LOOP 4
1. JAHRGANG

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter DM 3,–

In eigener Sache . . .

So langsam kriegen wir es in den Griff – diese *Loop 4* erscheint, wie in *Loop 3* versprochen, nunmehr rechtzeitig *im Zwei-Monats-Rhythmus!*

Sie haben uns dabei sehr geholfen – durch viele Artikel, die wir in diesem Heft veröffentlichen, und durch die steigende Zahl der Abonnenten, die uns helfen, *Loop* zu finanzieren.

Eins steht fest – *Loop wird die nächsten Jahre weiter erscheinen.* Wir müssen auch, denn wir haben bereits ein Abo für die nächsten Jahre angenommen . . . Wir planen, *Loop* mindestens alle zwei Monate erscheinen zu lassen.

Nun zu den Abos – viele Abos laufen mit Erscheinen der nächsten *Loop 5* aus. Wir wollen es Ihnen so leicht wie möglich machen: Ein Verlängerungs-Abo für die nächsten fünf Ausgaben kostet incl. Porto runde DM 20.—

Diese DM 20.— senden Sie uns per Scheck oder, falls Sie keinen Scheck besitzen, einfach in Form eines 20.— DM-Scheines. Sie erhalten dann eine Rechnung mit dem Vermerk – bezahlt; diese ist gleichzeitig Ihre Abo-Quittung.

Auf dem Aufkleber, auf dem Ihre Adresse vermerkt ist, ist markiert, bis zu welcher *Loop* Sie bezahlt haben. Nach dem Zahlungseingang müssen dann hier die neuen fünf bezahlten Ausgaben erscheinen.

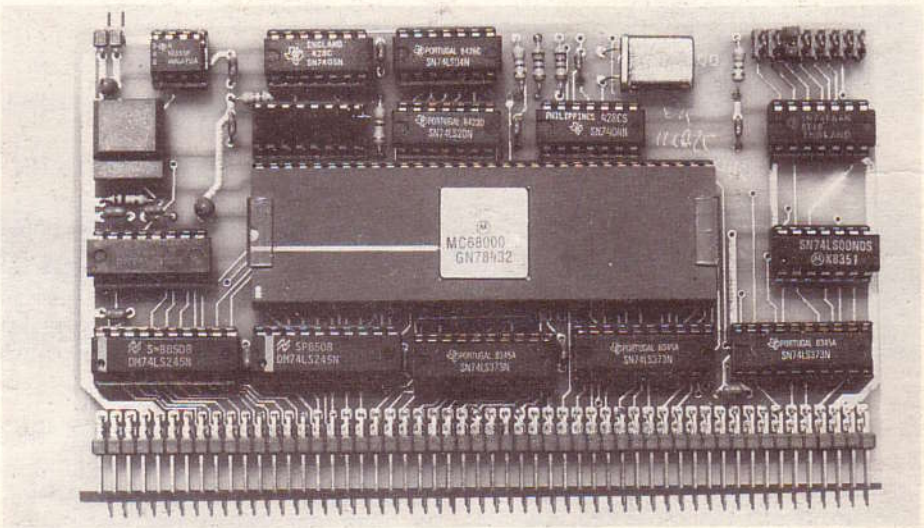
Wir bereiten uns schon auf die *Systems 85 in München* vor, bei der wir in Halle 22, OG, Stand B 7 (direkt neben Commodore . . .) ausstellen! Dort werden wir einige Überraschungen zeigen, die allerdings heute noch nicht verraten werden! Kommen Sie also zur Systems vom 28. 10. bis 1. 11. 85!

Vielen Dank für Ihre überwiegend positiven Zuschriften zur *Loop!* Der größte Wunsch, alles im Fotosatz drucken zu

(Fortsetzung nächste Seite)

Die CPU 68000 kommt!

16 bit – 12 MHz für den NDR-Computer



Rechtzeitig vor der System (Oktober 85) wird die CPU 68000 lieferbar sein, die Baugruppe mit der Motorola CPU 68000 – 12 MHz und 16 bit Datenbus.

Damit erhält der NDR-Computer eine „echte" 16 bit CPU.

Bei der CPU 68008, die in der Baugruppe CPU68K eingesetzt ist, wird zwar intern mit einem 16 bit breiten Datenbus gearbeitet, extern (also an den Beinchen der CPU) findet man nur einen 8 bit breiten Datenbus.

Die 16 bit Worte, die der 68008 verarbeitet, müssen also hintereinander im Speicher stehen. Daraus folgt, daß die CPU, um ein 16 bit Wort zu verarbeiten, zweimal in den Speicher greifen muß. Dies kostet natürlich Zeit.

Bei der CPU 68000 wird nun die echte 68000-CPU eingesetzt, die über einen 16 bit breiten Datenbus verfügt. Deshalb ist die CPU auch in einem größeren Gehäuse untergebracht. Um das modulare Konzept des NDR-Computers nicht zu durchbrechen, haben sich Herr Klein und Herr Graf folgenden Trick einfallen lassen:

Die Baugruppe CPU 68000 wird einfach in die Mitte des Busses (z. B. BUS3) gesteckt. Schauen Sie sich den BUS3 einmal genau an: Hier finden Sie in der Mitte eine zweireihige Leiste.

Die Datensignale D0 bis D7 und die Signale RD* und WR*, IORQ*, MREQ*, werden nun (durch aufkratzen) unterbrochen – sie liegen alle nebeneinander (Leitung 8 bis 19).

An diese Stelle wird nun die CPU 68000 gesteckt. Links und rechts der CPU 68000 kann man nun die normalen 8–bit breiten Speicher (ROA64), RAM64/256 usw.) stecken. Der Bus wird durch die in der Mitte steckende CPU einfach in eine untere und eine obere (Daten-) Hälfte aufgeteilt.

Mit diesem einfachen Trick ist eine Erweiterung des NDR-Computers auf 16 bit Datenbreite *ohne Einsatz neuer Baugruppen oder eines neuen Busses* problemlos möglich.

Fortsetzung: In eigener Sache

lassen, wurde mit diesem Heft erfüllt. Unser Wunsch an Sie: *Werben Sie Abonnenten!* Jeder neue *Loop*-Leser hilft uns bei der Finanzierung dieses Blattes.

Die Dokumentation des gesamten NDR-Computers wird derzeit auf ein DIN-A5-Ringbuchformat umgestellt. Die Handbücher werden also in Zukunft nicht mehr in geklammerter Form, sondern in gelochten Lose-Blatt-Sammlungen angeboten. Damit ist eine übersichtliche Ablage, das Einheften von eigenen Notizen und eine leichte Ergänzung und Erweiterung möglich. Natürlich werden auch die Ringbücher dazu angeboten.

Nochmals vielen Dank für Ihre Briefe, für Ihre Kritik und Zustimmung und für die vielen, teilweise ausgezeichneten Artikel, die wir erhalten haben und in dieser *Loop* abdrucken.

Rolf-Dieter Klein
Gerd Graf

Fortsetzung: Die CPU 68000 kommt

Damit können auch alle Ein-/Ausgaben weiter verwendet werden, die alle auf eine Hälfte des Busses gesteckt werden.

Was braucht also der „Aufsteiger", der von der CPU 68008 auf die CPU 68000 hochrüsten möchte?

Er braucht mindestens eine weitere ROA64, die Eproms des Grundprogrammes EG68000 und mindestens zwei R8-RAMs. Dieses Grundprogramm ist für den Benutzer *völlig identisch* mit dem EASS0-3, aber für den 16 bit breiten Datenbus ausgelegt.

Alle Ein-/Ausgaben sind auf gerade Adressen gelegt, wobei ein byteweiser Zugriff auf ungerade Adressen natürlich möglich ist. Auch hier gilt: Umtauschpauschale DM 10,— pro Eprom, also insgesamt DM 40,—.

Was bringt die CPU 68000 eigentlich?

Einen erheblichen Geschwindigkeitsvorteil und einen Adressraum von 2 MByte. Der Geschwindigkeitsvorteil ergibt sich einmal aus dem 16 bit Zugriff. Weiter läuft die CPU 68000 mit 12 MHz Taktfrequenz(!).

Nachteile?

Teuerer als die CPU 68008, sonst keine.

Tip

FLO2-Test mit GOSI

Zu unserem Schreiben vom 3. 5. 85 möchten wir ergänzen, daß die FLO2 jetzt auch mit der CPU68k läuft. Der Grund war die fehlende Beschreibung in Ihrem Handbuch FLO2 (Test mit CPU 68008). An dieser Stelle sei darauf hingewiesen, daß sich ein FL02-Test sehr gut mit GOSI realisieren läßt. Z.B:

Port 196 33
Port 195 32
Port 192 31
DZ: port 196
DZ: port 195
DZ: port 192

Man erhält dann gleich die Rückmeldung der Ports.

Heinz Voß
Fangkampstraße 19, D-4532 Mettingen,
Tel: 05452-1838

TEST: DAS ZEAT-BETRIEBSSYSTEM

Christiani Z80-Assembler

ZEAT – Ein Assemblerprogramm für den NDR-Klein-Computer

Das bekannte Lehrinstitut CHRISTIANI hat mit seinem ZEAT-Assemblerprogramm auf nur zwei 8 KB Eproms neue Zeichen gesetzt. Der Vorteil liegt in einem außergewöhnlichen Preis-/Leistungsverhältnis, einer vielseitigen Anwendungsmöglichkeit und hervorragender Dokumentation.

In den folgenden Zeilen werden grundsätzlich die Anwendungsmöglichkeiten aufgeführt, anschließend wir der Umgang mit dem System detaillierter beschrieben. Die bislang gebotenen Möglichkeiten, um einen Z80-Assembler zu betreiben, waren entweder ein 1-Pass-Zeilenassembler, der von CP/M gebotene Assembler mit Editor, oder einzeln kaufbare Software wie Macroassembler in Verbindung von Programmen wie Wordstar oder ähnlichem.

Der ZEAT-Assembler ist ein 2-Pass-Assembler, was bedeutet, daß die Festlegung aller Adressen vom Programm übernommen werden und somit viel Arbeit erspart bleibt. Der verwendbare Word-Star-Kompatible-Editor übertrifft den ED von CP/M insoweit wesentlich, daß hier Cursorsteuerungen, Löschfunktionen, Blockoperationen usw. möglich sind, welche sonst eigentlich nur von teueren Programmen angeboten werden. Außerdem können hier Z80-Mnemonic's (Mnemonic's = merkbare Kürzel, statt der Ziffern der Maschinensprache) verarbeitet werden, was sonst z. B. schon eines Macro-Assemblers bedarf. (Der CP/M-Assembler verarbeitet nur 8080 Befehle. Der Z80 stellt etwa doppelt soviele Befehle zur Verfügung als der 8080).

Ein sehr wichtiger Punkt ist der Disassembler im Einzelschritt-Modus, der ähnlich dem DDT-Programm vom CP/M arbeitet, aber sogar hier echte Z80 Mnemonic's verwendet.

Das ZEAT-Betriebssystem ist sowohl für den Assembler-Einsteiger wertvoll, da die erstellten Programme auf Kassetten abgelegt werden können, als auch für CP/M-Besitzer, die ihre Texte und Programme über Disketten verwalten (Zusatzprogramm auf Disketten notwendig).

Beide verfügen über die Möglichkeit, mit der Baugruppe „Promer" Eproms selbst zu „brennen".

Ein zusätzliches Angebot ist ein Modemprogramm, das Akustikkopplerbetrieb und somit Anschluß an die GES-Mailbox (siehe *Loop 3*) ermöglicht.

Hier noch einmal die Fähigkeiten in Kurzform:

- 2-Pass-Assembler
- Z 80 Mnemonic's
- Editor (Wordstar-ähnlich)
- Disassembler
- Z 80 Debugger (Fehlersuche)
- Kassetten- oder Diskettenbetrieb
- CP/M – „verträglich"
- Modemprogramm

Bis Herbst 1985 wird der 4 Teile umfassende Kurs „Microelektronik-Z80-Assembler" der Firma CHRISTIANI erwartet. Darin wird in hervorragender Didaktik das Programmieren in Assemblersprache, aber auch der Umgang mit dem ZEAT-Betriebsystem selbst ausführlich beschrieben. Der erste Teil „Microelektronik – Z80-Assembler-Programmierung mit dem NDR-Computer" ist bereits lieferbar. Er hat die Bestellnummer CASS1 und kostet DM 38.—.

Um für alle ZEAT-Benutzer die Zeit zu überbrücken, bis der oben erwähnte Kurs erhältlich ist, wird im nachfolgenden Teil, als Ergänzung zum beiliegenden Handbuch, ein Beispiel zur Anwendung des Betriebssystem beschrieben.

Die Grundkonfiguration (Ausstattung) besteht aus folgenden Baugruppen:

- CPU Z80
- BANKBOOT mit 3 ZEAT-Eproms und 1 R8 (Speicher) bestückt
- RAM 64/256 oder R0A 64 mit mind. 64 KB Arbeitsspeicher
- BUS, POW 5V oder NETZ 2, GDP 64k, KEY, Tastatur

Der Arbeitsspeicher von 64 KB dient später auch noch für den ohnehin benötigten Speicher für CP/M-Vollausbau. Das im ZEAT-Betriebssystem mitgelieferte FLOMON-Eprom ist eine angepaßte Form der bislang bekannten FLOMON-Version für CP/M-Betrieb.

Beispiel eines Assemblerprogramms:

Das CHRISTIANI-Betriebssystem wird aus dem FLOMON-Menü durch die Eingabe der Ziffer 3 gewählt. Es erscheint am Bildschirm:

CHRISTIANI
Z80
Edior
Assembler
Tester

Ein Programm entsteht folgendermaßen:

Das Assemblerprogramm wird im Editor (Quelle) geschrieben und anschließend assembliert. Sollte bis dahin ein Fehler aufgetreten sein, kann dieser durch ein Testprogramm gesucht werden. Ist der Fehler gefunden, muß dieser im Editor korrigiert und das ganze Programm neu assembliert werden. Wenn nun das Programm fehlerfrei ist, kann dieses abgespeichert und gestartet werden.

