

14

3. JAHRGANG

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter

DM 3,-

Das Microsoft-Basic- und -Assembler-Entwicklungspaket

Ein neuer Software-Knüller für CP/M 2.2-Anwender

Ein neues, im Preis fast unschlagbares Entwicklungspaket ist nun für den NDR-Computer mit Z80-Ausbau und CP/M 2.2 und für den mc-CP/M-Computer verfügbar. Es handelt sich um eine Kombination der bisherigen Einzelprodukte Microsoft-BASIC (MBASIC), den dazugehörigen Compiler BASCOM und den Macro-Assembler M80.

Die Einzelpreise der getrennten Produkte – die ab sofort einzeln nicht mehr lieferbar sind – lagen bei über DM 2000,-! Mit der Diskette wird die gesamte Original-Dokumentation im Ringbuch geliefert; etwa 1,25 Kilo dicht bedruckt. Die Beschreibung des Basic-Interpreters ist in deutsch; das zusätzliche Reference Manual in englisch, ebenso die Quick-Reference Guide und die Users Guide.

Der Basic-Compiler ist in englisch geschrieben, wie auch der Macro-Assembler und Linker. Mit im Lieferumfang sind MS-LINK, der Linking Loader, MS-LIB, der Library-Manager und MS-CREF, die Cross Reference Facility.

Der günstige Preis und die vollständige Dokumentation machen ein Raubkopieren dieser Produkte überflüssig. Der Anwender bleibt auf legaler Basis und hat die vollständige Dokumentation verfügbar.

Das Paket eignet sich besonders gut dazu, kommerzielle Software zu entwickeln. Viele Megabyte Programmcode Software sind schon in BASIC erstellt worden. Das BASIC-Programm wird compiliert und als übersetztes COM-File ausgeliefert. Der Anwender benötigt dann keinen Basic-Interpreter mehr, hat ein sehr schnell laufendes Programm. Der Programmierer behält seinen Quellcode; ein compiler-

tes Programm zu „knacken“ ist nahezu unmöglich.

Compiler und Linker können natürlich auch auf der Eprom-Floppy (mc-CP/M-Rechner) oder der RAM-Floppy (mc- und NDR-Rechner) ablaufen und sind so extrem schnell.

Mit diesem Produkt zeigt sich erneut: 8 bit Computer sind noch lange nicht tot!

Microsoft-BASIC-Interpreter und -Compiler

1975 entwickelte Microsoft den ersten BASIC-Interpreter für Mikrocomputer, heute ist Microsoft-BASIC auf mehreren Millionen Computersystemen installiert. Fast alle bekannten Mikrocomputer unterstützen Microsoft-BASIC. Es ist damit ganz eindeutig der Industriestandard.

Microsoft-BASIC ist eine der umfassendsten Implementationen der Sprache BASIC, die derzeit auf 8-Bit-Microcomputern zur Verfügung stehen. Sie entspricht den Anforderungen des ANSI-Standards für BASIC und unterstützt eine ganze Reihe von Eigenschaften, die bei anderen BASIC-Implementationen im allgemeinen nicht vorhanden sind, zum Beispiel leistungsfähige Programmfunktionen für die strukturierte Programmierung, die speziell für die Entwicklung von professionellen Anwendungsprogrammen konzipiert sind.

Microsoft-BASIC ist sowohl für Einsteiger als auch für erfahrene BASIC-Programmierer geeignet. Es ist einfach zu erlernen und anzuwenden, verfügt aber dennoch über alle Möglichkeiten, die für komplexere Programme benötigt werden.

Bei Microsoft-BASIC handelt es sich um

eine universell einsetzbare Programmiersprache. Das bedeutet: effizienter Einsatz in den verschiedensten Anwendungsbereichen, wie z. B. kaufmännische und technisch-wissenschaftliche Anwendungen, Schule und Ausbildung sowie Spielapplikationen.

