Herr Peter Gubela mit der Analogschaltung des Rechners ein wenig beschäftigt: der CAS-Baugruppe.

Vielen NDR-Computerbauern ist es trotz sorgfältigstem Aufbau dieser Baugruppe nicht gelungen, den eigenen Kassettenrekorder, wie versprochen, zur Datenaufzeichnung oder -wiedergabe zu bewegen.

Hier nun eine Lösung, die sich nun schon selbst bei den schwierigsten Fällen bewährt hat:



Der Schalter S2 entfällt, an dessen Stelle kommt ein Tantalkondensator 2,2uF, Plus an R2/R3, Minus an „AUS". Die Verbindung zwischen „M" und dem Knotenpunkt R10/R11 wird aufgetrennt und der Anschlußpunkt „M" direkt mit der Masse auf der Karte verbunden. Die Verbindung zwischen „EIN" und dem Knotenpunkt D1/D2/R7 wird auf der Bestückungsseite ca. 12 mm weggekratzt und ein neuer Widerstand 10 KOhm in Reihe mit einem Kondensator 100nF anstelle der Leiterbahn eingelötet. Von Knotenpunkt R10/R11 wird nun noch ein neuer Widerstand 100 KOhm nach Knotenpunkt D1/D2/R7 gelötet. Nun noch den schon gestrichelt eingezeichneten Kondensator 1nF an den invertierten (–) Eingang des Operationsverstärkers und Masse legen; fertig ist die Änderung.

Wen die senkrechten Streifen im Hintergrund auf dem Monitor stören, kann die GDP 64 K abändern: C1 gegen Elko 220uF tauschen, R5=150 Ohm in 100 Ohm ändern, R3= 470 Ohm in 510 Ohm ändern, zusätzlicher Elko mit Plus an Pin 14 und Minus an Pin 7 von IC 5.

Anmerkung der Redaktion:

Dieser Artikel hat uns ganz kurz vor Redaktionsschluß erreicht. Wir drucken ihn deshalb ab, ohne die Schaltung vorher im Labor zu testen!

Nachbauen sollten nur die *LOOP*-Leser, die tatsächlich Probleme mit ihrem Rekorder haben. Vielleicht könnten sie dann ihre Erfahrungen an die *LOOP*-Redaktion senden!

KONTAKTE

Suche Kontakte zu anderen begeisterten „NDR-Klein-Computer-Anwendern" im Raum Ravensburg (Schlier). Ich bin Schüler des Welfengymnasiums und 14 Jahre alt.

Tobias Schüle
Tödiweg 18, 7981 Schlier
Telefon (0 75 29) 71 18

Suche Kontaktaustausch zum 68008 PASCAL-System in der Umgebung von Hannover.

Carsten Malonnok
Wilh.-Raabe-Straße 16, 3257 Springe 2
Telefon (0 50 45) 63 69

Kontakt mit NDR-Klein-Rechnersystem-Anwendern Raum Frankfurt/Darmstadt.

Walter Wiewock
Kennedystraße 25, 6072 Dreieichenhain

Hinweis der *LOOP*-Redaktion: Solche Kontaktanzeigen werden von uns selbstverständlich kostenlos veröffentlicht. Wir behalten uns jedoch eine Kürzung, wie üblich, vor. Hinweise auf NDR-Benutzer-Clubs, Volkshochschulen und weiteres werden natürlich auch kostenlos veröffentlicht. Schreiben Sie uns!

Impressum:

LOOP, Zeitung für Computerbauer

Herausgeber: Gerd Graf

Redaktion: Rolf-Dieter Klein, Gerd Graf

Gestaltung und Druck:
Karl-Heinz Rieder, Kempten

Herstellung und Anzeigenverwaltung: GES GmbH
Magnusstraße 13, 8960 Kempten

Anzeigenpreisliste 1/84

Das geballte Wissen der 16-Bit-Technologie nutzen!

Neuerscheinung

Die Prozessoren 68000 und 68008

Rechnerarchitektur und Sprache im NDR-KLEIN-Computer. Von Rolf-Dieter **Klein.** Ca. 600 Seiten mit ca. 200 Abbildungen. Lwstr-gebunden. DM **78.–**
ISBN 3-7723-7651-7

Über das Buch:

Nach der Modernisierung und Erweiterung des NDR-KLEIN-Computers wartet die Fachwelt auf dieses neue Werk. Jeder kann sich jetzt die gebotene Leistungsfähigkeit der modernen 16-Bit-Technologie zunutze machen. Er benötigt dazu dieses neue Buch und die notwendigen Platinen.

Rolf-Dieter Klein führt in seiner bewährten Art in die Hardware der hochmodernen Prozessorfamilie 68000/68008 sowie der dazugehörigen Peripherietechnik ebenso ein, wie in die leistungsfähige Software. Der Leser lernt anfangs den Umgang mit dem Texteditor sowie dem Assembler und gelangt dann zu den höheren Programmiersprachen Pascal und Gosi. Die abschließende Erweiterung des Systems durch Floppy-Disk-Laufwerke führt hin zum Einsatz des professionellen Betriebssystems CP/M-68k.