EDIT

Die Cursor-, Scroll-, Lösch-, Block- . . . Funktionen sind im beigefügten Handbuch ausführlich beschrieben.

Bevor das Programm gestartet wird, sollte es auf Kassette abgespeichert werden, da es sich „im Sande verlaufen könnte" und zerstört wäre. Dies geschieht im Editor: Drücken der Tasten „ESC" und „S", es erscheint *SAVE? Jetzt muß eine 4-stellige Zahl als Name angegeben werden (z.B. „0001"), Recorder einschalten, Taste „Return" bzw. „CR". Einlesen durch die Tasten „ESC" und „A". Das Abspeichern auf Diskette wird durch eine Zusatzdiskette ermöglicht, welche auch erlaubt, CP/M-Files zu bearbeiten.

Zusammengefaßt bietet EZEAT für einen sehr niedrigen Preis (derzeit DM 198.—) die Möglichkeit, den NDR-Computer auch schon ohne Floppy-Disk-Laufwerke zu einem ernstzunehmenden Z80-Entwicklungssystem zu machen!

Das Arbeiten mit einem größeren Assembler (z.B. M80) ist für kleine Programme wesentlich mühsamer, da nicht nur die Assemblierungszeit wesentlich höher ist, sondern auch die Übergänge vom Assembler zum Texteditor sehr lange dauern. Diese Programme müssen ja immer von der Diskette geladen werden, während sie bei EZEAT ROM-resident vorhanden sind.

```
     TITLE          ADDITION - 1

     ORG 100H       ;nach diesem Zeichen ";" nur Kommentar
                    ;ORG bedeutet daß das Programm auf die
                    ;Adresse (0100H) geschrieben wird
                    ;TITLE bedeutet nur Überschrift

Start:              ;So werden Symbole (Label) definiert
     IN   A,(30)    ;Inhalt von Port 30H (IOE) in Akku
     LD   B,A       ;Inhalt von Register A (Akku) nach B
     IN   A,(31)    ;Inhalt von Port 31H in A
     ADD  A,B       ;Addition von Port 30H und Port 31H
     OUT  A,(31)    ;Ergebnis auf Ausgang Port 31H (IOE)
     JP Start       ;Rücksprung zu Start
     END            ;Ende
```

```
ESC                 ;Mit Taste "ESC" und "E" wird der Editor
*E                  ;verlassen. Es erscheint der Promt ">".

>
>ASM                ;Jetzt wird der Assembler aufgerufen
                    ;ASM/L bedeutet Ausdruck auf Drucker

Z80 Assembler       ;Dieser stellt nach zwei Durchläufen
pass 1:             ;einen Fehler fest
+
pass 2:
+

L    OUT  A,(31)    ;"L" steht für: falscher Opcode
                    ;Fehler hier: erst Ziel, dann Quelle
                    ; d,h.  OUT    (31),A

1 fatal error(s)    ;dieser Fehler wird im Editor korrigiert
                    ;Aufruf: Edit ...neu assemblieren

>RUN                ;das Programm wird gestartet
                    ;auch wenn die IOE richtig auf 30H einge-
                    ;stellt ist, läuft das  Programm noch nicht

>TEST               ;jetzt wird der Tester um Rat gefragt
*L 100              ;Programm disassembliert aufgelistet
                    ;ab 100H

START.
0100  DB 1E   IN    A,(1E)
0102  47      LD    B,A
0103  DB 1F   IN    A,(1F)
usw.
                    ;schon ab der ersten Zeile ist erkenn-
                    ;bar, daß Portangabe in Dezimal erfolgte.
                    ;richtig ist:  LD   A,(30H)

*^C                 ;mit der Taste "CTRL" und "C" zurück

>                   ;alles im Editor korrigieren, neu
                    ;assemblieren, Programm steht

Die wichtigsten Testerfunktionen sind:

*X                  ;alle Register ausgeben
*L                  ;auflisten (auch "von,bis" möglich)
                    ;des Speichers (sedez. und disassembl.)
*G                  ;Programmstart (auch "von,bis" möglich)
*T                  ;Trace (Einzelschritt)
*D                  ;Dump d.h. auflisten des Speichers
                    ;(sedezimal und ASCII-Zeichen)
*READ               ;Eproms lesen
*PROG               ;Eproms programmieren
```

Da alle Systemaufrufe an EZEAT CP/M-kompatibel sind, ist es für „Aufsteiger" problemlos möglich, die unter EZEAT erstellten Programme auch unter CP/M zu betreiben. Umgekehrt können CP/M-M80-Programme vom Debugger getestet werden!

EZEAT ist eine sinnvolle Investition für alle, die den sehr weit verbreiteten Z80-Assembler lernen wollen oder die bereits mit diesem Assembler arbeiten und ein preisgünstiges Entwicklungssystem suchen.

ES WAR EINMAL ...

Es waren einmal drei Gescheite, die besaßen jeder einen Commodore-Rechner. Der erste war stolzer Besitzer eines CBM 8032, der zweite besaß einen C64 und der dritte konnte einen VC20 sein eigen nennen. Eines Tages faßten sie den Entschluß, an vordersten Front der Technik mitzumischen und bauten den NDR-Klein Computer. Als sie ihre Geräte fertig aufgebaut hatten, stellten sie fest, daß ihnen nun das Geld ausgegangen war. Zum Betrieb ihrer Geräte aber fehlten ihnen noch die Tastaturen. „Was sollen wir bloß machen", fragte der erste und blickte traurig in die Runde. Nach einiger Zeit aber sprang er hoch und rief: „Ich hab's. Wir verwenden unsere Rechner als Tastatur". Gesagt, getan. Schnell löteten sie Verbindungskabel, die übrigens bei allen Rechnern die gleichen waren, und verkabelten ihre Computer, natürlich mit Kupferkabel vom Postminister, wie es dem neuesten Stand der Technik entsprach. Nach dem sie soweit waren, mußten sie noch ein Programm schreiben, um die Daten richtig zu ermitteln. Der Gescheite mit dem C64 war als erster fertig und rief: „So nun kann ich anfangen". Mit seinem Programm funktionierten sogar die Cursortasten und die Löschtaste einwandfrei. Die anderen zwei hatten es da schon etwas schwerer. Bei ihren Computern konnten sie keine Kontrollkodes per Tastendruck erzeugen. Und so dachte sich der erste: „Wenn ich auf ‚escape' drücke, so soll die nächste Taste in einen Kontrollkode umgewandelt werden. Der mit dem VC20 machte das gleiche mit einer Funktionstaste (er hatte zwar eine Control-Taste, aber diese erzeugte nicht den entsprechenden Kode). Gleichzeitig verbesserte er das Programm. Er speicherte nämlich alle Tastendrücke auf Kassettenrekorder, um sie später automatisch zu überspielen. So erhielt er auch ein Speichermedium. Seine Funktionstasten belegte er mit ganzen Befehlswörtern und machte so seinen Computer zur intelligenten Tastatur. Aber auch der mit dem CBM 8032 wollte ein Speichermedium. Da er eine Diskettenstation besaß, beschritt er einen anderen Weg. Mit seinem Programm konnte er jeden Tastendruck sofort zum NDR-Klein-Computer senden. Oder aber er erstellte den gesamten Programmtext auf dem Commodore. Diesen konnte er nun editieren, speichern, drukken oder zum NKC überspielen. Nun waren die drei Gescheiten glücklich, denn ihre Arbeit ging gut vonstatten.

Aus jener Zeit wurde uns die Schaltung für das Verbindungskabel und das kurze Programm vom C64 überliefert. Und hoffentlich ein paar Anregungen.

```
Zur Verbindung:
Verwendet wird der Userport (unten).
Userport                KEY
   A                    0 V (Masse)
   B                    ---
   C                    D0
   D                    D1
   E                    D2
   F                    D3
   H                    D4
   J                    D5
   K                    D6
   L                    Strobe
   M                    ---
   N                    ---
Die Kodierschlitze befinden sich zwichen A-B und L-M.

Das Programm:
100 b = 56577
110 poke b+1, 255
120 wait 198, 1 : get a$ : c = peek (203)
130 a = asc ( a$ )
140 if a = 20 then a = 127
150 if a=17 and c=7 then a=24
160 if a=145 and c=7 then a=5
170 if a=29 and c=2 then a=4
180 if a=157 and c=2 then a=19
190 if a>64 and a<96 then a=a+32
200 a= a and 127
210 poke b, a : poke b, a or 128 : poke b, a and 127
220 goto 120
```

Zuerst werden die Adresse der CIA festgelegt und alle Pins auf Ausgang programmiert. Danach (Zeile 120) beginnt eine Endlosschleife. Hier werden ein Zeichen von Tastatur (a) und ein Datenkode (c) geholt. In Zeile 140 wird Commodore-delete in NKC-delete umgewandelt. Da die Cursortasten die falschen Controlcodes liefern (z.B. Control-Q = Cursor hoch oder runter) müssen auch diese gewandelt werden (Zeile 150 ... 180). a liefert hier den ASC-Wert und c dient als Information ob es sich um eine Cursortaste oder einen anders erzeugten, gleichen Code handelt. Schließlich muß der Commodore-Code in ASCII gewandelt und zum NKC gesandt werden.

Michael Müller
Pingsdorfer Str. 143 a, 5040 Brühl
Tel. 02232/47105

CP/M68K

CP/M68K – was ist es, was kann es?

Seit einiger Zeit wird das Betriebssystem CP/M68K ausgeliefert. Hier nun eine Übersicht, was es kann und für welche Hardware es verwendet werden kann.

CP/M68K stellt das bekannte CP/M-Betriebssystem für die 68000er-Linie dar. Das von uns gelieferte CP/M68K funktioniert mit dem 68008 (CPU68K), mit dem 68000-Prozessor (CPU 68000) und wird mit dem 68020 (CPU 68020 gegen Ende des Jahres) funktionieren.

CP/M68K ist ein Single-User-Single-Tasking Betriebssystem. Dies bedeutet: ein Benutzer und eine Aufgabe zugleich. Besonders für CP/M80-Benutzer ist es sehr leicht zu erlernen. CP/M68K wird mit einem C-Compiler ausgeliefert, um auch vernünftige Software-Entwicklungen machen zu können.

Das Betriebssystem belegt etwa 64 KByte des RAM-Speichers. Es ist nur ablauffähig ab mindestens 128K; wir empfehlen 256 KByte (RAM64/256 mit 256K-Bausteinen ausgerüstet. Grundsätzlich kann das Betriebssystem 16 MByte-RAM und 16 Disketten-Laufwerke verwalten, die jedes bis zu 512 MByte (!) Speicherkapazität haben können.

Ein wichtiger Punkt ist die Dateiverträglichkeit von CP/M2.2 und CP/M68K. Dateien, die unter CP/M2.2 erstellt worden sind, können von CP/M68K gelesen werden und umgekehrt. Somit ist ein leichter Datei-Transfer möglich. Die eingebauten Kommandos von CP/M, wie DIR, TYPE, PIP, STAT, ED, verhalten sich bei beiden Systemen gleich. Auch das Software-Interface ist identisch. Die Aufrufe an CP/M68K sind mit den gleichen Funktions-Nummern versehen, wie bei CP/M (und bei CP/M68). CP/M68k kann von minimal 64KByte (empfohlen jedoch 128 K) bis 16MByte-RAM-Speicher verwalten. Der Speicher kann beliebig im Adressbereich des Rechners liegen. Der User-Programm-Bereich muß jedoch durchgängig sein.

Die Sprache C:

Das CP/M68k beinhaltet einen C-Compiler und eine RUN-TIME-LIBRARY! Dieses „C" ist eine Untermenge der UNIX-Version 7. Damit ist eine Verbindung zu UNIX-Systemen geschaffen – C-Programme, die unter UNIX erstellt worden sind, können leicht unter CP/M68k ablaufen. Das Betriebssystem wird mit kompletter Dokumentation ausgeliefert. Es ist derzeit ab Lager lieferbar.

Grafik auf dem NDR-Klein-Computer mit BASIC

von Carsten Denneburg, März 1985

Sind Sie „Profi"? Dann tun Sie mir und vor allen Dingen sich selbst einen Gefallen – lesen Sie diesen Beitrag nicht, sondern beschäftigen Sie sich mit etwas sinnvollerem!

Falls nein, darf ich Sie als „Mit-Anfänger" begrüßen! Die nachfolgenden Zeilen enthalten einige Tips für die Erstellung von Grafiken, die Ihnen die Versuche ersparen sollen, die ich und sicher viele andere auch machen mußten, bis „die paar Grundlagen" begriffen waren. Hier geht es also nicht um „die Belegung des Seitenports ...", das „Setzen des Bits ...", sondern um einige handfeste Hinweise, damit auf dem Bildschirm in Sachen Grafik ersteinmal etwas passiert. Die Beschäftigung mit der Tiefe der Materie macht dann umsomehr Spaß – wetten?!

1. Die GDP

(Besser eigentlich der GDP, der GRAFIK-DISPLAY-PROZESSOR!) Diese Baugruppe, darum vielleicht doch die (?), ermöglicht dem Anwender über einen Bildspeicher von insgesamt 64 KB auf dem Bildschirm vier Bildseiten (Ø, 1, 2, 3) einzeln oder in Kombination miteinander zu beschriften und/oder darzustellen. D. h. pro Bildseite stehen 16KB zur Verfügung.

Verdeutlichen läßt sich das so: Sie haben vier durchsichtige Folien im Bildschirmformat. Die Folien Ø-3 können Sie beschriften, mit Zeichnungen versehen und sie dann einzeln betrachten oder übereinanderlegen. Sie wissen natürlich, daß auf diese Weise Zeichentrickfilme hergestellt werden? Und genauso entsteht die bewegte Grafik auf dem NDR-KLEIN-COMPUTER; aber dazu später.

