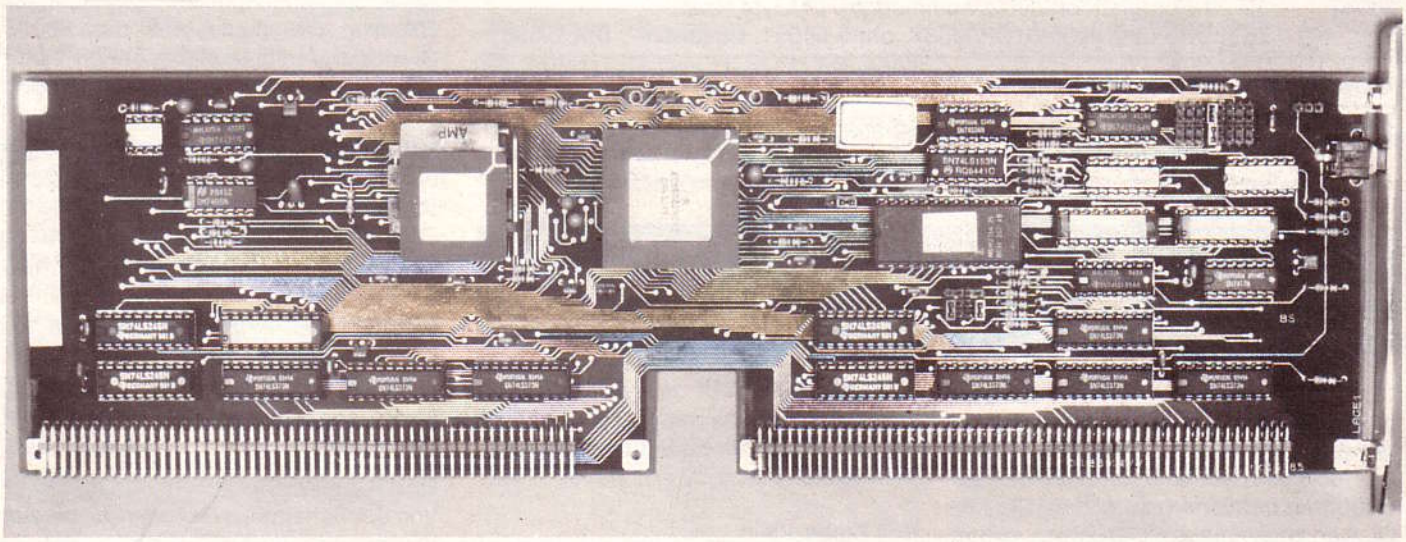
LOOP

10

2. JAHRGANG

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

Uwe Koch
Frankenstraße 25
5880 LÜDENSCHEID
Tel. (0 23 51) 2 61 92



Die CPU 68020 ist lieferbar!

von Axel Granel

Nur komplett aufgebaut als Fertiggerät lieferbar ist nun die erste 32-Bit Prozessorbaugruppe für den NDR-Computer!

Der 68020 ist der zur Zeit interessanteste 32-Bit-Mikroprozessor auf dem Markt, denn durch seine Aufwärtskompatibilität zum 68000 laufen die meisten, für den 68000 geschriebenen Programme auch auf dem 68020. Durch den internen und externen 32-Bit-Datenbus, neue leistungsstarke Befehle und einen kleinen superschnellen internen Speicher (Cache) ist er aber wesentlich schneller wie der 68000.

Daß auch eine 32-Bit CPU im NDR-Computer eingesetzt werden kann, ist seinem modularen Konzept zu verdanken: Die 32 Bit Datenleitungen werden einfach aufgeteilt in 4 Teile mit jeweils 8 Bit. Alle bisher für den NDR-Computer erhältlichen Speicher- und Peripheriekarten benutzen 8 Datenleitungen. Somit sind diese weiterhin verwendbar!

Für eine Speichererweiterung müssen allerdings immer vier neue Speicherkarten auf einmal eingesetzt werden, damit die CPU auch wirklich viermal 8 Datenbits bearbeiten kann.

Peripheriekarten brauchen, wie bisher, nur einmal vorhanden zu sein – die CPU kann hier auch 8 Bit Daten bearbeiten.

Man benötigt zu einem lauffähigen minimalen System zwei Busbaugruppen, die nebeneinander angeordnet werden. Empfehlenswert sind hier ein BUS3 und ein BUS4. Beide müssen in der Mitte doppelte Busleisten (wie für den 68000) haben, wobei die Leitungen D0 – D7 sowie IORQ, MREQ, RD und WR getrennt werden müssen. Ferner braucht man 4 ROA64 mit dem neuen Grundprogramm (in 8 EPROMs) und 4 RAM8. Bankboot-Baugruppen werden nicht benötigt, da die komplette BANKBOOT-Logik incl. BOOT-EPROM auf der CPU-Karte integriert ist. Nur noch eine KEY und eine GDP64 sind nötig, um mit dem Grundprogramm wie beim 68008 in Assembler programmieren zu können, allerdings jetzt auch mit den neuen 68020-Befehlen und auch den 68881-Befehlen, wenn dieser Chip eingesetzt ist.

Ein Sockel für den 68881 ist vorgesehen, er braucht also nur noch eingesteckt zu werden, damit man die neuen Befehle nutzen kann. Da der 68881 ein Mathematik-Coprozessor ist, und der 68020 so mit dem 68881 zusammenarbeitet, als wären sie ein (!) Chip, kann man die neuen Möglichkeiten wie ganz normale Assemblerbefehle nutzen. Es stehen dann neue Befehle zur Verfügung, um Integerzahlen und Floating-Point-Zahlen, z.B. 7.6 oder 1.2 E002 mit bis zu 80 Bit Genauigkeit zu bearbeiten, z.B. ADD, SUB, MUL, DIV, SIN, CON, TAN, LOG, Wurzel und vieles mehr. Diese Berechnungen werden mit sehr hoher Genauigkeit extrem schnell berechnet. Pro Gleitkomma-Operation werden – bei 80 bit Genauigkeit – 4 – 20 us benötigt.

In der Zeitschrift mc (ab Ausgabe 7/86) erscheint eine Artikelserie, die sich mit dem Aufbau und der Funktion der Baugruppe CPU 68020 und anschließend mit dem 68881 ausführlich beschäftigt.

Preise und Bestellnummern:

Bestell-Nr.	Bezeichnung	Preis
10176	CPU-Baugruppe mit der CPU XC-68020-12, 12 MHz, Boot-Eprom, komplett, geprüft, nur als Fertiggerät lieferbar (Multilayer). Sockel für Co-Prozessor 68881, Lieferung *ohne* Co-Prozessor	DM 1.698,–
10177	Handbuch der Baugruppe	DM 10,–
10772	Co-Prozessor FPU XC-68881-12, Sonderpreis, gilt nur in Verbindung mit 10176	DM 698,–
10564	Grundprogramm für 68020, aufwärtskompatibel zum 68008/68000-Grundprogramm, jedoch mit Befehlen für den Co-Prozessor und 68020, 8 Eproms, benötigt 4 ROA64	DM 185,–
---	Upgrade, falls schon 68008 oder 68000-Grundprogramm bei uns gekauft wurde	DM 80,–
Paketangebote:		
10047	32-Bit-Komplettcomputer, komplett mit 1 MByte RAM, zwei Disk-Laufwerken, CP/M68K, ohne 68851, **Bausätze**	DM 6.598,–
10048	wie 10047, jedoch Fertiggeräte	DM 7.498,–
10049	32-Bit Komplettsystem, funktionsbereit im Gehäuse, wie 10047, getestet	DM 7.998,–

In eigener Sache

Noch nie hat eine *LOOP* so viel Staub aufgewirbelt wie die letzte Doppelnummer! Wir wollen nochmals klarstellen: Es lag uns mit dieser Nummer fern, uns an Abo-Gebühren bereichern zu wollen! Die Kritik war zusammengefaßt: Wenn schon Doppelnummer, dann gefälligst auch doppelt so viele Seiten und einen Inhalt, der allen was bringt!

Ein weiterer Kritikpunkt: Bitte keine leeren Versprechungen, wie Gomoku-Listing oder Modem-Programm für den 68000!

Wir lösen mit dieser *LOOP* unsere leeren Versprechungen ein und bitten um Entschuldigung. Diese *LOOP* enthält das gesamte Gomoku-Listing. Alle Kunden, die ein Listing bereits bei uns gekauft haben, erhalten den Kaufpreis in voller Höhe zurückerstattet. Bitte senden Sie uns dazu einfach eine Kopie der Rechnung. Den Artikel: „Gomoku – endlich das Listing" finden Sie auf Seite 10, den Artikel: „Modem-Programm für den 68000" auf Seite 20.

Ein weiterer Kritikpunkt war: Doch bitte für *jeden* etwas! Wir haben dies auch erfaßt und haben schon diese *LOOP* mit klaren Unterrubriken ausgeführt. Beachten Sie „Für jeden etwas" auf Seite 2.

Dies funktioniert natürlich nur, falls Sie uns weiter so gut helfen und uns Ihre Artikel zusenden. Glauben Sie bitte nie, daß Ihre Arbeit oder Ihr, wenn auch noch so kleines Programm niemand interessieren könnte! Schreiben Sie uns auch, falls mal etwas nicht geklappt hat und wie Sie dem Problem auf die Spur gekommen sind!

Um unsere Absicht der Nichtbereicherung zu dokumentieren, hat diese *LOOP* einen Umfang von 32 Seiten: ohne Doppelnummer.

Rolf Dieter Klein
Gerd Graf

Für jeden etwas . . .

Das neue Erscheinungbild der *LOOP!*

von Gerd Graf

„. . . in der letzten *LOOP* war aber viel zu viel über C und Modula . . .", „. . . schreibt doch mehr für die Einsteiger . . .", „. . . warum steht in der *LOOP* nichts über den mc-CP/M-Computer . . .", „. . . viel zu viel über Basic, ich möchte etwas über Pascal lesen . . ." – so waren viele Meinungen, die während unserer letzten Messen in Hannover und Dortmund an mich herangetragen wurden.

Nun, ab dieser *LOOP* haben wir ein neues und klares Einteilungsschema, das hier nun vorgestellt werden soll.

Die Einteilung der neuen LOOP:

Titelgeschichte
Ein jeweils aktuelles Thema aus allen Bereichen.

In eigener Sache
Rolf-Dieter Klein und Gerd Graf geben Ihre nicht immer ungeteilte Meinung zum Besten.

Jetzt lieferbar
Neue Produkte zum NDR-Computer, mc-CP/M-Computer und Zubehör – ein bißchen Werbung und viel Produktinformation.

Für Einsteiger - Z80
Ihre Hardware: Einsteigerpaket mit Hexio oder Z80-Grundprogramm. Für Einsteiger und leicht Fortgeschrittene.

Z80-Vollausbau bis ZEAT
Für die etwas weiter Fortgeschrittenen auf dem Weg zum CP/M2.2.

Z80 – CP/M2.2
Für alle, die CP/M2.2 besitzen – Hinweise, Programme und Tips.

PASCAL und BASIC
Programme in PASCAL für das 68000 oder Turbo-Pascal, für das kleine BASIC des Z80 und für HEBAS.

Für 68000-Einsteiger
Für alle, die mindestens das 68008-Grundprogramm lauffähig haben. Natürlich auch für das Grundprogramm im 68000 oder im 68020.

68000-DOS ohne CP/M68K
Für die Benutzer des 68008 bis 68020 mit Diskettenlaufwerken und einem kleinen DOS (Disk Operating System) wie Jogi oder Jados, jedoch (noch) nicht CP/M68K.

CP/M68K, "C" und MODULA
Artikel – aber nicht nur für die Profis!

Der mc-CP/M-Computer
Noch lange nicht tot – der mc-CP/M-Computer.

Tips + Tricks
Von der Schaltungsveränderung bis zum Programmtrick – für alle Konfigurationen.

Aus der Technik
Technische Hinweise, Verbesserungen und Revisionen.

Briefe
Na klar: Ihre Leserbriefe, eventuell gekürzt, aber nicht beschönigt!

Kontakte, Kleinanzeigen
Kontaktwünsche, auch Informationen über Clubs, sind **kostenlos!** Kleinanzeigen kosten DM 3,– pro Zeile, eine Zeile hat 35 Zeichen. Bitte gleich beim Zusenden bezahlen!

Anzeigen
Wir möchten die Anzahl der Anzeigen klein, die Qualität jedoch hoch halten!

Kurz und Aktuell
Meldungen, die uns in letzter Minute erreichen.

Wir hoffen und wünschen, unter jeder Rubrik in jeder *LOOP* etwas berichten zu können! Helfen Sie uns dabei: Durch Ihre Artikel – die können abgedruckt werden – und durch Ihre Weiterempfehlung, besonders an die Mitleser, die *LOOP* zu abonnieren! Jedes Abo hilft uns, die *LOOP* noch dicker, besser und schöner zu machen!

Eine kleine Verbesserung am Rande: Vor jeder Seitennummer steht jetzt die Ausgabe (10/2: LOOP Heft 10, Seite 2). Dies erleichtert das Suchen und das Aufstel-

len eines Gesamt-Inhaltsverzeichnisses, das wir zum Jahresende geplant haben.

PS: Die *LOOP* bleibt immer aktuell – alle „alten" *LOOPs* sind weiter erhältlich und werden bei Bedarf auch nachgedruckt!

Software-Partner gesucht!

von Gerd Graf

Computer leben von der Software – von Programmen, die *Sie* in der Schublade haben und *wir* dringend benötigen! Ob die Programme nun echt vorhanden sind oder „nur" im Kopf, ist erst einmal egal.

Wir suchen *Sie* – als unseren Software-Partner! Denn eines ist klar: Unsere Kunden sind unsere besten Kritiker, aber auch unsere besten Entwickler.

Nun: Wir suchen nicht *nur* den Software-Crack, der unter 32 bit die Tastatur nicht mehr berührt! Wir suchen auch *und gerade* den Software-Ersteller für das Einsteigerpaket oder das Z80- oder 680xx-Grundprogramm!

Für welche Bereiche suchen wir?

Zur Erinnerung kurz die Programmsegmente der Hardware, für die wir Programme suchen:

Z80-Ausbau:
Einsteigerpaket
Grundprogramm (Bildschirm, Tastatur)
ZEAT
CP/M2.2
CP/M2.2 mit Festplatte

680xx-Ausbau:
680xx-Grundprogramm
Floppy-Ausbau mit Disketten-Betriebssystem (JOGI- oder JADOS)
Floppy-Ausbau mit CP/M68K
Festplatte
ACRTC-System
68020-Profi-System

Für alle diese Segmente suchen wir Software! Software, die vielleicht bei Ihnen schon seit Jahren auf Disketten schlummern, oder Software, die erst erstellt werden muß!

Was haben Sie davon?

Wir wollen und können Ihnen keine goldenen Berge versprechen.

Was wir machen können, ist aufzuführen, was unsere *bisherigen* Software-Partner von uns erhalten haben:

Unterstützung
Sie werden in unsere „VIP-Info" eingebunden und erhalten ab sofort einmal monatlich die aktuellsten Informationen über Updates, neue Produkte, Änderungen und vieles mehr.

Rat
Bei der Planung von neuen Produkten – und bei der Überarbeitung alter Versionen – fließen Ihre Vorstellungen ein.

Leihgeräte
Für interessante Software-Entwicklungen stellen wir kostenlos Leihsysteme (z.B. 68020, ACRTC) zur Verfügung.

Günstiger Einkauf
Als unser Software-Partner kaufen Sie im Einzelfall günstiger. Besonders bei Sonderaktionen werden Sie bevorzugt benachrichtigt.

Materielle Zuwendung
Natürlich gibts auch Geld – entweder in Form von einmaligen Zahlungen oder Gutschriften oder in Form eines Lizenzvertrages, der Ihnen einen bestimmten Prozentsatz vom Verkaufspreis sichert.

Einladung zum „Software-Workshop"

Wir planen im Herbst an einem Wochenende, hier in Kempten, einen Software-Workshop abzuhalten. Neben der Arbeit kommt auch das Vergnügen nicht zu kurz. Mehr wird noch nicht verraten, nur: Als Software-Partner sind Sie unser Gast!

Wie wird man Software-Partner?

Ganz einfach: Wenden Sie sich an uns! Schicken Sie uns, was Sie haben: Ihre Programme, Ihre Ideen, Ihre Vorschläge! Sie hören dann sofort wieder von uns!

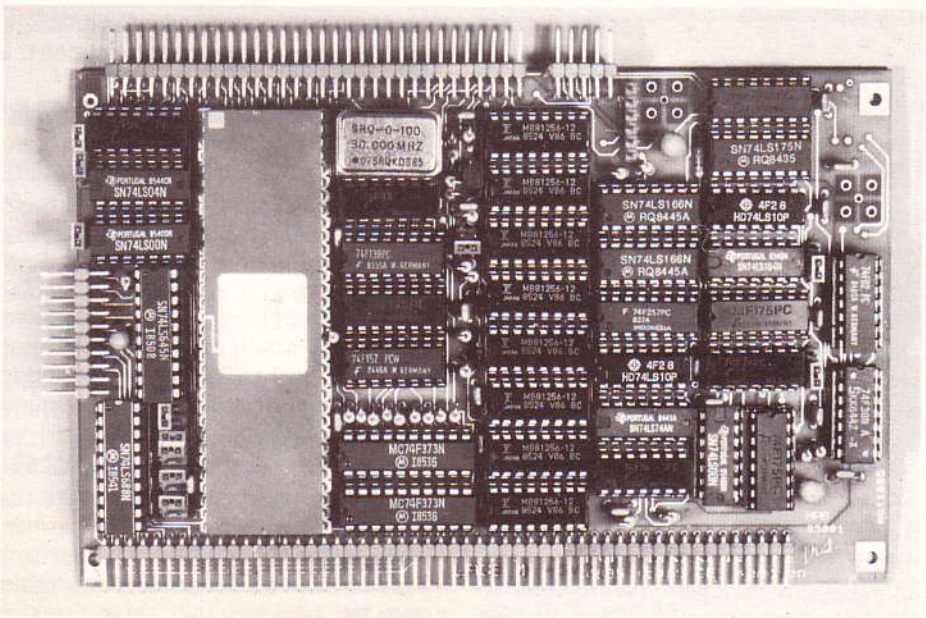
Adresse: GES, Herr Graf, Postfach 1610, 8960 Kempten (Allgäu).

P.S: Unsere bisherigen Software-Partner brauchen sich natürlich nicht nochmals zu melden!

Jetzt lieferbar - Jetzt lieferbar

Schon auf vielen Messen gezeigt, jetzt lieferbar:

ACRTC-Baugruppen für den NDR-Computer



Auf unseren letzten Messen (Systems München, Hannover-Messe) fand eine Baugruppe besonderes Interesse: das ACRTC-System für den NDR-Computer.

ACRTC heißt „Advanced Cathode Ray Tube Controller". Das System besteht aus zwei Baugruppen:

ACRTC-Basis, das komplette System mit schwarz-weiß Ausgang

und

ACRTC-Color, die Farberweiterung.

Beide Baugruppen werden mit einem 64poligen Flachbandkabel miteinander verbunden. Die ACRTC-Basis-Baugruppe enthält den Baustein HD63484ACRTC von Hitachi. Sie ist allein völlig funktionsfähig und enthält alle nötigen Timing- und Baugruppen-Elemente.

Möglichkeiten des Schwarz/Weiß-Systems ACRT-BASIS

Darstellbar sind 640 * 240 Punkte im normalen Darstellungsmodus (ohne Zeilensprung). Dies ist die Standard-Auflösung, die von der Hardware vorgegeben ist (Pixel-Frequenz 15MHz). Es ist möglich, mit dem Zeilensprung-Verfahren

640 x 480 Punkte darzustellen; dann wird jedoch ein Monitor mit langnachleuchtender Bildröhre empfohlen.

Es stehen 256KByte Bildspeicher zur Verfügung, die z.B. für 13 Darstellungsseiten genutzt werden können. Dies ermöglicht den Bildaufbau im Hintergrund (wichtig für bewegte flimmerfreie Darstellungen).

Der Bildschirm kann aufgeteilt werden in drei unabhängige Bereiche, deren Größe und Lage, mit Einschränkungen, im Bildspeicher frei wählbar sind. Es können so völlig unabhängige Statuszeilen am oberen und unteren Bildschirmrand eingeblendet werden. Des weiteren ist ein sogenanntes Window (Fenster) frei definierbar, es kann verdeckend oder durchscheinend zum sonstigen Bildinhalt dargestellt werden. Es ist auch als spezieller Cursor, als Informationstafel oder auch für unabhängige Beschriftungen beliebig verwendbar. Die Zeichengeschwindigkeit beträgt im normalen Darstellungsmodus (keine Bildstörungen) 533 ns pro Bildpunkt. Eine Verdoppelung der Zeichengeschwindigkeit ist möglich, allerdings können dann Bildstörungen auftreten; es können dann zwei Millionen Punkte pro Sekunde gezeichnet werden. Damit sollte der für manche Aufgaben benötigte Aufbau eines kompletten Vollbildes (z.B. Drahtmodell) in 20 ms möglich sein.

Für den Hauptbildbereich (BASE SCREEN) ist ein Hardware-Zoom möglich; dieser ist 1- bis 16fach wählbar. Außerdem kann jeder Bildschirmbereich getrennt in vier Richtungen verschoben werden („Smooth-Scroll"). Damit kann z.B. ein großes Darstellungsfeld vereinbart werden (z.B. 1024 * 1024), wobei dann der Bildschirm ein Fenster (eine Lupe) auf dieses Feld bildet.

Der ACRTC wandelt logische X-, Y-Koordinaten in die physikalischen Adressen im Bildschirm-RAM um, so daß der Anwender sich nicht um die Aufteilung des Speichers kümmern muß. Außerdem bietet der Baustein in Hardware realisierte Funktionen für das Zeichnen von Linien, Rechtecken, Polygonen, Kreisen, Ellipsen, Kreisbogen, Ellipsenbogen, gefülltes Viereck, Ausfüllen beliebiger Umrisse mit Mustern, Kopieren von Ausschnitten.

Diese Baugruppe kann direkt mit der ACRT-Color erweitert werden.

Das ACRT-Farbsystem

Wenn man die ACRT-BASIS mit der ACRT-COLOR erweitert, so erhält man das komplette Farbsystem. Es bestehen weiterhin alle beim ACRT-BASIS beschriebenen Möglichkeiten! Die Zeichengeschwindigkeit für Farbdarstellungen ist genauso hoch wie für Schwarz/Weiß – der ACRTC ist voll farbtauglich! Die Farbe eines Bildpunktes wird durch 4 Bit beschrieben. Je ein Bit enthält die Information für den Rot-, Grün-, Blau- und Helligkeitsanteil des Bildpunktes. Der ACRTC behandelt diese 4 Bit auch als Punkt und übernimmt die Speicherorganisation. Es können 16 Farben auf einem normalen Farbmonitor (auch IBM-kompatible Farbmonitore) dargestellt werden. Die Baugruppe ACRT-Color enthält 768KByte RAM (3 * 256KByte), so daß insgesamt 1 MByte RAM benutzbar ist. Ein Byte enthält die Information für zwei Punkte, der Anwender kann also auf über 2 Millionen Bildpunkte zugreifen.

Erweiterungen Mit der Color-Look-Up-Table (CLUT, Farbtabelle) können 16 aus 262144 Farben dargestellt werden. Verzichtet man auf die Anzeige des Windows, so können sogar 256 aus 262144 Farben gleichzeitig auf einem Farbmonitor (mit analogen Eingängen) dargestellt werden.

Das ACRTC-System ist als industrielles Grafik-System konzipiert. Wir haben es jedoch auf den NDR-Computer-BUS angepaßt, um auch professionell arbeitenden NDR-Computer-Benutzern die Möglichkeit zu geben, dieses System zu verwenden. Ebenso ist es sehr leicht möglich, mit NDR-Computer-Baugruppen ein intelligentes ACRTC-Subsystem aufzubauen. Wir haben ein solches Subsystem mit der CPU68008, einer ROA64 und einer SER-Karte bereits realisiert. Es ist damit an jeden Personal-Computer mit V24-Schnittstelle anzuschließen.

Die Preise:

ACRTC-Basis komplett, völlig aufgebaut, geprüft und funktionsfähig
Bestell-Nr. 10062 DM 1.300,–

ACRTC-Farbe, Farberweiterung, 8 Farben plus Intensity, Ausgang nach IBM-Belegung
Bestell-Nr. 10064 DM 1.300,–

ACRTC-Handbuch
Bestell-Nr. 10195 DM 20,–

Da die Baugruppen auf einer sechsfachen Multi-Layer-Baugruppe basieren, ist eine Bausatz- oder Platinenlieferung wenig sinnvoll und daher nicht möglich.

ACRTC-Library
Programm-Paket zur Ansteuerung unter 68000 – CP/M68K oder „C"
Bestell-Nr. 10561 DM 400,–

Handbuch der Library, einzeln
Bestell-Nr. 10562 DM 50,–

Modula und 16 bit-Erweiterung
Bestell-Nr. 10799 DM 149,–

COL 256: Bis zu 1280 x 600 Punkte

Auflösung:	Farben:	Syncron:	Bildfrequenz:
256 * 256	256		50 Hz
256 * 512	256	Interlace	25 Hz
320 * 200	256		50 Hz
320 * 400	256	Interlace	25 Hz
320 * 300	256		50 Hz
320 * 600	256	Interlace	25 Hz
320 * 256	256		50 Hz
320 * 512	256	Interlace	25 Hz
512 * 256	16		50 Hz
512 * 512	16	Interlace	25 Hz
640 * 200	16		50 Hz
640 * 400	16	Interlace	25 Hz
640 * 300	16		50 Hz
640 * 600	16	Interlace	25 Hz
640 * 256	16		50 Hz
640 * 512	16	Interlace	25 Hz
1024 * 256	4		50 Hz
1024 * 512	4	Interlace	25 Hz
1280 * 200	4		50 Hz
1280 * 400	4	Interlace	25 Hz
1280 * 300	4		50 Hz
1280 * 600	4	Interlace	25 Hz
1280 * 256	4		50 Hz
1280 * 512	4	Interlace	25 Hz

Anwenderbericht von Key Thomsen

Mit der COL 256 lassen sich nicht, wie anfangs vermutet nur 256 x 256 und 320 x 200 Punkte darstellen, sondern jede beliebige Auflösung bis 1280 x 600 Punkten. Die nebenstehenden Auflösungen lassen sich recht einfach darstellen.

Die COL 256 hat einen EURO-SCART Ausgang mit dem sich alle Auflösungen, die mit 256 Farben arbeiten, erzeugen lassen (z.B. 256 x 256).

Mit dem IBM-Ausgang lassen sich alle die Auflösungen mit 16 Farben erzeugen (z.B. 512 x 256).

Mit etwas Hardwareaufwand von 2 IC's lassen sich die Bilder mit 4 Farben oder 4 Grauzonen erzeugen (z.B. 1024 x 256).

Ein Nachteil ist allerdings, daß man für die höheren Auflösungen die etwas teureren RAM's mit 64K x 4 benötigt. Die RAM-Erweiterung wird von GES angeboten: Bestell-Nr. 10774. Man muß also die alten RAM's 4116 durch die RAM'S 4464 austauschen, was in 5 Minuten passiert ist. Damit sind dann auf der COL 256 genau 256KByte RAM.

Weitere Eigenschaften der COL 256.

Da die COL 256 nicht wie die GDP 64 auf Vektorbasis arbeitet, sondern nur den Bildspeicher verwaltet und diesen immer ausgibt, kann man auch auf den Bildspeicher frei zugreifen, um einen Punkt zu

modifizieren oder auszulesen. Damit ist es möglich, sehr schnell Bilder aufzubauen oder auszulesen. Es lassen sich **Windows** softwaremäßig durch einfaches Verschieben eines Speichers einblenden und auch wieder ausblenden. Weiterhin ist das Ausfüllen von Flächen in einem rasanten Tempo möglich.

Als ein weiteres großes Plus ist das hardwaremäßige Scrollen von Bildern zu nennen. Der CRT 6845 hat ein Register, mit dem man angeben kann, an welcher Speicherstelle die Bildausgabe beginnen soll. Durch dieses Register ist es möglich, in Viererschritten ein Bild in X- und Y-Richtung zu verschieben, ohne den Speicher zu verändern.