Immer steht das Lernen durch Versuche im Vordergrund. Der modulare Computer eignet sich dafür in besonderer Weise und stellt außerdem sicher, nicht zu veralten.

Es ist die bewährte Art des Autors, die Mikrocomputertechnik im Selbstbau zu vermitteln. Auf diesem sicheren Weg erlangt der Anwender ein Niveau, von dem aus er erfolgreich in die professionelle Computertechnik einsteigen kann.

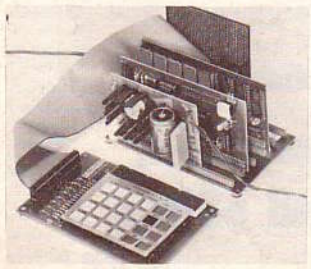
Franzis'

Franzis-Verlag, Postfach 37 01 20, München

Theorie und Praxis rund um den NDR-Computer

Mikroelektronik

Einführung



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

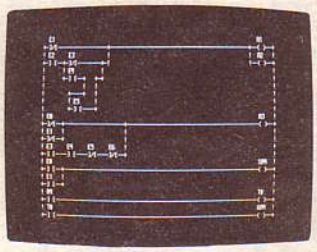
Der Kurs ist auf die HEXIO abgestimmt und ist für alle geeignet, die ihre ersten Schritte in Z 80-Maschinenprogrammierung machen.

Nach diesem Kurs sind Sie in der Lage, eigene Programme zu schreiben und die Arbeitsweise des Z 80 zu verstehen.

Der Kurs ist in verschiedene Fachgebiete aufgeteilt und bringt eine Menge Aufgaben, Beispielprogramme und Übungen.

Aus dem Inhalt: Was ist ein Mikroprozessor? * Inbetriebnahme des Computers * Planung von Programmen * Aufbau der CPU * Speicher und Adressen * Datentransfer * Lauflicht * Breakpoints * Hilfsfunktionen * Logo-Elemente * Strukturiertes Programmieren * Label & Call.

SPS-Programmierung



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Dieser Kurs zeigt Ihnen, wie SPS programmiert wird, die Normung, die Anwendungsmöglichkeiten und die verschiedenen Darstellungsarten.

Sie lernen spielend leicht, Relais- und Schützensteuerungen in SPS-Programme umzusetzen.

Beispielprogramme, Aufgaben und Übungen geben Ihnen die praktischen Erfahrungen und zeigen, wie SPS professionell eingesetzt wird. Nutzen Sie Ihren NDR-Computer für diese moderne Technik voll aus.

Der Kurs ist in folgende Fachgebiete gegliedert: Steuerungstechnik * Digitaltechnik * Methoden zur Beschreibung von Steuerungsaufgaben * Programmierung * Übungen und Tafeln.

ZEAT-Betriebssystem

Das Betriebssystem beeinhaltet in drei EPROMs: Z 80-2-Pass-Assembler, Disassembler, Editor, Debugger, Telefonmodem-Programm, FLOMON 1.5, ausserdem eine ausführliche Dokumentation zum Preis von DM 198,– .

Das Betriebssystem ZEAT benötigt 64-K-RAM (dynamische RAM-Karte). Die EPROMs werden in die BANKBOOT-Karte eingesteckt und sind sofort betriebsbereit. Programmieren Sie Ihren NDR-Computer mit einem Profi-Assembler.

Das Textverarbeitungsprogramm hat volle Bildschirmeditierung und kann neben der Programmeditierung auch zum Textschreiben eingesetzt werden.

Z 80-Assembler-Programmierung

4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,– je Kursteil

Der Kurs ist auf das ZEAT-Betriebssystem abgestimmt und zeigt Ihnen in leicht verständlicher Art, wie der NDR-Computer in Z 80-Assembler programmiert wird, bringt reichhaltig Übungsbeispiele und Anwendungen. Sie werden erstaunt sein, wie leicht diese Art der Programmerstellung ist. Und Sie lernen, wie man die serielle Schnittstelle bedient und Daten über Telefon übertragen kann.

Die Fachgebiete dieses Lehrgangs sind: Systembeschreibung * Betriebssystem * Programmierung * Testen * Modemprogramm * Listings, Tafeln und Tabellen.

Christiani

Hier abtrennen und im Umschlag einsenden an: Dr.-Ing. P. Christiani GmbH, Techn. Lehrinstitut und Verlag, Postfach 35 69189, 7750 Konstanz

Bestellcoupon

	Preis je Teil	Gesamtpreis
☐ Einführung mit dem NDR-Computer (4 Kursteile)	DM 38,--	DM 152,--
☐ Z 80-Assembler-Programmierung (4 Kursteile)	DM 38,–	DM 152,--
☐ SPS-Programmierung mit dem NDR-Computer (4 Kursteile)	DM 38,--	DM 152,--
☐ Kompakt-Kurs BASIC (angepasst an das RDK-BASIC)	DM 198,–	DM 198,--
☐ ZEAT-Betriebssystem (3 EPROMs mit Dokumentation)	DM 198,--	DM 198,--

Name, Vorname

Straße

PLZ, Ort

Datum Unterschrift 86189