Das BASIC des Rechners arbeitet stets mit den Seiten Ø+1. Wenn Sie dies nicht wollen und *immer* wenn Sie Grafik erzeugen wollen, müssen Sie mit dem Befehl *PAGE*, d. h. Seite, die Bildschirmseite (Folie!) vorwählen, die Sie beschriften wollen *und* gleichzeitig die Seite wählen, die abgebildet werden soll. Dies geschieht mit dem Befehl PAGE und dem Zusatz der Schreib- und Leseseite, also z.B. 10 Page 1,0

Das heißt: Schreibe (zeichne) ab Zeile 10 im Programm auf die Seite 1, stelle gleichzeitig die Seite 0 dar. Da vier Seiten zur Verfügung stehen, ist jede Kombination zwischen 0,0 und 3,3 möglich!

Wichtig ist, daß das BASIC bei Abarbeitung des von Ihnen erstellten Programms bei Erreichen bestimmter Befehle wie z.B. PRINT oder INPUT automatisch wieder auf die Seiten 0 und 1 im Wechsel umschaltet. Darum: Grafikprogrammteile immer an das Ende des Gesamtprogramms stellen. D.h. Programme nach Möglichkeit in der Reihenfolge Eingabeteil – Berechnungsteil – Ausgabeteil – Grafik aufbauen! Logischerweise darf im Grafikprogramm (Achtung, das kann bei Sprüngen oder Unterprogrammen schnell passieren!) oder nach Abschluß des Grafikteils niemals der Befehl CLRS = clear screen = Löschen des Bildschirms erscheinen.

2. Die Grafikelemente MOVETO und DRAWTO

Move to = bewege Dich nach
Draw to = zeichne nach

MOVETO bringt den gedachten Zeichenstift auf dem Bildschirm an die Stelle, an der es „losgehen" soll. Dieser Vorgang ist unsichtbar, d.h. auf unserer gedachten Folie setzen wir den Stift ab und bringen ihn – ohne zu zeichnen – von Position A nach B. Dort wird der Stift wieder aufgesetzt und harrt der Dinge, die da kommen sollen. In der Grundposition befindet sich der „Stift" unten links auf dem Bildschirm. Wie in einem Koordinatensystem stellt die untere, waagrechte Seite des Bildschirms die x-Achse mit den Werten unten links = 0, unten rechts = 511 und die linke, senkrechte Seite die y-Achse mit den Werten unten links = 0, oben links = 255, dar. Die Position des Zeichenstiftes bestimmen Sie nun mit MOVETO x, y. Ist der x-Wert größer als 511 oder der y-Wert größer als 255, liegt er außerhalb des Koordinatensystems und ist unsichtbar.

DRAWTO x, y (z.B. DRAWTO 100, 100) zeichnet eine Linie vom letzten Ort des Zeichenstifts zur mit DRAWTO angegebenen Position in Form einer Linie. Ohne ausdrücklichen neuen MOVETO-Befehl ist dieser Punkt dann der Ausgangspunkt für den nächsten DRAWTO-Befehl.

Ein Quadrat zeichnet sich also z.B. so:

10 CLRS	= Bildschirm löschen
20 PAGE 0,0	= Seite 0 beschreiben Seite 0 darstellen
30 MOVETO 100,50	= Ausgangspunkt x = 100, y = 50 einnehmen
40 DRAWTO 120,50	= 20 Schritte in x-Richtung zeichnen
50 DRAWTO 120,60	= 10 Schritte in y-Richtung aufwärts
60 DRAWTO 100,60	= 20 Schritte in x-Richtung zurück
70 DRAWTO 100,50	= 10 Schritte in y-Richtung 'runter und damit zum Ausgangspunkt

Daß die y-Werte in diesem Beispiel nicht 20, sondern nur 10 betragen, hängt damit zusammen, daß die y-Achse eben nur 255 Punkte, die x-Achse aber 511 Punkte, d.h. doppelt „so lang" ist. Der y-Wert muß bei Grafiken also in Gedanken immer halbiert werden!

3. Das „Flackern" der Grafik

Das Quadrat war ja schon sehr schön, nur dieses ewige Flackern im Rythmus des Cursors kann einen wahnsinnig machen – also Reset Taste drücken oder schnell den Befehl CLRS eingeben? Nein! Geben Sie zu Anfang eines Grafikprogramms immer zusammen mit dem Page-Befehl die Zeile

... POKE HEX ("87C5"), 0

ein. Also in unserem Beispiel in Zeilennummer 25. Damit greifen Sie unmittelbar auf Maschinenebene ein und können so – bis eine bessere BASIC-Version vorliegt – diese Macke des Systems beseitigen.

Übrigens, wenn eine Grafik bewußt „blinken" soll, können Sie die Geschwindigkeit (Frequenz) des „Blinkens" durch Veränderung des Wertes hinter dem Komma verändern, also z.B. POKE HEX ("87C5"), 2

4. Verschiedene Linien, Linien löschen

Ihr System verfügt über vier verschiedene Möglichkeiten, Linien zu zeichnen:

1. ——— = durchgehende Linie
2. - - - - - - - = kurz gestrichelte Linie
3. – – – – – = lang gestrichelte Linie
4. – · – · – · = Punkt-Strich Linie

Wenn Sie Erfahrungen mit technischen Zeichnungen haben, wissen Sie, wie wichtig das ist.

Um von einer Linienart in die andere umzuschalten, muß die GDP direkt von Ihnen beeinflußt werden. Dies geschieht mit dem Befehl OUT 114,.. also

OUT 114,1 = kurz gestrichelt
OUT 114,2 = lang gestrichelt
OUT 114,3 = Punkt-Strich
OUT 114,0 = durchgehende Linie

Mit OUT 114,0 wird der jeweilige Befehl wieder aufgehoben.

Als Beispiel:

```
10 CLRS
= Bildschirm löschen
20 PAGE 0,0
= Schreib- und Leseseite wählen
30 POKE HEX ("87C5"), 0
= „Flackern" beseitigen
40 MOVETO 100,50
= Ausgangsposition einnehmen
50 OUT 114,2
= Linienart wählen (gestrichelt)
60 DRAWTO 150,50
= zeichnen
70 OUT 114,0
= „normal" weiter
```

Das Löschen von gezeichneten Linien geht ähnlich einfach vor sich:

Auch hier ist es erforderlich, die GDP direkt anzusprechen. Die entsprechende Adresse mit dem Befehl OUT ist hier nicht 114, sondern 112.

OUT 112,1 schaltet auf Löschen. Der Zeichenstift wirkt dann wie ein Radiergummi (und zwar wie ein sehr guter!) und löscht alle Bildinformationen auf der vorgegebenen Linie aus. Mit dem Befehl OUT 112,0 wird der „Radiergummi" wieder ausgeschaltet und der Stift zeichnet wieder. Wenn wir das kleine „Quadrat-Programm" von vorhin nochmals eingeben und die Zeilen

80 MOVETO 120,50
= suche rechte, untere Seite auf

90 OUT 112,1
= schalte auf „radieren"

100 DRAWTO 120,60
= „radiere" rechte Seite aus

110 OUT 112,0
= schalte auf „schreiben"

ergänzen, wird es deutlich. Die rechte Seite des Quadrats wurde „ausradiert".

5. Verschiedene Schriften

Unser Wunderding von Rechner ist damit aber noch nicht am Ende. Als nächste Attraktion steht die Veränderung der Schrift ins Haus. Also Anschnallen und schon geht's los:

Wir benötigen wieder den Befehl OUT 114, .. Erneut ergeben sich vier Möglichkeiten:

OUT 114,1
= gerade Schrift

OUT 114,4
= Schrift nach rechts geneigt

OUT 114,10
= gerade Schrift „von unten nach oben"

OUT 114,12
= wie zuvor, ober nach rechts geneigt.

Ausprobiert? Der Ärger ist groß, die dritte und vierte Variante funktioniert nicht! Aber zunächst ein Beispiel für Variante 2:

```
10 CLRS
20 PAGE 0,0
(daran sollte man sich gleich gewöhnen!)
30 OUT 114,4
40 PRINT "Es klappt"
50 OUT 114,0
```

Es klappt tatsächlich. Und wenn Sie jetzt nochmals starten bzw. „listen" erscheint sogar wieder die „richtige" Schrift, weil wir intelligenterweise die Zeile 50 eingebaut und damit alles wieder in Ordnung, sprich in die richtige Schrift gebracht haben.

Aber warum nun Variante drei und vier nicht? Ganz einfach – BASIC beginnt immer „oben links" zu schreiben. Wenn wir aber schon ganz oben sind, können wir nicht noch höher, also auch nicht von „unten nach oben schreiben"! Wir suchen uns also für den Beginn der zu schreibenden Zeile eine Position, die weiter „unten" liegt. Wir müssen also wieder die x-, y-Position bestimmen und dorthin moven (MOVETO).

```
10  CLRS
20  POKE HEX ("87C5"), 0
30  X = 100 : Y = 50
40  Input "SCHRIFT"; S$
50  PAGE 0,0
60  OUT 114,10
70  MOVETO X, Y
80  FOR I = 1 TO LEN (S$)
90  OUT 112, ASC (MID$ (S$, I, 1))
100 NEXT
110 OUT 114,0
```

In Zeile 40 geben wir das, was geschrieben werden soll, als String-Variable, d.h. schlicht als Text ein.

Zeile 80: für die Länge dieser Variablen gebe in einer Schleife, von links beginnend, Buchstabe für Buchstabe des Textes an die GDP = Zeile 90. Die GDP ist nämlich so nicht in der Lage, den Text schlicht der Reihenfolge nach zu schreiben, sondern muß Buchstaben für Buchstaben genau nach Anweisung (ASCII-Code) malen. Na ja, wer schreibt auch schon von unten nach oben!

6. Vergrößern der Schrift

Funktioniert im Prinzip genauso wie die Veränderung der Schrift (nur hier OUT 115,.. usw.). Ich will darauf nicht näher eingehen, weil im Begleitheft der MC dargestellt. Ab Version 1.5 wird auch Scrollen usw. voll unterstützt

7. Bewegte Grafik

Nun wird es etwas komplizierter. Wir zeichnen ein Quadrat (wie gehabt) und lassen es von unten nach oben über den Bildschirm wandern. Im Prinzip funktioniert das so:

Der Zeichenvorgang läuft über das Unterprogramm ab Zeile 1000 ab.

Auf Seite 1 wird gezeichnet, Seite 0 dargestellt. Dann wird Seite 1 dargestellt und Seite und unmittelbar darauf Seite 1 gelöscht (Radiergummi!) und Seite 0 dargestellt. Der Y-Wert wird erhöht und es geht von vorne los. Das Programm:

```
10o  CLRS
110  POKE HEX("87C5"),0
120  X=200                 Ausgangspunkt für die
130  Y=10                  erste Zeichnung
140  FOR I= 1 TO 200       200 mal zeichnen
150  PAGE 1,0              Seite 1,0
160  OUT 112,1             löschen einschalten
170  GOSUB 1000            Unterprogramm und löschen
180  WAIT 112,2            warten bis fertig mit löschen
190  Y1=Y:Y2=Y+10          Hier werden die Werte für das
200  X1=X:X2=X+20          Quadrat festgelegt
210  Y=Y+1                 Y-Wert wird um 1 erhöht
220
230  OUT 112,0             auf Schreiben umschalten
240  GOSUB 1000            Quadrat zeichnen
250  PAGE 0,1              Seitenwechsel
260  WAIT 112,2            warten, b
270  NEXT                  Und von vorne!
280  END                   wichtig, da Unterprogramm!!
1000 MOVETO X,Y1           Grundposition des Quadrats
1010 DRAWTO X2,Y1          und dann zeichnen. Die Werte
1020 DRAWTO X2,Y2          für X bleiben gleich, die für
1030 DRAWTO X1,Y2          Y werden innerhalb der Schleife
1040 DRAWTO X1,Y1          erhöht.
1050 RETURN
```

Um die Geschwindigkeit der Bewegung zu verändern, muß lediglich der Y-Wert in Zeile 210 stärker verändert werden. Y = Y + 1 ist die langsamste Geschwindigkeit Y = Y + 30 erheblich schneller. Aber Achtung! Dann stimmt die Anzahl der Durchgänge in Zeile 140 nicht mehr, da das Quadrat dann aus dem Bild wandert. Dieser Wert ist also entsprechend zu verringern.

Durch entsprechende Veränderung der Zeilen 190 und 200 sowie des Unterprogramms lassen sich so alle nur denkbaren „bewegten" Figuren erstellen.

Ich würde mich freuen, wenn Sie von diesen Zeilen in der einen oder anderen Form profitieren könnten. Das war's – und nicht vergessen, ab und zu ausschalten, sonst gibt's viereckige Augen!

PROMER 80

von Rüdiger Bäcker

Schnelle Eprom-Programmierung mit dem NDR-Klein-Kleincomputer Z80

Für den NDR-Klein-Computer ist mit dem ZEAT-Betriebssystem ein neues „Grundprogramm" für die Z80 Konfiguration „Vollausbau CPU – Bankboot – 64k Ram" erhältlich. Es beinhaltet neben einigen anderen Routinen wie Editor, Modemprogramm etc. auch einen sehr guten Assembler. Mit diesem Assembler wird die Erstellung von Z80-Assemblerprogrammen zum Kinderspiel, zumal vom Lehrinstitut Christiani auch ein Kurs „Z80 Assemblerprogrammierung" angeboten wird.

Der Assembler ist eng an den „Profiassembler" Macro 80 angelehnt, der meiner Meinung nach das beste ist, was man zur Entwicklung von Assemblerprogrammen bekommen kann. Der Assembler Macro 80 läuft unter CP/M 2.2 und besitzt einige sehr leistungsfähige Pseudo-Befehle, die es ermöglichen, sehr gut strukturierte Programme zu entwickeln. So sind beispielsweise die Robotersteuerung, das Grundprogramm und viele andere Programme mit dem Macro 80 erstellt worden.