Nachteile hingegen ergeben sich bei schnellen Bildänderungen: So zum Beispiel kann die COL 256, im Gegensatz zu den hochgezüchteten Grafikchips, kein Hardware ZOOM oder Hardware Window erzeugen. Ein Hardware ZOOM zeichnet sich dadurch aus, daß die Bildvergrößerung sehr schnell vonstatten geht, im Gegensatz zur COL 256, bei der dies nur durch Speicherverschiebungen möglich ist.

Hardware Windows lassen sich sehr sauber und schnell bewegen, was bei der COL 256 im kleineren Maße bis 25 x 25 Bildpunkte auch noch recht flimmerfrei geht. Bei größeren Bildern hingegen ergeben sich leichte Störungen, die durch einige Tricks aber zu beheben sind.

Ein gewaltiger Vorteil gegenüber den Grafikchips besteht darin, daß man einzelne Bildpunkte schneller ansprechen kann, da der Datentransfer hier direkt ins RAM geschieht.

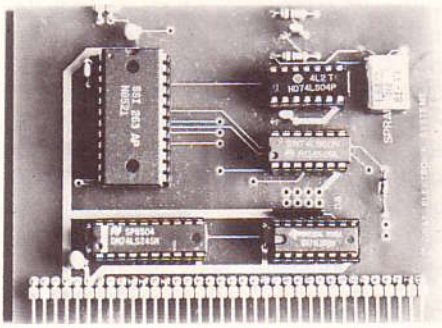
Weiterhin kann man bei der COL 256 recht einfach neue Befehle erzeugen, die in der Geschwindigkeit den übrigen Befehlen gleich kommen. Dies geht bei Grafikkarten mit Vektorgrafik nur recht schwierig und zeitraubend.

Die COL 256 ist für Anwendungen von bewegten Grafiken über Spiele und Daten/Textverarbeitung bis hin zu großen CAD-Programmen sehr gut zu gebrauchen.

Das demnächst erscheinende COL 256 Farbgrafikprogramm wird alle Auflösungen von 256 x 256 bis hin zu 1280 x 600 voll unterstützen. Es steht dann eine Befehlsliste von über 80 Befehlen zur Verfügung, die man natürlich selbst erweitern kann.

Sprache

Die Sprache-Baugruppe zum NDR-Computer ist ab Lager lieferbar



Die Sprache-Baugruppe dient, wie der Name schon sagt, zur Ausgabe von Sprache. Zu diesem Zweck wird auf der Baugruppe der Sprach-Synthese-Baustein SSI 263 A eingesetzt, der auch deutsche Laute kann. Dieser Baustein verfügt über 5 Register, die zur Programmierung der einzelnen Phoneme dienen. Unter „Phonemen" versteht man hörbare Laute, die zusammengesetzt Worte ergeben. Mit den 64 Phonemen, die der Sprache-Baustein zur Verfügung stellt, können alle Worte nachgebildet werden.

Außerdem können noch Lautstärke, Frequenz, Hüllkurve und 5 verschiedene Filter eingeschaltet werden, die der Sprache eine gewisse Individualität verleihen. Vor allem die Programmierung mit den Filtern macht viel Spaß und man kann mit etwas Erfahrung diese Filter dann sinnvoll einsetzen.

Die Sprache-Baugruppe hat keinen Verstärker und kann direkt an einen Kassettenrecorder oder eine Stereoanlage angeschlossen werden.

Technische Daten:
Spannung: + 5 V
Stromaufnahme: 100 mA
Leiterplattengröße: 75 x 100 mm
BUS: NDR-Bus

Daten zum Baustein SSI 263 A
- 64 Phoneme
- Lautstärke: 16 Stufen
- Sprachrate (Sprachgeschwindigkeit): 16 Stufen
- Sprachtraktfilter: 5

SPRACHEH
Handbuch zur SPRACHE
Bestell-Nr. 10403 DM 10,00

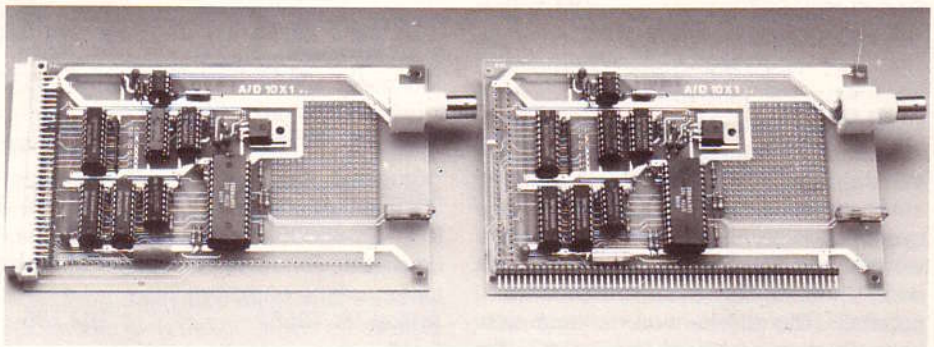
SPRACHEP
Leiterplatte SPRACHE
Bestell-Nr. 10404 DM 39,50

SPRACHEB
Bausatz SPRACHE
Bestell-Nr. 10401 DM 398,00

SPRACHEF
Fertiggerät SPRACHE
Bestell-Nr. 10402 DM 478,00

AD 10 x 1

Die AD-Wandler-Baugruppe zum NDR-Computer ist ab Lager lieferbar.



Die AD 10 x 1 ist eine Baugruppe, die analoge Spannungen in digitale Bitmuster wandelt. Dabei werden Spannungen von − 10 V bis + 10 V in 10-Bitworte gewandelt.

Diese Baugruppe ist in unserer neuen Norm ausgeführt, d.h. sowohl für den NDR- als auch für den ECB-Bus einsetzbar. Damit kann die Baugruppe auch für den mc-CP/M-Computer verwendet werden.

Die AD 10 x 1 ist in jeder Konfiguration des NDR- und des mc-CP/M-Computers einsetzbar. Im mitgelieferten Handbuch sind sowohl für 680xx Konfigurationen als auch Z80-Konfigurationen Abgleichprogramme und kleine Demoprogramme enthalten. Für Z80-Grundprogramme ist sogar ein kleines Oszilloskop-Programm enthalten, mit dem Oszilloskop (bis max. 6 – 10 kHz) mit dem AD 10 x 1 nachgebildet werden kann.

Technische Daten:
Spannung: + 5 V, + 12 V und − 12 V
(− 5 V wird auf der Baugruppe erzeugt)
Stromverbrauch: + 5 V: 250 mA
− 5 V: 35 mA
+ 12 V: 40 mA
− 12 V: 80 mA
Leiterplattengröße: 100 x 160
(Europakartenformat)
BUS: NDR-Bus oder ECB-Bus wahlweise!
Wandlungszeit: 20 us
Eingangsspannung:
unipolar: 20 V
bipolar: − 10 V bis + 10 V
Auflösung: 10 Bit = 20 m V/LSB

AD 10 x 1H
Handbuch zum AD 10 x 1
Bestell-Nr. 10069 DM 10,00

AD 10 x 1P
Leiterplatte AD 10 x 1
Bestell-Nr. 10070 DM 39,00

AD 10 x 1B
Bausatz AD 10 x 1
Bestell-Nr. 10067 DM 265,00

AD 10 X 1F
Fertiggerät AD 10 X 1
Bestell-Nr. 10068 DM 384,00

ATARI-Maus für Hardcopy-Maus nun verfügbar!

Die ATARI-Maus wurde für die Hardcopy des NDR- bzw. mc-Computers mit einem 9poligen CANON-Stecker versehen, der direkt in die Hardcopy-Maus einzustekken ist (Änderung aus *LOOP 7* eingebaut!). Somit ist eine einwandfreie Funktion ohne weitere Eingriffe gewährleistet. Die beiden Tasten können über Port 8BH Bit 6 und 7 eingelesen werden. Außerdem kann die umgebaute ATARI-Maus u.a. auch am APPLE-IIc und – Macintosh eingesetzt werden.
Bestell-Nr. 10570 DM 148,−

CP/M68K V1.3

Von der Firma Digital Research haben wir eine neue Version des Diskettenbetriebssystems CP/M68K bekommen. Ab sofort werden alle Disketten auch von uns an den Kunden in der neuen Version weitergegeben. Die alte Version meldete sich beim Booten mit CP/M-Version 1.2, die neue mit Version 1.3.

Mitgeliefert wird außerdem das neue BIOS (wie in der mc beschrieben), mit dem nun auch der Anschluß einer Festplatte (10 MByte) möglich ist. Eine RAM-FLOPPY wird automatisch gefunden und unterstützt (bis 1 MByte). Das Betriebssystem ist für 128 KByte RAM konfiguriert, mit beigefügten Programmen ist eine Anpassung für 256 oder 512 KByte möglich.

3 ½" für 68008 mit 1 BOOT-EPROM
Artikel-Nr. 10568 DM 775,−
5 ¼"
Artikel-Nr. 10163 DM 775,−
8"
Artikel-Nr. 10569 DM 775,−

3 ½" für 68000 mit 2 BOOT-EPROMS
Artikel-Nr. 10558 DM 795,−
5 ¼"
Artikel-Nr. 10559 DM 795,−
8"
Artikel-Nr. 10560 DM 795,−

3 ½" für 68020 ohne BOOT-EPROM, da dieses schon bei der CPU 68020 im Lieferumfang enthalten ist
Artikel-Nr. 10552 DM 755,−
5 ¼"
Artikel-Nr. 10571 DM 755,−
8"
Artikel-Nr. 10178 DM 755,−

Im Lieferumfang immer enthalten:
- komplette englische DR-Dokumentation
- deutsches Handbuch vom Franzis-Verlag

Update der V1.2 gegen 10,– DM pro Diskette möglich! Bitte alle Original-Disketten einsenden.

Erste TOOL-Diskette für den NDR-Computer mit CPU68K (CPU68000)

Das Programm MORX dient zum schnellen Durchblättern langer Dateien. Es zeigt sehr anschaulich, wie schnell die GDP64 sein kann, wenn man sie „richtig" programmiert (hier nur für 68008 gezeigt).

MORZ listet einen Text in Kurzvektor-Schrift (Plotschrift).

EDIT ist aus dem Programm EDITRDK entstanden. Folgende Verbesserungen wurden implementiert: Auflösung Tabulatorzeichen, Ladeadresse $1000, Abfrage ob geänderte Version gespeichert werden soll, Erstellung eines BAK Files (Sicherheitskopie der letzten bearbeiteten Version), ASSRDK LADE SAVE mit Auflösung Tabulatorzeichen, Ladeadresse $1000, STARTE auch mit Namen (Bibliothekseinträge).

BIBLMEN stellt die speicherresidenten Programme mit Bibliothekskopf in Form einer sortierten Liste dar. In dieser Liste kann vor- oder zurückgeblättert, alphabetisch positioniert und gestartet werden.

DISCCOPY kopiert Disketten im NDR-Format auf einem Laufwerk sektorenweise.

EDITERM bietet neue Routinen für den Grundprogramm-Editor, mit diesen können dann Texte über 64KByte editiert werden; außerdem ist die Scroll-Geschwindigkeit auch bei langen Texten erhöht. Funktioniert nur mit dem Grundprogramm V4.3 und erfordert ein zusätzliches EPROM.

Das Programm INITIAL erstellt über eine oder mehrere Tabellen symbolische Zuweisungen für I/O-Adressen, selbsterstellte Unterprogramme und speicherresidente Programme mit Bibliothekskopf.

Neue Unterprogramme DRAWTO, DRAWMOV, MOVETO, BEGRXY, ZEICHEN, WRITE, SCHREIBTE, SCHR16TEL.

Alle Programme werden in dem SUBMIT-File INFO ausführlich beschrieben und probegestartet. Für den 68000 gibt es das File INFO1, allerdings müssen die meisten Programme erst an diesen Prozessor angepaßt werden.

TOOL01 3 ½"
Bestell-Nr. 10555 DM 39,00
TOOL01 5 ¼"
Bestell-Nr. 10556 DM 39,00
TOOL01 8"
Bestell-Nr. 10557 DM 39,00

ZEAT-Kassettenprogramm auf Diskette?

Da immer mehr ZEAT-Anwender auf den Betrieb mit Diskettenlaufwerken umsteigen, ergibt sich der Wunsch, die alten Kassettenprogramme auf Disketten abzuspeichern.

Selbstverständlich soll der Text dann auch wieder im EDITOR stehen, um ihn weiter zu bearbeiten.

Die Antwort darauf, und vieles mehr wie Diskettenbetrieb unter CP/M, CP/M-Kommandos, Formatieren von Disketten, Systemfunktionen usw. finden Sie im neuen ZEAT-Lehrbrief 5 unter der Bezeichnung „ZEAT-Diskettenbetrieb mit CP/M".
Bestell-Nr. 10563 DM 38,−

DISKDOC - Ein Editor für schlecht zugängliche Bytes

Neue Diskette – geschrieben in TURBO-Pascal mit Quellisting

Im Umgang mit Disketten unter z. B. CP/M-Betriebssystem, merkt man recht wenig von den Seiten, Spuren und Sektoren, in die eine Diskette aufgeteilt ist.

Grund dafür ist die Hauptaufgabe eines Betriebssystems, ein komfortables Arbeiten für den Anwender zu ermöglichen.

Es gibt aber auch Fälle, bei denen man sich um spezielle (kranke) Bytes auf der Diskette kümmern muß und der „Disketten-Doktor" DISKDOC.COM eingesetzt werden muß.

Das Programm DISKDOC.PAS ist in TURBO-Pascal geschrieben und in der Lage, einen frei wählbaren Sektor von der Diskette zu lesen und nach Bearbeitung wieder in diesen Sektor zurückzuschreiben.

Eine wichtige Anwendung des DISKDOC.COM ergibt sich z.B. bei einer nicht gewollten Löschung von Files, da sie durch Änderung eines Bytes im richtigen Sektor rückgängig gemacht werden kann.

Nach dem Aufruf meldet sich das Programm mit einem Menue und einem Sektor als HEX-Dump, z.B.:

```
↑I    = Info      ↑Q = ENDE                          BUFFERSEKTOR: 0
↑P    = Seite / Spur / Sektor neu waehlen            SEITE  : 0
↑Z    = aktuellen Sektor auf Diskette schreiben      SPUR   : 0
↑K/↑L = Vorhergehenden / Naechsten  Sektor lesen     SEKTOR : 1
↑A/↑H = ASCII($20..$7F) / HEX Eingabe
        Cursorsteuerung wie Turboeditor = WS

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000:>31<00 01 21 00 D4 06 07 16 00 1E 02 E5 D5 C5 0E   1..!.T.....eUE.
010: D0 06 01 CD 27 F0 C1 D1 E1 DA 1E F0 D5 11 00 04   P..M'pA@aZ.pU...
020: 19 D1 1C 7B FE 06 DA 2C FC 1E 01 14 05 C2 0C FC   .Q.{~.Z,|....B.|
030: C3 00 EA F5 CD FD 00 F5 3E 80 D3 C8 F1 ED 7B 4D   C.juM}.u>.SHqm{M
040: F1 C9 AF D3 C8 ED 73 4D F1 31 56 F5 CD 07 01 F5   qI/SHmsMq1VuM..u
050: 3E 80 D3 C8 F1 ED 7B 4D F1 C9 AF D3 C8 ED 73 4D   >.SHqm{Mq1/SHmsM
060: F1 31 56 F5 CD 65 02 F5 3E 80 D3 C8 F1 ED 7B 4D   q1VuMe.u>.SHqm{M
070: F1 C9 AF D3 C8 ED 73 4D F1 31 56 F5 CD 66 04 3E   qI/SHmsMq1VuMf.>
080: C3 5C D7 C3 58 D7 7F 00 20 20 20 20 20 20 20 20   C\WCXW▒.
090: 20 20 20 20 20 20 20 20 43 4F 50 59 52 49 47 48           COPYRIGH
0A0: 54 20 28 43 29 20 31 39 37 39 2C 20 44 49 47 49   T (C) 1979, DIGI
0B0: 54 41 4C 20 52 45 53 45 41 52 43 48 20 20 00 00   TAL RESEARCH  ..
0C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Hier kann mit den üblichen CURSOR-Funktionen der Sektor wahlweise durch HEX- oder ASCII-Eingabe geändert und auf einer beliebigen Seite, Spur und Sektor wieder eingetragen werden.

Der Sektor kann fortlaufend „umgeblättert", aber auch wie die Seiten und Spur frei gewählt werden. Die Bildschirmmaske beschreibt die Anwendung der Steuerzeichen.

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00: 00 50 49 50 20 20 20 20 20 20 43 47 4D 20 20 20 30      .PIP COM . . . .
01: 00 usw.
```

Da im Lieferumfang ein ausführliches Textfile zur Anwendung des Programmes (Beispiel: gelöschtes File retten) enthalten ist, wird der Einsatz ausgesprochen problemlos.

Auch die Source, das Programm DISKDOC.PAS wird mitgeliefert!

Diskette 5¼", 80 Spuren
Bestell-Nr. 10550 DM 39,–

Diskette 3½", 80 Spuren
Bestell-Nr. 10549 DM 39,–

Diskette 8", SS SD
Bestell-Nr. 10092 DM 39,–

Lieferzeit: Ab Lager.

Z 80-Vollausbau bis ZEAT

Rekursives Programmieren – schon mit der SBC 2

von Jens Decker

Zum Programm:

Dieses Programm stellt eine in reinem Z80-Code geschriebene Version des Baumprogramms aus *LOOP* 6 („Einführung in C, Teil 3") dar. Die Baum-Konstruktion entspricht der Konstruktionsvorlage von Rolf-Dieter Klein, außer, daß die Winkel, die die Äste mit der Stammrichtung verbinden, nicht symmetrisch gewählt wurden, um das Bäumchen etwas lebensnäher wirken zu lassen.

Obwohl dieses Programm für Fortgeschrittene sicherlich mühelos nachvollzogen werden kann, möchte ich im Interesse der Anfänger, die zum ersten Mal rekursiv programmieren, einige Erläuterungen geben.

Rekursive Programmierung bedeutet, daß sich ein Programmteil selbst aufruft, quasi sein eigenes Unterprogramm ist. Dadurch ergeben sich zwei Probleme:

- Das Programm muß wissen auf welcher Ebene es sich befindet (wie oft es sich bereits aufgerufen hat), und
- Variablen (Daten in Registern oder Speicherzellen) müssen nach der Rückkehr in die nächst höhere Ebene oft wieder den Wert annehmen, den sie vor dem Abstieg in die tieferen Ebenen hatten.

Das erste Problem wird dadurch gelöst, daß man einen Merker einführt. Dieser wird vor der Rekursion auf den Wert der untersten Ebene gesetzt (hier zum Beispiel 15, also 15 Ebenen). Bei jedem Abstieg in eine tiefere Ebene wird der Merker um 1 verringert, bis man auf der untersten Ebene bei Merker = 0 ankommt. Da das Programm bei jedem Abstieg überprüft, auf welcher Ebene es sich befindet, kann es nun das letzte Ästchen zeichnen und danach wieder „auftauchen". Dabei wird der Merker wieder erhöht. Dies ist nötig, falls wie hier, nochmals abgestiegen werden muß, um den zweiten Ast zu zeichnen, wofür es ja wieder wichtig ist, zu wissen auf welcher Ebene man liegt.

Eng damit verbunden ist das zweite Problem: Wert der Variablen. Nach der Rückkehr müssen wieder die alten Variablen

verfügbar sein, da im Unterprogramm diese zerstört wurden (so wird ja die Länge des Stamms kleiner!).

Dies geschieht hier dadurch, daß die alte Länge im BC-Register auf den Stack kopiert wird und nach der Rückkehr von dort wieder ins BC-Register gelesen wird. Da der Stack vom Mikroprozessor verwaltet wird, um darin die Rückkehradresse vom Unterprogramm zu speichern, kommen die Daten automatisch in der richtigen Ebene wieder heraus, nachdem der Prozessor die Rückkehradresse vom Stack nimmt.

Symbole:

```
87fe merker
8800 div16
8818 neulang
8826 aestchen
8841 baum
888d aufruf
```

Programmcode:

```
div16     21 00 00
          06 10
          cb 11
          cb 17
          ed 6a
          ed 52
          30 01
          19
          3f
          10 f2
          cb 11
          cb 17
          c9
neulang   11 03 00
          78
          60
          69
          29
          7c
          4d
          cd div16
          47
          c9
aestchen  c5
          e1
          cd schreite
          21 b4 00
          cd drehe
          c5
          e1
          cd schreite
          21 b4 00
          cd drehe
          21 merker
          34
          c9
baum      21 merker
          35
          3a merker
          fe 00
          20 08
          cd aestchen
          21 merker
          34
          c9
          c5
          e1
          cd schreite
          21 1e 00
          cd drehe
          c5
          cd neulang
          cd baum
          21 merker
          35
          21 ce ff
          cd drehe
          c1
          c5
          cd neulang
          cd baum
          21 c8 00
          cd drehe
          e1
          cd schreite
          21 b4 00
          cd drehe
          21 merker
          34
          c9
aufruf    21 00 01
          11 00 00
          01 5a 00
          cd set
          21 0f 00
          22 merker
          01 a0 00
          cd baum
          c9
```

Fractals mit SBC2 und Grundprogramm

von Jens Decker

Dieses Programm ist das erste einer Folge von Programmen zur Darstellung von Fractals. Ein Fractal ist ein Gebilde, das sich beliebig vergrößern läßt. Dies wäre zunächst nichts besonderes, läßt sich doch jedes Gebilde vergrößern. Das Besondere jedoch ist, daß auch nach einer Vergrößerung um den Faktor 1 zu 1000000000 (wobei man natürlich nur noch einen Ausschnitt sehen kann) immer neue Details auftauchen, die noch dazu Selbstähnlichkeit mit dem Ganzen besitzen. Und wie wird das nun erreicht?

Ganz einfach – durch rekursive Anwendung einfacher Regeln, das jedoch (theoretisch) unendlich tief ist.

Da die Praxis meist einfacher ist wie die Theorie, nehme der Leser ein Blatt Papier und falte es in der Mitte. Dieses gefaltete Blatt wird nun wieder geöffnet, so daß ein 90°-Winkel entsteht. Dies ist eine Dragonkurve der Tiefe 1. Und wie erhält man nun eine Dragonkurve der Tiefe 2, 3, 4 usw.?

Nun, man lege das Blatt wieder zusammen und falte es nochmals in der Mitte, so daß die alte Faltkante auf die Enden des Blattes kommt, und nach dem Öffnen drei Kanten vorhanden sind. Ist der Winkel an diesen Kanten wiederum jeweils 90°, so hat man eine Dragon-Kurfe der Tiefe 2 vor sich. Man muß also einfach die alte Kante auf die Enden des Blattes legen und dies immer wieder wiederholen. Nach dem Öffnen ergeben sich dann mit wachsender Tiefe immer kompliziertere Gebilde. Bei unendlich oftmaligem Falten erhielte man dann das eigentliche Fractal. Da Papier jedoch nicht dazu angetan ist, unendlich oft, oder doch wenigstens bis zu einer ästhetisch annehmbaren Tiefe gefaltet zu werden, ist es sinnvoll, mit dem Computer eine Näherung des Fractals durchzuziehen, die die Auflösung des Bildschirms erreicht.

Dabei sieht man das Fractal dann quasi unscharf, eben mit der Auflösung des Bildschirms. Will man etwas schärfer sehen, so muß man eben bei gleicher Schrittweite eine höhere Tiefe wählen und die Turtle eventuell an eine andere Anfangsstelle positionieren, um das gewünschte Teil zu sehen.

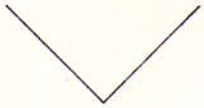
Manch einer wird nun fragen, wie man dem Computer das Papierfalten beibringt, noch dazu am Bildschirm. Hier wird ein kleiner Trick angewandt:

Man malt einfach den Winkel, doch statt Schreite – Drehe – Schreite läßt man

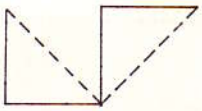
rechter Winkel – Drehe – linker Winkel durchführen. Ebenso wird beim rechten bzw. linken Winkel verfahren, so daß man immer tiefer kommt (Routinen RECHTS bzw. LINKS). Ist man in der gewünschten Tiefe angekommen, wird schließlich der Winkel gezeichnet (Routinen RE bzw. LI). Wer dies nachvollziehen will, kann ja bei TIEFE die Tiefe 4 eintragen und bei Einzelschritt das Programm durchlaufen lassen.

Will man – was sehr empfehlenswert ist – mit Tiefen von 14 und mehr arbeiten, so sollte man in den Routinen RE und LI die Befehle 21 06 00 durch 21 01 00 ersetzen, wodurch man eine Dragonfläche erhält, und in den Spiralarmen deutlich sieht, was Selbstähnlichkeit bedeutet.

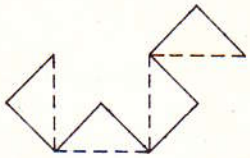
Doch Vorsicht! Ebenso wie beim Papierfalten die Dicke, wächst hier die Rechenzeit, und zwar exponentiell zur Tiefe (eine Dragonkurve der Tiefe 14, aus 0,1 mm dikkem Papier gefaltet, wäre zusammengelegt 1,6 m dick!!!).