Der Microsoft-BASIC-Interpreter ist interaktiv, der Benutzer kann damit Berechnungen durchführen und Programme austesten, ohne daß er dazu spezielle Programme schreiben muß. Der Microsoft-BASIC-Compiler setzt mit dem Inter-

Aus dem Inhalt:

- Neue Version RL-BASIC 2.4..... 14/4
- JADOS-Tool-Diskette ist da..... 14/6
- NDR-Computer mit IBM-Diskettenformat..... 14/6
- Vorankündigung: ROA 256/1M.. 14/7
- BIOS, BIOS 14/8
- Für Sie gelesen 14/9
- Software für die COL256..... 14/10
- CLUT – die Farbtabelle 14/11
- Quell-Code für HEBAS..... 14/14
- Motorensteuerung mit dem Einsteigerpaket..... 14/15
- Tips + Tricks in dBasell..... 14/16
- Routinen für Smart Watch 14/19
- Apfelmännchen mit der COL256 14/22
- Funktionenplot mit RL-BASIC .. 14/23
- Sonderzeichen auf der GDP ... 14/26
- Transfer C64 – NDR 14/29
- Selbstbau der 16bit-Erweiterung für die ACRTC..... 14/33

preter geschriebene Programme in drei- bis zehnmal schnelleren Maschinencode um (bei einigen Programmen sogar 30mal schneller). Der Compiler unterstützt alle Möglichkeiten des Interpreters, bis auf interpreterspezifische Befehle wie SAVE, EDIT, RENUM usw.

Interpreter und Compiler sind zusammen eine unschlagbare Kombination aus interaktiver Programmier- und Testumgebung und schneller Ausführung. Sie können Ihre Programme komfortabel und ohne Wartezeiten unter dem Interpreter entwickeln und sie dann kompilieren, um ein wesentlich schnelleres und kompakteres Maschinenprogramm daraus zu machen. Zusätzlich können Sie die kompilierten Programme vertreiben, ohne den Quellcode und damit Ihr Know-how aus der Hand zu geben.

Microsoft-Makroassembler und Utilities

Das Microsoft-Makroassembler-Paket ist ein flexibles und leistungsfähiges Assembler-Entwicklungssystem für die

Programmierung in 8080- und Z80-Assembler. Das Paket beinhaltet den Microsoft-Makroassembler M80, der relokierbaren Code erzeugt, den Linker L80, das Querverweisprogramm CREF80 und den Bibliotheksverwalter LIB.

Der Assembler erhöht durch seine ausgefeilten Makrofähigkeiten die Programmiereffektivität. Häufig benötigte Programmsequenzen können damit zusammengefaßt werden. Durch bedingte Kompilierung können verschiedene Programme aus einer Quelldatei entwickelt werden. Aus diesem Assembler entwickelte Routinen können mit anderen Microsoft-Sprachen, wie zum Beispiel dem BASIC-Compiler, verknüpft werden.

Die Dokumentation

Die Dokumentation umfaßt alle englischen Handbücher und zusätzlich ein umfangreiches deutsches Handbuch. Die gesamte Dokumentation hat einen Umfang von über 650 Seiten. Sie berücksichtigt sowohl die Bedürfnisse von Einsteigern als auch die von erfahrenen BASIC- und Assembler-Programmierern.