Nun stehen also mit den beiden Assemblern Entwicklungswerkzeuge zur Verfügung, die fast keine Wünsche mehr offen lassen. Was noch fehlt, ist eine Möglichkeit, die entwickelten Programme in ein Eprom zu „bannen". Will man beispielsweise ein Programm entwickeln, das später auf einem anderen System (z.B. SBC2) laufen soll, so muß das Programm in einem Eprom auf das andere System übernommen werden.

Das ZEAT-Betriebssystem benutzt die selben Routinen des Flomon zur Ein- und Ausgabe auf dem Bildschirm wie CP/M. Daher ist es möglich ein Programm auf dem einen System zu entwickeln und auf dem anderen System ablaufen zu lassen.

Doch zurück zur Eprom-Programmierung. Das nachstehende Programm ermöglicht das Auslesen und Programmieren von Eproms. Es läuft sowohl unter CP/M als auch mit dem ZEAT-Betriebssystem. Zur Programmierung wird ein sehr schneller Algorithmus verwendet, der es ermöglicht, ein Eprom vom Typ 2764 in ca. 30 Sekunden vollständig zu programmieren. Mit dem Standardalgorithmus würde dies über 7 Minuten dauern. Ein 2732 wird in ca. 15 Sekunden programmiert.

Der Aufruf des Programmes erfolgt unter CP/M mit „qprom". Man kann dann wählen zwischen „Eprom lesen", „Eprom programmieren" und „Return". Mit Return gelangt man zum Betriebssystem zurück.

Beim Lesen bzw. Programmieren eines Eproms werden die Startadresse und Endadresse innerhalb des Eproms ingegeben. Hier ist wichtig, daß die Eingaben in sedezimaler Form erfolgen, wobei Buchstaben groß geschrieben werden müssen. Beispiel: „bis = 1FFF".

Nach erfolgter Eingabe erscheint eine weitere Menuezeile. Hier wird mit „s" der Programmiervorgang gestartet. Hat man einen Fehler bei der Eingabe gemacht, so kann man mit „f" wieder an den Anfang der Eingaberoutine gelangen und die Eingabe wiederholen. Mit „m" gelangt man zum Hauptmenue zurück. Die Taste „w" hat hier noch keine Bedeutung. Nach erfolgter Programmierung erhält man eine OK-Meldung. Es erscheint dann auch wieder die Menuezeile. Drückt man nun „w", so kann ein weiteres Eprom programmiert werden. Zu beachten ist noch, daß vor der Programmierung der dem Eprom entsprechende DIL-Stecker in die Fassung gesteckt werden muß.

Das Listing zeigt die Version für CP/M. Wie bereits erwähnt ist es ohne Änderungen unter ZEAT lauffähig. Man kann es beim Zeatsystem auf Kassette abspeichern und bei Bedarf laden. Sollte das Programm in ein Eprom programmiert werden, so ist darauf zu achten, daß die Pufferspeicher in einem Rambereich liegen.

Hinweis: Dieses Programm ist auf der Diskette „CP/M80-Werkzeuge" enthalten.

```
Prommer 80 - V 1.0 - 26.06.85  MACRO-80 3.4   29/12/84       Seite  1


                        title   Prommer 80 - V 1.0 - 26.06.85

                        ;       P R O M M E R   8 0
                        ;       Routinen zum programmieren und lesen
                        ;       von Eproms mit dem NDR-Klein-Computer
                        ;       unter CP/M 2.2
                        ;       Fastprogrammroutine
                        ;       COPYRIGHT (C) 1985 by
                        ;       Ruediger Baecker - Postfach 4111 - 5820 Gevelsberg 11

                         .Z80

0000'                   aseg

                        org 100h

0100                    begin:
0100    C3 0103         jp entry

                        ; Konstantenvereinbarungen

00B0                    proad           equ             0B0h    ; Portadressen Prommer
00B1                    proma1          equ             0B1h
00B2                    proma2          equ             0B2h
0000                    sysret          equ             0       ; Systemresetadresse
0002                    conout          equ             2       ; Code fuer Consolausg. BDOS
0001                    conin           equ             1       ; dto. Eingabe
0005                    bdos            equ             5       ; Einsprungadresse BDOS

0103                    entry:
0103    21 02CD         ld      hl,entrytxt
0106    CD 01CA         call    textaus
0109    CD 0207         call    textein                 ; Zeichen holen
010C    FE 31           cp      '1'
010E    CA 020F         jp      z,promread
0111    FE 32           cp      '2'
0113    CA 0120         jp      z,pwr
0116    FE 33           cp      '3'
0118    FE 72           cp      'r'
011A    CA 0202         jp      z,retcpm
011D    C3 0103         jp      entry                   ; wenn keine gueltige Eingabe

0120                    pwr:                    ; hier Entry des Prg.
0120    21 0275         ld      hl,opentext     ; Adresse des openertextes in hl
0123    CD 01CA         call    textaus         ; und ausgeben
0126    CD 01DD         call    getpar          ; von, bis und nach holen
0129                    pwr2:
0129    21 03E1         ld      hl,menuetxt     ; s: Start f: Fehler w: weiter m: Return
012C    CD 01CA         call    textaus         ; ausgeben
012F                    pwr1:
012F    CD 0207         call    textein         ; Zeichen holen
0132    FE 73           cp      's'             ; s, dann Programmierstart
0134    CA 0149         jp      z,progst
0137    FE 66           cp      'f'             ; Fehler, ebenfalls zu Start
0139    CA 0120         jp      z,pwr
013C    FE 77           cp      'w'
013E    CA 0120         jp      z,pwr
0141    FE 6D           cp      'm'             ; m = zum Hauptmenue
0143    CA 0103         jp      z,entry         ;
0146    C3 012F         jp      pwr1            ; sonst Eingabe nochmal

0149                    progst:
0149    21 03B7         ld      hl,progtxt      ; 'Bitte warten'

Prommer 80 - V 1.0 - 26.06.85  MACRO-80 3.4   29/12/84       Seite  1-1


014C    CD 01CA         call    textaus
014F    3E 00           ld      a,0h            ; 0 in a
0151    32 042C         ld      (loopm),a       ; und in Impulszaehler
0154    2A 042D         ld      hl,(vonmerk)    ; von in hl
0157    ED 5B 042F      ld      de,(bismerk)    ; bis in de
015B    ED 4B 0431      ld      bc,(nachmerk)   ; nach in bc

015F                    prglp:                  ; Programmierschleife
                                                ; hl = von, de = bis, bc = nach, in a ist Byte
015F    3E 80           ld      a,10000000b
0161    D3 82           out     (proma2),a      ; tristate

0163                    prglp1:
0163    7E              ld      a,(hl)
0164    D3 80           out     (promd),a       ; Byte raus
0166    FE FF           cp      0ffh            ; FF ?
0168    CA 019F         jp      z,nextbyte      ; ja, dann naechstes Byte
016B    79              ld      a,c
016C    D3 81           out     (proma1),a      ; Adresse im Eprom
016E    78              ld      a,b
016F    E6 1F           and     00011111b       ; LED an, Enable aus, kein prgm Puls
0171    F6 80           or      10000000b       ; nun enable
0173    D3 82           out     (proma2),a      ; und an Steuerport
0175    F6 20           or      00100000b       ; Programmierimpuls
0177    D3 82           out     (proma2),a      ; ausgeben
0179    E6 DF           and     11011111b       ; Trigger ruecksetzen
017B    D3 82           out     (proma2),a
017D    F5              push    af              ; retten
```

```
017E                    trready:            ; wartet, bis prgapuls zuende
017E  DB 81             in      a,(proma1)  ; trigger testen
0180  CB 47             bit     0,a         ; "1" = warten
0182  C2 017E           jp      nz,trready  ; bis "0"
0185  F1                pop     af          ; zurueckholen
0186  E6 1F             and     00011111b   ; lesen
0188  D3 82             out     (proma2),a
018A  DB 80             in      a,(promd)   ; Byte holen
018C  BE                cp      (hl)        ; Byte ok ?
018D  CA 019F           jp      z,nextbyte  ; dann weiter
0190  3A 042C           ld      a,(loopm)   ; schleifenzaehler holen
0193  FE 03             cp      03h         ; schon dreimal und noch nicht ok ?
0195  CA 01C1           jp      z,error     ; dann Fehler
0198  3C                inc     a           ; sonst Schleifenzaehler erhoehen
0199  32 042C           ld      (loopm),a   ; und zurueckschreiben
019C  C3 0163           jp      prglp1      ; dann Byte nachprogrammieren


019F                    nextbyte:
019F  23                inc     hl          ; Adresse 'von' erhoehen
01A0  03                inc     bc          ; Adresse 'nach' ebenso
01A1  E5                push    hl          ; hl und de retten
01A2  D5                push    de
01A3  ED 52             sbc     hl,de       ; fuer Vergleich ob alles progrm.
01A5  7D                ld      a,l
01A6  B4                or      h
01A7  CA 01B4           jp      z,ende      ; ja, alles fertig -> ende
01AA  D1                pop     de          ; nein, weiter - holen de und hl
01AB  E1                pop     hl
01AC  3E 00             ld      a,0h        ; 0 in a
01AE  32 042C           ld      (loopm),a   ; und in Impulszaehler
01B1  C3 0163           jp      prglp1      ; und weiter programmieren
```



```
01B4                    ende:
01B4  21 03AA           ld      hl,oktext   ; Adresse fuer Oktext in hl
01B7  CD 01CA           call    textaus     ; und ausgeben
01BA  3E 40             ld      a,01000000b ; Prommer neutral stellen
01BC  D3 82             out     (proma2),a
01BE  C3 0129           jp      pwr2    ; und zum Start

01C1                    error:
01C1  21 0387           ld      hl,errortxt ; Adresse fuer Fehlertext in hl
01C4  CD 01CA           call    textaus     ; und ebenfalls ausgeben
01C7  C3 0129           jp      pwr2        ; und zum Start

01CA                    textaus:
01CA  7E                ld      a,(hl)      ; Zeichen holen
01CB  CB 7F             bit     7,a         ; Bit 7 gesetzt ?
01CD  C2 01DC           jp      nz,textende ; ja, dann Endezeichen
01D0  5F                ld      e,a         ; Zeichen fuer BDOS-Aufruf laden
01D1  E5                push    hl          ; retten
01D2  0E 02             ld      c,conout    ; code fuer Consolausgabe in c
01D4  CD 0005           call    bdos        ; BDOS aufrufen
01D7  E1                pop     hl          ; hl zurueck
01D8  23                inc     hl          ; auf naechstes Zeichen zeigen
01D9  C3 01CA           jp      textaus     ; und weiter ausgeben

01DC                    textende:
01DC  C9                ret                 ; zurueck zum aufrufenden Programm

01DD                    getpar:             ; holt Adressen vom user - Uprg. auch fuer
                                            ; Promread !
01DD  21 0362           ld      hl,vontxt   ; von ?
01E0  CD 01CA           call    textaus
01E3  CD 0241           call    wert        ; Wert holen
01E6  22 042D           ld      (vonmerk),hl ; und retten
01E9  21 036D           ld      hl,bistxt   ; bis ?
01EC  CD 01CA           call    textaus
01EF  CD 0241           call    wert        ; holen
01F2  22 042F           ld      (bismerk),hl ; retten
01F5  21 037A           ld      hl,nachtxt  ; nach ?
01F8  CD 01CA           call    textaus
01FB  CD 0241           call    wert        ; holen
01FE  22 0431           ld      (nachmerk),hl ; auch retten
0201  C9                ret                 ; als Unterprogramm

0202                    retcpm:             ; springt zum System zurueck
0202  0E 00             ld      c,sysret    ; Befehlscode in C
0204  CD 0005           call    bdos        ; BDOS aufrufen

0207                    textein:
0207  E5                push    hl          ; wird in BDOS-Routine veraendert
0208  0E 01             ld      c,conin     ; Zeichen von Tastatur holen
020A  CD 0005           call    bdos        ; jetzt Zeichen in a
020D  E1                pop     hl
020E  C9                ret

020F                    promread:
020F  21 0419           ld      hl,readtxt
0212  CD 01CA           call    textaus
0215  CD 01DD           call    getpar
0218  2A 042D           ld      hl,(vonmerk)
021B  ED 5B 042F        ld      de,(bismerk)
021F  ED 4B 0431        ld      bc,(nachmerk)

0223                    reloop:
```



```
0223  79                ld      a,c
0224  D3 81             out     (proma1),a
0226  78                ld      a,b
0227  E6 1F             and     00011111b   ; lesen
0229  D3 82             out     (proma2),a
022B  DB 80             in      a,(promd)
022D  02                ld      (bc),a      ; in Speicher ablegen
022E  23                inc     hl
022F  03                inc     bc
0230  E5                push    hl
0231  D5                push    de
0232  ED 52             sbc     hl,de
0234  7D                ld      a,l
0235  B4                or      h
0236  CA 023E           jp      z,readend   ; wenn Ende
0239  D1                pop     de          ; sonst weiter
023A  E1                pop     hl
023B  C3 0223           jp      reloop

023E                    readend:
023E  C3 0103           jp entry            ; zum Hauptmenue


0241                    wert:               ; jetzt Wert nach hl holen
0241  21 0000           ld      hl,0        ; Anfangswert
0244  CD 0207           call    textein     ; ein Zeichen holen
0247  47                ld      b,a         ; retten
0248  CD 0265           call    nibble      ; Zahl ?
024B  78                ld      a,b         ; alten Wert
024C  30 02             jr      nc,ex1      ; weiter, wenn ok
024E  37                scf                 ; Carry = keine gueltige Zahl
024F  C9                ret                 ; in a zeichenwert

0250                    ex1:
0250  47                ld      b,a
0251  CD 0265           call    nibble      ; test in schleife
0254  38 0B             jr      c,ex2       ; ok, exit ausfuehren
0256  29                add     hl,hl
0257  29                add     hl,hl
0258  29                add     hl,hl
0259  29                add     hl,hl       ; hl * 16 = hl
025A  B5                or      l
025B  6F                ld      l,a         ; wert von akku
025C  CD 0207           call    textein
025F  18 EF             jr      ex1

0261                    ex2:
0261  78                ld      a,b
0262  37                scf
0263  3F                ccf
0264  C9                ret                 ; in a = terminatorzeichen

0265                    nibble:             ; test ob Zahl
0265  D6 30             sub     '0'
0267  D8                ret     c
0268  FE 17             cp      'G'-'0'
026A  3F                ccf
026B  D8                ret     c
026C  FE 0A             cp      0ah
026E  3F                ccf
026F  D0                ret     nc
0270  D6 07             sub     'A'-'9'-1
0272  FE 0A             cp      0ah
```



```
0274  C9                ret

                        ; Texte

0275                    opentext:
0275  1B 1B 47 5A       defb    1bh,1bh,'GZA',1eh   ; Bildschirm loeschen und HOME
0279  41 1E
027B  51 20 55 20       defb    'Q U I C K P R O M  B 0 - V 1 . 1',0dh,0ah,0ah,0ah
027F  49 20 43 20
0283  4B 20 50 20
0287  52 20 4F 20
028B  4D 20 20 20
028F  38 20 30 20
0293  20 2D 20 20
0297  56 20 20 31
029B  20 2E 20 31
029F  0D 0A 0A 0A
02A3  43 6F 70 79       defb    'Copyright (C) 1985 by Ruediger Baecker',0dh,0ah,0ah,80h
02A7  72 69 67 68
02AB  74 20 28 43
02AF  29 20 31 39
02B3  38 35 20 62
02B7  79 20 52 75
02BB  65 64 69 67
02BF  65 72 20 42
02C3  61 65 63 6B
02C7  65 72 0D 0A
02CB  0A 80

02CD                    entrytxt:
02CD  1B 1B 47 5A       defb    1bh,1bh,'GZA',1eh       ; Bildschirm loeschen und HOME
02D1  41 1E
02D3  50 20 52 20       defb    'P R O M M E R  B 0',0dh,0ah,0ah,0ah
02D7  4F 20 4D 20
02DB  4D 20 45 20
02DF  52 20 20 38
02E3  20 30 0D 0A
02E7  0A 0A
02E9  43 6F 70 79       defb    'Copyright (C) 1985 by Ruediger Baecker',0dh,0ah,0ah
02ED  72 69 67 68
02F1  74 20 28 43
02F5  29 20 31 39
02F9  38 35 20 62
```

```
02FD   79 20 52 75
0301   65 64 69 67
0305   65 72 20 42
0309   61 65 63 6B
030D   65 72 0D 0A
0311   0A
0312   42 69 74 74          defb    'Bitte waehlen :',0dh,0ah,0ah
0316   65 20 77 61
031A   65 6B 6C 65
031E   6E 20 3A 0D
0322   0A 0A
0324   3C 31 3E 09          defb    '<1>    =       Eprom lesen',0dh,0ah
0328   3D 09 45 70
032C   72 6F 6D 20
0330   6C 65 73 65
0334   6E 0D 0A
0337   3C 32 3E 09          defb    '<2>    =       Eprom programmieren',0dh,0ah
033B   3D 09 45 70
033F   72 6F 6D 20
0343   70 72 6F 67

Prommer B0 - V 1.0 - 26.06.85  MACRO-80 3.4   29/12/84       Seite  1-5

0347   72 61 6D 6D
034B   69 65 72 65
034F   6E 0D 0A
0352   3C 72 3E 09          defb    '<r>    =       System',0dh,0ah,'>',80h
0356   3D 09 53 79
035A   73 74 65 6D
035E   0D 0A 3E B0

0362                        vontxt:
0362   76 6F 6E 20          defb    'von ==> ',80h
0366   20 3D 3D 3E
036A   20 20 B0

036D                        bistxt:
036D   0D 0A 62 69          defb    0dh,0ah,'bis ==> ',80h
0371   73 20 20 3D
0375   3D 3E 20 20
0379   B0

037A                        nachtxt:
037A   0D 0A 6E 61          defb    0dh,0ah,'nach ==> ',80h
037E   63 68 20 3D
0382   3D 3E 20 20
0386   B0

0387                        errortxt:
0387   0D 0A 0A 0A          defb    0dh,0ah,0ah,0ah,'Eprom nicht zu programmieren !',80h
038B   45 70 72 6F
038F   6D 20 6E 69
0393   63 68 74 20
0397   7A 75 20 70
039B   72 6F 67 72
039F   61 6D 6D 69
03A3   65 72 65 6E
03A7   20 21 B0

03AA                        oktext:
03AA   0D 0A 0A 0A          defb    0dh,0ah,0ah,0ah,'Eprom ok',80h
03AE   45 70 72 6F
03B2   6D 20 6F 6B
03B6   B0

03B7                        progtxt:
03B7   0D 0A 0A 45          defb    0dh,0ah,0ah,'Eprom wird programmiert - bitte warten'
03BB   70 72 6F 6D
```

```
03BF   20 77 69 72
03C3   64 20 70 72
03C7   6F 67 72 61
03CB   6D 6D 69 65
03CF   72 74 20 2D
03D3   20 62 69 74
03D7   74 65 20 77
03DB   61 72 74 65
03DF   6E
03E0   80                   defb    80h

03E1                        menuetxt:
03E1   0D 0A 0A 0A          defb    0dh,0ah,0ah,0ah
03E5   73 3A 20 53          defb    's: Start   f: Fehler   w: weiter   a: Hauptmenue'
03E9   74 61 72 74
03ED   20 20 20 66
03F1   3A 20 46 65
03F5   68 6C 65 72

Prommer B0 - V 1.0 - 26.06.85  MACRO-80 3.4   29/12/84       Seite  1-6

03F9   20 20 20 77
03FD   3A 20 77 65
0401   69 74 65 72
0405   20 20 20 6D
0409   3A 20 48 61
040D   75 70 74 6D
0411   65 6E 75 65
0415   0D 0A 3E B0          defb    0dh,0ah,'>',80h

0419                        readtxt:
0419   0D 0A 0A 0A          defb    0dh,0ah,0ah,0ah
041D   45 70 72 6F          defb    'Eprom lesen',0dh,0ah,0ah,80h
0421   6D 20 6C 65
0425   73 65 6E 0D
0429   0A 0A B0

                            ; Pufferspeicher - Wichtig, muss RAM sein !

042C                        loopm:      ds 1    ; Merker fuer Anzahl der Prgmversuche
042D                        vonmerk:    ds 2    ; Merker Adresse von
042F                        bismerk:    ds 2    ; Merker Adresse bis
0431                        nachmerk:   ds 2    ; Merker Adresse nach

                            end begin

Prommer B0 - V 1.0 - 26.06.85  MACRO-80 3.4   29/12/84       Seite  S

Macros:

Symbole:
BDOS    0005   BEGIN   0100   BISMER  042F   BISTXT  036D
CONIN   0001   CONOUT  0002   ENDE    01B4   ENTRY   0103
ENTRYT  02CD   ERROR   01C1   ERRORT  0387   EX1     0250
EX2     0261   GETPAR  01DD   LOOPM   042C   MENUET  03E1
NACHME  0431   NACHTX  037A   NEXTBY  019F   NIBBLE  0265
OKTEXT  03AA   OPENTE  0275   PRGLP   015F   PRGLP1  0163
PROGST  0149   PROGTX  03B7   PROMA1  0081   PROMA2  00B2
PROMD   00B0   PROMRE  020F   PWR     0120   PWR1    012F
PWR2    0129   READEN  023E   READTX  0419   RELOOP  0223
RETCPM  0202   SYSRET  0000   TEXTAU  01CA   TEXTEI  0207
TEXTEN  01DC   TRREAD  017E   VONMER  042D   VONTXT  0362
WERT    0241


Keine  Fehler
```