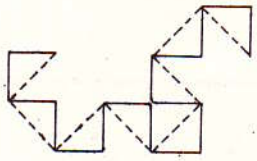
Tiefe 1



Tiefe 2



Tiefe 3



Tiefe 4

Dragon-Kurve

```
RE      := 8800H
LI      := 8813H
MERKER  := 8850H
RECHTS  := 8860H
LINKS   := 8890H
GANZ    := 88D0H
TIEFE   := 88D0H

RE      : 21 06 00      ld hl, 0006H
          CD SCHREITE   call SCHREITE     um 6 schreiten
          21 A6 FF      ld hl, -90D
          CD DREHE      call DREHE        um 90° nach rechts drehen
          21 06 00      ld hl, 0006H
          CD SCHREITE   call SCHREITE     um 6 schreiten
          C9            ret               Ende rechter Winkel

LI      : 21 06 00      ld hl, 0006H
          CD SCHREITE   call SCHREITE     um 6 schreiten
          21 5A 00      ld hl, 90D
          CD DREHE      call DREHE        um 90° drehen
          21 06 00      ld hl, 0006H
          CD SCHREITE   call SCHREITE     um 6 schreiten
          C9            ret               Ende linker Winkel

RECHTS  : 21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          35            dec (hl)          Merker um eins erniedrigen
          3A MERKER     ld a, (MERKER)    a aus Merker laden
          FE 01         cp 01H            mit 1 vergleichen
          20 08         jr nz,+8          falls ungleich Rekursion
          CD RE         call RE           sonst rechter Winkel
          21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          34            inc (hl)          Merker um eins erhöhen
          C9            ret               Aufstieg in die nächst höhere
                                            Ebene

          CD RECHTS     call RECHTS       rekursiver Abstieg
          21 A6 FF      ld hl, -90D
          CD DREHE      call DREHE        um 90° drehen
          CD LINKS      call LINKS        rekursiver Abstieg
          21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          34            inc (hl)          Merker um eins erhöhen
          C9            ret               Aufstieg in die nächst höhere
                                            Ebene

LINKS   : 21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          35            dec (hl)          Merker um eins erniedrigen
          3A MERKER     ld a, (MERKER)    a aus Merker laden
          FE 01         cp 01H            mit 1 vergleichen
          20 08         jr nz,+8          falls ungleich Rekursion
          CD LI         call LI           sonst linker Winkel
          21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          34            inc (hl)          Merker um eins erhöhen
          C9            ret               Aufstieg in die nächst höhere
                                            Ebene

          CD RECHTS     call RECHTS       rekursiver Abstieg
          21 5A 00      ld hl, 90D
          CD DREHE      call DREHE        um 90° drehen
          CD LINKS      call LINKS        rekursiver Abstieg
          21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          34            inc (hl)          Merker um eins erhöhen
          C9            ret               Aufstieg in die nächst höhere
                                            Ebene

GANZ    : 21 50 00      ld hl, 0050H
          11 2C 01      ld de, 012CH
          01 5A 00      ld bc, 005AH
          CD SET        call SET          Turtle passend positionieren
TIEFE-1 : 3E 0C         ld a, 0CH         a mit der gewünschten Tiefe
          21 MERKER     ld hl, MERKER       laden
          77            ld (hl),a         Merker auf die Tiefe setzen
          CD RECHTS     call RECHTS       rechter Winkel als Superstrukt.
          C9            ret               Ende des Programms

Merker  := 87FEH
Neulang := 8818H
Div16   := 8800H
Aestchen := 8826H
Baum    := 8841H

Div16   : 21 00 00      ld hl,0000H       Lösche Akkumulator
          06 10         ld b,10H          Bitzähler auf 16D
          CB 11         rl c
          CB 17         rl a              Dividend raus-Ergebnis rein
          ED 6A         adc hl,hl         links schieben
          ED 52         sbc hl,de         Subtraktion
          30 01         jr nc,+1          möglich?
          19            add hl,de         Akkumulator wieder herstellen
          3F            ccf               Ergebnisbit erzeugen
          10 F2         djnz,-14          Ende der Schleife
          CB 11         rl c
          CB 17         rl a              letztes Ergebnisbit rein
          C9            ret               Ende der Divisionsroutine

Neulang : 11 03 00      ld de,0003H       Divisor für Div16
          78            ld a,b
          60            ld h,b
          69            ld l,c            bc nach hl
          29            add hl,hl         verdoppeln
          7C            ld a,h
          4D            ld c,l            und nach ac (=Dividend)
          CD DIV16      call DIV16        dividieren
          47            ld b,a            und Ergebnis nach bc
          C9            ret               Ende der Längenberechnung
                                          Länge = Länge * 2/3

Aestchen : C5           push bc
          E1            pop hl            bc nach hl
          CD SCHREITE   call SCHREITE     Zeichnen des Ästchens
          21 B4 00      ld hl, 180D       um 180° drehen
          CD DREHE      call DREHE        "
          C5            push bc
          E1            pop hl            bc nach hl
          CD SCHREITE   call SCHREITE     zurück
          21 B4 00      ld hl, 180D       in die Ausgangsrichtung drehen
          CD DREHE      call DREHE        "
          21 MERKER     ld hl, MERKER     hl mit Merkeradresse laden
          34            inc (hl)          Merker um eins erhöhen und
          C9            ret               eine Ebene höher
```

Z 80 - Vollausbau bis ZEAT

```
Baum    : 21 MERKER     ld hl, MERKER       hl mit Merkeradresse laden
          35            dec (hl)            Merker um eins erniedrigen
          3A MERKER     ld a,(MERKER)       a mit Merker laden
          FE 00         cp 00H              mit 0 vergleichen
          20 0B         jr nz,+8            falls nicht, dann Baum
          CD AESTCHEN   call AESTCHEN       sonst Aestchen
          21 MERKER     ld hl, MERKER       hl mit Merkeradresse laden
          34            inc (hl)            Merker um eins erhöhen und
          C9            ret                 eine Ebene höher
          C5            push bc
          E1            pop hl              bc nach hl
          CD SCHREITE   call SCHREITE       Stamm malen
          21 1E 00      ld hl, 30D          um 30° drehen
          CD DREHE      call DREHE          "
          C5            push bc             bc retten, da abgestiegen wird
          CD NEULANG    call NEULANG        neue Länge
          CD BAUM       call BAUM           Selbstaufruf = Abstieg
          21 MERKER     ld hl, MERKER       hl mit Merkeradresse laden
          35            dec (hl)            Merker um eins erniedrigen
          21 BA FF      ld hl, -50D         um 50° nach rechts drehen
          CD DREHE      call DREHE          "
          C1            pop bc              bc wieder holen
          C5            push bc             und eine Kopie retten
          CD NEULANG    call NEULANG        neue Länge
          CD BAUM       call BAUM           Selbstaufruf = Abstieg
          21 C8 00      ld hl, 200D         um 200° drehen
          CD DREHE      call DREHE          "
          E1            pop hl              Länge wieder holen
          CD SCHREITE   call SCHREITE       zurückgehen
          21 B4 00      ld hl, 180D         um 180° in Ausgangslage
          CD DREHE      call DREHE          drehen
          21 MERKER     ld hl, MERKER       hl mit Merkeradresse laden
          34            inc (hl)            Merker um eins erhöhen
          C9            ret                 Ende Baum
```

Vor dem Aufruf von Baum muß MERKER mit einem Wert versehen werden (freie Wahl) Ebenso können die Winkel 50° und 200° geändert werden, es muß nur die Differnz gleich 150° sein.

Gomoku: Endlich das Listing!

Spieleprogramm für Z80-Systeme – 1. Preis im *LOOP*-Wettbewerb.

Hardware: Z80-System mit Grundprogramm, CPU SBC2, CPU Z80, SBC3 oder CPU 64180 (kommt bald).

Hier ist es endlich, das versprochene GOMOKU-Listing!

Die nun abgedruckte Version läuft auf der SBC2, natürlich aber auch auf allen anderen Z80-Baugruppen. Das Grundprogramm muß ab Adresse 0 (Null) verfügbar sein, da Gomoku direkt auf Grundprogrammroutinen zugreift.

Das Spiel selbst wurde bereits in *LOOP* 3 vorgestellt. Es ist ein Brettspiel aus Japan, das am Bildschirm auf einem 16 mal 16 großen Spielfeld gegen den NDR-Computer gespielt wird. Ziel des Spieles ist es, fünf Steine seiner Farbe in eine Reihe zu bringen. Achtung: Der NDR-Computer spielt wirklich gut . . .

Zum Listing:

Die entsprechenden Grundprogramm-Einsprünge sind zu Beginn als EQU-Anweisungen deklariert. Bei einer anderen Lage des Grundprogrammes hier eventuell ändern. Gomoku muß aber in jedem Fall auf der gleichen Bank wie das Grundprogramm liegen!

Danach sind die Variablen, die das Programm benötigt, definiert. Man kann diese erst zuweisen und tut sich dann mit dem Eintippen etwas leichter.

Für Einsteiger: Der linke Block des Listings zeigt die ADRESSE, auf der die Bytes, die im nächsten Block ausgedruckt sind, eingegeben werden müssen. Beispiel: In die Speicherzelle mit der Adresse 8800 muß 21H (Hexadezimal) eingegeben werden, in 8801 "BFH", in 8802 "8DH", in 8803 (diese Adresse erscheint im Listing wieder) "11H" und so weiter. „H" steht für „Hexadezimal".

Rechts steht dann der sogenannte „Quelltext", danach, mit ";" getrennt, ein Kommentar.

Sobald Sie einen Assembler, z.B. mit ZEAT oder CP/M 2.2/M80 zur Verfügung haben, können Sie direkt den Quelltext eingeben – der Assembler übersetzt dann das Programm in Maschinensprache.

Für ZEAT-Anwender: Eine ZEAT-Version des Gomoku-Programmes ist in unserer Datenbank, Tel.: 0831/ 69330 abrufbereit. Sie können ja unter ZEAT direkt von der Modem-Betriebsart in Ihren ZEAT-Texteditor laden.

Für Kunden, die das Gomoku-Listing schon bei uns gekauft haben: Senden Sie uns eine Kopie Ihrer Rechnung, Sie erhalten Ihr Geld zurück! Versprochen ist versprochen . . .

Und nun das Listing! Wir wünschen Ihnen viel Spaß beim Abtippen!

```
GOMOKU                                  PAGE  1


                TITLE GOMOKU
;***************
;***  Gomoku  ***
;***************
;
; -  Gomoku wird nicht wie vorgesehen
;    mit RUN oder GOMOKU gestartet,
;    sondern mit Adresse 8800H.
; -  Gomoku greift auf Routinen im
;    Grundprogramm zurück, kann also
;    nur mit Grundprogramm-EPROM
;    gespielt werden.
;
CLR     EQU     0030H
SCHREIT EQU     0003H
DREHE   EQU     0006H
HEBE    EQU     0009H
SENKE   EQU     000CH
SCHLEIF EQU     000FH
ENDSCHL EQU     0012H
SET     EQU     0015H
MOVETO  EQU     001BH
DRAWTO  EQU     001BH
WRITE   EQU     001EH
CI      EQU     0024H
CSTS    EQU     0027H
WAIT    EQU     003BH
CLPG    EQU     0033H
;
;
;
; Variable
MODE    EQU     87D7H           ;Merker für Betriebsart
                                ;01=S, 10=W
ZW      EQU     87C6H           ;Zwischenspeicher
ST2     EQU     87C5H           ;Register für Spielebene
M1      EQU     87C3H           ;Register, Merker
M0      EQU     87C2H           ;Register, Merker
KOR     EQU     87C0H           ;Register für Position
P1      EQU     87BEH           ;Register für Wert
P2      EQU     87BCH           ;Register für Wert
M2      EQU     87BBH           ;Register, Merker
KOR1    EQU     87B9H           ;Register
X       EQU     87B7H           ;X Cursor
Y       EQU     87B5H           ;Y Cursor
PHI     EQU     87B3H           ;Cursor-Winkel
ZAE1    EQU     87B2H           ;Zähler für Schritte
ZAE2    EQU     879CH           ;Spielstand
AB      EQU     87F0H           ;Zwischenspeicher
AC      EQU     87F2H           ;Zwischenspeicher
AD      EQU     87F4H           ;Zwischenspeicher
;
;
        ORG     0B800h
GOMOKU:
RUN:
8800  21 BF 8D           LD      HL,COR          ;Die Speicher werden
8803  11 1F 87           LD      DE,871FH        ;vorbelegt
8806  01 B9 00           LD      BC,00B9H
```

```
8809  ED B0          LDIR
880B  21 4C 86       LD    HL,864CH
880E  22 B9 87       LD    (KOR1),HL
8811  21 3C 86       LD    HL,863CH
8814  22 C0 87       LD    (KOR),HL
8817  3E 00          LD    A,00H
8819  32 BB 87       LD    (M2),A
```



```
881C  32 BC 87       LD    (P2),A
881F  32 D7 87       LD    (MODE),A
8822  32 C3 87       LD    (M1),A
8825  32 C2 87       LD    (M0),A
8828  32 C6 87       LD    (ZW),A
882B  32 C5 87       LD    (ST2),A
882E  32 B3 87       LD    (PHI),A
8831  C3 17 8D       JP    RUN1
                     ;
                     SET1:
8834  D5             PUSH  DE            ;setzen des
8835  CB 1A          RR    D             ;Steins
8837  CB 1B          RR    E
8839  CD 18 00       CALL  MOVETO
883C  D1             POP   DE
883D  CD F3 04       CALL  04F3H
8840  22 59 80       LD    (8059H),HL
8843  EB             EX    DE,HL
8844  CD F3 04       CALL  04F3H
8847  22 5B 80       LD    (805BH),HL
884A  69             LD    L,C
884B  60             LD    H,B
884C  CD 80 02       CALL  0280H
884F  22 5D 80       LD    (805DH),HL
8852  AF             XOR   A
8853  32 58 80       LD    (8058H),A
8856  3E 00          LD    A,00H
8858  32 48 80       LD    (8048H),A
885B  C9             RET
                     Feld:
885C  21 30 00       LD    HL,CLR        ;Bewegen
885F  11 D0 01       LD    DE,01D0H      ;im
8862  01 00 00       LD    BC,0000H      ;Feld
8865  CD 34 88       CALL  SET1
8868  21 10 00       LD    HL,0010H
886B  CD 0F 00       CALL  SCHLEIF
886E  21 10 00       LD    HL,0010H
8871  CD 0F 00       CALL  SCHLEIF
8874  21 04 00       LD    HL,0004H
8877  CD 0F 00       CALL  SCHLEIF
887A  21 1A 00       LD    HL,001AH
887D  CD 03 00       CALL  SCHREIT
8880  21 5A 00       LD    HL,005AH
8883  CD 06 00       CALL  DREHE
8886  CD 12 00       CALL  ENDSCHL
8889  21 1A 00       LD    HL,001AH
888C  CD 03 00       CALL  SCHREIT
888F  CD 12 00       CALL  ENDSCHL
8892  21 60 FE       LD    HL,0FE60H
8895  CD 03 00       CALL  SCHREIT
8898  21 A6 FF       LD    HL,0FFA6H
889B  CD 06 00       CALL  DREHE
889E  CD 09 00       CALL  HEBE
88A1  21 1A 00       LD    HL,001AH
88A4  CD 03 00       CALL  SCHREIT
88A7  CD 0C 00       CALL  SENKE
88AA  21 5A 00       LD    HL,005AH
88AD  CD 06 00       CALL  DREHE
88B0  CD 12 00       CALL  ENDSCHL
88B3  C9             RET
                     ;
                     AUSWS:
88B4  21 00 86       LD    HL,8600H      ;Feld zeichnen
```



```
88B7  22 F4 87       LD    (AD),HL
88BA  21 35 00       LD    HL,0035H
88BD  22 F0 87       LD    (AB),HL
88C0  21 EB 00       LD    HL,00EBH
88C3  22 F2 87       LD    (AC),HL
88C6  3E 42          LD    A,042H
88C8  CD 43 02       CALL  0243H
88CB  21 10 00       LD    HL,0010H
88CE  CD 0F 00       CALL  SCHLEIF
88D1  21 10 00       LD    HL,0010H
88D4  CD 0F 00       CALL  SCHLEIF
88D7  2A F4 87       LD    HL,(AD)
88DA  7E             LD    A,(HL)
88DB  23             INC   HL
88DC  22 F4 87       LD    (AD),HL
88DF  FE 01          CP    01H
88E1  C2 0A 89       JP    NZ,L1
88E4  2A F2 87       LD    HL,(AC)
88E7  29             ADD   HL,HL
88E8  EB             EX    DE,HL
88E9  2A F0 87       LD    HL,(AB)
88EC  01 00 00       LD    BC,0000H
88EF  CD 34 88       CALL  SET1
88F2  21 04 00       LD    HL,0004H
88F5  CD 0F 00       CALL  SCHLEIF
88F8  21 0F 00       LD    HL,SCHLEIF
88FB  CD 03 00       CALL  SCHREIT
88FE  21 5A 00       LD    HL,005AH
8901  CD 06 00       CALL  DREHE
8904  CD 12 00       CALL  ENDSCHL
8907  C3 20 89       JP    L2
                     L1:
890A  FE 10          CP    10H
890C  C2 20 89       JP    NZ,L2
890F  2A F0 87       LD    HL,(AB)
8912  ED 5B F2 87     LD    DE,(AC)
8916  CD 18 00       CALL  MOVETO
8919  3E 0B          LD    A,0BH
891B  CD 3B 00       CALL  WAIT
891E  D3 70          OUT   (70H),A
                     L2:
8920  2A F0 87       LD    HL,(AB)
8923  11 1A 00       LD    DE,001AH
8926  19             ADD   HL,DE
8927  22 F0 87       LD    (AB),HL
892A  CD 12 00       CALL  ENDSCHL
892D  2A F2 87       LD    HL,(AC)
8930  11 F3 FF       LD    DE,0FFF3H
8933  19             ADD   HL,DE
8934  22 F2 87       LD    (AC),HL
8937  21 35 00       LD    HL,0035H
893A  22 F0 87       LD    (AB),HL
893D  CD 12 00       CALL  ENDSCHL
8940  C9             RET
                     ;
                     ;
                     CHECK:
8941  E5             PUSH  HL
8942  F5             PUSH  AF            ;Register retten
8943  E1             POP   HL
8944  CB 65          BIT   4,L           ;Halbbit?
8946  28 01          JR    Z,L3          ;wenn Null dann L3
8948  37             SCF                 ;setze Carry-Flag
```



```
                     L3:
8949  E1             POP   HL
894A  C9             RET
                     ;
                     POS0:
894B  E5             PUSH  HL            ;Register
894C  D5             PUSH  DE            ;retten
894D  C5             PUSH  BC
894E  06 04          LD    B,04H         ;4 mal
8950  DD 56 00       LD    D,(IX+00H)    ;get Wert für Richtung
8953  DD 23          INC   IX
8955  DD 5E 00       LD    E,(IX+00H)
8958  DD 23          INC   IX
895A  7D             LD    A,L
895B  92             SUB   D             ;subtrahieren
895C  CD 41 89       CALL  CHECK         ;an den Rand gekommen
895F  38 11          JR    C,L4
8961  83             ADD   A,E           ;addieren
8962  CD 41 89       CALL  CHECK         ;an den Rand gekommen
8965  38 0B          JR    C,L4
8967  6F             LD    L,A           ;in HL Position von
                                         ;von neuen Platz
8968  7E             LD    A,(HL)
8969  B7             OR    A
896A  20 0A          JR    NZ,L5         ;JR if not zero
896C  3E 10          LD    A,010H        ;leerer Platz
896E  06 01          LD    B,01H         ;Ende
8970  18 0D          JR    L6
8972  3E 00          LD    A,00H         ;0 Punkt
8974  18 F8          JR    L7
8976  4F             LD    C,A
8977  3A D7 87       LD    A,(MODE)
897A  B9             CP    C
897B  20 F5          JR    NZ,L4         ;JR wenn Stein von
                                         ;Gegner
897D  3E 01          LD    A,01H         ;eigener Stein
897F  4F             LD    C,A
8980  3A C6 87       LD    A,(ZW)
8983  81             ADD   A,C           ;Summieren
8984  32 C6 87       LD    (ZW),A
8987  10 D1          DJNZ  L50           ;4 mal
8989  C1             POP   BC
898A  D1             POP   DE            ;get Register
898B  E1             POP   HL
898C  C9             RET
                     ;
                     POS1:
898D  DD E5          PUSH  IX            ;Register retten
898F  C5             PUSH  BC
8990  11 00 00       LD    DE,0000H
8993  06 04          LD    B,04H
8995  DD 21 67 8E    LD    IX,TAB        ;Set Anfang Tab.
8999  3E 00          LD    A,00H
899B  32 C6 87       LD    (ZW),A        ;Clear ZW
899E  CD 4B 89       CALL  POS0          ;1. Richtung
```

Z 80 - Vollausbau bis ZEAT

```
89A1  CD 4B 89              CALL    POS0        ;Richtung 180 Grad
89A4  3A C6 87              LD      A,(ZW)
89A7  E6 0F                 AND     0FH
89A9  FE 04                 CP      04H         ;4 Steine?
89AB  38 1A                 JR      C,L8
89AD  3A C5 87              LD      A,(ST2)
89B0  B7                    OR      A
89B1  20 0F                 JR      NZ,L9       ;JR wenn nicht Null

GOMOKU                                          PAGE  5

89B3  FD 75 00              LD      (IY+00H),L  ;Set Position
89B6  FD 23                 INC     IY
89B8  3A D7 87              LD      A,(MODE)    ;Set Mode
89BB  FD 77 00              LD      (IY+00H),A
89BE  FD 23                 INC     IY
89C0  18 4A                 JR      L10         ;ret
89C2  11 FF FF        L9:   LD      DE,0FFFFH   ;größter Wert
89C5  18 45                 JR      L10
89C7  3A C6 87        L8:   LD      A,(ZW)
89CA  E6 F0                 AND     0F0H
89CC  28 3C                 JR      Z,L11
89CE  3A C6 87              LD      A,(ZW)
89D1  E6 0F                 AND     0FH
89D3  FE 03                 CP      03H         ;3 in einer Reihe
89D5  20 0F                 JR      NZ,L12
89D7  7A                    LD      A,D
89D8  C6 20                 ADD     A,20H
89DA  57                    LD      D,A
89DB  3A C6 87              LD      A,(ZW)
89DE  CB 6F                 BIT     5,A
89E0  28 2B                 JR      Z,L11
89E2  CB E2                 SET     4,D
89E4  18 24                 JR      L11
89E6  FE 02           L12:  CP      02H         ;2 in einer Reihe
89E8  20 0F                 JR      NZ,L13
89EA  7A                    LD      A,D
89EB  C6 02                 ADD     A,02H
89ED  57                    LD      D,A
89EE  3A C6 87              LD      A,(ZW)
89F1  CB 6F                 BIT     5,A
89F3  28 15                 JR      Z,L11
89F5  CB C2                 SET     0,D
89F7  18 11                 JR      L11
89F9  FE 01           L13:  CP      01H         ;Einer?
89FB  20 0D                 JR      NZ,L11
89FD  7B                    LD      A,E
89FE  C6 20                 ADD     A,20H
8A00  5F                    LD      E,A
8A01  3A C6 87              LD      A,(ZW)
8A04  CB 6F                 BIT     5,A
8A06  28 02                 JR      Z,L11
8A08  CB E3                 SET     4,E
8A0A  10 BD           L11:  DJNZ    L14
8A0C  C1              L10:  POP     BC
8A0D  DD E1                 POP     IX
8A0F  C9                    RET
                      ;
                      POS2:
8A10  FD 7D                 DB      0FDH,07DH
8A12  32 BB 87              LD      (M2),A
8A15  3E 00                 LD      A,00H
8A17  32 C2 87              LD      (M0),A
8A1A  00                    NOP
8A1B  00                    NOP
8A1C  00                    NOP
8A1D  23              L15:  INC     HL          ;neue Position
8A1E  7C                    LD      A,H
8A1F  FE 87                 CP      087H
8A21  C8                    RET     Z           ;ret wenn alle
                                                ;Positionen gecheckt
8A22  7E                    LD      A,(HL)      ;get Platz
8A23  B7                    OR      A
8A24  20 F7                 JR      NZ,L15      ;JR wenn nicht Null

GOMOKU                                          PAGE  6

8A26  CD 8D 89              CALL    POS1        ;Wert berechnen
8A29  7A                    LD      A,D
8A2A  B3                    OR      E
8A2B  28 F0                 JR      Z,L15       ;JR if Wert Null
8A2D  FD 7D                 DB      0FDH,07DH
8A2F  00                    NOP
8A30  4F                    LD      C,A
8A31  3A BB 87              LD      A,(M2)
8A34  B9                    CP      C
8A35  20 E6                 JR      NZ,L15      ;Wenn 4 in einer Reihe
8A37  E5                    PUSH    HL          ;nur noch alle Positionen
8A38  2A BE 87              LD      HL,(P1)     ;auf neue 4. Reihe checken
8A3B  AF                    XOR     A
8A3C  ED 52                 SBC     HL,DE
8A3E  E1                    POP     HL
8A3F  30 19                 JR      NC,L16      ;JR wenn neuer Wert kleiner
8A41  ED 53 BE 87           LD      (P1),DE     ;P1 ist neuer Wert
8A45  3E 01                 LD      A,01H
8A47  32 C2 87              LD      (M0),A      ;größerer Wert wurde gefunden
8A4A  3E 00                 LD      A,00H
8A4C  32 C3 87              LD      (M1),A      ;clear, weil 2. Ebene noch
                                                ;nicht gecheckt
8A4F  3A C5 87              LD      A,(ST2)
8A52  B7                    OR      A
8A53  20 C8                 JR      NZ,L15
8A55  22 C0 87              LD      (KOR),HL    ;Position nur abspeichern
                                                ;wenn Stufe = 0.
8A58  18 C3                 JR      L15
8A5A  37              L16:  SCF
8A5B  C8                    RET     Z           ;Ret wenn beide Werte
                                                ;gleich und in zweiter
                                                ;Ebene geprüft werden muß.
8A5C  18 BF                 JR      L15
                      ;
                      CH1:                      ;für zweite Ebene
8A5E  E5                    PUSH    HL          ;rette akt. Position
8A5F  3A D7 87              LD      A,(MODE)
8A62  77                    LD      (HL),A      ;Position besetzen
8A63  3E 01                 LD      A,01H
8A65  32 C5 87              LD      (ST2),A     ;zweite Ebene
8A68  2A BE 87              LD      HL,(P1)
8A6B  E5                    PUSH    HL          ;rette P1
8A6C  2A BC 87              LD      HL,(P2)
8A6F  22 BE 87              LD      (P1),HL
8A72  21 FF 85              LD      HL,085FFH   ;set Anfang
8A75  CD 10 8A        L17:  CALL    POS2
8A78  38 FB                 JR      C,L17       ;JR wenn gleicher Wert
                                                ;gefunden
8A7A  E1                    POP     HL
8A7B  22 BE 87              LD      (P1),HL     ;get old P1
8A7E  E1                    POP     HL          ;clear
8A7F  3E 00                 LD      A,00H       ;reset Position
8A81  77                    LD      (HL),A
8A82  C9                    RET
                      ;
                      POS3:
8A83  3E 00                 LD      A,00H
8A85  32 C3 87              LD      (M1),A
8A88  21 FF 85              LD      HL,085FFH
8A8B  CD 10 8A        L19:  CALL    POS2
8A8E  D0                    RET     NC          ;Ret wenn alle Positionen
                                                ;gecheckt
8A8F  3A C3 87              LD      A,(M1)

GOMOKU                                          PAGE  7

8A92  B7                    OR      A
8A93  20 13                 JR      NZ,L18      ;JR wenn Wert schon mal
                                                ;getestet
8A95  3E 01                 LD      A,01H       ;jetzt getestet
8A97  32 C3 87              LD      (M1),A
8A9A  E5                    PUSH    HL
8A9B  21 00 00              LD      HL,0000H
8A9E  22 BC 87              LD      (P2),HL     ;P2 ist Null
8AA1  2A C0 87              LD      HL,(KOR)    ;für Position besetzen
8AA4  CD 5E 8A              CALL    CH1
8AA7  E1                    POP     HL
8AA8  CD 5E 8A        L18:  CALL    CH1         ;Test 2. Wert
8AAB  3E 00                 LD      A,00H
8AAD  32 C5 87              LD      (ST2),A
8AB0  3A C2 87              LD      A,(M0)
8AB3  B7                    OR      A
8AB4  28 D5                 JR      Z,L19
8AB6  22 C0 87              LD      (KOR),HL    ;wenn zweiter Wert größer
8AB9  18 D0                 JR      L19         ;bis alle Pos. gecheckt
                                                ;erster Wert zählt
                      ;
                      CHWS:                     ;bei Carry ok
8ABB  21 01 00              LD      HL,0001H
8ABE  22 BE 87              LD      (P1),HL
8AC1  2A B9 87              LD      HL,(KOR1)
8AC4  23                    INC     HL
8AC5  22 C0 87              LD      (KOR),HL    ;definiere Koordinaten
                                                ;für ersten Zug
8AC8  FD E5                 PUSH    IY
8ACA  CD B3 8A              CALL    POS3        ;für Computer
8ACD  FD 7D                 DB      0FDH,07DH   ;LD A,Y Befehl nicht definiert
                                                ;aber möglich
8ACF  C1                    POP     BC
8AD0  B9                    CP      C
8AD1  C0                    RET     NZ          ;wenn 4 in einer Reihe
                                                ;gewonnen
8AD2  2A C0 87              LD      HL,(KOR)
8AD5  E5                    PUSH    HL
8AD6  3A D7 87              LD      A,(MODE)
8AD9  77                    LD      (HL),A      ;eigene Position setzen
                                                ;damit Vergleich korrekt
8ADA  0F                    RRCA                ;01 = 10 und 10 = 01
8ADB  0F                    RRCA
8ADC  0F                    RRCA
8ADD  0F                    RRCA
8ADE  32 D7 87              LD      (MODE),A    ;Mode vom Gegner
8AE1  2A BE 87              LD      HL,(P1)
8AE4  E5                    PUSH    HL
8AE5  FD E5                 PUSH    IY
8AE7  CD B3 8A              CALL    POS3        ;für Gegner
8AEA  DD E1                 POP     IX
8AEC  E1                    POP     HL
```