Der NDR-Klein-Computer – Zukunftsansichten

von Rolf-Dieter Klein

Damit der NDR-Computer nicht veraltet, ist es nun an der Zeit ein paar grundlegende Erweiterungen vorzunehmen. So wird es nötig sein, insbesondere für die 680xx Benutzer, die Bedienoberfläche und das Betriebssystem neu zu überarbeiten. Geplant ist eine Mausoberfläche, wie man sie von anderen 68000er Computern her kennt. Die läßt sich aber mit der bestehenden GDP-Baugruppe nur sehr mühsam erreichen. Daher soll die GDP langfristig durch eine neue Baugruppe ersetzt werden. Zunächst einmal wird die bestehende ACRTC-Baugruppe dafür verwendet werden. Die ACRTC bietet alles, was die GDP kann, und ist obendrein erheblich vielseitiger. So ist sie für Farbe zu gebrauchen, aber was viel wichtiger ist, man kann Daten aus dem Bildspeicher direkt rücklesen. Der ACRTC-Controller verfügt über die sogenannten Bitblit-Operationen: Er ist damit in der Lage, Flächen zu verschieben und zu bearbeiten, eine Fähigkeit, die man für eine Mausoberfläche benötigt. Geplant ist ferner ein Betriebssystem im Grundprogramm zu integrieren, das MS-DOS-kompatible Dateien lesen und schreiben kann.

Für die Grundkonfiguration braucht man dann also eine Maus-Baugruppe (HCOPY-MAUS) und die ACRTC-Basisbaugruppe.

Alle Grundprogramm-Traps werden erhalten bleiben und nur durch die neuen Maus-, Window- und DOS-Befehle ergänzt. Wer also ausschließlich die TRAPs des Grundprogramms verwendet, bekommt keine Probleme bei der Umstellung.

Gefährlich ist es, die GDP direkt anzusprechen, da bei der ACRTC zwar ähnliche Befehle vorhanden sind, doch die Art und Weise des Aufrufs sich sehr unterscheidet. So gibt es auch keine Kurzvektorbefehle mehr, sie werden einfach durch Linienbefehle ersetzt.

Da die ACRTC Linienbefehle richtungsunabhängig (einfach x-, y-Koordinate angeben) verarbeiten kann, sind solche Spezialbefehle nicht mehr nötig. Der entsprechende TRAP-Aufruf wird im Grundprogramm natürlich nachgebildet.

Weiter geplant ist es, für den NDR-Computer einen Logiksimulator anzubieten, der in der Lage ist, auch reale Schaltungen anzuschließen.

Die Eingabe erfolgt mit der Maus und auf grafischem Wege. Es wird versucht, hierfür noch die GDP zu verwenden, wobei man aber die HCOPY/MAUS-Baugruppe benötigt.

Unterstützt werden alle IO-Baugruppen, die man dann grafisch ansprechen kann. Dazu aber Näheres in einer der nächsten Ausgaben.

NDR-Computer steuert Betonmischanlage –

300 t Beton in 8 Stunden

von H. Hopp, Betonwerk Schröder

Hiermit möchte ich Ihnen kurz beschreiben, wie ich mit dem NDR-Computer eine Steuerung einer Mischanlage realisiert habe. Diese Anlage kann bis zu 300 t Beton in acht Stunden mischen. Bei dieser Anlage benötige ich 64 Eingänge und 48 Ausgänge. Da mir aber nur ein SPS-Programm mit 16 Ein- und Ausgängen zur Verfügung steht, habe ich 5 SPS-Programme geschrieben, in dem ich alle Sprungadressen und Speicheradressen geändert habe.

Weiter habe ich das Grundprogramm geändert, mit Speicheradressen ab Adresse F000. Somit ist es mir möglich, daß ich vom Grundprogramm aus jedes einzeln programmieren kann. Anschließend lasse ich sie zusammen laufen.

Ich benötige 64 Eingänge, da ich die einzelnen Programme beim Programm-schreiben nicht zusammenfügen konnte. Das heißt: IOE-Karte mit der Jumpadresse 20 Eingang 0, mit der IOE-Karte mit der Jumpadresse 40 Eingang 0.

So benötigte ich noch einige Eingänge mehr, die ich dann mit anderen Eingängen parallel geschaltet habe. So benötigte ich anstatt 64 Eingängen 80 Eingänge. Ein weiterer Nachteil ist, daß ich mir die Timerzeiten neu ausmessen mußte. Au-

ACRTC billiger

Die professionelle Graphik für den NDR-Computer nun zum erheblich günstigeren Preis.