Zugriff auf Flomonroutinen unter CP/M

Wenn man mit dem Z80-System des NDR-Klein-Computers unter dem Betriebssystem CP/M 2.2 arbeitet, so übernimmt das Flomon auf der Bank-Boot-Karte das Lesen und Schreiben von der Floppy, die Bildschirmausgabe, Zeicheneingabe von der Tastatur etc.

Außerdem sind dort auch eine große Anzahl sehr leistungsfähiger Unterprogramme für die Grafikausgabe vorhanden. Diese Befehle werden mittels Steuerzeichen übergeben. Diese Steuerzeichen können von beliebigen Programmen, also auch von Hochsprachen aus übergeben werden. Man kann hier beispielsweise von Turbo-Pascal oder von Hebas auf die Grafik zugreifen.

Aber auch im sogenannten Alpha-Modus sind einige interessante Routinen im Flomon erreichbar. Als Beispiel ist die Routine zur Umschaltung zwischen deutschem und internationalem Zeichensatz zu nennen.

Was muß man also tun, um dem Flomon mitzuteilen, daß man ihm bestimmte Zusatzfunktionen „entlocken" möchte?

Man muß nur die der Funktion entsprechenden Befehle übergeben. Das geschieht genauso wie die Umschaltung verschiedener Schriftarten und Sonderfunktionen bei Druckern. Die Steuerzeichen werden als sogenannte Escapesequenzen übergeben, das sind Zeichenfolgen, die durch das Zeichen ESC eingeleitet werden. Zum Umschalten der Zeichensätze wird die Sequenz „ESC z n" übergeben, wobei für n eine 1 einzusetzen ist, wenn man den deutschen Zeichensatz benutzen will und 0, wenn mit den amerikanischen Zeichen gearbeitet werden soll.

Man kann das einmal im Testmodus des Flomon probieren. Drücken Sie nach dem Einschalten des Rechners die Tasten CONTROL und C, um in den Testmodus zu gelangen und dann die obenstehende Steuerzeichenfolge.

Von Hochsprachen aus kann man diese Zeichenfolge mit den Befehlen übergeben, die ein Zeichen auf dem Bildschirm darstellen. Als Beispiel finden Sie hier zwei kleine Routinen, die mit Turbo-Pascal erstellt wurden.

```
program geinit (output);
begin
write (chr(27), 'z1');
end.
```

```
program usinit (output);
begin
write (chr(27), 'z0');
end.
```

Dabei dient das Programm geinit zum Umschalten auf deutsche Zeichen und das Programm usinit zum Umschalten auf internationale Zeichen. In beiden Routinen wird mit dem Befehl write die Zeichenfolge ausgegeben. chr(27) ist hierbei das Zeichen ESC, dessen dezimaler Wert 27 ist. Nun „weiß" das Flomon, daß nun noch Steuerzeichen folgen. Es werden dann noch ein kleines z und, abhängig vom gewünschten Zeichensatz eine 0 oder eine 1 übergeben.Turbo-Pascal bietet nun die Möglichkeit, COM-Files zu erzeugen, das sind Files, die von CP/M direkt ausführbar sind. Wir können also die Programme compilieren (übersetzen) lassen und auf beliebige andere Disketten copieren. Man kann dann beispielsweise von Wordstar aus über die Funktion „R" = aufrufen anderer Programme durch „geinit" auf den deutschen Zeichensatz umschalten.

Natürlich kann auch von anderen Hochsprachen oder aus Assemblerprogrammen auf die Steuerfunktionen zugegriffen werden. In Basic sähe das dann so aus "PRINT CHR$(27); "z"; "0". In Assembler sieht das so aus.

```
title Umschaltung Zeichensatz US/GE
 .z80

system    equ  0005h     ; Adresse für BDOS-Aufrufe
warmst    equ  0000h     ; Adresse für Warmstart
console   equ  2         ; Funktionscode für Consolausgabe

org  100h                ; für diese Adresse assemblieren

start:
ld   e,1bh               ; 1Bh = Code für ESC
call zeiout              ; dann Zeichen ausgeben
ld   e,'z'               ; z in Register e
call zeiout              ; und ebenfalls ausgeben
ld   e,'1'               ; 1 = deutsche Zeichen
call zeiout              ; ausgeben
ld   c,warmst            ; Befehlscode für Warstart in Register C
call system              ; und dann wieder zum CP/M zurück

zeiout;                  ; gibt Zeichen aus e auf Console
ld   c,console           ; Befehlscode für Consolausgabe laden
call System              ; ausgeben
ret                      ; und Return

end  start
```

Dieses Programm wurde mit dem Assembler Macro 80 erstellt, mit dem die Entwicklung von Assemblerprogrammen recht komfortabel möglich ist.

Will man nun komplexere Datenfolgen übergeben, so ist dies auch nicht problematisch, wie Sie aus dem kleinen Basicprogramm ersehen können. Es handelt sich dabei um ein kleines Spiel, bei dem mit einer Kanone ein Ziel zu treffen ist. Um die Sache ein wenig zu erschweren, steht zwischen Kanone und Ziel ein Berg als Hindernis. Position von Berg und Ziel werden bei jedem Start zufällig gewählt, ebenso die Größe des Berges.