Z 80 - Vollausbau bis ZEAT

```
8AED  C1                  POP   BC
8AEE  3E 00               LD    A,00H
8AF0  02                  LD    (BC),A        ;Position wieder löschen
8AF1  3A D7 87            LD    A,(MODE)
8AF4  0F                  RRCA
8AF5  0F                  RRCA
8AF6  0F                  RRCA
8AF7  0F                  RRCA
8AF8  32 D7 87            LD    (MODE),A      ;alter Mode
8AFB  FD 7D               DB    0FDH,07DH
8AFD  DD 5D               DB    0DDH,05DH
```



```
8AFF  BB                  CP    E
8B00  C0                  RET   NZ            ;wenn 4 in einer Reihe
8B01  ED 5B BE 87         LD    DE,(P1)
8B05  CB 62               BIT   4,D           ;Dreierfolge?
8B07  20 13               JR    NZ,L20
8B09  CB 64               BIT   4,H           ;Dreierfolge vom Computer?
8B0B  20 0B               JR    NZ,L21
8B0D  CB 42               BIT   0,D           ;Zweierfolge beim Gegner?
8B0F  28 07               JR    Z,L21
8B11  7A                  LD    A,D
8B12  E6 0F               AND   0FH
8B14  FE 04               CP    04H
8B16  30 04               JR    NC,L20
8B18  ED 43 C0 87   L21:  LD    (KOR),BC      ;ok Position vom Computer
8B1C  37            L20:  SCF
8B1D  C9                  RET
                    ;
                    ZUGGEN:                   ;wenn Carry Spiel zu Ende
                                              ;in Mode steht der Gewinner
8B1E  CD BB 8A            CALL  CHWS
8B21  FD 7D               DB    0FDH,07DH
8B23  B7                  OR    A
8B24  C8                  RET   Z             ;Ret wenn kein Wert da
8B25  DD 21 00 87         LD    IX,8700H      ;Anfang der Werte
8B29  DD 7D         L24:  DB    0DDH,7DH
8B2B  FD 45               DB    0FDH,045H
8B2D  B8                  CP    B
8B2E  C8                  RET   Z
8B2F  DD 6E 00            LD    L,(IX+00H)    ;get Wert Position
8B32  DD 23               INC   IX
8B34  DD 46 00            LD    B,(IX+00H)    ;get Wert (Mode)
8B37  DD 23               INC   IX
8B39  26 86               LD    H,86H
8B3B  7E                  LD    A,(HL)
8B3C  B7                  OR    A
8B3D  28 20               JR    Z,L22         ;wenn Platz = 0 dann L22
8B3F  B8                  CP    B
8B40  20 06               JR    NZ,L23        ;JR wenn Platz = Mode
8B42  78                  LD    A,B
8B43  32 D7 87            LD    (MODE),A
8B46  37                  SCF                 ;Gegner gewonnen
8B47  C9                  RET
8B48  FD 2B         L23:  DEC   IY
8B4A  FD 2B               DEC   IY
8B4C  DD E5               PUSH  IX
8B4E  E1                  POP   HL
8B4F  00                  NOP
8B50  DD 2B               DEC   IX
8B52  DD 2B               DEC   IX
8B54  DD 54               DB    0DDH,054H
8B56  DD 5D               DB    0DDH,05DH
8B58  01 15 00            LD    BC,SET
8B5B  ED B0               LDIR                ;lösche Wert
8B5D  18 CA               JR    L24           ;bis alle Werte gecheckt
8B5F  3A D7 87      L22:  LD    A,(MODE)
8B62  B8                  CP    B
8B63  28 05               JR    Z,L25
8B65  22 C0 87            LD    (KOR),HL      ;def. Kor
8B68  18 BF               JR    L24           ;bis alle Werte gecheckt
8B6A  77            L25:  LD    (HL),A
8B6B  37                  SCF                 ;Computer hat gewonnen
8B6C  C9                  RET
                    ; wenn Carry: Spiel zu Ende, d.h. 5 in einer Reihe
```



```
                    ; in Kor Position für den errechneten Zug
                    ; wenn Spiel zu Ende steht die Steinfarbe in Mode
                    ; (weiß oder schwarz) des Gewinners
                    ;
                    KOREIN:
8B6D  3E 00               LD    A,00H
8B6F  32 48 B0            LD    (B048H),A
8B72  18 0E               JR    L26           ;am Anfang kein Flip
8B74  2A B7 87      L35:  LD    HL,(X)
8B77  ED 5B B5 87         LD    DE,(Y)
8B7B  ED 4B B3 87         LD    BC,(PHI)
8B7F  CD 15 00            CALL  SET           ;mit Flip
8B82  CD 24 00      L26:  CALL  CI            ;Bewegung nach oben
8B85  FE 77               CP    77H
8B87  20 10               JR    NZ,L27
8B89  3A B9 87            LD    A,(KOR1)
8B8C  D6 10               SUB   10H
8B8E  32 B9 87            LD    (KOR1),A
8B91  21 5A 00            LD    HL,005AH      ;90 Grad
8B94  22 B3 87            LD    (PHI),HL
8B97  3E 00               LD    A,00H
8B99  FE 79         L27:  CP    79H           ;Bewegung nach unten?
8B9B  20 10               JR    NZ,L28
8B9D  3A B9 87            LD    A,(KOR1)
8BA0  C6 10               ADD   A,10H
8BA2  32 B9 87            LD    (KOR1),A
8BA5  21 A6 FF            LD    HL,0FFA6H     ;-90 Grad
8BA8  22 B3 87            LD    (PHI),HL
8BAB  3E 00               LD    A,00H
8BAD  FE 73         L28:  CP    73H           ;Bewegung nach rechts?
8BAF  20 10               JR    NZ,L29
8BB1  3A B9 87            LD    A,(KOR1)
8BB4  C6 01               ADD   A,01H
8BB6  32 B9 87            LD    (KOR1),A
8BB9  21 00 00            LD    HL,0000H      ;0 Grad
8BBC  22 B3 87            LD    (PHI),HL
8BBF  3E 00               LD    A,00H
8BC1  FE 61         L29:  CP    61H           ;Bewegung nach links?
8BC3  20 10               JR    NZ,L30
8BC5  3A B9 87            LD    A,(KOR1)
8BC8  D6 01               SUB   01H
8BCA  32 B9 87            LD    (KOR1),A
8BCD  21 B4 00            LD    HL,00B4H      ;180 Grad
8BD0  22 B3 87            LD    (PHI),HL
8BD3  3E 00               LD    A,00H
8BD5  F5            L30:  PUSH  AF
8BD6  21 DD 01            LD    HL,01DDH
8BD9  22 B5 87            LD    (Y),HL
8BDC  21 3D 00            LD    HL,003DH
8BDF  22 B7 87            LD    (X),HL
8BE2  3A B9 87            LD    A,(KOR1)
8BE5  E6 0F               AND   0FH
8BE7  28 14               JR    Z,L31         ;wenn Null sonst Error
8BE9  00                  NOP
8BEA  26 00               LD    H,00H
8BEC  6F                  LD    L,A
8BED  CD 0F 00            CALL  SCHLEIF
8BF0  11 1A 00            LD    DE,001AH
8BF3  2A B7 87            LD    HL,(X)        ;bestimmt x
8BF6  19                  ADD   HL,DE
8BF7  22 B7 87            LD    (X),HL
8BFA  CD 12 00            CALL  ENDSCHL
8BFD  3A B9 87      L31:  LD    A,(KOR1)
```



```
8C00  CB 3F               SRL   A
8C02  CB 3F               SRL   A
8C04  CB 3F               SRL   A
8C06  CB 3F               SRL   A
8C08  28 13               JR    Z,L32         ;wenn Null sonst Error
8C0A  26 00               LD    H,00H
8C0C  6F                  LD    L,A
8C0D  CD 0F 00            CALL  SCHLEIF
8C10  11 E6 FF            LD    DE,0FFE6H
8C13  2A B5 87            LD    HL,(Y)        ;bestimmt y
8C16  19                  ADD   HL,DE
8C17  22 B5 87            LD    (Y),HL
8C1A  CD 12 00            CALL  ENDSCHL
8C1D  F1            L32:  POP   AF
8C1E  F5                  PUSH  AF
8C1F  FE 66               CP    66H           ;Flip?
8C21  20 10               JR    NZ,L33
8C23  3E 00               LD    A,00H
8C25  CD 3B 00            CALL  WAIT
8C28  D3 60               OUT   (060H),A
8C2A  DB 68         L34:  IN    A,(68H)
8C2C  CD 27 00            CALL  CSTS
8C2F  FE FF               CP    0FFH
8C31  20 F7               JR    NZ,L34
8C33  DB 68         L33:  IN    A,(68H)
8C35  F1                  POP   AF
8C36  FE 0D               CP    0DH           ;Carriage Return? Ret?
8C38  C2 74 8B            JP    NZ,L35
8C3B  2A B9 87            LD    HL,(KOR1)
8C3E  7E                  LD    A,(HL)
8C3F  B7                  OR    A
8C40  C2 74 8B            JP    NZ,L35
8C43  3A D7 87            LD    A,(MODE)
8C46  0F                  RRCA
8C47  0F                  RRCA
8C48  0F                  RRCA
8C49  0F                  RRCA
8C4A  77                  LD    (HL),A
8C4B  3E 00               LD    A,00H
8C4D  CD 3B 00            CALL  WAIT
8C50  D3 60               OUT   (060H),A
8C52  CD B4 8B            CALL  AUSWS
8C55  C9                  RET
                    ; mit w,y,a,s steuert man den Cursor
                    ; das Bild flimmert dann
                    ;
                    ZAEHLER:                  ;zählt Schritte
8C56  21 B2 87            LD    HL,ZAE
8C59  7E                  LD    A,(HL)
8C5A  3C                  INC   A             ;+1
```

```
8C5B  27                  DAA             ;Dezimalzahl
8C5C  77                  LD    (HL),A
8C5D  3E 30               LD    A,30H
8C5F  ED 6F               RLD
8C61  32 AD 87            LD    (87ADH),A   ;Zehner
8C64  ED 6F               RLD
8C66  32 AE 87            LD    (87AEH),A   ;Einer
8C69  ED 6F               RLD             ;ok
8C6B  21 9E 87            LD    HL,0879EH
8C6E  CD 1E 00            CALL  WRITE       ;Write
8C71  C9                  RET
                  ;
                  PLAY:                     ;Spiel gegen Computer
```



```
8C72  21 F3 00            LD    HL,00F3H
8C75  22 B7 87            LD    (X),HL      ;definiere x
8C78  21 27 01            LD    HL,0127H
8C7B  22 B5 87            LD    (Y),HL      ;definiere y
8C7E  21 77 86            LD    HL,8677H
8C81  22 B9 87            LD    (KOR1),HL   ;definiere Anfang
8C84  21 00 86            LD    HL,8600H
8C87  11 01 86            LD    DE,8601H
8C8A  01 00 01            LD    BC,0100H
8C8D  3E 00               LD    A,00H
8C8F  32 00 86            LD    (8600H),A
8C92  32 B2 87            LD    (ZAE1),A    ;Reset Zähler
8C95  ED B0               LDIR              ;Clear Feld
8C97  CD 5C 88            CALL  FELD        ;Print Feld
8C9A  FD 21 00 87         LD    IY,8700H
8C9E  3A D7 87            LD    A,(MODE)
8CA1  FE 10               CP    10H         ;weiß fängt an
8CA3  28 06               JR    Z,L36
8CA5  CD 6D 8B      L35:  CALL  KOREIN
8CA8  CD 56 8C            CALL  ZAEHLER
8CAB  21 10 00      L36:  LD    HL,0010H
8CAE  11 28 00            LD    DE,0028H
8CB1  CD 18 00            CALL  MOVETO      ;Position Zeichen
8CB4  3E 0B               LD    A,0BH
8CB6  CD 3B 00            CALL  WAIT
8CB9  D3 70               OUT   (70H),A     ;Zeichen markieren
8CBB  CD 1E 8B            CALL  ZUGGEN
8CBE  F5                  PUSH  AF
8CBF  CD 21 02            CALL  0221H       ;clear Mode
8CC2  21 10 00            LD    HL,0010H
8CC5  11 28 00            LD    DE,0028H
8CC8  CD 18 00            CALL  MOVETO
8CCB  3E 0B               LD    A,0BH
8CCD  CD 3B 00            CALL  WAIT
8CD0  D3 70               OUT   (70H),A     ;Zeichen löschen
8CD2  CD 1D 02            CALL  021DH
8CD5  F1                  POP   AF
8CD6  F5                  PUSH  AF
8CD7  38 07               JR    C,L37
8CD9  2A C0 87            LD    HL,(KOR)
8CDC  3A D7 87            LD    A,(MODE)
8CDF  77                  LD    (HL),A      ;set Position
8CE0  CD B4 8B      L37:  CALL  AUSWS       ;Print
8CE3  F1                  POP   AF
8CE4  D2 A5 8C            JP    NC,L38      ;Carry?
8CE7  C9                  RET               ;wenn dann Ende
                  ; Weiß fängt an. Wenn der Computer rechnet erscheint
                  ; links unten ein weißer Punkt. Ist der Computer fer-
                  ; tig wird der Punkt gelöscht. Wird eine der Tasten
                  ; a,s,w oder y gdrückt so erscheint ein Pfeil. Drückt
                  ; man dann "CR" so wird auf die Position in der sich
                  ; der Pfeil befindet ein Stein gesetzt.
                  ;
                  ;
                  STATUS:                   ;zählt Spielstand
8CE8  B7                  OR    A
8CE9  28 16               JR    Z,L39       ;JR wenn Punkt für
                                            ;Computer
8CEB  21 9D 87            LD    HL,879DH
8CEE  7E                  LD    A,(HL)
8CEF  3C                  INC   A           ;+1
8CF0  27                  DAA
8CF1  77                  LD    (HL),A
```



```
8CF2  3E 30               LD    A,30H
8CF4  ED 6F               RLD
8CF6  32 98 87            LD    (8798H),A   ;Zehner
8CF9  ED 6F               RLD
8CFB  32 99 87            LD    (8799H),A   ;Einer
8CFE  ED 6F               RLD             ;ok
8D00  C9                  RET
                  ;
8D01  21 9C 87      L39:  LD    HL,ZAE2
8D04  7E                  LD    A,(HL)
8D05  3C                  INC   A           ;+1
8D06  27                  DAA
8D07  77                  LD    (HL),A
8D08  3E 30               LD    A,30H
8D0A  ED 6F               RLD
8D0C  32 95 87            LD    (8795H),A   ;Zehner
8D0F  ED 6F               RLD
8D11  32 96 87            LD    (8796H),A   ;Einer
8D14  ED 6F               RLD             ;ok
8D16  C9                  RET
                  ;
                  RUN1:
8D17  21 BF 8D            LD    HL,COR
8D1A  CD 1E 00            CALL  WRITE
8D1D  21 00 30            LD    HL,3000H
8D20  CD 0F 00            CALL  SCHLEIF
8D23  CD 12 00            CALL  ENDSCHL
8D26  21 99 99            LD    HL,9999H
8D29  22 9C 87            LD    (ZAE2),HL
8D2C  3E 01               LD    A,01H
8D2E  CD E8 8C            CALL  STATUS
8D31  3E 00               LD    A,00H
8D33  CD E8 8C            CALL  STATUS
8D36  CD 33 00      L43:  CALL  CLPG
8D39  CD 1D 02            CALL  021DH
8D3C  21 74 00            LD    HL,0074H
8D3F  22 80 87            LD    (8780H),HL
8D42  21 B4 00            LD    HL,00B4H
8D45  22 7E 87            LD    (877EH),HL
8D48  21 64 00            LD    HL,0064H
8D4B  22 7C 87            LD    (877CH),HL
8D4E  21 7C 87            LD    HL,877CH
8D51  CD 1E 00            CALL  WRITE       ;Print Gomoku
8D54  21 F8 8D            LD    HL,ANW
8D57  CD 1E 00            CALL  WRITE       ;Print Anweisung
8D5A  3E 01               LD    A,01H
8D5C  32 D7 87            LD    (MODE),A
8D5F  CD 24 00            CALL  CI
8D62  FE 73               CP    073H        ;schwarz oder weiß?
8D64  20 05               JR    NZ,L40
8D66  3E 10               LD    A,10H
8D68  32 D7 87            LD    (MODE),A
8D6B  21 2B 00      L40:  LD    HL,002BH
8D6E  22 7C 87            LD    (877CH),HL
8D71  21 3A 00            LD    HL,003AH
8D74  22 7E 87            LD    (877EH),HL  ;definiere Pos + Form
8D77  21 53 08            LD    HL,0853H
8D7A  22 80 87            LD    (8780H),HL
8D7D  CD 33 00            CALL  CLPG        ;Erstaal Seite löschen
8D80  CD 1D 02            CALL  021DH
8D83  21 7C 87            LD    HL,877CH
8D86  CD 1E 00            CALL  WRITE
8D89  21 8B 87            LD    HL,878BH
```



```
8D8C  CD 1E 00            CALL  WRITE       ;Print Spielstand
8D8F  3A D7 87            LD    A,(MODE)
8D92  F5                  PUSH  AF
8D93  CD 72 8C            CALL  PLAY
8D96  F1                  POP   AF
8D97  47                  LD    B,A
8D98  3A D7 87            LD    A,(MODE)
8D9B  90                  SUB   B
8D9C  F5                  PUSH  AF
8D9D  CD E8 8C            CALL  STATUS
8DA0  21 8B 87            LD    HL,878BH
8DA3  CD 1E 00            CALL  WRITE
8DA6  F1                  POP   AF
8DA7  28 08               JR    Z,L41       ;gewonnen oder verloren?
8DA9  21 DD 8D            LD    HL,WIN
8DAC  CD 1E 00            CALL  WRITE
8DAF  18 06               JR    L42
8DB1  21 EB 8D      L41:  LD    HL,LOSE
8DB4  CD 1E 00            CALL  WRITE
8DB7  CD 24 00      L42:  CALL  CI
8DBA  FE 6E               CP    6EH         ;Nochmal?
8DBC  CA 36 8D            JP    Z,L43
8DBF  C9                  RET               ;Ende
                  ;
                  ORG   RUN+05BFH
                  ;
                  COR:
8DBF  46 00 78 00         DW    046H,78H
8DC3  33 00 28 43 29 20 46 72   DB  33H,00H,'(C) Frederik Siegmund',00h
                  ;
                  ORG   COR+01EH
                  WIN:
8DDD  90 01 10 00         DW    190H,10H
8DE1  21 00 57 49 4E 00   DB    21H,00H,'WIN',00H
                  ;
                  ORG   COR+02CH
                  ;
                  LOSE:
8DEB  90 01 10 00         DW    0190H,10H
8DEF  21 00 4C 4F 53 45 00      DB  21H,00H,'LOSE',00H
                  ;
                  ;
                  ORG   COR+039H
                  ANW:
8DF8  32 00 64 00         DW    32H,64H
8DFC  22 00 53 3D 73 63 68 77   DB  22H,00H,'S=schwarz W=weiss N=nochmal',00H
                  ;
                  ORG   COR+050H
```

```
                        GOMOKU1:
8E1C  2B 00 3A 00               DW      2BH,3AH
8E20  53 08 47 6F 6D 6F 6B 75   DB      53H,08H,'Gomoku',00H
                        ;
                                ORG     COR+06CH
                        STAND:
8E2B  00 01 10 00               DW      100H,10H
8E2F  21 00 43 2F 53 3A 30 30   DB      21H,00H,'C/S:00601',00H
                        ;
                                ORG     COR+07FH
                        OUT1:
8E3E  10 00 10 00               DW      10H,10H
8E42  21 00 53 63 68 72 69 74   DB      21H,00H,'Schritte:17',00h
                        ;
                                ORG     COR+0ABH

GOMOKU                                                  PAGE 14


                        TAB:
8E67  00 01 01 00 00 11 11 00   DB      00H,01H,01H,00H,00H,11H,11H,00H,00H,10H,10H,00H
8E73  10 01 01 10               DB      10H,01H,01H,10H

GOMOKU                                                  PAGE 15


0030 CLR                        0003 SCHREIT
0006 DREHE                      0009 HEBE
000C SENKE                      000F SCHLEIF
0012 ENDSCHL                    0015 SET
0018 MOVETO                     001B DRAWTO
001E WRITE                      0024 CI
0027 CSTS                       0038 WAIT
0033 CLPB                       87D7 MODE
87C6 ZW                         87C5 ST2
87C3 M1                         87C2 M0
87C0 KOR                        87BE P1
87BC P2                         87BB M2
87B9 KOR1                       87B7 X
87B5 Y                          87B3 PHI
87B2 ZAE1                       879C ZAE2
87F0 AB                         87F2 AC
87F4 AD                         8800 GOMOKU
8800 RUN                        8834 SET1
885C FELD                       8884 AUSWS
890A L1                         8920 L2
8941 CHECK                      8949 L3
894B POS0                       895A L50
896E L7                         8972 L4
8976 L5                         897F L6
898D POS1                       8999 L14
89C2 L9                         89C7 L8
89E6 L12                        89F9 L13
8A0A L11                        8A0C L10
8A10 POS2                       8A1D L15
8A5A L16                        8A5E CH1
8A75 L17                        8A83 POS3
8A8B L19                        8A8B L18
8ABB CHWS                       8B18 L21
8B1C L20                        8B1E ZUGGEN
8B29 L24                        8B48 L23
8B5F L22                        8B6A L25
8B6D KOREIN                     8B74 L35
8B82 L26                        8B99 L27
8BAD L28                        8BC1 L29
8BD5 L30                        8BFD L31
8C1D L32                        8C2A L34
8C33 L33                        8C56 ZAEHLER
8C72 PLAY                       8CA5 L38
8CAB L36                        8CE0 L37
8CE8 STATUS                     8D01 L39
8D17 RUN1                       8D36 L43
8D6B L40                        8DB1 L41
8DB7 L42                        8DBF COR
8DDD WIN                        8DEB LOSE
8DF8 ANW                        8E1C GOMOKU1
8E28 STAND                      8E3E OUT1
8E67 TAB

no fatal error(s)

SUM=8715

GOMOKU                                                  PAGE 16

CRC=0C15
```

Z 80 – CP/M 2.2

Z80 – CP/M2.2 – deutsche Umlaute für CP/M

Will man die deutschen Umlaute (ö, ä, ü ...) unter CP/M verwenden, gibt es eine schnelle Möglichkeit:

Im Flomon-Menue nach dem Einschalten gibt man die Tasten "CTRL C" "ESC" "Z1" "CTRL C" ein

Mit "CTRL C" gelangt man in den Testmodus, mit "ESC" "Z1" (nacheinander) wird auf deutschen Zeichensatz umgeschaltet und mit einem weiteren "CTR C" (gleichzeitig) wird auf Flomon-Menue zurückgesprungen.

Anmerkung: Nach jedem Reset oder durch die Eingabe wie oben jedoch "ESC" "z" "0" ist wieder der amerikanische Zeichensatz eingeschaltet.

Nähere Erläuterungen finden Sie in der *LOOP* 4, Seite 9 und im mc-Sonderheft „Schritt für Schritt 2", Seite 88.

PASCAL und BASIC

Tips und Tricks bei HEBAS, Nr. 4

von Dr. Hans Hehl

Es liegen einige Leseranfragen vor, die bei HEBAS.COM dies und jenes geändert haben möchten. Hier sind nun die Verbesserungen, die gewünscht wurden. Außerdem soll die Verwendung von Dateien mit wahlfreiem Zugriff erläutert werden. Vielleicht gibt es Anwender, die selbst Verbesserungen durchgeführt haben. Wie wäre es mit einer kurzen Mitteilung in *LOOP*?

1. Der Befehl DATE$ bei HEBAS.COM unterbricht innerhalb eines Programmes die Programmausführung und ergibt die Meldung „Fertig". Um einen Programmabbruch zu vermeiden, müssen folgende Bytes geändert werden (Bild 1). Dies kann wie üblich mit einem Debugger oder mit einem Diskketteneditor geschehen. Nun wird das beim Kaltstart eingegebene Datum mit dem DATE$-Befehl ausgegeben und die nächste BASIC-Zeile (nicht nach gosub) abgearbeitet (siehe Musterprogramm Bild 1).

2. Beim Befehl LIST kann die Ausgabe des Programms mit den Tasten (CTRL-S) und (CRTL-Q) angehalten bzw. fortgesetzt werden. Wem das zu viel Fingerakrobatik bedeutet, kann dies auf die Leertaste umstellen. Erstmaliges Drükken stoppt die Ausgabe, nochmaliges

```
Adresse      altes Byte      neues Byte

370E         C3              E1
370F         F8              E1
3710         01              C3
3711         00              2E
3712         0A              03

3727         A0              8A


10 PRINT "TESTPROGRAMM"
20 PRINT DATE$ : PRINT
30 PRINT "TESTENDE"
```

Bild 1: BYTE-Tabelle und Musterprogramm

Drücken setzt die Ausgabe fort. Die zu ändernden Bytes sind in Bild 2 angegeben. Auch bei RUN gilt dies.

```
Adresse      altes Byte      neues Byte

3F8          13              20
403          11              20
```

Bild 2: Bytetabelle für Leertastenfunktion bei LIST und RUN

3. Im Handbuch ist der Abschnitt „wahlfrei schreiben " und „wahlfrei lesen" etwas kurz weggekommen. Hier nun eine ausführlichere Beschreibung zu dem Beispiel einer sog. RANDOM-Datei (Bild 3), die auch auf der HEBAS-Diskette enthalten ist (Record 4.BAS).

Was ist das Besondere daran? Bei sequentiellen Dateien sind alle Daten hintereinander abgelegt, z.B. Mitgliedsdaten eines Vereins. Möchte man einzelne Daten verändern, sollten alle Daten im Kernspeicher zur Verfügung stehen. Bei einer Datei mit wahlfreiem Zugriff können z.B. nur der 2. und dann gleich der 200. Datensatz bestehen und auch bearbeitet werden, ohne daß dazwischen Datensätze vorhanden sein oder gelesen werden müssen. Man spricht von virtuellen Dateigrößen, da nur die tatsächlich vorhandenen Daten Speicherplatz auf der Diskette beanspruchen, also bei unserem Beispiel nur 2 Datensätze. Bei kleineren Diskettenkapazitäten gibt es jedoch dann erhebliche Probleme, wenn die virtuellen Dateien zu groß werden. Bei dem Beispielprogramm (Bild 3) wird daher die Datei in voller Größe angelegt.

Eine wahlfreie Datei enthält Datensätze mit fest vorgegebener Satzlänge. Ein Datensatz aus einem Adressenprogramm kann z.B. so aufgebaut sein:

Anrede (max. 10 Zeichen), Name und Vorname (max. 40 Zeichen), Straße (max. 30 Zeichen), Postleitzahl und Wohnort (max. 50 Zeichen).