Die ACRTC-Baugruppe mit dem modernen Hitachi-Graphik-Prozessor hat aufgrund ihrer enormen Leistungsfähigkeit schon viele Anwender gefunden. Sie wird derzeit vom ACRTC-Treiberprogramm, das wiederum in Assembler- oder Modula-Programme eingebunden wird, unterstützt.

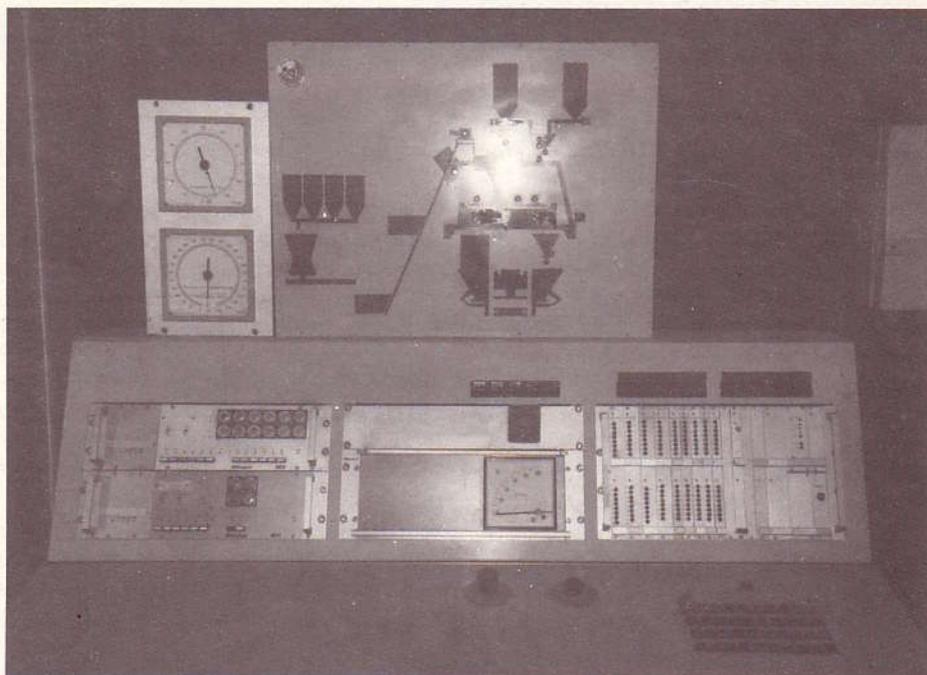
Wir haben nun den Preis dieser Baugruppe (nicht zuletzt aufgrund des günstigeren Preises für den ACRTC und die Speicher) auf fast die Hälfte senken können. Dies gilt sowohl für die Standard-ACRTC mit Schwarz-Weiß-Ausgang wie auch für die Farberweiterung.

Damit ist die ACRTC auch für den „professionellen Hobby-Anwender“ erschwinglich geworden! Zur Erinnerung: Die Baugruppe gibt es als leere Leiterplatte (für den Profi, da Multi-Layer) und als geprüftes Fertiggerät!

Berdem habe ich dann die fertigen Programme mit Hilfe des Grundprogrammes auf Kassette gespeichert. Das Ende eines einzelnen Programmes wird mit 80 beendet. Dann habe ich mit einem Zusatzprogramm die 5 Programme auf ein Eprom ab Adresse 0000 gebrannt, so daß beim Einschalten des Computers die einzelnen Programme in die richtigen Speicheradressen geladen werden.

Beim Systemaufbau habe ich die IOE-Karten auf Platinen gesetzt, so daß ich sie in ein 19" Rack einstecken kann. Ich habe mir selber einen neuen Bus mit fest vorgegebenen Jump-Adressen angefertigt.