Die Grafikausgabe übernimmt hier das FLOMON, dem mit der Übergabe der Zeichenkette G$ mitgeteilt wird, daß nun Grafikbefehle folgen. Dem String G$ wurde im Vereinbarungsteil des Programmes die Zeichenfolge "ESC ESC G" zugewiesen. Man kann so sehr leicht öfter wiederkehrende Befehlsfolgen definieren und im Programm dann aufrufen. Dies funktioniert in Assembler ebenso. Eine Liste aller Befehle des Flomon finden Sie im MC-Sonderheft „Mikrocomputer Schritt für Schritt 2". Die Vielfalt der Befehle läßt erkennen, welche Möglichkeiten im FLOMON stecken!

```
PAGE  1


6 REM COPYRIGHT (C) 1985 BY : RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
7 REM SOURCE LANGUAGE : HEBAS V 1.1
10 PRINT CHR$(27);CHR$(27);"GZA"
15 LET G$=CHR$(27)+CHR$(27)+"G"
20 VG=0
30 N=INT(RND(1)*250)
40 PT=100000
50 DIM P(512)
60 GOSUB 610
70 N=0
80 XZ=INT(RND(1)*512)
90 IF XZ <450THEN 80
100 IF XZ+5>512 THEN 80
110 PRINT G$
120 PRINT "M";XZ;YZ
130 PRINT "D";XZ;3
160 PRINT "D";XZ+5;3
170 PRINT "D";XZ+5;0
180 PRINT "A"
200 PRINT G$
210 PRINT "M";0;0
220 PRINT "D";512;0
230 PRINT "A"
240 X0=0
250 Y0=0
260 PRINT CHR$(30); "ENTFERNUNG DES ZIELES IN METERN: ";XZ*50;"-";(XZ+5)*50
270 PRINT AW,:INPUT"WINKEL";W
280 AW=W
290 MZ=(XZ+2.5)*50
300 W=W*3.14/360
310 PRINT V0,:INPUT"STAERKE";V0
320 PV=V0/2
330 VX=V0*COS(W)
340 VY =V0*SIN(W)
345 PRINT G$;"M";0;0
350 G=15
360 T=0
370 H=20
380 VG=VG+1
390 Y=-G/2*T*T+VY*T
400 IF Y<0 AND X<(XZ*50) THEN ZD = (XZ*50)-X
410 IF Y<0 AND X> (XZ*50) THEN ZD = X-(XZ*50)
420 IF Y<0 THEN GOSUB 770
430 IF Y<0 AND X/50 > XZ AND X/50 < XZ+5 THEN 570
440 IF Y<0 THEN 260
450 X=VX*T
460 N =INT(X/50)
470 IF INT(X/50) >511 THEN 520
480 IF INT(Y/50) <P(N) THEN GOSUB 770
490 IF INT(Y/50) <P(N) THEN 260
500 PRINT G$
515 PRINT "D";INT(X/50);INT(Y/50)
520 L=X
530 H=Y
540 T=T+1
550 N=N+1
560 GOTO 390
570 PRINT "GETROFFEN":PRINT "VERSUCHE GESAMMT : ";VG
580 INPUT "NOCHMAL (J/N) ";A$
590 IF A$="J" THEN RUN
600 STOP
610 REM BERG
620 FOR K = 1 TO INT(RND(1)*150)
630 HA=INT(RND(1)*(K/2)+K)


PAGE  2


640 PRINT G$
650 PRINT "M";N;0;"D";N;HA
660 P(N)=HA
670 N=N+1
680 NEXT K
690 FOR K = K TO 0 STEP - 1
700 HA = INT(RND(1)*(K/2)+K)
710 PRINT "M";N;0;"D";N;HA
720 P(N)=HA
730 N=N+1
740 NEXT K
750 PRINT "A"
760 RETURN
770 PRINT "A";CHR$(30);TAB(50);"EINSCHLAG BEI: ";INT(X);" M"
775 ZD = INT(ZD)
780 PRINT TAB(50);"ENTF. ZUM ZIEL: ";ZD;" M"
790 PT=PT-(ZD*VG)-PV
800 PRINT TAB(50);"PUNKTE : ";PT
810 RETURN
```

ASSEMBLERPRINT 68

von Rüdiger Bäcker

Programm zum Ausdrucken von 68008-Assemblerlistings mit Epson-Druckern

Mit dieser kleinen Routine werden einige kleine Hindernisse beim Ausdruck von 68008 Assemlerlistings beseitigt.

Da beim Ausdruck von Listings oftmals mehr als 80 Zeichen pro Zeile auf den Drucker bzw. Bildschirm ausgegeben werden, kommt es oft vor, daß bei der Verwendung von Endlospapier, auf der Abreißkante geschrieben wird.

Für die Benutzer von Epsondruckern bzw. kompatiblen Geräten kann mit einer kleinen Routine leicht Abhilfe geschaffen werden.

Assemblerlistings werden dann im Schmalschriftmodus ausgedruckt, so daß nun alle Zeichen in einer Zeile stehen. Ist die Ausgabe des Listings beendet, so meldet sich der Drucker durch ein Akustisches Signal.

Das Programm ist relokativ und kann aus der Bibliothek aufgerufen werden.

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


020000                  ORG $20000
020000
020000                  * A S S E M P R I N T  6 8
020000                  * DRUCKROUTINE FUER EPSONDRUCKER
020000                  * COPYRIGHT (C) 1985 BY :
020000                  * RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG 11
020000
020000                  KOPF:
020000  55AA0180        DC.L $55AA0180
020004  412D5072696E74  DC.B 'A-Print
02000B  20
02000C  0000002B        DC.L Q-KOPF
020010  000000D6        DC.L ENDEA-KOPF
020014  01              DC.B 1
020015  00 00 00        DC.B 0,0,0
020018  00000000        DC.L 0,0
02001C  00000000
020020
020020
= 0000001B              ESC     EQU     $1B
020020
020020                  LO:
020020  3E3C 0016       MOVE #!LO,D7
020024  4E41            TRAP #1
020026  4E75            RTS
020028
020028                  Q:                      * ENTRY
02002B  303C 001B       MOVE #ESC,D0            * DRUCKER AUF NORMALMODUS
02002C  6100 FFF2       BSR LO
020030  303C 0040       MOVE #'@',D0
020034  6100 FFEA       BSR LO
020038
020038                  INITZE:                 * ZEICHENSATZ AUF AMERIKANISCH
020038  303C 001B       MOVE #ESC,D0
02003C  6100 FFE2       BSR LO
020040  303C 0052       MOVE #'R',D0
020044  6100 FFDA       BSR LO
020048  303C 0000       MOVE #0,D0
02004C  6100 FFD2       BSR LO
020050
020050                  INITSI:                 * DRUCKER AUF SCHMALSCHRIFT
020050  303C 001B       MOVE #ESC,D0
020054  6100 FFCA       BSR LO
020058  303C 000F       MOVE #15,D0
02005C  6100 FFC2       BSR LO
020060
020060                  INITZA:                 * ZEILENABSTAND VORBELEGEN
020060  303C 001B       MOVE #ESC,D0
020064  6100 FFBA       BSR LO
020068  303C 0001       MOVE #1,D0              * 1/8 * ZEILENABSTAND
02006C  6100 FFB2       BSR LO
020070
020070                  INITFL:                 * FORMULARLAENGE SETZEN
020070  303C 001B       MOVE #ESC,D0
020074  6100 FFAA       BSR LO
020078  303C 0043       MOVE #'C',D0
02007C  6100 FFA2       BSR LO
020080  303C 0048       MOVE #72,D0
020084  6100 FF9A       BSR LO
020088
020088                  INITLR:                 * LINKEN RAND SETZEN
020088  303C 001B       MOVE #ESC,D0
02008C  6100 FF92       BSR LO


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

020090  303C 006C       MOVE #'l ',D0
020094  6100 FF8A       BSR LO
020098  303C 000F       MOVE #15,D0
02009C  6100 FF82       BSR LO
0200A0
0200A0  3E3C 0032       MOVE #!LST,D7           * UMSCHALTEN AUF DRUCKERAUSGABE
0200A4  4E41            TRAP #1
0200A6  3E3C 0041       MOVE #!ASSEMBLE,D7      * DANN ASSEMBLER AUFRUFEN
0200AA  4E41            TRAP #1
0200AC  303C 0007       MOVE #7,D0              * PIEPSER ALS FERTIGMELDUNG
0200B0  6100 FF6E       BSR LO
0200B4  303C 000C       MOVE #12,D0             * FORMULAR FEED
0200B8  6100 FF66       BSR LO
0200BC  303C 001B       MOVE #ESC,D0            * DRUCKER AUF NORMALMODUS
0200C0  6100 FF5E       BSR LO
0200C4  303C 0040       MOVE #'@',D0
0200C8  6100 FF56       BSR LO
0200CC  3E3C 0034       MOVE #!NIL,D7           * NUR FEHLERAUSGABE
0200D0  4E41            TRAP #1
0200D2  4E75            RTS                     * ZURUECK ZUM GRUNDPROGRAMM
0200D4
0200D4
0200D4
0200D4
0200D4  4E75            RTS
0200D6
0200D6                  ENDEA:
0200D6

0EBB02  Ende-Symboltabelle
```

Die Sprache C

Wenn man mit CP/M68K arbeitet, so kommt man nicht umhin, sich auch mit der Sprache C zu beschäftigen, denn neben Programmen, wie Assembler, Editor usw. wird auch ein kompletter C-Compiler mit CP/M68K geliefert. C ist eine höhere Programmiersprache, wie zum Beispiel auch PASCAL eine ist, die es erlaubt, mit komfortablen Befehlen Programme zu schreiben.

Was ist der Unterschied zwischen Assemblersprache und einer höheren Programmiersprache? Beim Assembler wird Befehl für Befehl in entsprechenden Maschinencode übertragen. Zum Beipiel der Assemblerbefehl RTS in den Maschinencode 4E75. Ein Assemblerbefehl entspricht dabei im wesentlichen immer einem bestimmten Maschinencode.

Umgekehrt gibt es für jeden Maschinencode des Prozessors auch einen entsprechenden Assemblerbefehl. Jeder Prozessor besitzt eine eigene Assemblersprache, so der Z80, der 68000 usw. Bei höheren Sprachen ist das anders. Zunächst einmal ist eine höhere Sprache unabhängig vom verwendeten Prozessor. Damit werden Programme übertragbar. In einer höheren Programmiersprache erzeugt ein Befehl meist eine Gruppe von Maschinenbefehlen. Auch gibt es nicht für alle vorhandenen Maschinencodes einen entsprechenden Befehl in der Hochsprache, aber dadurch wird die Hochsprache unabhängig vom jeweils verwendeten Prozessor.

Die Sprache C enthält viele Ideen von anderen Hochsprachen, wie z.B. PASCAL. C besitzt aber auch Befehle, die sich insbesondere für Steueraufgaben gut eignen.

C verbreitet sich zunehmend, und Programme die in C geschrieben wurden, laufen nach der Übersetzung auch schnell ab.

Nun in die Praxis. Zum Arbeiten mit C benötigen Sie das CP/M68k-Betriebssystem (für den Z80 gibt es natürlich auch C-Compiler, die dann unter CP/M80 laufen) und zwei Diskettenlaufwerke. Zunächst formatieren Sie eine neue Diskette mit Hilfe des Programms UFORMCPM, das sich auf der Systemdiskette befindet. Dann kopieren Sie die wichtigsten C-Programme auf diese neue Diskette, sowie ein paar Programme der Systemdiskette.

```
CP/M-68K(tm) Version 1.2  03/20/84
Copyright (c) 1984 Digital Research, Inc.


A>dir
A: CLINK  ✓SUB : C     ✓  SUB : CE    ✓  SUB : CP68  ✓68K : C068  ✓68K
A: C168   ✓68K : LIBF      A  : GSATTR    H  : GSIF      H  : LIBE      A
A: CPM      H  : PORTAB  ✓H  : SETJMP   ✓H  : ERRNO   ✓H  : STDIO   ✓H
A: ASSERT ✓H   : OSIFERR ✓H  : SIGNAL   ✓H  : SGTTY   ✓H  : OPTION  ✓H
A: CTYPE  ✓H   : BDOS      S  : CBIOS     S  : LOADR     0  : S       0
A: OVHDLR   0  : W         0  : AS68INIT  ✓  : UFORMCPM 68K : CLIB    ✓
A: FLOATDEM C  : AS68SYMBOL DAT : TIME     S  : AS68    ✓68K : LO68  ✓68K
A: STAT     68K : PIP      68K : SD       68K : EDITPROK✓68K : EDITPROK S
A: CPM      SYS : CLINK    BAK : CLINKE  ✓SUB : CLINKF  ✓SUB
A>
```

Bild 1 zeigt das Inhaltsverzeichnis der so erstellten Diskette. Achtung, das Programm SD.68K ist bei der Original-CP/M68K-Diskette nicht vorhanden und muß auch nicht da sein.

Bei den gelieferten Disketten haben die meisten Dateien die Endung „REL". Mit Hilfe des Programms RELOC kann man sie in „68K"-Dateien umwandeln. Um z.B. das Programm C168.REL in C168.68K umzuwandeln, tippen Sie RELOC C 168.REL C168.68K ein. Wenn das Pro-

gramm RELOC selbst noch mit „REL“ endet, können Sie das Programm sich selbst umwandeln lassen, indem Sie den Befehl RELOC.REL RELOC.REL RELOC.68K eingeben. Dabei ist das erste Wort der Befehl, und dadurch, daß man REL an den Befehl anhängt, lädt CP/M68K das Programm auch in dieser Form.

Man könnte alle Programme auch mit REL belassen, jedoch verkürzt sich ein Programm ca. auf die Hälfte des ursprünglichen Umfangs, wenn es umgewandelt wird. Alle „REL“-Programme enthalten nämlich zusätzliche Informationen, die es gestatten, das Programm beliebig zu verschieben. Die „68K“-Programme sind dagegen absolut.

Nun heran ans Programmieren. Zunächst müssen noch ein paar Vorbereitungen getroffen werden. Dazu geben Sie einmal den Befehl „DIR *.SUB“ auf der neu erstellten Diskette ein. Die SUB-Dateien sind sogenannte Kommando-Dateien. Darin stehen Befehle, die nach Aufruf einfach vom CP/M68K ausgeführt werden.

Bild 2 zeigt die beiden Dateien. C.SUB

```
A>
A>
A>
A>type c.sub
a:cp68 -i  a: $1.c $1.i
a:c068 $1.i $1.1 $1.2 $1.3 -f
era  $1.i
a:cl68 $1.1 $1.2 $1.s
era  $1.1
era $1.2
a:as68 -l -u -s a: $1.s


A>type clink.sub
a:lo68 -r -u+nofloat -o $1.68k a:s.o $1.o $2.o $3.o $4.o $5.o $6.o $7.o $8.o $9.
o a:clib


A>
A>
A>
A>
A>
```

und CLINK.SUB und Bild 3 die Dateien CE.SUB, CLINKE.SUB und CLINKF.SUB.