Ein Datensatz (auch Record genannt) enthält dann immer 130 Zeichen. Für jede Adresse gibt es einen Datensatz. Die Satzlänge kann nachträglich nur durch Übertragen der Daten in eine größere Datei abgeändert werden. Auch bekannte Datenbanksysteme wie DBASE arbeiten

```
10  REM Record-Beispiel  HEBAS CP/M Vers. 2.2
20  ' copyright Dr.Hehl
30  CLEAR 5000:A$ = CHR$(64)                    ' 5000 Byte für String
40  GOSUB 440                                    ' Bildschirm frei
50  PRINT"Anlegen einer Datei = 1":PRINT
60  PRINT"Daten schreiben     = 2":PRINT
70  PRINT"Daten lesen         = 3":PRINT
80  PRINT"Ende                = 0":PRINT
90  B = VAL(BYTE$(&0)):RL=0                      ' Tastaturabfrage
100 :   ON B GOTO 130,210,330                    ' Verteiler
110 CLOSE:END
120 :
130 ' alle Records anlegen und auffüllen
140 GOSUB 460:OPEN$20,"R",NA$,L1                 ' L1 =RL+4
150 FOR RN = 1 TO RZ:PRINT RN;
160 :   S$ =STR$(RN)+STRING$(A$,RL)
170 :   S1$=FIX$(MID$(S$,2,LEN(S$)),RL)          ' Länge konstant
180 :   PRINT$205RN,S1$
190 NEXT RN:CLOSE$20:GOTO40
200 :
210 ' Record schreiben
220 GOSUB 440:PRINT"Record schreiben":PRINT
230 :   IF RL=0 THEN GOSUB 460                   ' Recorddaten
240 OPEN$20,"R",NA$,L1
250 :   INPUT"Record-Nr.    0=Ende";RN
260 :   IF RN = 0 THEN 310
270 :   IF RN>RZ THEN 250                    ' Recordnummer begrenzt
280 :   INPUT"Eingabe:";E$
290 :   IF LEN(E$)) RL THEN 280                   ' Eingabe begrenzt
300 :   PRINT$205RN,E$:PRINT:GOTO 250
310 CLOSE$20:GOTO 40
320 :
330 ' Record lesen
340 GOSUB 440:PRINT"Record lesen":PRINT
350 :   IF RL=0 THEN GOSUB 460                   ' Recorddaten
360 OPEN$20,"R",NA$,L1
370 :   INPUT"Record-Nr.   0=Ende";RN:GOSUB 440
380 :   IF RN>RZ THEN GOSUB 440:PRINT"kein Record":GOTO 370
390 :   IF RN=0 THEN 420
400 :   INPUT$205RN,E$: IF E$="" THEN PRINT"Keine Daten"
410 :   PRINTE$:PRINT:GOTO 370
420 CLOSE$20:GOTO 40
430 :
440 PRINT CHR$(27);CHR$(69):RETURN               ' Bildschirm frei
450 :
460 ' Recorddaten
470 INPUT"Dateiname ohne Dateityp";NA$:PRINT
480 INPUT"Laufwerk";LW$:PRINT
490 NA$=LW$ + ":" +NA$ +".rec"
500 :   IF LOOKUP(NA$) THEN 570
510 IF B=3 THEN GOSUB 440:PRINT"Datei nicht vorhanden":PRINT:GOTO 50
520 INPUT"Recordanzahl";RZ:PRINT
530 INPUT"Recordlänge";RL:L1=RL+4
540 OPEN$20,"R",NA$,13:D$=FIX$(STR$(RZ),5)+FIX$(STR$(L1),4)
550 PRINT$205O,D$:CLOSE$20:RETURN
570 :   IF B=1 THEN GOSUB 440:PRINT"Datei vorhanden":PRINT:GOTO 50
580 OPEN$20,"R",NA$,13:INPUT$205O,D$
590 :   RZ = VAL(LEFT$(D$,5)):L1=VAL(RIGHT$(D$,3)):RL=L1-4
600 :   PRINT"Recordzahl:";RZ;" Recordlänge:";RL:PRINT
610 CLOSE$20:RETURN
620 END
```

Bild 3: Beispielprogramm für wahlfreien Zugriff

nach diesem Prinzip. Jede Teilinformation, z.B. Straße, beginnt immer an der gleichen Stelle im Datensatz. Eine andere Möglichkeit ist die Verwendung eines Markierungszeichens nach jeder Teilinformation, das dann allerdings mit dem INSTRING-Befehl gesucht werden muß.

Mit dem Beispielprogramm als Kernstück kann man z.B. ein Adressenprogramm aufbauen. Vom Autor wird es bei einem großen Vereinsprogramm mit 500 Mitgliedern verwendet. Eine sortierte sequentielle Datei enthält nur die Namen mit einer Schlüsselnummer, die die Satznummer der einzelnen Random-Dateien Adresse, Bankverbindung, Mitgliedsdaten etc. enthält. Auch die Auswertung der Bundesjugendspiele kann mit einer RANDOM-Datei, die die Punkte enthält, durchgeführt werden.

Programmbeschreibung:

Das Programm ist in 5 Abschnitte gegliedert.

1. Abschnitt: Zeile 10 – 110 ✶Menue✶
Nach Speicherreservierung und Bildschirmlöschung wird ein Menueplan angeboten.

2. Abschnitt: Zeile 130 – 190 ✶Record anlegen✶
Nach Eingabe und Überprüfung der Recorddaten im Abschnitt Nr. 5 wird die Datei vollständig aufgefüllt, wobei die Recordnummer mit dem Zeichen CHR$(64) = „Klammeraffe" auf die eingegebene Recordlänge ergänzt wird. Somit werden bei einem Lesezugriff keine sinnlosen Daten gelesen. Die Recorddaten werden zusätzlich auf der Diskette im Record Nr. 0 abgespeichert. Der Drucker gibt an Stelle des „Klammeraffen" ein Paragraphenzeichen (§) aus.

3. Abschnitt: Zeile 210 – 310 ✶Record schreiben✶
Nur bei Programmstart (RL=0) werden die Recorddaten im Abschnitt Nr. 5 entweder von der Diskette gelesen oder eingegeben. Die eingegebenen Daten werden vor dem Schreibvorgang überprüft. Hier könnte man auch einen Lesezugriff einfügen, um bei vorhandenen Dateien nicht Daten durch versehentliches Überschreiben zu vernichten.

4. Abschnitt: Zeile 330 – 420 ✶Record lesen✶

Dieser Abschnitt ist dem 3. Abschnitt ähnlich aufgebaut, nur erfolgt jetzt ein Lesezugriff. Ist die Datei anfangs mit dem Programmabschnitt Nr. 2 angelegt worden, so werden jetzt definierte Daten gelesen.

5. Abschnitt: Zeile 460 – 620 * Recorddaten *
Dies ist eigentlich der wichtigste Abschnitt. Zunächst werden Dateiname und Laufwerk erfragt. Mit dem Befehl „Look-up(Dateiname)" wird überprüft, ob die Datei existiert. Wenn nicht, wird bei einem Lesezugriff (b=3) ins Menue zurückgesprungen bzw. bei einem Schreibzugriff werden erst die Recorddaten ermittelt.

Recordanzahl und Recordlänge richten sich nach der Diskettenkapazität. Bei 141 KByte sind z.B. über 9000 Records mit 10 Zeichen Stringlänge oder auch 1300 Records mit 100 Zeichen möglich, wobei die Stringlänge maximal 250 Zeichen betragen kann.

Die Records werden mit den Bytes 22h, 0Dh, 0Ah und 22h miteinander verbunden. Daher muß die eingegebene Länge der Records um 4 erhöht werden (Zeile 530: L1=RL+4). Recordanzahl (RZ) und Recordlänge (L1) werden als String (D$) mit konstanter Länge (9 Zeichen) abgespeichert. Ist die Datei vorhanden, so darf sie nicht aus Versehen mit Menueteil Nr. 1 durch Neuanlegen vernichtet werden; es erfolgt ein Rücksprung ins Menue (b=1).

Nun kann Record Nr. 0 gelesen werden. Aus dem String (D$) werden Recordanzahl und Recordlänge ermittelt, danach erfolgt der Rücksprung in den jeweiligen Programmabschnitt.

HEBAS-Grundlagen

Der Autor des HEBAS-Handbuches bietet eine 60seitige Beschreibung der grundlegenden Maschinensprache-Routinen von HEBAS als Textdatei HEBASMA.DOC auf Diskette an (siehe u.a. Musterseite). In der Beschreibung enthalten sind die BDOS- und BIOS-Sytemroutinen, die wichtigen Unterprogramme wie Ein-und Ausgabe von Zeichen, Verarbeitung der eingegebenen BASIC-Befehle, Befehlsschlüsseltabellen und Einsprungadressen, Ausgabe von Fehlermeldungen, Zahlen- und Variablenverarbeitung, Kalt- und Warmstart, Aufbau wichtiger Direktbefehle etc. zum Preis von DM 40,— inkl. Mehrwertsteuer, Diskette (5,25 oder 8 Zoll) und Versandkosten.

Weiterhin sind für den Anfänger eine Debugging-Anleitung zur Analyse des Interpreters und eine kleine Programmbibliothek mit grundlegenden BASIC-Routinen für Dateiverarbeitung dabei. Ab Herbst 86 wird dann der Quellcode auf Diskette mit ausführlichem Kommentar als Ergänzung der Beschreibung zur Verfügung stehen; bei Bestellung an u.a. Adresse bitte das Diskettenformat angeben.

Folgende Formate stehen z.Zt. zur Verfügung: 5,25-Zoll ECMA 70 (40 Spuren), 8-Zoll IBM (einfache Dichte sowie doppelte Dichte: 8*1024, skew = 0), 5,25-Zoll NDR-Computer (80 Spuren), zweiseitig). Das Format bitte angeben und den Absender nicht vergessen!

Bestellung bitte an folgende Adresse:

Dr. H. Hehl, Lindenstraße 20,
8059 Wartenberg

Es gibt folgende Zahlungsweisen:
a) Beilage von 40,— DM
b) Beilage eines Verrechnungsschecks
c) Einzugsverfahren bei Angabe von Konto-Nr., Bankname und Bankleitzahl

Nach Gutschrift des Betrages erfolgt die Zusendung der Diskette.

D) Musterseite aus der Textdatei: HEBASMA.COC

Maschinenprogramme der BASIC-Befehle

RESET

Der Befehl ist in der Tokens-Tabelle ab 1BFD enthalten und besitzt den Schlüssel 80 8F. In der Sprungtabelle steht ab 1CF die Adresse 3606h der Unterroutine. Es werden die Standard-BDOS-Funktionen von CP/M2.2 benützt, nämlich die Funktionen Nr. 25 (Aktuelles Laufwerk melden), Nr. 13. (Rücksetzen des Disk-Systems) und Nr. 14 (Laufwerk selektieren). Verwendet wird der BDOS-Einsprung (siehe dort) bei Adresse 3770h. Die aktuelle Laufwerksnummer ist in Adresse 0004h enthalten.

```
Einzelschritte:

3606    3E 19       LD A,19         BDOS-Funktion Nr. 25, aktuelles
                                    Laufwerk melden
3608    CD 3770     CALL 3770       BDOS-Aufruf
360B    F5          PUSH AF         Aktuelles Laufwerk auf Stapel
360C    3E 0D       LD  A,13        BDOS-Funktion Nr.  13,
                                    Rücksetzen  des Disk-Systems
360E    CD 3770     CALL 3770       BDOS-Aufruf

3611    F1          POP AF          aktuelle Laufwerks-Nr. holen
3612    4F          LD C,A          lade Register C mit diesem Wert
3613    3E 0E       LD A,0E         BDOS-Funktion Nr. 14,
                                    Laufwerk selektieren
3615    C3 3770     JP 3770         BDOS-Aufruf, dann zurück
                                    nach 032E

****    ****                    ****

032E    3A 0FD0     LD A,(FD0)
0331    3C          INC A           Akku-Wert + 1, Z = 0
0332    C4 3756     CALL NZ,3756 BIOS-Einsprung bei EA06
                                    (Konsolenstatus)

0335    von da ab üblicher BASIC-Eintritt über 0414h mit Aus
        gabe der Meldung "Fertig" (wie beim STOP-Befehl).
```

„Drehender Würfel"

von Willi Beer, 8271 Grassau

Mit diesem Schreiben erhalten Sie ein BASIC-Programm, das ich auf dem NDR-Klein-Computer mit SBC-2-Platine erstellt habe. Das Programm zeichnet einen Würfel, der sich schrittweise um die durch Y1, X1 und Y4, X8 gehende Raumdiagonale dreht.

Der grundsätzliche Aufbau des Programms entspricht dem in *LOOP* 4 (1. Jahrgang) auf Seite 6 vorgeschlagenen Grafik-Programm. Wichtig ist die in den Zeilen 330 und 340 eingefügte Warteschleife, weil sonst das Bild zwischen den relativ langen Löschvorgängen nicht recht

```
100 CLRS
110 POKE HEX("87C5"),0
120 Y1=50
130 Y2=100
140 Y3=150
150 Y4=200
160 X1=255
170 X8=255
180 FOR I=1 TO 360
190 PAGE 1,0
200 OUT 112,1
210 GOSUB 1000
220 WAIT 112,2
230 X6=X1-INT(120*SIN(I*3.1415/180))
240 X5=X1-INT(120*SIN((I+120)*3.1415/180))
250 X7=X1-INT(120*SIN((I+240)*3.1415/180))
260 X2=X1-INT(120*SIN((I+60)*3.1415/180))
270 X4=X1-INT(120*SIN((I+300)*3.1415/180))
280 X3=X1-INT(120*SIN((I+180)*3.1415/180))
290 OUT 112,0
300 GOSUB 1000
310 PAGE 0,1
320 WAIT 112,2
```

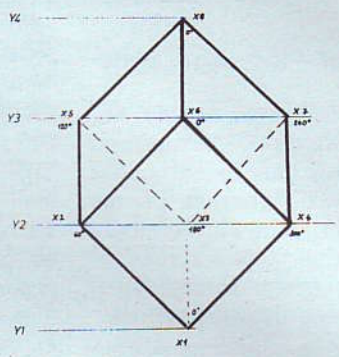
sichtbar wird. N kann jedoch nach Geschmack verringert werden, um die Drehzahl zu erhöhen, bis das Bild undeutlich wird.

Es läßt sich auch die Schrittgröße in Zeile 180 z.B. wie folgt vergrößern: 180 FOR I=1 T/ 360 STEP 2.

Und wer Spaß daran hat, kann versuchen, dieses Programm so zu erweitern, daß unsichtbare Kanten gestrichelt gezeichnet werden. Es können auch mehrere Bildschirmseiten verwendet werden.

Bei der Gestaltung dreidimensionaler, bewegter Figuren läßt sich also schon mit der SBC-2-Platine ein Hauch von CAD (Computer Aided Design) erleben.

```
330 FOR N=1 TO 750      :REM Warteschleife, da sonst Bild schlecht
340 NEXT N              :REM sichtbar, oder teilweise unsichtbar
350 NEXT I
360 END
1000 MOVETO X1,Y1
1010 DRAWTO X2,Y2
1020 DRAWTO X5,Y3
1030 DRAWTO X8,Y4
1040 DRAWTO X7,Y3
1050 DRAWTO X4,Y2
1060 DRAWTO X1,Y1
1070 DRAWTO X3,Y2
1080 DRAWTO X5,Y3
1090 MOVETO X7,Y3
1100 DRAWTO X3,Y2
1110 MOVETO X2,Y2
1120 DRAWTO X6,Y3
1130 DRAWTO X4,Y2
1140 MOVETO X6,Y3
1150 DRAWTO X8,Y4
1160 RETURN
```



FÜR 68000-EINSTEIGER

Mehr Software zum NDR-Computer

Folkert Hallenga, Hufelandstr. 16
3000 Hannover 01, Tel.: (0511) 2133239

Hallo Mitbastler und -programmierer!

Seit etwa einem Jahr bin ich stolzer Besitzer des NKC. Gerade in Hinsicht auf mein späteres Berufsziel (Berufschullehrer) eignet sich dieser Computer wegen seiner Modularität und Durchschaubarkeit ganz besonders für mich (solange weiterhin alle Hintergrundinformationen wie Systemlistings, Systemschnittstellen und Schaltpläne mit veröffentlicht werden). Diese Gründe sind es auch, die meiner Meinung nach diesen Computer gegenüber anderen Fertigcomputern auszeichnet und bei nicht weiter steigenden Preisen konkurrenzfähig bleiben läßt. Leider leidet diese Konkurrenzfähigkeit einerseits unter dem Preisgefälle (Preiserhöhungen beim NKC und Preisnachlässe bei Fertigcomputeranbietern) und andererseits unter der nur arg schwachen Softwareunterstützung. Auf den ersten Teil der Kritik sollten eventuell einmal die Lizenznehmer des Systems antworten, aber zum zweiten Teil ist nicht nur der Anbieter aufgefordert, sondern auch die zahlreichen User des Systems. Die Offenlegung aller Schaltpläne und Listings ist doch geradezu eine Aufforderung. Es ist etwas schade, daß viele gute verwirklichte Ideen bei einigen in der Schublade liegen, während andere diese zwar gut gebrauchen könnten aber sich erst mit anderen Problemen zum NKC beschäftigt haben, die ihrerseits wieder für viele User interessant sind. Es sollte meiner Meinung nach möglich sein, die Erfahrungen und Tips, kleinere Hilfsprogramme und größere Dienstprogramme, Hard- und Softwaretips allen zugänglich zu machen. Ein Forum dafür wird uns ja sogar durch die *LOOP* und die Clubzeitschrift des Braunschweiger Userclubs geboten. Das Rumbasteln macht vielmehr Spaß, wenn es jemand gibt, dem man das Erfolgserlebnis mitteilen kann, bzw. der einem beim Fehlschlag weiterhelfen kann. Auch das Übernehmen von Programmteilen für sein eigenes Programm sei hier als Möglichkeit einmal aufgeführt oder ein Leser findet eine bessere Lösung für eine veröffentlichte Problemlösung . . .

Es gibt viel zu veröffentlichen – also ran!!

Ich möchte in dieser Ausgabe mit ein paar Vorschlägen für Änderungen am Grundprogramm selbst beginnen; für die nächsten Ausgaben habe ich verschiedene Programme in GOSI, PASCAL und ASSEMBLER vorbereitet. Wer einen Eprombrenner sein eigen nennt und nicht zufrieden ist, was das Grundprogramm ihm vorsetzt, dem kann geholfen werden. Ich möchte aber zuvor ausdrücklich darauf hinweisen, daß ich die Leistung von Rolf-Dieter Klein anerkenne und nicht vorhabe, sein Copyright zu unterlaufen. Dennoch stört es mich z. B. wenn 2 Sekunden lang eine Copyrightmeldung auf dem Bildschirm steht; oder daß beim Einschalten der Editor auf 40 Zeichen eingestellt ist oder daß bei den Assembleroptionen die Bildschirmausgabe voreingestellt ist oder der Eprommer etwas arg langsam ist. Alle diese aufgeführten Punkte lassen sich ändern, wobei die Tatsache zu Hilfe kommt, daß die zu ändernden Stellen in nur einem Eprom vorzunehmen sind.

Ich möchte hier einmal einen Änderungsvorschlag unterbreiten, der folgende Grundprogrammänderung vorsieht:

Statt der 2 Sekunden Copyright-Meldung ein kurzes freundlicheres ‚Hallo' gerade noch so lange, daß man es lesen kann; die Voreinstellung beim Einschalten auf „80 Zeichen" und „nur Fehlerausgabe"; und einer erheblich schnelleren Epromprogrammierroutine, die sich an den Empfehlungen der Epromhersteller orientiert (siehe auch INTEL Datenblatt).

Für diesen schnelleren Programmieralgorithmus ist eine kleine Hardwareänderung auf der Prommerkarte nötig, die aber erst nach dem Umbrennen des Grundprogramms vorgenommen werden sollte: Der 10 uF Kondensator C1 muß gegen einen 220nF Kondensator ausgetauscht werden. Besitzer einer Prommerkarte für den Z80-Betrieb (100 ns Kondensator und 20 KOhm Trimmpoti) brauchen keinerlei Änderungen vorzunehmen. Durch diese Änderung wird nur noch ein Programmierimpuls von etwa 1 ms erzeugt.

Bei der hier vorgestellten Programmierroutine wird das Eprom mit einem kurzen Impuls „programmiert" und gleich anschließend der vorhandene Wert ausge-

lesen. Ist der gewünschte Wert noch nicht erreicht, so wird dieser Vorgang wiederholt bis der Wert tatsächlich gebrannt ist. Wenn dieser Fall nach 15 Impulsen noch nicht eingetreten ist, gilt das Eprom als fehlerhaft. Ansonsten wird der Wert zusätzlich noch mit der vierfachen der bis dahin benötigten Zeit zur Sicherheit nachgebrannt. Da $FF Werte nicht noch extra gebrannt werden brauchen, werden diese Werte beim Brennen übersprungen. Da jedes Byte gleich nach dem Brennen auf Korrektheit überprüft wird, kann beim Auftreten eines eventuellen Fehlers dieser sofort gemeldet und der Programmiervorgang abgebrochen werden. Bei meinen Messungen der Programmierdauer ermittelte ich mit diesem Algorithmus für ein 8 K Eprom eine durchschnittliche Programmierzeit von 56 Sekunden. Wenn allerdings auf die Bildschirmausgabe des Mitzählers verzichtet wird, verringert sich diese Zeit auf nur noch 38 Sekunden. Da bei diesem Tempo die Ausgabe eines Zählers sowieso nicht mehr sinnvoll ist, habe ich sie zugunsten einer höheren Programmiergeschwindigkeit durch eine „bitte warten . . .“ Meldung ersetzt. Im Fehlerfall wird allerdings die Adresse im Eprom mit ausgegeben, bei der der Fehler auftrat.

Bei dieser Gelegenheit sei auf einen häufig gemachten Fehler bei der Eingabe der Startadresse (von) und der Endadresse (bis) hingewiesen:

Wenn z. B. für die Startadresse $A000 und für die Endadresse $C000 angegeben wird, wird der Wert in $C000 wieder in das Eprom auf die Adresse $0 gebrannt (eine korrekte Eingabe hätte hier $A000 bis $BFFF sein müssen). Leider ließ sich ein Abfangen dieses Fehlers nicht auch noch in dieser Routine mit unterbringen, die ja nicht mehr Platz als die Originalroutine von Rolf-Dieter Klein im Eprom belegen darf.

Um die geschilderten Änderungen am Grundprogramm vornehmen zu können, sind nacheinander folgende drei Schritte erforderlich:

1. Kopieren des Grundprogramms in das Ram.

```
*****************************************************************
VERSCHIEB EQU $10000            * hier bitte die Adresse angeben, auf die das
                                * Grundprogramm im Ram verschoben wurde
*****************************************************************
* Voreinstellung auf 80 Zeichen Modus
ORG Verschieb+$3020-$2000
 DC.B $11                       * siehe auch WRITE Routine des Grundprogramms
*****************************************************************
* Voreinstellung auf "nur Fehlerausgabe"
ORG Verschieb+$3047-$2000
 DC.B 1                         * siehe auch Menüpunkt ASSEMBLEROPTIONEN
*****************************************************************
* Startmeldung beim RESET       ACHTUNG bei der Eingabe, da mal DC.W und mal DS.W
ORG Verschieb+$3422-$2000
 DC.W $40                       * X-Koordinate für Ausgabe
 DS.W 1
 DC.W $5b                       * Y-Koordinate für Ausgabe
 DS.W 1
 DC.W $CC                       * Schriftgröße
 DS.W 3
 DC.W $8                        * Länge der Ausgabedauer in 20ms Stufen
ORG Verschieb+$303C-$2000
 DC.B 'Hallo',0                 * Ausgabetext
 DC.L 0,0,0,0,0                 * mit Nullen auffüllen
*****************************************************************
* Schnellprogrammierroutine für den Eprommer

ORG Verschieb+$1500-$2000
PRINT4X:
ORG Verschieb+$1386-$2000
TEXTAUS:
ORG Verschieb+$2922-$2000
FINMENUE:
ORG Verschieb+$3D64-$2000
TXTP5:
ORG Verschieb+$3D76-$2000
TXTP6:
DC.B 'EPROM FEHLER  bei $',0,0

AUSBUF  EQU $70
PROMD   EQU $FFFFFF60
PROMA1  EQU $FFFFFF61
PROMA2  EQU $FFFFFF62

ORG Verschieb+$25DC-$2000
* in D6.L steht VON
* in D7.L steht BIS
* in D5.L steht NACH

PROGPROM:
 LEA    TXTWAIT(PC),A0          * Wartemeldung
 BSR    ANZEIGEN                * ausgeben
 MOVEA.L D5,A0                  * ab hier sollen die Werte programmiert werden
LOOPWRITE:
 CLR.B  D2                      * Zähler für Programmierimpulse auf 0 setzen
 MOVE.B (A0)+,D1                * zu programmierenden Wert holen
LOOPPRG:
 BSR.S  WRITEHDL                * Programmierroutine aufrufen
 ADDQ.B #1,D6                   * Zähler für Programmierimpulse um 1 hochzählen
 AND.B  #%00011111,D0           * wieder auf laser
 MOVE.B D0,PROMA2               *   stellen
 CMP.B  PROMD,D1                * ist Wert schon gebrannt?
 BEQ.S  SAVETY                  *   dann noch Sicherheitsprogrammierung
 CMPI.B #15,D2                  * sonst waren es schon 16 Programmierimpulse ?
 BNE.S  LOOPPRG                 * wenn nicht, dann noch mal probieren
 FEFIN:                         * sonst Fehler --> Abbruch
  LEA    AUSBUF(A6),A0
  MOVE.L D5,D0                  * Fehleradresse im Eprom
  BSR    PRINT4X                * hexadezimal
  LEA    AUSBUF(A5),A0
  MOVEQ  #$22,D0
  MOVE   #314,D1
  MOVEQ  #80,D2
  BSR    TEXTAUS                * ausgeben
  LEA    TXTP6(PC),A0           * "EPROM FEHLER" Meldung
  BRA.S  GIBAUS
 OKFIN:
  LEA    TXTP5(PC),A0           * "EPROM OK" Meldung
 GIBAUS:
  MOVE.B #%01000000,PROMA2      * Prommer Baugruppe neutral stellen
  BSR.S  ANZEIGEN               * Meldung ausgeben
  BRA    FINMENUE

 SAVETY:
  BSR.S  WRITEHDL               * Programmierroutine und
  BSR.S  WRITEHDL               *   doppelte
  BSR.S  WRITEHDL               *     dreifache
  BSR.S  WRITEHDL               *       vierfache Zeit nachprogrammieren
  SUBQ.B #1,D3                  * Zähler wieder um 1 runterzählen
  BNE.S  SAVETY                 * wenn noch nicht fertig, dann weitermachen
  CMP.L  A0,D7                  * gesamter Bereich programmiert?
  BMI.S  OKFIN                  * wenn ja, dann OK-Meldung
  ADDQ.L #1,D5                  * sonst mit nächstem Wert
  BRA.S  LOOPWRITE              * weitermachen

 WRITEHDL:
  MOVE.B D1,PROMD               * Wert an Prommer Baugruppe
  MOVE   D5,D0                  * Adresse im Eprom
  MOVE.B D0,PROMA1              * unteren Teil der Adresse laden
  ROR    #8,D0                  * oberen Teil der Adresse holen
  CMPI.B #$FF,D1                * ist der zu brennende Wert $FF ?
  BEQ.S  SCHONOK                * der braucht nicht noch gebrannt zu werden
  AND.B  #%00011111,D0          * nur unteren 5 Bits auswerten
  OR.B   #%10000000,D0          * ENABLE an, LED an, kein Programmierimpuls
  MOVE.B D0,PROMA2              *   an Prommer Baugruppe ausgeben
  BSET   #5,D0                  * und nun Triggerimpuls
  MOVE.B D0,PROMA2              *   setzen
  BCLR   #5,D0                  * Triggerimpuls wieder zurück-
  MOVE.B D0,PROMA2              *   setzen
 TRIGGER:
  BTST   #0,PROMA1              * abwarten, bis eingestellte Zeit
  BNE.S  TRIGGER                * das Programmierimpuls zu Ende
 SCHONOK:
  RTS

 ANZEIGEN:
  MOVEQ  #$22,D0                * Textgröße
  MOVEQ  #30,D1                 * Ausgabekoordinaten
  MOVEQ  #20,D2
  BRA    TEXTAUS                * Grundprogrammroutine beinhaltet das RTS

 TXTWAIT:
  DC.B 'bitte warten...',0      * Wartemeldung
  DS   0
  DC.W 0

END
```

Dazu wird folgendes Programm eingegeben, assembliert und im STARTENmenü mit "Start" gestartet:

```
START:
LEA     $E0000,A0       * bitte geben Sie statt $E0000 die Hexadresse
                        * an, auf der bei Ihnen das Grundprogramm liegt
LEA     $10000,A1       * wenn Sie bei $10000 keine 8 K Ram haben,
                        * setzen Sie statt der $10000 eine entsprechend
                        * andere Ramadresse (mindestens $2000) ein
ADDA.L #$2000,A0        * es interessiert nur das zweite Eprom
MOVE    #$2000-1,D0     * insgesamt 8 K
SCHLEIFE:
MOVE.B (A0)+,(A1)+      * rüberkopieren
DBRA    D0,SCHLEIFE
RTS
```

2. Abändern der entsprechenden Stellen Programmlisting (am Schluß) eingeben und dann nur! assemblieren.
3. Brennen des Eproms (2. Eprom des Grundprogramms). Zum Brennen (das letzte Mal noch so langsam) geben Sie bitte als Startadresse den gleichen Wert wie im obigen Programm für A1 an. Für die Angabe der Endadresse rechnen Sie $1FFF zur Startadresse dazu und geben diesen Wert an.

Ein Eprom, auf dem die geschilderten Grundprogrammänderungen schon vorgenommen wurden, ist auch gegen eine Vorauszahlung von DM 20,– bei mir zu erhalten (Adresse siehe oben).