So kann ich die einzelnen IOE-Karten wahlweise umstecken und erweitern. Die CPU und ROA 64 sitzen zusammen auf einer Platine. Weiter habe ich die GDP, KEY- und CAS-Karten auf eine Platine gesetzt. So besitze ich ein SPS-System mit vielen Vorzügen. Steht ein Programm, ziehe ich die Karte mit GDP, KEY und CAS heraus und das Programm läuft. Müssen Änderungen vorgenommen werden, nehme ich die Karte mit GDP, KEY und CAS, stecke sie in das System und ich kann in kürzester Zeit das Programm direkt am Einsatzort ändern und ausprobieren.



Ich hoffe, daß Ihnen mein Artikel gefällt und er interessant genug ist, um ihn in Ihrer LOOP-Zeitschrift zu veröffentlichen. Falls Sie noch Fragen haben sollten, ste-

he ich Ihnen jederzeit für Auskünfte zur Verfügung.

D. Hopp, Betonsteinwerk Schröder, Postfach 1769, 2240 Heide

Softwarepartner für IEEE-Schnittstelle gesucht!

Mittlerweile ist für den NDR-Computer eine IEEE-Schnittstelle auf dem Markt, allerdings ohne sinnvolle Anwendersoftware. Wir würden uns freuen, wenn ein

NDR-Anwender bereit wäre, Anwendersoftware für diese Schnittstelle zu schreiben.

Die Baugruppe basiert auf dem IEEE-

Schnittstellen-Prozessor uPD7210. Wenn Sie Interesse haben, erhalten Sie nähere Unterlagen.

Kontakt: GES, H. Bischof

In eigener Sache

Zunächst einmal Dank an all die vielen Autoren, die uns mit ihren sehr guten Artikeln geholfen haben, diese LOOP wieder interessant – hoffentlich für jeden Leser – zu machen!

Das Erscheinungsbild der LOOP hat sich ein klein wenig geändert. Wir haben die Preis- und Bestellinformationen von den Artikeln weggenommen und in eine GES-Anzeige gesetzt. Damit wollen wir eine klarere Trennung zwischen redaktionellem Teil und Anzeigen/Preisteil schaffen.

Für die Zukunft haben wir vor, auch andere Firmen neben den „bekannteren“ Partnern Christiani und Franzis-Verlag, zur Insertion in der LOOP zu gewinnen. Das klappt natürlich nur dann, wenn die LOOP

nicht ein „Graf Verkaufsblatt“ ist. Das ist sie sicher nicht; natürlich sind wir stolz, wenn wir Sie über neue oder verbilligte Produkte informieren können.

Der NDR-Computer und der mc-Computer wächst ja mit der Anzahl an Hardware, viel mehr aber mit der Breite der Software-Unterstützung. Hierzu zählt auch „Teachware“, also Papier zur Aus- und Weiterbildung. Davon stellen wir in dieser LOOP zwei interessante Bücher vor.

Das neue RL-BASIC erfüllt nun wohl fast alle Wünsche, die man an eine höhere Programmiersprache stellt. Durch die Strukturbefehle ist der Abstand zu PASCAL nicht mehr so groß, und durch die Kombination mit JADOS werden nun alle

CPU-Typen, natürlich auch der (billiger gewordene!) 68020 mit Floating Point unterstützt.

Diese „Software-Unterstützung“ haben wir uns für die Zukunft fest vorgenommen. Was nützt das beste Produkt, wenn es an der Software dazu mangelt! Abhilfe ist in Sicht: Für die (ebenfalls enorm verbilligte) COL 256 sind in dieser LOOP gleich zwei Artikel enthalten, die diese Baugruppe unterstützen, der oben erwähnte BASIC-Interpreter und die Treiberprogramme von Kei Thomsen.

Z80-Anwender, nicht verzweifeln! In Zusammenarbeit mit Markt und Technik können wir Ihnen auch für diesen Rechner – in dieser LOOP – einen Schlager

vorstellen: Das Microsoft-Entwicklungs-paket mit MBASIC-Interpreter und Compiler sowie M80 und Linker zu einem tollen Preis.