```
A>type ce.sub
a:cp68 -i a: $1.c $1.i
a:c068 $1.i $1.1 $1.2 $1.3 -e
era  $1.i
a:cl68 $1.1 $1.2 $1.s
era  $1.1
era  $1.2
a:as68 -l -u -s a: $1.s
era  $1.s


A>type clinke.sub
a:lo68 -r -o $1.68k a:s.o $1.o $2.o $3.o $4.o $5.o $6.o $7.o $8.o $9.o a:clib a:
libe.a

A>type clinkf.sub
a:lo68 -r -o $1.68k a:s.o $1.o $2.o $3.o $4.o $5.o $6.o $7.o $8.o $9.o a:clib a:
libf.a

A>
A>
A>
A>
A>
```

Auf der CP/M68K-Diskette sehen die Dateien ähnlich aus (mit TYPE name. name ansehen), jedoch steht anstatt A: der Wert 0: Der Unterschied ist, daß bei A: die Befehle und Dateien immer auf Laufwerk A gesucht werden, bei 0: auf dem aktuellen Laufwerk. Wenn Sie den C-Compiler und alle Dateien auf A haben, aber das Anwenderprogramm auf B, so sollen Sie die abgedruckte Version verwenden. Geben Sie dazu die neuen Dateien mit dem Editor ein. Achtung, wenn Sie die alten Dateien einfach verbessern wollen, müssen Sie das Programm ED.68K (ED.REL) verwenden, da CLINK, CLINKE und CLINKF Zeilen mit mehr als 80 Zeichen enthält.

Ansonsten können Sie das Programm EDITRDK benutzen. Dabei wird nämlich der im Grundprogramm eingebaute Bildschirmeditor aufgerufen. Der Aufruf lautet dann zum Beispiel: „EDITRDK C.SUB“.

Nun geht es daran, das erste Programm in C zu schreiben. Dazu rufen Sie den Editor mit „EDITRDK DEMO1.C“ auf. Bild 4

```
/* kleines C-Beispiel   */

#include <stdio.h>

main()
{
 int i;
 for (i=1;i(=36;i++)
 { printf("\001 E @schreite 10\n");
   printf("\001 E @drehe 10\n");
 }
printf("\001 E @setflip 0 0\n\001 E @newpage 0 0\n\001 E @grapoff\n");
}

Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

zeigt ein Beispiel. Tippen Sie das Beispiel sorgfältig ab, und achten Sie darauf, daß keine Strichpunkte usw. fehlen.

Das Zeichen „\“ entsteht, wenn man SHIFT und „Ö“ drückt, und die geschweiften Klammern sind das kleine „ä“ und „ü“, wenn Sie eine deutsche Tastatur besitzen.

Den Editor verläßt man wie gewohnt mit CTRL-K und X.

Nun muß das C-Programm übersetzt werden. Dazu tippt man einfach „C name“ ein. Als „name“ gibt man den Namen der Datei an, also z.B. „DEMO1“, da die Datei „DEMO1“ heißt. Nach einer Weile erscheinen die Befehle aus der SUB-Datei automatisch auf dem Bildschirm und werden nacheinander ausgeführt. Wenn der Cursor wieder blinkt, können Sie den nächsten Befehl eingeben. Jetzt muß das übersetzte Programm noch mit der C-Bibliothek zusammengebunden werden. Dazu gibt man den Befehl „CLINK DEMO1“. Wenn der Cursor erneut erscheint, kann man einfach DEMO1 eintippen und Bild 5 muß nun insgesamt auf dem Bildschirm erscheinen. Ein Kreis wurde gezeichnet. Wenn eine Fehlermeldung auf dem Bildschirm erscheinen sollte, müssen Sie erneut das Programm aufrufen und nochmals genau vergleichen.

```
B>a:c demo1

B>A:CP68 -I A: DEMO1.C DEMO1.I

B>A:C068 DEMO1.I DEMO1.1 DEMO1.2 DEMO1.3 -f

B>ERA DEMO1.I

B>A:C168 DEMO1.1 DEMO1.2 DEMO1.S

B>ERA DEMO1.1

B>ERA DEMO1.2

B>A:AS68 -L -U -S A: DEMO1.S

B>a:clink demo1

B>A:LO68 -R -U+NOFLOAT -O DEMO1.68K A:S.O DEMO1.O .O .O .O .O .O .O .O A:CLIB

B>demo1

B>
```

Der C-Compiler hat auch die Möglichkeit mit Gleitkommazahlen zu arbeiten. Dabei gibt es zwei verschiedene Formate. Das Motorola-Format, das besonders schnell ist und das IEEE-Format, das international genormt ist. Die Formate beschreiben, wie die Gleitkommazahlen im Rechner dargestellt werden. Der C-Compiler kann mit beiden Formaten arbeiten, man muß ihm aber angeben mit welchem. Wenn man „C.SUB“ zur Übersetzung verwendet, wird das Motorola-Format verwendet und bei „CE.SUB“ das IEEE-Format.

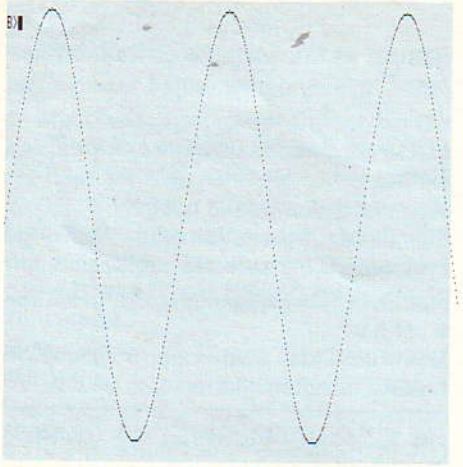
Beim Linken muß man dann „CLINKF.-SUB“ für das Motorola-Format angeben und „CLINKE“ für das IEEE-Format.

```
/* Gleitkommaarithmetik mit C */

#include <stdio.h>

EXTERN FLOAT sin();

main()
{
 FLOAT winkel;
 int x,y;
 printf("\001E @clrscreen\n\001E @setflip 0 0\n\001E @newpage 0 0\n");
 printf("\001E @curoff\n");
 for (winkel=0;x=0; x(=511; x++,winkel=winkel+3.1415/100)
 {
  y = (sin(winkel)*120+128);
  printf("\001E @moveto 0 %d %d\n\001E @drawto 0 %d %d\n",x,y,x,y);
 }
 printf("\001E @curon\n");
}

Textstart=009000 Fenster=009000 Tor=009002      amer CTRL-J=Hilfe
```

Bild 6 zeigt ein weiteres C-Programm, diesmal zum Test der Gleitkommaarithmetik. In Bild 7 ist der Übersetzungsvor-

```
B>a:c demo2

B>A:CP68 -I A: DEMO2.C DEMO2.I

B>A:C068 DEMO2.I DEMO2.1 DEMO2.2 DEMO2.3 -F

B>ERA DEMO2.I

B>A:C168 DEMO2.1 DEMO2.2 DEMO2.S

B>ERA DEMO2.1

B>ERA DEMO2.2

B>A:AS68 -L -U -S A: DEMO2.S

B>a:clinkf demo2

B>A:LO68 -R -O DEMO2.68K A:S.O DEMO2.O .O .O .O .O .O .O .O A:CLIB A:LIBF.A

B>demo2
```

gang gezeigt und Bild 8 zeigt das Ergebnis auf dem Bildschirm.



In allen unseren C-Programmen hatten wir Unterprogramme aus dem Grundprogramm aufgerufen. Der Aufruf geschah dabei über einen PRINTF-Befehl und den im Grundprogramm eingebauten Mechanismus (siehe Beschreibung Grundprogramm 4.3 und PASCAL-Beispiele).
Der Aufruf ist natürlich nicht sehr effizient und daher langsam, denn das Grundprogramm muß den Befehl ja jedesmal neu entschlüsseln.

Besser ist es, Unterprogramme in Assembler zu verwenden und diese direkt aufzurufen.

```
* Assemblerprogramm zum Aufruf von Grundprog.
* Funktionen von C aus.
 .globl +init,+moveto,+drawto
+init:
  movem.l d3-d7/a3-a6,-(a7)
  move #23,d0    * Spezial Bios Einsprung
  trap #3        * Holt Grundprog. Start
  move.l a4,merker
  movem.l (a7)+,d3-d7/a3-a6
  rts

+moveto:
  link a6,#0
  movem.l d3-d7/a3-a6,-(a7)
  move.w 8(a6),d1
  move.w 10(a6),d2
  move #8,d7
  movea.l merker,a0
  jsr $420(a0)
  movem.l (a7)+,d3-d7/a3-a6
  unlk a6
  rts
```

Textstart=009000 Fenster=009000 Tor=009100 amer CTRL-J=Hilfe

```
+drawto:
  link a6,#0
  movem.l d3-d7/a3-a6,-(a7)
  move.w 8(a6),d1
  move.w 10(a6),d2
  move #9,d7
  movea.l merker,a0
  jsr $420(a0)
  movem.l (a7)+,d3-d7/a3-a6
  unlk a6
  rts


fehler: rts

merker: dc.l fehler
 end
```

Textstart=009000 Fenster=0091EC Tor=0091EC amer CTRL-J=Hilfe

Die Bilder 9 und 10 zeigen dazu das Assemblerprogramm. Tippen Sie das Programm unter dem Namen GRUPROG.S ein und übersetzen es mit "A: AS68 –S A: GRUPROG.S". Der Aufruf der Unterprogramme soll von C aus geschehen.

```
/* Direkter Einbau von Grundprog. Routinen */

#include <stdio.h>

EXTERN init();
EXTERN moveto();
EXTERN drawto();

main()
{
 int x,y;
 init();
 printf("\001E @setflip 0 0\n\001 E @newpage 0 0\n");
 for (x=0;x<=500;x=x+20) {
  moveto(x,0); drawto(x,250);
 }
 for (y=0;y<=250;y=y+10) {
  moveto(0,y); drawto(500,y);
 }
}
```

Textstart=009000 Fenster=009000 Tor=009000 einf amer CTRL-J=Hilfe

Dazu zeigt Bild 11 das C-Programm. Durch die EXTERN-Befehle werden die im Assemblerteil vorhandenen Unterprogramme definiert.

```
B)a:as68 -s a: gruprog.s

B)a:c demo3

B)A:CP68 -I A: DEMO3.C DEMO3.I

B)A:C068 DEMO3.I DEMO3.1 DEMO3.2 DEMO3.3 -F

B)ERA DEMO3.I

B)A:C168 DEMO3.1 DEMO3.2 DEMO3.S

B)ERA DEMO3.1

B)ERA DEMO3.2

B)A:AS68 -L -U -S A: DEMO3.S

B)a:clink demo3 gruprog

B)A:LO68 -R -U+NOFLOAT -O DEMO3.68K A:S.O DEMO3.O GRUPROG.O .O .O .O .O .O .O .O
 A:CLIB

B)
```

Bild 12 zeigt den kompletten Übersetzungsvorgang. Das C-Programm wurde unter dem Namen „DEMO3.C" abgelegt.

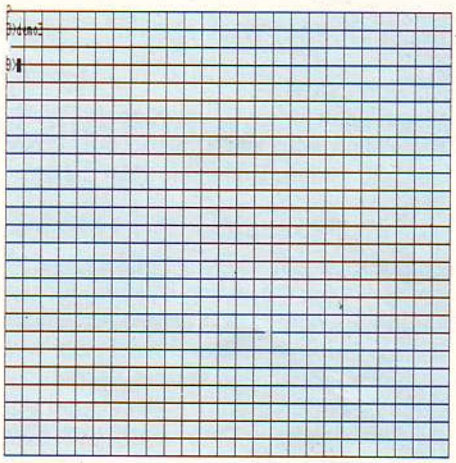


Bild 13 zeigt das Ergebnis nach einem Programmstart.

(Fortsetzung folgt).

Rolf-Dieter Klein

Produkte

TEXTEY, DATEY und DOSEY

68008-Programme von Horst Günter Eysel

Seit geraumer Zeit werden von Herrn Eysel (Piepmäkerstraße 4, 3380 Goslar) die Programme TEXTEY, DATEY und DOSEY angeboten.

Textey ist eine kleine Textverarbeitung, die auf dem Texteditor des 68008-Grundprogrammes basiert. Datey, eine kleine Dateiverwaltung, ermöglicht den Aufbau von Dateien mit beliebig langen Datensätzen, die leicht manipuliert werden können.

Dosey stellt ein Mini-DOS dar, unter dem Daten und Programme auf Disketten gespeichert und gelesen werden können.

Diese Programme sind direkt bei H. Eysel erhältlich.

Aus PRISMA 17/85 (von G. Mersson, Dortmund)

„Ich bin etwas beunruhigt — ein Junge aus meiner Computer-Klasse hat mich heute nach meiner Kontonummer gefragt . . .!"

FLOH MARKT

Verkaufe:
SBC2 voll aufgebaut mit Monitorprogramm für DM 100,—.
U. Wiesel,
Memmelsdorfer Straße 117,
8600 Bamberg

Betreff: Austausch wegen NDR-Klein-Computer. Bitte melden bei
Jochen Tofahrn,
Pappenheimer Straße 72,
8500 Nürnberg 60

IN & OUT

Leser fragen – Fachleute antworten

Stellen Sie auch Ihre Fragen an „loop"

Sehr geehrte *LOOP*-Redaktion,

zunächst einmal möchte ich Ihnen danken, daß der NDR-Klein-Computer von einer eigenen Zeitschrift begleitet wird.