Anmerkung des Herstellers:

Herr Hallenga spricht uns aus dem Herzen: Bitte beachten Sie den Artikel „Software-Partner gesucht"!

Einige Worte zur Preissituation: Die angesprochene Verteuerung bezieht sich nur auf einige Leiterplatten, die von uns in kleineren Stückzahlen aufgelegt wurden und deshalb etwas teurer wurden. Wir stehen hier immer vor der Entscheidung: Große Serien auflegen, deshalb billiger, aber Fehler bleiben über einen längeren Zeitraum drin oder: kleinere Serien auflegen, lieber etwas teurer, aber möglichst bald fehlerfrei.

Da wir (im Gegensatz zu vielen anderen Anbietern) die Leiterplatten (außer Multilayer) auch leer verkaufen, müssen wir schon in den Leiterplattenpreis zwangsläufig unsere Entwicklungskosten einkalkulieren. Man kann es drehen und wenden wie man will: Eine neue Baugruppe kostet zwischen 10.000.– und 50.000,– DM, bis sie serienreif ist. Wir haben es früher auch nicht geglaubt – aber es ist so.

Da die Innovationsgeschwindigkeit sehr hoch ist, müssen die Kosten bereits bei ca. 200 verkauften Einheiten eingespielt sein: DM 10.000.– div. DM 200.– = DM 50.– Entwicklungskosten pro Baugruppe. Incl. der Herstellkosten und des normalen Aufschlages sollte also eine Leiterplatte nicht unter DM 100.– kosten, und dies für eine „einfache" Baugruppe... Wir sind also doch noch billig!

Natürlich bleiben wir mit dem Ohr am Markt und mit den Preisen am Boden – vergleichen Sie auch unsere Anzeige: Sonderangebote – auf Seite 31.

Wilfried Oehrle
Philosophenweg 19, 7400 Tübingen

Sehr geehrte *LOOP*-Redaktion,

falls Sie das beiliegende kleine Programm für brauchbar halten, mögen Sie es gelegentlich in die *„LOOP"* aufnehmen. Ich zähle mich zu denjenigen NDR-Klein-Computerfreunden, die noch keine so guten Programme zustande bringen; trotzdem sende ich Ihnen dieses Programm, um es den anderen, falls sie es nicht schon herausbekommen haben, zu ermöglichen, von einem eigenen Programm aus (mit oder ohne eigenem Bibliotheksmenue) an eine Stelle des RDK-Grundprogramms zu springen.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* VON EINEM EIGENEN PROGRAMM KANN IN DAS BIBLIOTHEKSMENUE DES RELOKATIVEN *
* RDK-GRUNDPROGRAMMS GESPRUNGEN WERDEN.DAS VORLIEGENDE PROG.IST RELOKATIV *
* UND BERECHNET DIE EINSPRUNGADRESSE INS GRUNDPROGRAMM. WENN DER MARKE    *
* BIBO EIN ANDERER WERT ( = ADR. DES BEI $0000 BEGINNENDEN GRUNDPROGRAMMS)*
* ALS ADR. ZUGEORDNET WIRD KANN EINE ANDERE EINSPRUNGADRESSE GEWAEHLT     *
* WERDEN. AUF SEITE 211 DES HANDBUCHS IST BEI ADRESSE $0027CA DIE MARKE   *
* bibo ZU FINDEN. DORTHIN WIRD VON DIESEM PROGRAMM GESPRUNGEN.            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ORG $0
OFFSET $10000        * CODE WIRD BEI ADR $10000 ABGELEGT.VERAENDERBAR.
DC.L $55AA0180
DC.B 'IN BIBO '      * TITEL, 8 ZEICHEN LANG INCL LEERZEICHEN
DC.L BIBSTART        * STARTADRESSE. NUR ZUR INFORMATION IM MENUE.
DC.L BIBENDE-BIBSTART * LAENGE
DC.B 1               * RELOKATIV
DC.B 0,0,0
DC.L 0,0
DC.L 0
* * * * * * * * * * *
BIBSTART:            *
move #!clr,d7        * BILDSCHIRM LOESCHEN
trap #1              * DER EINSPRUNG INS RDK-GRUNDPROGRAMM
BIBO EQU $27CA       * LIEGT IM ABSTAND VON $27CA VOM
                     * BEGINN DES GRUNDPROGRAMMS AUS. SIEHE LISTING
                     * IM HANDBUCH BEI ADR $27CA.
move #$12,d0         * DIE MARKE "BIBO" ERHAELT DEN WERT $0027CA ZUGEORDNET
move #10,d1          * EIGENEN TEXT ZUSAETZLICH ZUM BIBLIOTHEKSMENUE AUSGEBEN
move #10,d2          *
lea text(pc),a0      *
move #!write,d7      *
trap #1              *
move #!GETBASIS,d7   * ADRESSE DES BEGINNS DES RDK - GRUNDPROGRAMMS IN A0
trap #1              *
LEA BIBO(A0),A1      * ADDIERE ZUM WERT VON BIBO DIE ADRESSE IN A0 UND LEGE DAS
                     * ERGEBNIS IN A1 AB.
JSR BIBO,A1          * SPRINGE ( MIT JSR SPRUENGE UEBER GESAMTEN 1  MBYTE -
                     * ADRESSBEREICH MOEGLICH ! ) ZUR MARKE BIBO, DEREN ADRESSE
                     * IM REGISTER A1 LIEGT. ( BSR WUERDE ZUM FEHLER FUEHREN.)
MOVE #!SETA5,D7      * ERKLAERUNG SIEHE HANDBUCH, BEFEHL NUMMER 91 ( REGISTER
trap #1              * A5 RESTAURIEREN, WENN ES Z.B.DURCH EIGENES PROGRAMM ODER
                     * DEN JSR - BEFEHL VERAENDERT WURDE.)
move #$32,d0         * WIEDER IM EIGENEN PROGRAMM UND DAMIT MAN'S MERKT
move #10,d1          * NOCHMAL EINE TEXTAUSGABE ZUR VERGEWISSERUNG OB ZURUECK
move #20,d2          * IM EIGENEN PROGRAMM.
lea endtext(pc),a0   *
move #!write,D7      * RELOKATIV MIT DEM TRAP - BEFEHL
trap #1              *
MOVE #!SYMCLR,D7     * SYMBOLTABELLE LOESCHEN, DA BEI ERNEUTEM ASSEMBLIEREN
TRAP #1              * FEHLER AUFTRETEN KOENNEN (ADRESSERROR)
rts                  *
* * * * * * * * * * *
text: dc.b 'von eigenem Programm ins RDK - Bibliotheksmenue',0
endtext: dc.b 'Wieder im eigenen Programm',0
DS 0                 * FALLS UNGERADE ADRESSE, DANN AUSGLEICH; NOETIG, WENN
BIBENDE: END         * DIESES PROGRAMM ALS UNTERPROGRAMM VERWENDET WIRD.
```

Rolf-Dieter Klein
Modem-Programm für den 680xx

Will man auf dem 680xx ein Modemprogramm mit dem Grundprogramm realisieren, so tritt ein Problem auf. Wenn der Bildschirm gescrollt wird, geht recht viel Zeit verloren, und somit auch Zeichen, die in dieser Zeit empfangen werden. Will man das verhindern, so gibt es zwei Möglichkeiten: Man verwendet einen Interrupt, den die SER-Baugruppe abgibt, dann aber muß man andere Dinge beachten, wenn man z. B. mit CP/M arbeitet. Die andere Möglichkeit wird hier gezeigt: man schreibt einen neuen Bildschirmteil, der auch während der Bildlaufzeit die serielle Schnittstelle abfragt. Das Scrollen wurde auch verbessert, so daß man nun auch schnell arbeiten kann.

Das Programm eignet sich für alle CPUs 680xx, beim 68020-Betrieb kann man sogar mit 9600 Baud Übertragungsrate arbeiten, und der Bildschirm ist schnell genug, um den Zeichen zu folgen. Beim 68000 sind es etwa 4800 Baud und beim 68008 ca. 2400 Baud. Also immer noch mehr als 300 Baud der Rate, die man für eine normale Modemverbindung braucht.

Im Programmteil START wird zunächst die Baudrate initialisiert. Dazu werden hier die Werte $16 und $09 geladen. Wenn man hier andere Werte einsetzt, so lassen sich andere Übertragungsraten realisieren.

Als nächstes erfolgt der Aufruf des Unterprogramms OGETBASIS, es liefert die Grundprogrammanfangsadresse. Mit einem Offset von $414 kann man dort die CPU-Größe erfahren. Damit wird dann die Portadresse der GDP multipliziert. Bei einer 8-Bit-CPU also z. B. 68008 wird der Wert 1 als Multiplikator verwendet, beim 68000 mit 16-Bit Datenbus wird der Wert 2 verwendet und beim 68020 wird der Wert 4 genommen.

Das Hauptprogramm hat nun zwei Aufgaben: Zum Einen wird ständig die Tastatur abgefragt, wenn man ein Zeichen eingibt so wird es über die serielle Schnittstelle übertragen. Zum Anderen wird die serielle Schnittstelle abgefragt. Wenn von dort ein Zeichen ankommt, wird es auf den Bildschirm ausgegeben, nachdem es in einem Puffer abgelegt wurde. Dazu werden nun neue Unterprogramme verwendet. Die Ausgabe geschieht mit dem Programmteil CRT. Im Gegensatz zum Grundprogramm wird hier in jeder Warteschleife die serielle Schnittstelle abgeprüft. Wenn ein Zeichen vorhanden ist,

wird es zunächst in einen sogenannten Ringpuffer gelegt. Damit wird erreicht, daß keine Zeichen verloren gehen, auch wenn die Bildschirmroutine mal längere Zeit beschäftigt sein sollte.

Das Ablegen im Ringpuffer geschieht mit Hilfe des Programms PUT. Das Auslesen aus dem Ringpuffer wird von der Routine GET übernommen. Die Routine CRT kennt nicht so viele Steuercodes wie CO, jedoch ausreichend viele, um über ein Modem arbeiten zu können, also z. B. Bildschirmlöschen CR, LF, Backspace, Forward.

Der Bildschirminhalt wird hier übrigens nicht um eine Zeile nach oben gescrollt, sondern gleich um 10, um Zeit zu sparen.

Die Routine XCRT entspricht der Routine, wird jedoch von dieser aufgerufen, immer wenn ein Scrollen stattfindet. Dazu werden auch alle Steuersequenzen mit gespeichert. Die Speicherung der Information für die Neuausgabe geschieht in einem weiteren Ringpuffer.

```
*********************************
* Universal Modem Programm      *
* mit intergriertem CRT-Treiber *
* fuer ueberlappenden Betrieb   *
* (C) 1986 Rolf-Dieter Klein    *
* Rev 1.0      860124           *
*********************************

org $10000              * Programmstart

xon equ $11             * Fuer Protokoll
xoff equ $13

start:                  * Start des Programms
 move #$16,d0     * 300  Baud
 move #$09,d1     * 2 Stop 8 Bit
 jsr @siinit            * SER initialisieren
*
 jsr @getbasis          * Startadresse Grundprog.
 move.l $414(a0),d0     * CPU =1,2,4
 muls #$ff70,d0         * IO-Port GDP
 move.l d0,gdpadr       * somit unabhaengig von CPU-Typ
 jsr @clr               * Bildschirm loeschen
 clr d0
 clr d1
 jsr @newpage           * Bildschirmseite lesen=0, schreiben=0
 clr d0
 clr d1
 jsr @setflip           * keine automatische Bildschirmseiten-
*                       * Umschaltung.
 bsr curon              * Cursor darstellen (Achtung, eigene Routine)
schleife:
 jsr @csts        * Tastatur abfragen, Zeichen da ?
 beq st1          * nein, dann weiter bei st1
 bsr warte        * Polling durchfuehren
 jsr @ci          * Zeichen lesen
 cmp.b #0,d0      * Abbruch ist CTRL-A, Ende MODEM
 bne.s st1a       * Sonst weiter  bei st1a
 rts              * Ende, zurueck ins System.
st1a:
 jsr @so          * Zeichen ueber die serielle Schnittstelle
*                 * ausgeben
st1:              * Ausgabe
 jsr @sists       * Status Eingabe der Schnittstelle
 beq st2          * kein Zeichen vom Modem, dann weiter bei st2
 jsr @si          * Zeichen holen sonst
 bsr put          * Ablegen in Ringbuffer fuer Anzeige vorbereiten
st2:
 bsr get          * Ist was im Ringbuffer
 beq schleife     * Kein Zeichen da, dann zurueck zur schleife
 bsr crt          * D1=Wert , Ausgabe auf dem Bildschirm
 bra schleife     * und auch zurueck dannach.
*
warte:            * Warten bis GDP bereit und Polling.
wa1:
 jsr @sists       * Zeichen da, nein, dann weiter bei wa1a
 beq.s wa1a
 jsr @si          * sonst Zeichen holen
 bsr put          * und im Ringpuffer ablegen.
wa1a:
 movea.l gdpadr,a1    * Adresse GDP holen
 btst #2,(a1)         * GDP schon fertig,
 beq.s wa1        * nein, dann warten und si pollen,
 rts              * so dass keine Zeichen verloren gehen.


crt:              * Zeichen in d1. BILDSCHIRMAUSGABE
 bsr curoff       * Cursor ausblenden. (Achtung eigene Routine)
 and.w #$7f,d1    * Bit 7 ignorieren.
*
 lea screen,a0    * Auch im Bildwdh-Speicher ablegen.
 move scrpoip,d2
 move.b d1,0(a0,d2.w)
 add #1,d2          * neuer Pointer.
 and #scrmask,d2 * Groesse Bildwdh.Speicher
 move d2,scrpoip * neuer Pointer merken.
*
 cmp.b #$8,d1     * bs  , Zeichen zurueck
 bne crt1
 sub #6,xadr      * Zeichen nach links.
 bpl crtee        * bis linker Rand erreicht.
 clr xadr
 bra crtee
*
crt1:
 cmp.b #$9,d1   * Forward, Zeichen nach rechts
 bne crt2
 add #6,xadr    * neue x-Koordinate.
 bra crtee
*
crt2:
 cmp.b #$b,d1     * Up, Zeile nach oben.
 bne crt3
 sub #1,lfcount   * abziehen. Anzahl Linefeeds.
 add #10,yadr     * neue y-Position.
 bra crtee
*
crt3:
 cmp.b #$a,d1     * LF, Zeile nach unten.
 bne crt4
 add #1,lfcount   * Distanz zaehlen, Anzahl Linefeeds
 cmp #10,lfcount * bei Zeile 10 Spezialbehandlung.
 bne crt3b        * Dort Startposition fuer Vorschub.
 move scrpoip,scrpoig  * Merken 10. Zeile  Position
 move xadr,xadrline    * Auch den Start merken x-Koordinate
crt3b:
 sub #10,yadr     * neue y-Adresse
 cmp #10,yadr     *
 bpl xcrtee
 *
 bsr clear        * Seitenvorschub durchfuehren
 move xadrline,xadr     * Wieder zurueck n.Zeile
 move #240,yadr  * neue y-Position .
 clr lfcount      * Anzahl Zeilen.
 lea screen,a3
 move scrpoig,d4
crt3c:
 cmp scrpoip,d4   * Alten Bildschirminhalt ausgeben ab Zeile 10.
 beq crt3d        * Bis Ende erreicht.
 move.b 0(a3,d4.w),d1
 add.w #1,d4
 and #scrmask,d4
 bsr xcrt         * Ausgabe aller Zeichen incl. Steuerzeichen in Loop
 bra crt3c        *
*
crt3d:
 bra crtee        * Ende.
*
crt4:
 cmp.b #$1a,d1  * CLEAR , Bildschirm loeschen.
 bne crt5
 bsr clear        * Loeschen.
 clr lfcount      * Zeile oben.
 clr xadr         * Links, x=0
 move #240,yadr * y=240.
 bra crtee
crt5:
 cmp.b #$d,d1     * CR, Zeile an linken Anfang.
 bne crt6
 clr xadr
 bra crtee
*
crt6:
 cmp.b #$20,d1
 blt crtee          * Steuerzeichen. Undef.
* Zeichenausgabe
 movem.l d1-d2,-(a7)
 bsr warte          * Warten bis GDP fertig.
 move xadr,d1
 move yadr,d2
 jsr @moveto        * Positionieren
 bsr warte
 movem.l (a7)+,d1-d2
 move.b d1,d0
 jsr @cmd           * Zeichen ausgeben.
 add #6,xadr      * Neue Position
crtee:
 bsr curon          * Cursor dann wieder an.
 rts

*
* Routine aehnlich zu crt, jedoch fuer Neuausgabe.
xcrt:    * Zeichen in d1, d4=Pointer auf screen.
 and.w #$7f,d1
 cmp.b #$8,d1  * bs
 bne xcrt1
 sub #6,xadr
 bpl xcrtee
 clr xadr
 bra xcrtee
*
xcrt1:
 cmp.b #$9,d1   * Forward
 bne xcrt2
 add #6,xadr
 bra xcrtee
*
xcrt2:
 cmp.b #$b,d1     * Up
 bne xcrt3
 sub #1,lfcount  * abziehen.
 add #10,yadr
 bra xcrtee
*
xcrt3:
 cmp.b #$a,d1     * LF
 bne xcrt4
 add #1,lfcount  * Distanz zaehlen
 cmp #10,lfcount * > dann Verschiebedistanz gefunden
 bne xcrt3b
 move d4,scrpoig       * Merken n-te Zeile
 move xadr,xadrline    * D4=Pointer lokal
xcrt3b:
 sub #10,yadr
 bra xcrtee
*
xcrt4:
 cmp.b #$1a,d1   * CLEAR
 bne xcrt5
 bsr clear
 clr lfcount
 clr xadr
 move #240,yadr
 bra xcrtee
xcrt5:
 cmp.b #$d,d1     * CR
 bne xcrt6
 clr xadr
 bra xcrtee
*
xcrt6:
 cmp.b #$20,d1
 blt xcrtee         * Steuerzeichen. Undef.
* Zeichenausgabe
 movem.l d1-d2,-(a7)
 bsr warte
 move xadr,d1
 move yadr,d2
 jsr @moveto
 bsr warte
```

```
movem.l (a7)+,d1-d2
move.b d1,d0
jsr @cmd
add #6,xadr      * Neue Position
xcrtee:
rts
*
*
curon:           * Cursor einschalten und anzeigen
movem.l d1-d2,-(a7)
bsr warte
move xadr,d1
move yadr,d2
sub #1,d2        *
jsr @moveto
bsr uline        * Untertreichungsstrich.
movem.l (a7)+,d1-d2
rts

curoff:          * Cursor ausblenden.
movem.l d1-d2,-(a7)
bsr warte
move xadr,d1
move yadr,d2
sub #1,d2
jsr @moveto
jsr @erapen
bsr uline        * Unterstreichungsstrich.
jsr @setpen
movem.l (a7)+,d1-d2
rts

uline:           * Unterstreichen.
lea datau(pc),a2
ulp:
bsr warte
move.b (a2)+,d0
beq ulfin
jsr @cmd
bra.s ulp
*
ulfin:
rts

datau:   * Kurzvektortabelle.
dc.b %11111000,%11111000,%10101000,0
*

clear:           * Bildschirm loeschen.
```



```
bsr warte        * Warten bis GDP bereit.
move #$4,d0
jsr @cmd
rts

get:             * Zeichen aus Ringbuffer holen, falls da
                 * Ergebnis in d1
lea ring,a0      * Get Pointer
move ripoig,d0
cmp ripoip,d0    * kein Zeichen da.
beq nein
move.b 0(a0,d0.w),d1
add #1,d0
and #rimask,d0   * Groesse Ringpuffer
move d0,ripoig
ja:

move #$ffff,d0   * Zeichen war da.
rts

nein:            * Nein, kein Zeichen da.
clr d0
rts

put:                * In den Ringbuffer legen
lea ring,a0
move ripoip,d1      * Put Pointer
move.b d0,0(a0,d1.w)* Zeichen ablegen
add #1,d1           * Weiterruecken
and #rimask,d1      * Buffergroesse.
move d1,ripoip
rts

rimask equ $fff   * 4K Byte  EinlesePuffer
scrmask equ $fff  * 4K Byte  ScreenPuffer

lfcount: dc.w 0  * Fuer Scroll
xadrline: dc.w 0 * Fuer Scroll
xadr: dc.w 0     * Spalte 0..79*6     In Bildschirm koor.
yadr: dc.w 240   * Zeile 240-0..23*10
gdpadr: dc.l 0
scrpoig: dc.w 0
scrpoip: dc.w 0
ripoig:  dc.w 0
ripoip:  dc.w 0


screen: ds.b 4096  * Bildschirm Ringspeicher
ring: ds.b 4096    * Dateneingabespeicher

end
```

68000 DOS ohne CP/M68k

Das Arbeiten mit JADOS

von Klaus Janßen

In der *LOOP*-Ausgabe 8/9 wurde das Diskettenbetriebssystem JADOS vorgestellt und seine wichtigsten Eigenschaften und Funktionen beschrieben. Dabei wurde ein sehr wichtiger Aspekt vergessen. JADOS ist nämlich so aufgebaut, daß es *ohne Änderungen auch mit den Prozessoren 68000 und 68020 zusammenarbeitet.* Wer also von der bewährten 68008 CPU „aufsteigt", kann sein JADOS weiterhin einsetzen, ohne ein neues kaufen zu müssen!

Im Folgenden soll am Beispiel einer Programmentwicklung gezeigt werden, wie man mit JADOS arbeitet. Zum besseren Verständnis sei zunächst die Speicherbereichsaufteilung erklärt, die im Handbuch etwas zu kurz gekommen ist.

Man muß hier zwischen zwei verschiedenen Speichermodellen unterscheiden, abhängig davon, ob man die Bankbootkarte verwendet oder nicht.

Die Abb. 1 zeigt das Speichermodell einer typischen Konfiguration mit 96 KByte RAM ohne Einsatz der Bankbootkarte:

Abb. 1: Speichermodell ohne Bankbootkarte

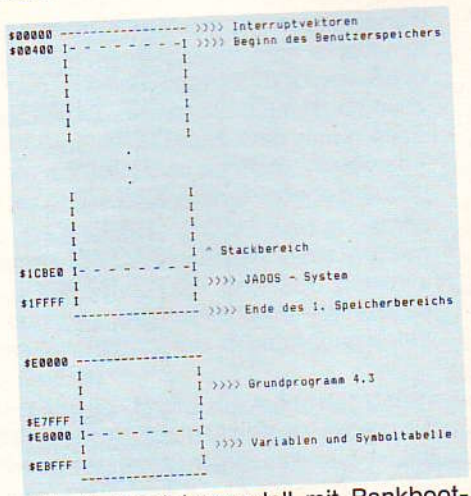


Abb. 2. Speichermodell mit Bankbootkarte

Auf Adresse $00000 bis $07FFF liegt das Grundprogramm. Danach folgen die Variablen des Grundprogramms ab $08000. Der Bereich ab der letzten Grundprogrammvariablen bis zum Beginn des JADOS-Benutzerspeichers auf Adresse $0C000 ist für Symbole reserviert. Der Bereich für die Symbole umfaßt etwa 12 KByte oder rund 720 Symbole!

Die Adresse $0C000 ist die Ladeadresse für die .68K-Dateien.

Am Ende des 1. zusammenhängenden Speicherbereichs liegt das JADOS mit seinen Variablen. Davor beginnt der Stack, welcher in Richtung niedrigerer Adressen wächst. Kollisionen zwischen dem Stack und dem Benutzerspeicher werden von JADOS rechtzeitig erkannt und gemeldet.

Die Abb. 2 zeigt das Speicherabbild einer typischen Konfiguration mit Bankbootkarte, 128 KByte RAM ab Adresse 0 und 16 KByte RAM hinter dem Grundprogramm.

Wie man sieht, sind bei dieser Konfiguration zwei Speicherbereiche interessant. Im 1. Speicherbereich liegen der JADOS-Benutzerspeicher und das JADOS-System selbst. *Die Ladeadresse für .68K-Dateien liegt hierbei auf Adresse $00400.*

Im Speicherbereich hinter dem Grundprogramm werden die Grundprogrammvariablen und die Symboltabelle verwaltet.

Wer übrigens mit den weiter oben erwähnten 720 Symbolen nicht auskommt, der kann den RAM-Speicher hinter dem Grunprogramm auf 32 KByte ausbauen und hat dann Platz für fast 1500 Symbole.

Anhand eines kleinen Programms soll nun die Arbeitsweise mit JADOS gezeigt werden. Das eingebaute Kommando „TYPE" lädt immer eine komplette Textdatei in den Speicher und gibt sie dann auf den Bildschirm aus. Wenn aber die Datei größer als der freie Benutzerspeicher ist, bricht JADOS den Ladevorgang mit der Meldung „SPEICHER VOLL" ab, und die Datei kann nicht ausgegeben werden. Das kleine Dienstprogramm, das wir nun entwickeln, umgeht die Problematik eines zu kleinen Speichers, indem es immer nur jeweils einen Sektor lädt, dann die Ausgabe macht, dann den nächsten Sektor lädt usw.

Um das Programm zu erstellen, wird zunächst „EDIT" aufgerufen. Da ein ganz neues Programm erstellt werden soll, muß der Menuepunkt "1" angewählt werden. Das Quellprogramm wird nun komplett eingegeben. Danach fordert JADOS dazu auf, dem eingetippten Programm einen Namen zu geben. Wir wählen z.B. den Namen „AUSGABE.ASM". Danach wird das Kommando „ASS" eingegeben. Da der Maschinencode für das hier vorgestellte Programm sicher nicht über 4 KByte liegen wird, können wir die Textladeadresse mit (CR) quittieren.

Im anschließend erscheinenden Textlademenue wählen wir den Punkt "1" und geben den Datennamen „AUSGABE.ASM" ein. Im danach abermals erscheinenden Textlademenue beenden wir das Textladen mit "9". Als Assemblerausgabe wählen wir "1", damit der Assembliervorgang schnell erfolgt. Wenn keine Fehler gemacht wurden, fordert JADOS nach einiger Zeit dazu auf, dem Maschinencode einen Programmnamen zu geben.

Wir wählen den Namen „AUSGABE". Die Dateikennung .68K wird von JADOS automatisch vergeben und das Programm auf die Diskette gespeichert. Anschließend tippen wir „AUSGABE" und unser kleines Programm wird geladen und ausgeführt.

Damit unser Programm universell verwendbar wird, müssen wir darauf achten, daß der Maschinencode relokativ ist, d.h., daß das Programm nicht an eine feste Anfangsadresse gebunden ist. Dies ist sehr wichtig, wenn Programme einem größeren Benutzerkreis zugänglich gemacht werden sollen. Im JADOS-System gibt es ja, wie weiter oben erwähnt, die zwei unterschiedlichen Ladeadressen $400 und $C000 für .68K-Dateien.

Um Fehler beim Aufruf von JADOS-Unterprogrammen zu vermeiden, muß unbedingt darauf geachtet werden, daß das Adreßregister A6 nicht verändert wird, da es als globaler Zeiger auf die Variablen von JADOS benutzt wird!!!

Für die Universalität der Programme ist es sehr wichtig, die Verschiebbarkeit des Grunprogramms zu beachten. Es dürfen daher keine JSR (dXXXX-Aufrufe verwendet werden, sondern nur der TRAP #1-Aufruf oder der Sprungmechanismus, der unter CP/M 68K verwendet werden muß. Aber aufgepaßt! Die beiden letzteren Methoden verändern das Adreßregister A6. Dieses muß daher vor dem Aufruf einer Grundprogrammroutine gerettet werden, z.B. so:

```
CO2:    MOVE.L   A6,-(A7)
        MOVE     #!CO2,D7
        TRAP     #1
        MOVEA.L  (A7)+,A6
        RTS
```

Wer direkt auf I/O-Ports zugreift, sollte bedenken, daß sich die Portadressen bei 68008- und 68020-Systemen unterscheiden. Der Direktzugriff auf Portadressen sollte daher vermieden werden.

Die obigen Regeln zur Entwicklung universell verwendbarer Programme lassen sich sehr schön an unserem Beispielprogramm studieren.

Der Aufruf der Grundprogrammroutine CO2 erfolgt mit dem TRAP # 1. Wichtig! Das Register A6 wird vor dem TRAP gerettet und danach wieder hergestellt.

Folgende Maßnahmen wurden ergriffen, um das Programm relokativ zu machen:

- Die Texte werden relativ zum Programmzähler adressiert.
- Die Variablen werden relativ zum Adreßregister A4 adressiert. A4 selber wird zur Programmlaufzeit berechnet (LEA Var(PC),A4).
- Unterprogrammaufrufe erfolgen mit BSR.

Jede absolute Adressierung muß also unbedingt vermieden werden!!!

Abschließend noch ein Hinweis: In der aktuellen JADOS-Version 1.2a befindet sich ein kleiner „Bug". Der Befehl "ERA" zum Löschen von Dateien funktioniert nur auf dem jeweils aktuellen Laufwerk einwandfrei. Dies liegt an einem Fehler im Unterprogramm 17 ("ERASE"). Wenn z.B. das Laufwerk 1 aktuell ist – was durch das Systemprompt 1) angezeigt wird – darf keine Datei auf dem Laufwerk 2 gelöscht werden!!!

Hinweis: JADOS kostet DM 140,— und ist ab Lager lieferbar!

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

000400                  ****************************************
000400                  *  NAME: AUSGABE                       *
000400                  *                                      *
000400                  *  ZWECK: GIBT TEXTDATEI AUF BILDSCHIRM*
000400                  *         AUS, WOBEI IMMER NUR 1 SEKTOR*
000400                  *         GELADEN WIRD                 *
000400                  *  VERSION: 1.0                        *
000400                  *  DATUM  : 06.05.86                   *
000400                  *  AUTOR  : KLAUS JANSEN               *
000400                  *                                      *
000400                  ****************************************
000400
000400
000400                  * KONSTANTENDEFINITIONEN

= 00000400              SECLEN    EQU 1024   * SEKTORLAENGE
= 00000000              TRUE      EQU 0      * OK-MELDUNG
= 000000FF              FALSE     EQU 255    * FEHLERCODE
= 00000001              EOF       EQU 1      * END OF FILE - CODE
= 00000063              MEMFULL   EQU 99     * SPEICHERVOLL-CODE
000400
000400
000400                  * VARIABLENDEFINITIONEN
000400                  * ALS OFFSET ZUM ADRESSREGISTER A4
000400
= 00000000              USERFCB   EQU 0            * DATEISTEUERBLOCK
= 00000030              DATBUF    EQU USERFCB+48   * LADEPUFFER
= 00000430              LASTVAR   EQU DATBUF+1024  * LETZTE VARIABLE
000400
000400
000400                  * PROGRAMMSTART
```

```
000400  6000 0506          BRA     START
000404
000404
000404                  * UNTERPROGRAMME DES GRUNDPROGRAMMS
000404
000404                  CO2:
000404  48E7 0002         MOVEM.L A6,-(A7)
000408  3E3C 0021         MOVE    #!CO2,D7
00040C  4E41              TRAP    #1
00040E  4CDF 4000         MOVEM.L (A7)+,A6
000412  4E75              RTS
000414
000414
000414                  * UNTERPROGRAMME VON JADOS
000414
000414                  CLOSE:
000414  3E3C 000E         MOVE    #14,D7
000418  4E46              TRAP    #6
00041A  4E75              RTS
00041C
00041C                  GETNAME:
00041C  3E3C 0001         MOVE    #1,D7
000420  4E46              TRAP    #6
000422  4E75              RTS
000424
000424                  OPEN:
000424  3E3C 0013         MOVE    #19,D7
000428  4E46              TRAP    #6
00042A  4E75              RTS
00042C
00042C                  READSECT:
00042C  3E3C 0014         MOVE    #20,D7
```

68000 DOS ohne CP/M68k

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

000430  4E46              TRAP    #6
000432  4E75              RTS
000434
000434                  SCHREIBE:
000434  3E3C 0007          MOVE    #7,D7
000438  4E46              TRAP    #6
00043A  4E75              RTS
00043C
00043C                  SETDTA:
00043C  3E3C 0016          MOVE    #22,D7
000440  4E46              TRAP    #6
000442  4E75              RTS
000444
000444
000444                  * TEXTE
000444  42697474652044  TGETNAME: DC.B 'Bitte Dateinamen eingeben:  ',0
00044B  617465696E6160
000452  656E2065696E67
000459  6562656E3A2000
000460  53706569636865  TMEMFULL: DC.B 'Speicher voll ',13,10,0
000467  7220766F6C6C20
00046E  0D 0A 00
000471  46616C73636865  TNAMFALS: DC.B 'Falscher Dateiname ',13,10,0
000478  722044617465569
00047F  6E616D65200D
000485  0A 00
000487  44617465692064E  TNOTFND:  DC.B 'Datei nicht gefunden ',13,10,0
00048E  69636874206765
000495  66756E64656E20
00049C  0D 0A 00
00049F  0D 0A 0A 0A 3E  TREADY:   DC.B 13,10,10,10,'>>>>>>> Dateiausgabe '
0004A4  3E3E3E3E3E2044
0004AB  617465696175573
0004B2  6761626520
0004B7  6265656E646574            DC.B 'beendet <<<<<<',13,10,0
0004BE  203C3C3C3C3C3C
0004C5  0D 0A 00
0004C8  4C6573656666568  TUNDEF:   DC.B 'Lesefehler ',13,10,0
0004CF  6C65722000 0A
0004D5  00
0004D6
0004D6                  * ANGLEICH
0004D6                    DS 0
0004D6
0004D6                  * VARIABLENBEREICH
0004D6                  VAR:      DS.B    LASTVAR+2
0004D6
000908
000908                  * PROGRAMMSTART
000908                  START:
000908  49FA FBCC          LEA     VAR(PC),A4      * ADRESSZEIGER AUF VARIABLEN
00090C  43EC 0000          LEA     USERFCB(A4),A1  * ZEIGER AUF DATEISTEUERBLOCK
000910  41FA FB32          LEA     TGETNAME(PC),A0 * ZEIGER AUF TEXT
000914  6100 FB06          BSR     GETNAME         * DATEINAMEN EINLESEN
000918  0C00 0000          CMP.B   #TRUE,D0        *
00091C  6600 0054          BNE     NAMERROR        * FALSCHER DATEINAME
000920
000920  6100 FB02          BSR     OPEN            * DATEI OEFFNEN
000924  0C00 0000          CMP.B   #TRUE,D0        *
000928  6600 004E          BNE     NOTFND          * DATEI NICHT GEFUNDEN
00092C
00092C                  LOOP:
00092C  41EC 0030          LEA     DATBUF(A4),A0   * LADEADRESSE FESTLEGEN

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

000930  6100 FB0A          BSR     SETDTA
000934  6100 FAF6          BSR     READSECT        * SEKTOR LESEN
000938  0C00 0001          CMP.B   #EOF,D0         *
00093C  6728               BEQ.S   FILEEND         * DATEIENDE ERREICHT
00093E  0C00 0063          CMP.B   #MEMFULL,D0     *
000942  6728               BEQ.S   MEMORY          * SPEICHER VOLL
000944  0C00 0000          CMP.B   #TRUE,D0        *
000948  6616               BNE.S   ERROR           * UNDEF. FEHLER
00094A
00094A  323C 03FF          MOVE    #SECLEN-1,D1    * SCHLEIFENZAEHLER FUER AUSGABE
                        INNER:                      * AUSGABESCHLEIFE
00094E  1018                MOVE.B  (A0)+,D0        *
000950  6714                BEQ.S   FILEEND         * DATEIENDE
000952  0200 007F          AND.B   #$7F,D0         * BIT 7 AUSBLENDEN
000956  6100 FAAC          BSR     CO2             * ZEICHEN AUSGEBEN
00095A  51C9 FFF2          DBRA    D1,INNER        * NAECHSTES ZEICHEN HOLEN
00095E
00095E  60CC               BRA.S   LOOP            * NAECHSTEN SEKTOR LESEN
000960                                              * DISKETTENFEHLER
000960                  ERROR:
000960  41FA FB66          LEA     TUNDEF(PC),A0
000964  6016               BRA.S   MELDE
000966                                              * DATEIENDE
000966                  FILEEND:
000966  41FA FB37          LEA     TREADY(PC),A0
00096A  6010               BRA.S   MELDE
00096C                                              * SPEICHER VOLL
00096C                  MEMORY:
00096C  41FA FAF2          LEA     TMEMFULL(PC),A0
000970  600A               BRA.S   MELDE
000972
000972                  NAMERROR:                   * FALSCHER DATEINAME
000972  41FA FAFD          LEA     TNAMFALS(PC),A0
000976  6004               BRA.S   MELDE
000978
000978                  NOTFND:                     * DATEI NICHT GEFUNDEN
000978  41FA FB0D          LEA     TNOTFND(PC),A0
00097C
00097C                  MELDE:
00097C  6100 FAB6          BSR     SCHREIBE        * MELDUNG AUSGEBEN
000980                  ENDE:
000980  6100 FA92          BSR     CLOSE           * DATEI SCHLIESSEN
000984  4E75               RTS                     * RUECKSPRUNG INS JADOS
000986
000986

0EBC8E  Ende-Symboltabelle
```

CP/M 68K, C UND MODULA

Gleitkommaarithmetik für MODULA-2

Elmar Henne

Für den in der letzten Ausgabe der *LOOP* vorgestellten MODULA-2 Compiler ist jetzt auch die Gleitkommaarithmetik verfügbar. Besitzer des MODULA-2 Compilers können diese Aufrüstung kostenlos nachfordern.

Die Gleitkommaarithmetik entspricht dem IEEE-Standard und ist kompatibel mit der FPU 68881 für den 68020. Es wurden zwei Datenformate realisiert:

REAL:
32 Bits (8Bits Exponent, 23 Bits Mantisse), Bereich: 1.2E-38 .. 3.4E+38, (FPU single Format)

LONGREAL:
96 Bits (15 Bits Exponent, 64 Bits Mantisse, 16 Bits unbenutzt), Bereich 8E-4933 .. 6=+4931, (FPU extended Format)

Der Compiler erzeugt für die Gleitkommaoperationen immer FPU-Instruktionen, die bei Systemen ohne FPU (68008 bzw. 68000 Prozessoren) durch ein spezielles Modul abgefangen und ausgewertet werden (FPU-Befehle erzeugen einen Line-F-Emulator-Trap). Auf diese Weise ist es möglich, daß das gleiche Programm sowohl mit der FPU auf einem 68020-System, als auch auf Systemen ohne FPU mit einer programmierten Gleitkommaarithmetik ablauffähig ist. Ein Beispiel für ein solches Programm ist der MODULA-2 Compiler selbst.

Man kann Programme natürlich auch ohne die programmierte Gleitkommaarithmetik binden, dann sind sie allerdings nur noch auf 68020-Systemen mit FPU 68881 lauffähig.

Diese neue Version des Compilers arbeitet auch mit dem neuen Uhrenbaustein (Smart Watch DS1216). Es erscheint dann im Listing das richtige Datum und am Ende die Übersetzungszeiten. Außerdem enthält die mitgelieferte neue Bibliothek ein Modul zur Ansteuerung des Uhrenbausteins und ein Programm zum Stellen der Uhr.

Hinweis: Der erwähnte Uhrenbaustein ist bei uns erhältlich.

Uhrenbaustein DS1216
Bestell-Nr. 10702 DM 149,–

LOOP nun auch im Bankeinzug

Viele *LOOP*-Abonnenten vergessen die erneute Zahlung der DM 20,– und würden somit keine neue *LOOP* mehr erhalten! Deshalb haben wir für die *LOOP*-Zahlung das Bankeinzugsverfahren neu aufgenommen.

Es funktioniert wie folgt: Sie ermächtigen uns auf beiliegendem Formular, jeweils DM 20,– für fünf *LOOP*-Ausgaben von Ihrem Konto abzubuchen. Diese Ermächtigung erlischt automatisch mit Kündigung Ihres Abos, die Sie jederzeit vornehmen können. Den zuviel abgebuchten Betrag erhalten Sie dann von uns zurück!

Maus-Editor PEDIT für den NDR-Computer mit CP/M68K

Elmar Henne

Der in dem MODULA-2 Artikel der letzten Ausgabe angekündigte mausgesteuerte Editor ist inzwischen lieferbar. Durch eine optimale Ausnutzung der GDP-Befehle war es möglich, auch auf dem NDR-Computer einen Editor anzubieten, der dem entspricht, was man von anderen mausorientierten Systemen, wie Macintosh, Amiga oder GEM, gewöhnt ist. Das Ganze geht nun nicht, wie man vielleicht zunächst glauben könnte, zu Lasten der Geschwindigkeit, sondern einige Operationen (z.B. Scrollen) gehen sogar deutlich schneller als bei anderen NDR-Editoren. Den einzigen Unterschied den man im Vergleich zu echten Bitmap-Grafik-Rechnern sieht, ist ein Flimmern des Cursors bei schnellen Mausbewegungen.

Doch nun zu den Möglichkeiten, die PEDIT bietet:

- Cursorsteuerung durch Maus (Atari oder Microsoft) oder über Tasten (frei definierbar)
- Befehlseingabe über (popup-)Menue
- Befehle und Hinweise in Deutsch
- Scrollbar für relatives und absolutes Positionieren
- Mehrere Fenster (Windows) für eine oder mehrere Dateien
- invertierter Text zur Darstellung von Selektionen
- Parameterdatei zur Konfiguration von PEDIT
- Makrodefinition im Lernmodus
- Kurzeingabe von Löschen, Kopieren und Transportieren über Maustasten
- Umbruch bei Zeilen mit mehr als 80 Zeichen
- Lesen von Dateien aus anderen User-Bereichen
- Erzeugen einer Backup-Datei
- Möglichkeit zur Erzeugung einer Replay-Datei zum automatischen Wiederholen der gemachten Änderungen nach einem Systemabsturz

PEDIT ist selbst in MODULA-2 programmiert und läuft unter CP/M68K. Im Paketangebot PEDIT + MODULA-2 werden auch die Moduln für die Grafik- und Mausansteuerung mitgeliefert.

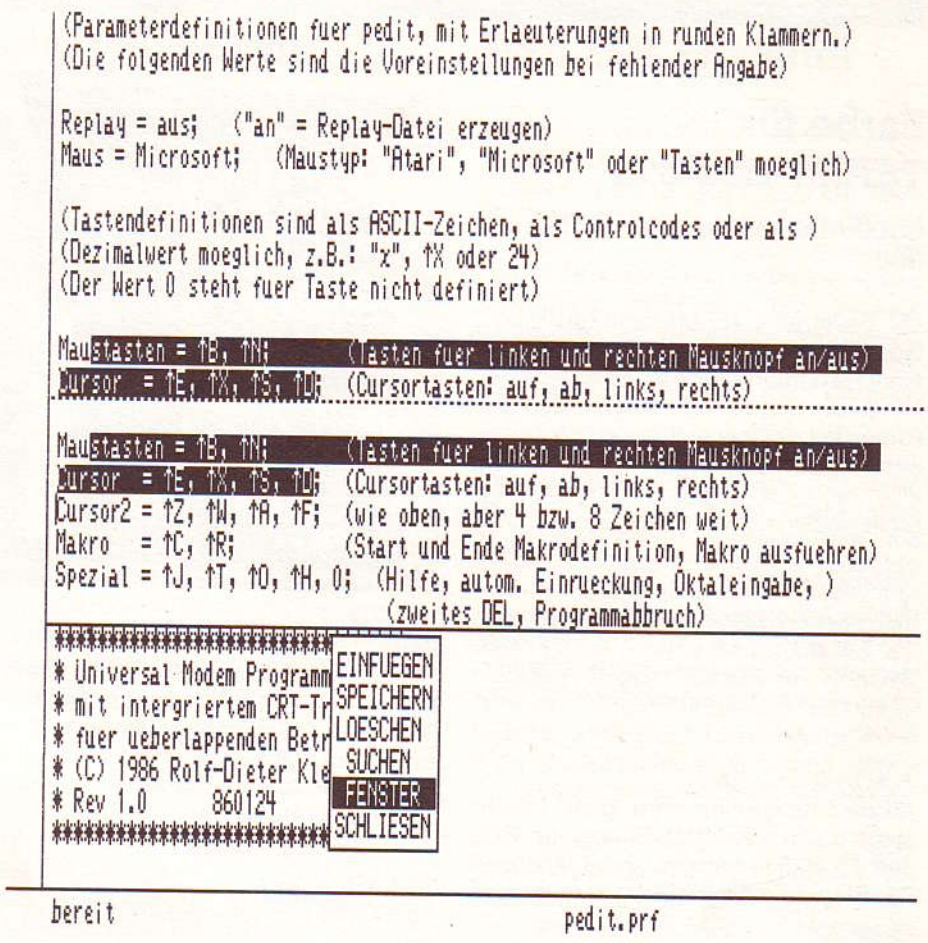


Bild 1: Dieses Bild zeigt PEDIT in Aktion mit drei Fenstern für zwei Dateien. Die beiden oberen Fenster zeigen die Parameterdatei von PEDIT mit den Voreinstellungen.

Der Editor ist ab Lager lieferbar bei:
GES, Postfach 1610, 8960 Kempten
Telefon: (0831) 6211

p1, Gesellschaft für Informatik mbH,
Gotthardstraße 99, 8000 München 21
Telefon: (089) 5806099

PEDIT Text-Editor für CP/M68K
5 ¼", 80 Spuren
Bestell-Nr. 10565 DM 250,—
3 ½", 80 Spuren
Bestell-Nr. 10566 DM 250,—
8" SS50
Bestell-Nr. 10567 DM 250,—

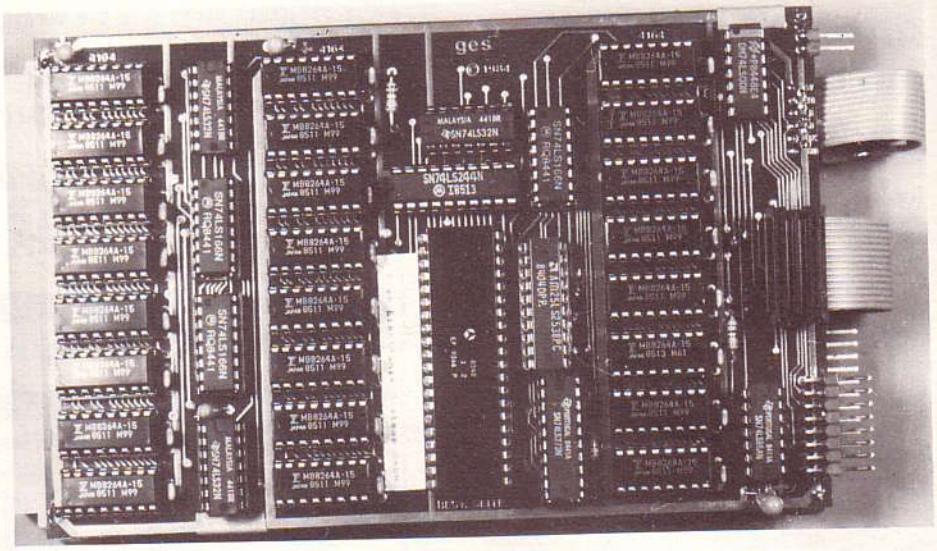
MODULA-2-Compiler mit Text-Editor,
5 ¼", 80 Spuren
Bestell-Nr. 10337 DM 598,—
3 ½", 80 Spuren
Bestell-Nr. 10553 DM 598,—
8", SS50
Bestell-Nr. 10554 DM 598,—

Der mc-CP/M-COMPUTER

Farbe für TERM1-Besitzer

COLOUR-Erweiterung für TERM1 und GSS

Die Baugruppe TERM1 kann relativ einfach mit einer Aufsteckbaugruppe zu einer Farbgraphikkarte erweitert werden. Mit dieser Doppelkarte können dann 16 Farben bei gleicher Auflösung dargestellt werden. Ab der Monitorversion „TERM V4.0" sind bereits die Routinen für die Farberweiterung enthalten.



Technische Daten

Bildwiederholspeicher:
192 KByte RAM (je 64K für R, G und B). Vier umschaltbare Bildseiten je Grundfarbe mit einer Auflösung von 256 * 512 Bildpunkten

Anschluß an TERM1:
Aufsteckbaugruppe wird über Durchsteck-IC-Sockel (Wrap-Sockel) für GDP und 25LS2538 und mit einem 14poligen DIL-Stecker mit der TERM1 verbunden.

Ausgänge:

R	rot
G	grün
B	blau
H	Horizontal Synchron Signal
V	Vertikal Synchron Signal
D	Hintergrund
GND	Masse
BAS	Bild-, Austast-, Synchronsignal

Abmessungen
160 * 100 mm (Europakarte)

Farben
8 Farben + Hintergrundebene

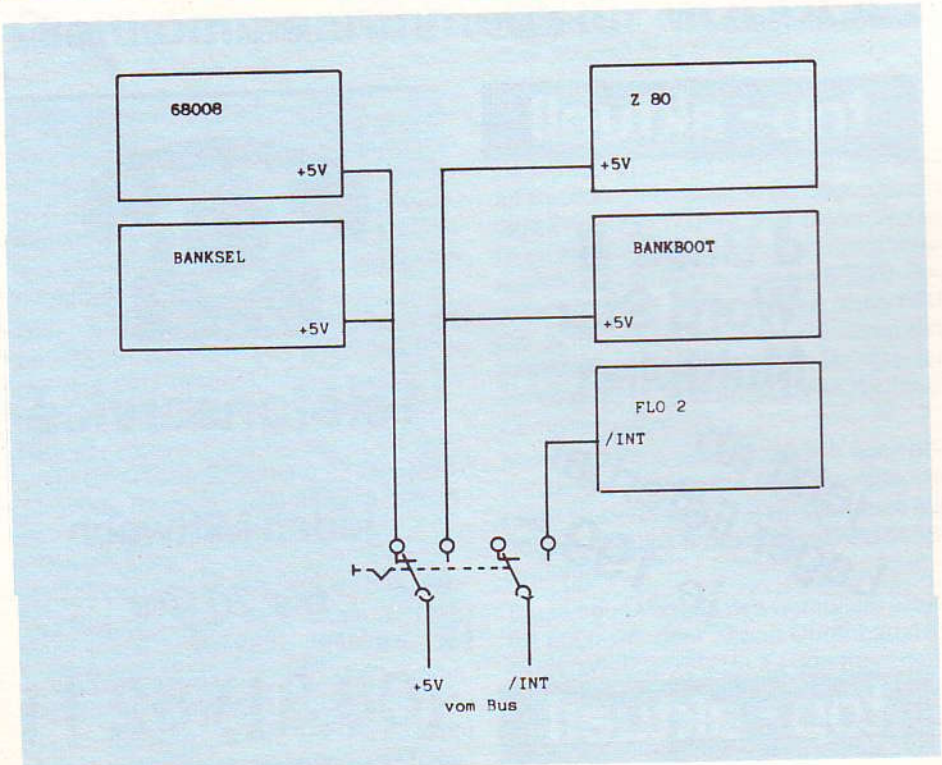
TERMCH
Handbuch zur Colour-Erweiterung TERM
Bestell-Nr. 10537 DM 20,–

TERMCP
Leiterplatte Colour-Erweiterung TERM
Bestell-Nr. 10538 DM 99,–

TERMCPE
Leiterplatte und EPROM
Bestell-Nr. 10539 DM 149,–

TERMCB
Bausatz Colour-Erweiterung TERM
Bestell-Nr. 10536 DM 498,–

TIPS + TRICKS – TIPS + TRICKS



Zwei auf einen Streich – Z80 + 68008 gleichzeitig auf dem Bus

von Rüdiger Bäcker

Viele Benutzer des NDR-Klein-Computers haben ihren Computer über die SBC II/CPUZ80 hin zum 68008-System ausgebaut. Da man aber meistens noch so einiges an Software für die Z80-CPU hat, will man noch so ab und an auf die Z80-CPU zurückgreifen. Hier wird dann die CPU-68008 und das 68008-Grundprogramm vom Bus entfernt und dafür die CPU-Z80 mit dem entsprechenden Betriebssystem eingesteckt.

Diese umständliche Prozedur ist für die Benutzer, die ihrem Computer schon ein „Zuhause" in Form eines Gehäuses spendiert haben, wohl fast unmöglich.

Es geht aber auch anders! Ein gleichzeitiger Betrieb von Z80 und 68008 ist möglich, wenn man wie folgt vorgeht:

Das 68008-Grundprogramm ist ab Adresse $e0000 untergebracht und wird von einem Boot-Eprom (EBOOT 68) auf einer Bank-Sel gesucht und gestartet. Die Spannungsversorgung von + 5 V für die Bank-Boot oder Bank-Sel sowie für die 68008 CPU werden über einen Umschalter an die Karten geführt, ebenso die Versorgung der CPU Z80 und einer weiteren Bank-Boot mit dem Flomon und optional dem Z80-Grunprogramm in einer Version für Adresse $2000 oder EZEAT-Betriebssystem. Die Beine der Steckerleisten werden bei den betroffenen Karten einfach so umgebogen, daß sie nicht mit den + 5 V vom Bus in Berührung kommen. Nun kann man mit dem Umschalter zwischen Z80 und 68008 umschalten. Die beiden Systeme benutzen dann die gesamte Peripherie gemeinsam, die Speicherinhalte ab Adresse $400 – $7fff und ab ca. $8100 (hängt von der Länge des BOOT-Programmes ab) bleiben beim Unschalten erhalten.

Will man mit den Betriebssystemen CP/M 80 und CP/M68K arbeiten, so muß beim Floppycontroller noch bei 68008-Betrieb die Leitung /INT zum Bus unterbrochen werden. Dies geschieht durch einen weiteren Kontakt des Umschalters. Auch hier kann man wieder einfach das entsprechende Beinchen der Stiftleiste (Pin 32) umbiegen.

Einen kleinen Nachteil hat diese Lösung jedoch noch, beim Umschalten wird natürlich jedesmal ein POWER-On-Reset erfolgen.

Die genaue Verdrahtung geht aus dem Schaltplan hervor.

Hinweis der Redaktion: Wir finden diese Lösung sehr elegant! Um den Wermutstropfen – Anschaffung einer BANKSEL – etwas zu lindern, bieten wir die BANKSEL für LOOP-Leser zu folgendem Sonderpreis an:

BANKSEL: Bausatz statt DM 79,95 = DM 35,–

Bestell-Nr. 10088.

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

Computer und Finanzamt

von Gerd Simon

Immer wieder steht man vor der Frage: „Kann ich meinen NDR-Klein-Computer von der Steuer absetzen?"

Aus meiner Erfahrung möchte ich an dieser Stelle etwas dazu beitragen. Gleich vorweg, ich bin kein Steuerexperte – gebe nur wieder, was ich so erlebte.

Grundsätzlich muß man sagen, daß wohl in allen Berufsgruppen der „Micro-Computer" Einzug gehalten hat und somit jeder etwas davon zu spüren bekommt. Direkt Betroffene ändern ihr Berufsbild. So werden aus Technikern mit internen Umschulungen Anlagen-Elektroniker. Kfz-Elektriker müssen sich mehr und mehr mit den Bordcomputern rumschlagen, die Sonderfahrzeuge steuern und von den Flugzeug-Elektrikern gar nicht zu reden. Da gehts ja schon lange nicht mehr ohne.

Im allgemeinen hängt aber die Ausbildung für die schon im Beruf stehenden hinterher. So muß sich jeder bemühen, hinter die „Machenschaften" eines Computers zu kommen. Dies erfordert Geld und Zeit. Der Gesetzgeber sagt nun, daß man für seine Fort- und Weiterbildung Kosten geltend machen kann. Am einfachsten ist es, wenn man einen artverwandten Beruf hat (dann fällt es den Finanzbeamten leichter). Leute mit nicht so viel Glück, müssen sich von ihrem Arbeitgeber eine Bescheinigung geben lassen. Diese sollte aber unbedingt den Hinweis tragen, daß die Weiterbildung in "Soft- und Hardware" für das berufliche Fortkommen förderlich ist. Damit werden ja auch die Anteile für die Hard- und Software bestimmt. Und wenn Sie Rechnungen sammeln, denken Sie auch an die Schrauben für Gehäuse, den Alu-Winkel und das Klebeband zur Kabelhalterung – alles sind Kosten.

Machen Sie es wie bei der Programm-Entwicklung. Erst einen Plan, so wie Sie Ihren Computer aufbauen wollen, und nach diesem Plan setzen Sie alles ab.

Dies sind nun meine praktischen Erfahrungen – erkundigen Sie sich bitte bei Ihrem zuständigen Finanzamt oder bei Ihrem Steuerberater.

Rechnen Sie nach, man kann bis zu 1/4 zurückbekommen – oder 1/4 größer ausbauen. Je nach Geschmack.

Hinweis: Gerd Simon stellt sich als Kontaktmann für den Frankfurter Raum zur Verfügung; hier seine Anschrift:
Gerd Simon, Görlitzer Straße 9,
6203 Hochheim, Tel.: (06146) 7384 (nicht nach 20.30 Uhr! – da kleine Kinder!)

Tips zum Z80-Emulator

Als Besitzer von CP/M68K steht man des öfteren vor dem Problem, daß benötigte Software nicht für den 68000 erhältlich ist. Dagegen könnte man beim Z80 auf Standardprogramme zurückgreifen. Um nun nicht immer die Hardware seines Computers umzurüsten, gibt es eine Möglichkeit in Software: Der Einsatz eines Z80-Emulators. Dieser kann zum Preis von DM 598,– von GES bezogen werden.

Der Emulator vermittelt dem Anwender den Eindruck, daß er auf einem (allerdings langsamen) Z80-Computer mit CP/M2.2 arbeitet. WordStar, Multiplan und auch DBASE II und viele andere Programme sind sofort lauffähig. Wenn man aber ein Programm speziell für den NDR-Computer besitzt, welches direk auf I/O-Adressen zugreift (z.B. auf GDP64K), funktioniert es nicht. Der Emulator kennt ja nicht die Abbildungsvorschrift (Z80 I/O-Adressen ⟶ 68008 I/O-Adressen).

Allerdings gibt es eine Möglichkeit, diese Abbildungsvorschrift dem Emulator mitzuteilen. Dazu muß ein Programm mit dem Namen EMUIO.REL auf der Emulator-Diskette voranden sein. Dies ist ein 680xx Programm im relocativen Format, welches Sie auf folgende Weise erzeugen können:

Geben Sie folgendes Programm ein, nennen Sie es EMUIO.S.