Die angesprochenen Themen stellen meiner Meinung nach einen großen Anwenderkreis zufrieden. Vielleicht ist es möglich, evtl. noch mehr kleine Anregungen zum Programmieren in den zur Verfügung stehenden Sprachen zu geben.

Hier in Berlin haben sich einige NDR-Computer-Bastler zusammengefunden. Da es sich ausnahmslos alles um „blutige Laien" handelt, stellen sich von Zeit zu Zeit einige Fragen ein, die uns leider bis zum jetzigen Zeitpunkt niemand beantworten konnte. Wir hoffen, daß uns vielleicht die Redaktion *LOOP* weiterhelfen kann und würden uns über eine Antwort sehr freuen.

1. Kann auf der CPU68K-Karte auch ein 12 MHz Quarz ohne Probleme verwendet werden?

2. Wie sieht die Jumperstellung bei Verwendung eines 12 MHz Quarzes zusammen mit der Baugruppe FL02 aus? (Waitzyklen).

3. Warum ist ein korrektes Listing per Drucker beim BASIC nicht möglich? (Ausdruck sonst einwandfrei. – Siehe beiliegendes kurzes BASIC-Listing).

Für eine Beantwortung der Fragen wären wir Ihnen sehr dankbar. Wir hoffen, daß die *LOOP* auch weiterhin ihre Verbreitung findet.

Mit freundlichem Gruß

Werner Wolff
Weinmeisterhornweg 132,
1000 Berlin 20

Antwort LOOP:

1. Wie Radio Eriwan: Prinzipiell ja. Allerdings müßen Sie eine CPU L12 (12 MHz Version) einsetzen. Wir würden ihnen aber, falls Sie Wert auf höhere Geschwindigkeit legen, die neue Baugruppe CPU 68000 empfehlen, die mit der 16 bit CPU 68000 und 12 MHz läuft. Sie wird ab Herbst lieferbar sein.

2. 3 – 4 Waitzyklen, ausprobieren.

3. Listing ist schon möglich. Der von Ihnen geschilderte Fehler deutet auf ein Hardware-Problem hin, z.B. Kurzschluß einer Datenleitung.

4. Besuchen Sie bitte unseren Händler in Berlin, die Firma Jörg Korb, Elektronik Systeme, Budapester Straße 39A/I, 1000 Berlin 30. Herr Korb kennt den NDR-Computer bis zum letzten IC und kann Ihnen sicher immer und gern Auskunft geben. Er vermittelt auch gerne Kontakte zu erfahrenen Bastlern für Sie!

Sehr geehrte Damen und Herren,

herzlichen Glückwunsch zu *LOOP*. Die Zeitung wird immer besser. Sie könnte sich zur wertvollsten Informationsquelle für den NDR-Computer entwickeln. Man kann nur hoffen, daß die Seitenzahl steigt (16 sind doch etwas dürftig) und dabei der Informationsinhalt gleich gut bleibt.

Mein Vorschlag dazu wäre, die Vorstellung neuer Produkte für den NDR-Computer auszudehnen.

Ehrlich gesagt, mit dem was in *LOOP 3* unter „Was gibts Neues" stand, war nicht viel anzufangen. Das waren doch nur allgemein gehaltene Stichworte ohne großen Informationswert.

Bitte beschreiben Sie diese Dinge noch einmal etwas genauer, sagen Sie bitte mehr über die Eigenschaften, Fähigkeiten und Vor-/Nachteile, z.B. von Jogidos oder Eproms.

Vielen Dank für Ihre Mühe mit den vielen Fragen.

Michael Tremel
Reuth 4A, 3641 Weißenbrunn

Antwort LOOP:

Vielen Dank für die Glückwünsche. Wahrscheinlich wird der Umfang von *LOOP* bald 20 Seiten betragen. Nun zu Ihren Fragen:

Was kann Jogidos?

Jogidos ist ein kleines Betriebssystem (DOS = Disk Operating System) für den 68008. Es richtet sich an alle Anwender, die mit Floppy-Laufwerken arbeiten wollen, sich jedoch das teurere Betriebssystem CP/M68K (noch) nicht zulegen wollen. Jogidos kann Daten und Texte (Programme) auf Diskette speichern und von Diskette lesen.

Kann es formatieren?

Ja, verschiedene Formate. Im Normalfall braucht man also das EUFORM-Eprom nicht.

Führt es ein Directory?

Ja, Einträge kann man löschen, Zugriff erfolgt über Dateinamen.

Benutzt es Routinen des Grundprogrammes?

Ja.

EUFORM: Braucht Jogidos Euform?

Nein.

Wozu ist Euform dann nötig?

Für nicht Jogidos-Benutzer. EUFORM kennt mehr Formate als Jogidos.

Hat EUFORM Fähigkeiten, die CP/M68K nicht hat?

Nein; es findet sich sogar auf den CP/M68K-Disketten wieder.

Hat EUFORM Fähigkeiten, die CP/M68K nicht hat?

Nein; es findet sich sogar auf den CP/M68K-Disketten wieder.

EFLOMON: Ist es Ergänzung oder Bestandteil des CP/M80?

Nein. EFLOMON muß vorhanden sein, um das (Disketten-)betriebssystem CP/M80 a) zu laden, und b) diesem die Grundroutinen zur Terminal-Bedienung und zur Diskettensteuerung zur Verfügung zu stellen.

Gibt es etwas Entsprechendes für den 68008-Ausbau?

Ja, das Grundprogramm V 4.3.

Oder eine erweiterte Grundprogrammversion mit Floppy-Routinen?

Version 4.3 enthält alle Floppy-Routinen. Ältere Versionen werden, wie immer, für DM 10,– pro Eprom (also DM 40,–) umgetauscht. Voraussetzung: Die Eproms müssen bei uns gekauft sein und unsere Aufkleber tragen.

CPU 68000 – die 16 bit CPU – was kann sie?

Siehe Artikel: Die CPU 68000 kommt.

Bugs im BASIC

Natürlich bleiben auch unsere Programme nicht von Bugs verschont und im BASIC für den Z80 haben sich ein paar Bugs eingeschlichen.

1. Die RND-Funktion. Manchmal tritt ein OV-Fehler auf, wenn man die RND-Funktion verwendet. Dies kommt daher, daß nach dem Einschalten im Speicher ein zufälliger Wert steht und dieser Wert von der RND-Funktion verwendet wird. Abhilfe schafft folgende Sequenz:

10 A = RND (– 1)

Damit wird der Zufallszahlengenerator mit einem festen Wert initialisiert. Alle nachfolgenden Aufrufe von RND(1) dürfen nun keinen Fehler mehr bringen. Übrigens, wer es noch nicht weiß, RND(0) liefert immer die letzte Zufallszahl, die mit RND(1) erzeugt wurde.

Ab Version 1.5 wird der Zufallszahlengenerator standardmäßig initialisiert.

2. LPRINT TAB (oder LPRINT SPC) führen zu einem Syntaxfehler. Abhilfe schafft folgende Sequenz:

LPRINT ""; TAB (. . . oder
LPRINT ""; SBC (. . .

auch dieser Fehler wird beseitigt.

3. Manchmal springt der Cursor bei der Eingabe von Programmzeilen einfach in die nächste. Man kann dann aber getrost weiter eingeben. Ursache dafür ist ein fehlerhafter Zeichenzähler, der immer nur aufwärts zählt, aber nicht abwärts, wenn man z.B. die BS- oder DEL-Taste verwendet.

4. Eigentlich kein Fehler, aber immer wieder häufig falsch gemacht wird folgendes:

Es tritt der Fehler OM auf (out of memory) oder Speicherüberlauf. Wenn man das BASIC neu startet, werden nämlich nur 50 Bytes für Strings (Zeichenketten) reserviert. Wenn man dann häufig mit Strings (A$ . . .) arbeitet, läuft der Speicher über und oft hängt sich alles auf.

Abhilfe schafft der Befehl CLEAR. Man kann mit diesem Befehl den Stringspeicher reservieren, dazu schreibt man einfach die Anzahl der benötigten Bytes dahinter: Beispiel: CLEAR 1000, reserviert 1000 Bytes für Zeichenketten.

Übrigens, sollte sich das BASIC einmal aufhängen, vielleicht wegen des OM-Fehlers oder weil man Unterprogramme in Maschinensprache aufruft, so gibt man nach dem RESET einfach ein großes „W" ein (SHIFT W), anstelle eines „C". Dann wird ein Warmstart durchgeführt und das alte BASIC-Programm ist noch vorhanden, falls es nicht durch das Programm selbst zerstört wurde. Dabei kann es ggf. wegen eines weiteren Fehlers zu einem OM-Fehler kommen, der allerdings nach auftreten sofort wirkungslos wird.

Neues bei der Version 1.5: Neben der Beseitigung der oben genannten Fehler gibt es zwei Neuerungen. Zunächst einmal kann nun die Schriftgröße in zwei Werten verändert werden. Dazu gibt man den Befehl OUT 115,n – n bestimmt die Größe und errechnet sich wie folgt: n = groessex* 16 + groessey.

Dabei kann groessex die Werte 1 und 2 annehmen und groessey ebenfalls. Es gibt also vier verschiedene Kombinationen. Bei allen vier Kombinationen wird das Bildschirmscrollen usw. richtig ausgeführt. Insbesondere eignet sich die Version OUT 115,16*2+2 für Unterrichtszwecke, da dann die Schrift doppelt hoch und doppelt breit ausgegeben wird.

Für Fernseher ist die Kombination OUT 112,16*2+1 günstiger, bei der doppelt breite Schrift verwendet wird.

In der Version 1.5 wurde auch ein neuer Befehl eingebaut: GET$.

Damit ist es möglich, einzelne Zeichen von der Tastatur zu lesen und auch abzufragen, ob eine Taste gedrückt wurde. Dies war vorher auch mit Hilfe von Maschinenprogrammen nicht möglich.

GET$ wird wie eine Stringvariable verwendet. Sie liefert den Wert "", wenn keine Taste gedrückt wurde, sonst wird der Wert geliefert.

Beispiel: 10 PRINT GET$; 20 GOTO 10

Das Programm läßt sich natürlich nach wie vor durch ESC abbrechen.

Beispiel:

```
10 A$ = GET$
20 IF A$ = ""THEN 10
30 PRINT ".", A$; ".";
40 GOTO 10
```

Wenn man nun eine Taste drückt, wird das Zeichen zwischen "." ausgegeben. GET$ liest auch Steuerzeichen wie CR ein. Beispiel:

```
10 B$ = ""
20 A$ = GET$
30 IF A$ = CHR$(13) THEN 100
40 B$ = B$+A$
50 GOTO 20
100 PRINT B$
```

Man kann eine Textzeile eingeben, die allerdings erst nach Eingabe von CR auf dem Bildschirm ausgedruckt wird.

Ein größeres Anwendungsbeispiel zeigt das Listing. Hier wird ein kleines Spiel dargestellt. Eine Linie wandert über den Bildschirm, wenn man die Leertaste drückt, so wird ein „Schuß" abgefeuert.

In der nächsten *LOOP* wird gezeigt, wie man unter BASIC Daten auf Kassette speichern und von Kassette laden kann.

Die neue Version 1.5 ist im Handel gegen den Umsatz von 10,– DM pro EPROM erhältlich.

Rolf-Dieter Klein

```
10 REM Erst ab Version 1.5 lauffaehig
20 CLRS                          : REM Bildschirm loeschen
30 S=0                           : REM Merker fuer Schuss
40 POKE HEX("87C5"),0            : REM Flip ausschalten
50 PAGE 0,0                      : REM Bildseite 0,0
60 FOR X=0 TO 511                : REM alle x-Koordinaten
70 MOVETO X,0:DRAWTO X,10        : REM Linie zeichnen
80 IF GET$=" " THEN S=1:HX=X:HY=11 : REM schuss starten
90 GOSUB 2000                    : REM Schuss ausfuehren
100 OUT HEX("70"),1              : REM Loeschstift
110 MOVETO X,0: DRAWTO X,10:     REM Linie loeschen
120 OUT HEX("70"),0              : REM Schreibstift
130 NEXT X
140 END
2000 IF S=0 THEN RETURN          : REM nur wenn Schuss an
2010 OUT HEX("70"),1             : REM Loeschstift
2020 MOVETO HX,HY:DRAWTO HX,HY+10 : REM Linie
2030 OUT HEX("70"),0             : REM wieder ausschalten
2040 HY=HY+5                     : REM neue Koordinate fuer Schuss
2050 MOVETO HX,HY:DRAWTO HX,HY+10
2060 RETURN
```


IMPRESSUM

LOOP Zeitung für Computerbauer

Herausgeber: Gerd Graf
Redaktion: Rolf-Dieter Klein, Gerd Graf, Rüdiger Bäcker
Druck und Gestaltung: Rieder, Kempten
Herstellung und Anzeigenverwaltung: GES GmbH
Magnusstr. 13 8960 Kempten
Anzeigenpreisliste 1/84

Theorie und Praxis rund um den NDR-Computer

Mikroelektronik
Einführung

NEU

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.

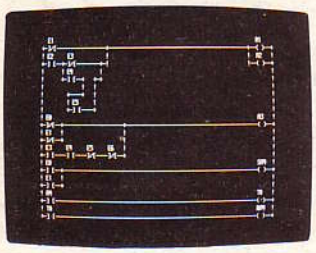
Nach diesem Kurs sind Sie in der Lage, eigene Programme zu schreiben und die Arbeitsweise des Z 80 zu verstehen.

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

NEU



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.

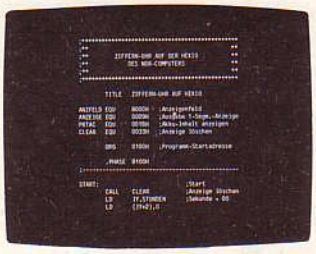
Sie lernen spielend leicht, Relais- und Schützensteuerungen in SPS-Programme umzusetzen.

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

NEU



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