```
iobase: .equ $ffffff00
    *                   für 68000 hier $ffffffe00
io:
    jmp input
    jmp output
input:
    movea.l #iobase,a0
    and.w #$00ff,d0
    *                   für 68000 hier noch 'add.w d0,d0' einfügen
    move.b 0(a0,d0.w),d0
    rte
output:
    movea.l #iobase,a0
    and.w #$00ff,d0
    *                   für 68000 hier noch 'add.w d0,d0' einfügen
    move.b d1,(a0,d0.w)
    rte
```

Assemblieren Sie dieses Programm mit dem AS68 und linken Sie dann mit dem LO68:

as68 emuio.s
lo68 –r –o emuio.rel emuio.o

Jetzt haben Sie das EMUIO.REL erzeugt, welches der Emulator dazulädt. Jetzt sollten alle auf dem Z80 laufenden Programme auch auf dem Emulator laufen. Dies gilt nicht für das MODEM7 auf der Emulator-Diskette, welches nicht auf den NDR-Computer angepaßt ist.

ROA-Akku – gepuffert RAM-Floppy verliert keine Daten

von Dr. Jürgen Gabriel, 4270 Dorsten 1

Die Akkupufferung auf der SBC3 bringt für den CP/M-Betrieb keine Vorteile.

Sehr vorteilhaft wäre aber eine akkugepufferte RAM-Floppy mit ROA64-Platinen. Alle auf Laufwerk E enthaltenen Programme wären auch nach dem Ausschalten noch im Rechner und dami sofort nach dem Wiederanschalten verfügbar.

Um dies zu erreichen, hatte ich zunächst überlegt, ob man das Schaltungsprinzip der SBC3 für die Akkupufferung übernehmen könnte. Dazu wäre aber ein Schalttransistor für jeden –CS-Eingang der RAM- notwendig gewesen.

In Anlehnung an eine Schaltung aus C'T 5/86, S 98 habe ich folgende Lösung gefunden, die mit sehr wenigen Änderungen auf der ROA-Platine auskommt, aber dennoch zufriedenstellend arbeitet:

Änderung der ROA64 zur Akkupufferung aller acht 6264-RAMs

Die noch freie Busleitung 54 wird zur Bereitstellung der gepufferten + 5 V eingesetzt.

Bestückungsseite:

1. Pin 16 von J10 (74LS138) wird von der breiten 5 V-Leiterbahn getrennt.
2. Die schmale 5 V-Leiterbahn links neben "RDK ges" wird unterbrochen.

Lötseite:

1. Die 5 V-Leiterbahn wird unterhalb des Aufdrucks "RDK ges" unterbrochen.
2. Pin 54 der eingelöteten Stiftleiste wird durch isolierten Schaltdraht mit Pin 28 von J8 verbunden.
3. Pin 16 von J10 wird durch isolierten Schaltdraht mit Pin 28 von J4 verbunden.

Damit der Akku nicht zu viel Strom liefern muß und alle –CS-Eingänge bei Akkubetrieb auch sicher gesperrt (HIGH) sind, muß J10 statt eines 74LS138 ein 74HC138 sein.

Nach Durchführung dieser Änderungen, die auch noch an einer voll bestückten ROA64K möglich sind, behalten die ROA64K-Karten auch nach dem Abschalten des Netzstromes ihre Informationen.

Dies ist wie schon weiter oben angedeutet, besonders beim Betrieb einer RAM-Floppy sehr vorteilhaft. Alle häufig benötigten Programme können auf Laufwerk E gespielt werden und sind nach dem Wiedereinschalten des Rechners in Sekundenbruchteilen abrufbar.

An Pin 54 muß dafür natürlich eine gepufferte 5 V-Spannung anliegen. Entweder findet man für die Akkus und die wenigen Bauteile im Gehäuse einen Platz oder, wenn auf dem Bus noch Platz zur Verfügung steht, kann man folgende Schaltung auf einer Lochrasterplatine aufbauen:

Bauteile:

3	1,2 Volt Akku 500 mA
2	Germaniumdiode z.B. AA113
1	Widerstand 1 kOhm
1	Keramikkondensator 100 nF
1	Tantal-Kondensator 10 uF

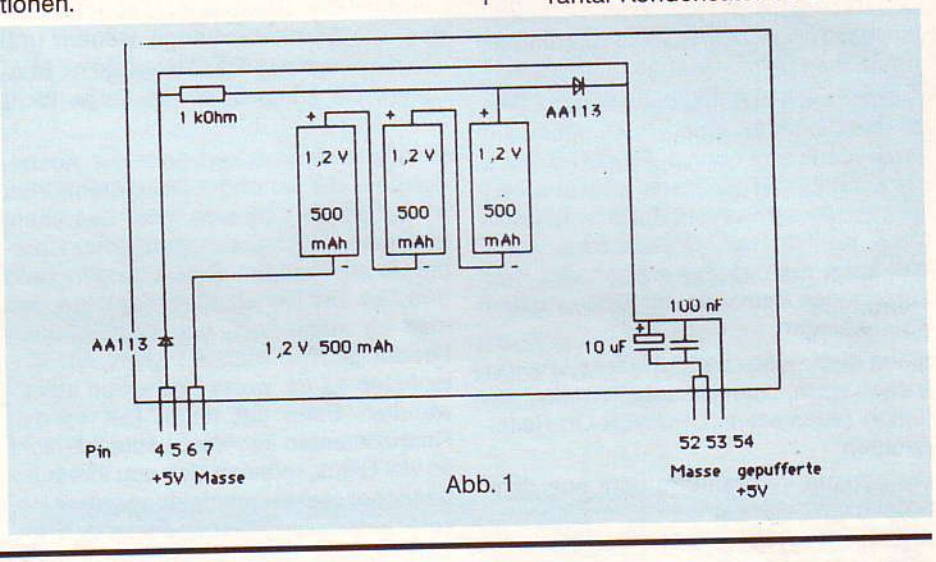


Abb. 1

Anpassung dBase II, WordStar, Multiplan

von Markt & Technik, H. Teller

Sehr wichtig ist es bei den Produkten darauf zu achten, daß möglichst viele handelsübliche Drucker unterstützt werden, was durch das Fehlen des dBase-Anpassungsprograms EINST erschwert wird. Hierbei kann man sich im DBase mit CHR-Codes behelfen.

Eine gut zu propagierende Version einer Druckeranpassung bei Wordstar ist die, daß man auf die erste freidefinierbare Funktion nur den Escape-Code legt, d.h. nur hex 1B.

Folgende Codes, wie "W" bei EPSON für EXPANDED, können direkt eingegeben werden. Somit lassen sich theoretisch alle Druckercodes ausnutzen.

Erklärlich muß auch folgender Umstand gemacht werden: Tief- bzw. Hochstellen sind Modis, die nur auf Typenraddruckern anzuwenden sind. Sub- und Superscript müssen dementsprechend selbst definiert werden.

Unter Lizenz genommen wurde auch das Spellstar nicht, somit fallen die Textkorrekturen weg.

Zur dBase-Version 2.41 ist noch zu sagen, daß der Report-Generator nicht einwandfrei funktioniert. Bei der Abfrage nach Gesamtsummen muß man amerikanisch mit "Y" antworten, damit auch die Frage nach Zwischensummen beantwortet werden kann.

Außerdem wird "GESAMMT" mit DOPPEL-M geschrieben, was man durch einen SID-Patch bei $1900 abändern kann.

Bei Multiplan ist es von Vorteil zu erklären, wie bei „Druck Operation" die Sequenzen eingegeben werden:
Escape entspricht der Eingabe "^Ä", jeder folgende Code wird im ASCII-Format übergeben.

Dies sind die wichtigen allgemeinen Merkmale, die eine vermehrte Nachfrage nach sich zieht.

Aus der Technik

COL 256:

Aus technichen Gründen mußte die Adresse der COL256 geändert werden. Die alte Adresse, die auf alle Platinen COL 256 r2 voreingestellt ist, wählte die I/O-Ports $CC, $CD, $CE und $CF aus. Dort liegt jedoch auch die SASI-Schnittstelle.

Die neuen I/O-Adressen der COL256 sind $AC, $AD, $AE und $AF. Bitte beachten Sie bei der Erstellung neuer Software diese neue Adresse.

Die Umstellung kann wie folgt vorgenommen werden:

JMP2 (Ansicht von der Lötseite, Bausanschluß zum Betrachter zeigend),

Lage: rechts unten auf der Leiterplatte

alt:

```
o o o o o o
    | |
o o o o o o
```

neu:

```
o o o o o o
  |   |
o o o o o o
```

Nun sollte man zum Test die im Handbuch beschriebene Initialisierungsroutine aufrufen, aber mit neuen Definitionen:

```
crt     equ     $ffffffac
crtd    equ     $ffffffac
crtb    equ     $ffffffae
```

Fehler bei der HCOPY/MAUS: Das verbogene Fadenkreuz

Grund der Änderung:

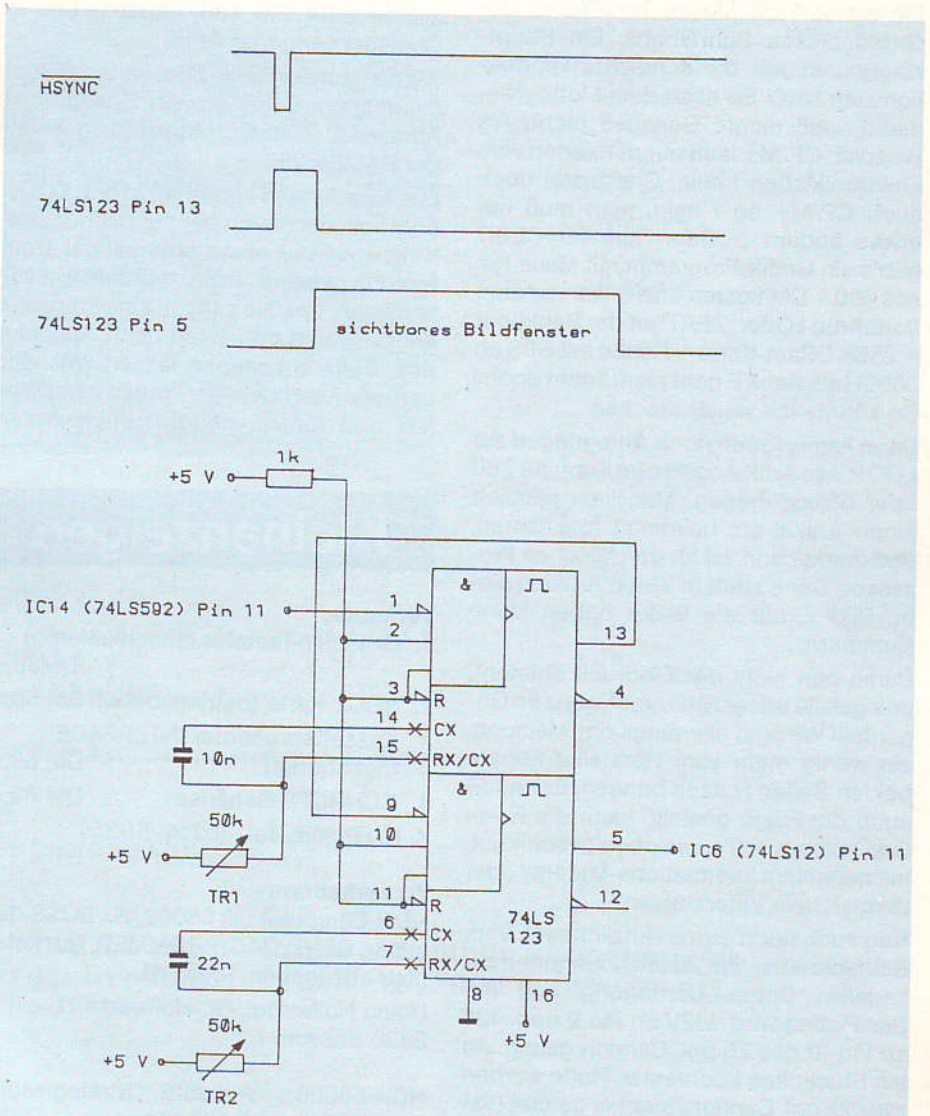
Die HCOPY/MAUS-Baugruppe bietet unter anderem die Möglichkeit, ein Fadenkreuz einzublenden. Die Einblendung erfolgt durch direkte Beeinflussung des BAS-Signals der GDP-Baugruppe am NDR-Klein-Computer bzw. der TERM1-Baugruppe am mc-CP/M-Computer. Die horizontale Linie des Fadenkreuzes entsteht durch ständige Aktivierung des Video-Signals während der Ausgabe der betreffenden Zeile (ca. 63 us bei 14 MHz). Einige Monitore reagieren auf die ständige Aktivierung des Video-Signals innerhalb einer Zeile allerdings sauer. Dies macht sich durch Verzerrungen und Bildstörungen bemerkbar. Zur Behebung dieser Unsauberheit bedarf es eines zusätzlichen Monoflops. Dieses Monoflop dient zur Sperrung des Video-Signals außerhalb des sichtbaren Bildbereichs. Die Abbildung zeigt die dafür nötige Schaltungserweiterung. Der Baustein 74LS123 beinhaltet zwei Monoflops. Die Triggerung des ersten Monoflops erfolgt durch die negative Flanke des HSYNC-Signals. Am Ausgang (Pin 13) erscheint ein positiver Impuls mit einer Impulsdauer, die der Zeit entspricht, während sich der Elektronenstrahl im unsichtbaren Bereich des Zeilenanfangs befindet. Die abfallende Flanke des Impulses triggert dann das zweite Monoflop. Die Impulsdauer des zweiten Monoflops entspricht dann dem sichtbaren Bildbereich. Durch Abtrennen von Pin 11 des IC6 (74LS12) vom Pin 12 des gleichen Bausteins und der Erstellung einer Verbindung zum Ausgang des zweiten Monoflops (Pin 5) kann die Einblendung des Fadenkreuzes ziemlich genau auf den sichtbaren Bereich des Bildfensters beschränkt werden.

Durchführung der Änderung:

Beim Auftreten unakzeptabler Bildstörungen bei der Einblendung des Fadenkreuzes sollte die kleine Zusatzschaltung entsprechend dem abgebildeten Schaltplan auf einer kleinen Lochrasterplatte aufgebaut und entsprechend angeschlossen werden. Bei der Wahl der Kondensatoren sollte nicht am falschen Ende gespart werden. Für zeitbestimmende Kondensatoren eines Monoflops sollten immer gute Wickelkondensatoren Verwendung finden. Bei der Verwendung billiger Keramikkondensatoren verändert sich während des Betriebs ständig die Impulsdauer des Monoflops und das Bildfenster driftet ab. Für die Trimmer (einstellbarer Widerstand) sollten Spindel-Potentiometer verwendet werden.

Einstellung der Hilfsschaltung:

Der Trimmer TR1 dient zur Einstellung des linken Randes des sichtbaren Bildbereichs, während der Trimmer TR2 die Dauer des sichtbaren Bereichs bestimmt. Zu Beginn der Einstellung sollte der Trimmer TR1 den minimalsten Widerstand und TR2 den maximalsten Widerstand besitzen. Im Normalfall bleibt der Bildschirm dann beim Einschalten dunkel. Am Trimmer 1 wird dann solange gedreht (Zunahme des Widerstands) bis das Bild am Bildschirm erscheint. Nach dem Einblenden des Fadenkreuzes (siehe Handbuch HCOPY/MAUS) erfolgt dann die Endeinstellung. Durch Drehen am Trimmer 2 (Verkleinerung des Widerstands) kann dann der rechte Teil des horizontalen Strichs des Fadenkreuzes annähernd auf den sichtbaren Bereich beschränkt werden. Durch Veränderung der Einstellung am Trimmer 1 gilt es dann den Anfang des linken Teils des Fadenkreuzes zu bestimmen. Eine Nachkorrektur des rechten Randes beendet dann den Ausgleich.



Pin 11 des IC 74LS12 hochbiegen und ausserhalb der Fassung direkt anschliessen. Grund: Bruecke zwischen Pin 11 und Pin 12 auf der Bestuechungsseite (unter dem IC-Sockel) !

BRIEFE

Sehr geehrte *LOOP-Redaktion*,

Durch das veröffentlichen meines Briefes durch Sie, und einer Anzeige in einer Fachzeitschrift durch mich zum Erfahrungsaustausch über den NKC hier in unserem Raum (Ruhrgebiet), hat sich einiges bewegt.

Es setzte ein heftiges Telefonieren, Schreiben und Besuchen ein. Wir waren sehr erstaunt, wieviele NKC's es in nächster Nähe gibt. Um nun mal Leute miteinander bekannt zu machen, haben wir am 05. 4. 86 ein Treffen veranstaltet, meine Firma hatte uns freundlicherweise einen Raum zur Verfügung gestellt. Trotz schlechtem Wetter haben einige Leute eine weite Anreise nicht gescheut, so daß dann 19 Mann zusammen kamen, 8 Rechner aufbauten und dann ca. 6 Stunden nicht mehr zur Ruhe kamen.

Ein erstes Fazit dieses Treffens ist der Grund dieses Schreibens. Ein Hauptklagepunkt war die schlechte Information zum NKC. So nach dem Motto: Niemand weiß nichts Genaues nicht. Als Beispiel: CP/M+ läuft nur mit Karten vom Elektronikladen ! nein, Graf bietet doch auch CP/M+ an ! nein, man muß nur etwas ändern ! „Oder: auf der CEBIT gab's ein Grafik-Programm mit Maus ! ja, soll 600.– DM kosten ! nein, war nur zum Vorführen ! Oder: ZEAT auf der Bankboot + 256K DRam-Karte + Roa64 mit SPS ab 0000H als Bank E geht nicht ! geht doch ! So könnte ich weiter machen.

Dann kamen auch noch Anregungen zur *LOOP*. Alle Artikel sollten im Kopf „für Z80 oder 68xxx" tragen. Man liest nämlich einen Artikel an, bekommt rote Ohren, und merkt dann, ist für den falschen Prozessor. Dann steht in vielen Artikeln siehe Bild x, nur die Bilder haben keine Nummern.

Damit nun nicht der Eindruck entsteht, uns gefalle unser NKC nicht, ganz im Gegenteil! Wir sind allerdings der Meinung, ein wenig mehr vom Hersteller könnte beiden Seiten Nutzen bringen. Es wurde auch die Frage gestellt, kann die Firma Graf solche Treff's, wie oben geschildert, mit neuestem Informations-Material oder dergleichem unterstützen?

Nun auch noch etwas Nützliches: 1. Vom Briefschreiber. Ein Zusatz-Platinchen zur seriellen Daten-Übertragung! Auf der Ser-Platine wird +12V an Pin 9 und -12V an Pin 10 des 25 pol. Cannon gelegt. An ein Stückchen Lochraster-Platte werden zwei 25 pol. Cannon-Stecker gelötet (Vater und Mutter). Auf die Platte die paar Bauteile und dann auf die Ser gesteckt. Jetzt läßt sich die Schnittstelle auch im TTY (20 mA) Modus betreiben (Sender aktiv, Empfänger passiv). Als zweites ein Beitrag zur Diskussion oder Probieren: Zwei Systeme auf einem Bus, von Herrn Holger Lech, Dahlmannsweg 24e, 4390 Gladbeck, Tel. 0 20 43/32 989 einem Teilnehmer unseres Treff's.

Willi Wegemann
Bredenscheiderstr. 28, 4320 Hattingen

Antwort LOOP:

Wir freuen uns, daß sich so viele NKC-Interessenten bei Ihnen getroffen haben. Wir werden versuchen, die offenen Fragen im Rahmen unseres neuen, umfangreichen Kataloges zu klären, der bis spätestens Herbst verfügbar sein wird.

Zu Ihren konkreten Fragen:

1. CP/M+ bieten wir heute noch nicht an. Wir stehen jedoch mit dem Elektronikladen in Verhandlung, dieses CP/M+ auf der neuen CPU64/180 anzubieten.

2. Das Grafikprogramm mit der Maus für die COL256 wird derzeit fertiggestellt und vermutlich ab Juli verkauft. Es wird sicher unter DM 100,– kosten und im Quelltext verfügbar sein.

3. ZEAT auf der Bank-Boot verlangt einen mindestens 64KB großen Speicher ab Adresse 0. Natürlich funktioniert es mit der RAM64/256.

Die Anregungen haben wir in die *„LOOP"* übernommen, auch bei der Bildnumerierung wollen wir etwas ordentlicher arbeiten. Für weitere Treffs würden wir vorschlagen, daß Sie einfach eine Frageliste erstellen und uns diese nach Abschluß des Treffs zukommen lassen. Wir werden, wie jetzt auch, die Fragen beantworten und Ihnen wieder zurücksenden.

Kleinanzeigen

Verkaufe:

1. CHERRY-Tastatur (anschlußfertig) DM80,—
2. SBC2-Karte (betriebsbereit) DM 50,—
3. Kassettenrecorder INTERFACE triebsbereit) DM 50,—
4. SCHROFF-Gehäuse DM 80,—

K. H. Findeis, Tel.: (0214) 59257

Zu verkaufen:

NDR-Computer mit 68008, 3 x BUS3, Tastatur, CENT, CAS, PROMMER. Nur komplett abzugeben. Preis VB.

Heino Hollwedel, Gr.-Hollwedel 11,
2830 Bassum 1

NDR-68008: Verkaufe Strategiespiel Reversi, sehr spielstark!
Auf EPROM oder Diskette (unter JADOS). DM 29,— per NN.

K. Janßen, Hanninxweg 74, 4150 Krefeld 1

68000-System:
Grundprogamm-EPROMs selbst herstellen: 68008-System mit Prommer, 64KByte, freies RAM. Komfortables Umwandlungsprogramm auf EPROM für DM 20,— bei mir erhältlich. Info gegen Freiumschlag.

E. Walter, Vermehrenrich 11f,
2400 Lübeck

Progr. zum Generieren des Eproms der ELZET80 DIN-Tast Hilfsprogramme für 68008: RAM-Vergleich, -Durchsuch, -Kopie, HEX-Eingabe, Summe für DATEY, Schnellprom. Textopti, Biorhythmus. Info gegen Rückumschlag.

Achim Scheffel, Boelckestraße 8,
6503 Mainz-Kastel, Tel.: (06134) 64944

Verkaufe: 3 x ROA64 ohne RAM à DM 40,—; 2 x DRAM 128 voll bestückt à DM 300,—; 1 x RAM 64/256 mit 256KB bestückt zu DM 250,—; 1 x ROA64 unbestückt zu DM 20,—.

Andreas Krutof, August-Bebel-Str. 132,
2050 Hamburg 80, Tel.: (040) 7206196

NDR 68008: Grafik-Schaltzeichen DM 80,—.

Info: K. Hahn, Kreuzlach 19,
8806 Neuendettelsau

KONTAKTE

Suche Kontakt zu NDR-68008-Usern im Großraum Düsseldorf.

Telefon: (02151) 303676

Suche Kontakt zu NDR-68008-Usern im Raum Wiesbaden/Mainz.

Roland Steyer, Kappenbergweg 2,
6200 Wiesbaden-Bierstadt

Suche Kontakt im Raum Landshut zu Z80-Anwendern. – Verkaufe: SBC2 mit den Eproms Grundprogramm, Basic, Ampelsteuerung und Musik mit Ram.
Tel.: (08705) 868

Suche im Raum Pforzheim weitere NDR-Computer-Bauer und -Anwender.

Horst Klittich, Ersingerstraße 31,
7530 Pforzheim, Tel.: (07231) 42430

Suche NDR-Computer-Anwender der bereit ist, sein Wissen zu teilen – Raum (MG – VIE).

Herbert Hardt, Tel.: (02161) 51383

ges gmb, magnusstr. 13, 8960 kempten, tel. 0831/6211

an die
loop-redaktion kempten
telex-nr. 50680 - 19.06.1986

preissenkungen

an alle loop-leser

wir haben folgende ndr-computer-teile im preis gesenkt.

produkt	alt	neu
tastatur 1 cherry	195,--	148,--
tast gehaeuse cherry	32,--	29,--
tastatur 2 preh	395,--	295,--
roa 16 platine	20,--	15,--
roa 16 bausatz	59,95	45,--
pow 5v platine	20,--	15,--
pow 5v bausatz	59,--	35,--
pow 26/22 platine	20,--	10,--
pow 26/22 v.1 bausatz	49,50	29,50
pow 26/22 v.2 bausatz	159,--	89,--
laufwerk 3zoll eme 101	444,--	248,--
fischertechnik roboter baukasten	248,--	198,--
ergotilt monitoruntersatz	69,--	40,--
cpu68008 prozessor 8 mhz	-,--	39,--
gdp 9366 graphikprozessor	-,--	45,--
2143 - flol	-,--	35,--
251s2538 (fuer term 1 und gdp 64K)	-,--	5,90
z80 sti (fuer term 1)	-,--	24,--
sony-farbmonitor		
cpd 1301/e, 13zoll fuer alle graphikkarten	1.898,--	1.698,--
softvare fuer den halben preis		
egosi 68 - egosicomp		
epascal 68 - epascal je	185,--	92,50
fuer den mc-cp/m-computer		
flol platine	59,--	20,--
rflo-ramfloppy bausatz mc	389,--	289,--
ram 16 platine mit smp-bus	49,--	20,--

alle preise sind freibleibend und ab lager kempten

mfg
graf elektronik systeme gmbh

ps: natuerlich auch in unseren filialen erhaeltlich:
filiale muenchen, georgenstraße 61,8000 muenchen 40
filiale hamburg,ehrenbergstraße 56,2000 hamburg 50

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.

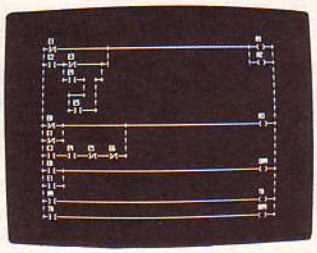
Nach diesem Kurs sind Sie in der Lage, eigene Programme zu schreiben und die Arbeitsweise des Z 80 zu verstehen.

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