Nochmals zu den Anzeigen: Der redaktionelle Teil der LOOP wird natürlich den gleichen Umfang behalten; die Zeitschrift soll dicker werden, soll Sie über günstige Einkaufsmöglichkeiten informieren und soll uns die Möglichkeit geben, die LOOP monatlich erscheinen zu lassen.

Deshalb bitte, wie immer: LOOP weiterempfehlen; jedes neue Abo hilft!

Desktop Publishing hat auch bei uns Einzug gehalten. Die nächste Nummer wird, zumindest in Teilbereichen, mit dem PageMaker gesetzt werden. Dazu eine Bitte an alle Artikellieferanten:

Wir benötigen den Text auf Diskette, entweder im NDR-Format (bevorzugt NDR-Standard, 5 1/4", 80 Spuren) oder im IBM-Format (MCOPI). Alle Programme benötigen wir entweder auch auf Diskette oder, was die Angelegenheit einfacher und schneller macht, auf Papier, mit möglichst einem Farbband ausgedruckt. Bitte die Listings nicht breiter als ca. 60 Zeichen Normalschrift!

In letzter Zeit wurde der Wunsch an uns herangetragen, alle LOOP-Programme auch auf Diskette anzubieten oder diese Diskette der LOOP gleich beizulegen. Das mit dem Beilegen klappt nicht, da die Kosten für die Verpackung (und Diskette) die LOOP unnötig verteuern würde. Eine „LOOP-Diskette“, die man extra bestellen kann, wäre eher möglich; man könnte auch die Programme aus zwei LOOPS sammeln etc. Schreiben Sie uns Ihre Meinung!

Eine weitere Bitte von den Lesern, die wir hiermit an die Autoren weiterreichen: Alle

Artikel sollten mit der vollständigen Anschrift des Autors plus Telefonnummer gekennzeichnet sein. Wir würden folgenden Weg vorschlagen (Datenschutz!):

Falls wir von Ihnen als Autor nicht anders unterrichtet werden, geben wir die vollständige Adresse und, sofern uns mitgeteilt, die Telefonnummer im Artikel mit. Gleich eine Bitte an LOOP-Leser: Auch die Autoren haben ein Recht auf Privatleben. Selbst wenn das Programm gar nicht funktioniert: Ein Telefonat nach 20.00 Uhr oder am Wochenende sollte nicht unbedingt erfolgen . . .

Wir verstehen uns als Mittler zwischen Ihnen, den Lesern. Hierbei helfen uns auch die Adressen der Leser und die Möglichkeit, zwischen Lesern zu vermitteln. Sie haben uns ja während der letzten LOOP-Leseraktion geholfen, indem Sie zugestimmt haben, daß Ihre Adresse weitergegeben werden darf. Ab Ende des Jahres werden wir mit unserem Rechner so weit sein, daß wir auf Anfrage, in Ihrer unmittelbaren Nähe (sofern vorhanden), einen NDR- oder mc-Anwender nennen können. Warum erst ab Jahresende? Weil dann erst unser Netzwerk steht und der einzige Rechner, in dem diese Adressen derzeit gespeichert sind, rund um die Uhr ausgelastet ist.

Graf Elektronik und das Telefon

Vielleicht ist es Ihnen schon mal so gegangen: Sie riefen während der normalen Arbeitszeit unter der Nummer 6211 bei Graf Elektronik an, Freizeichen kam, aber niemand meldete sich.

Nun, wir sind dann nicht etwa beim Feiern oder Kaffeetrinken: Das Problem liegt in unserer Telefonanlage.

Unter der Nummer 6211 sind mehrere Amtsleitungen zusammengefaßt, die auf verschiedene Abfrage-Telefone aufgeschaltet werden. Sind nun alle Abfrage-Stationen besetzt, aber noch Amtsleitungen frei, so wird der Ruf zwischengespeichert. Es kann nun bis zu einer Minute dauern, bis auf eine weitere Abfragestation durchgeschaltet wird; dies ist für Anrufe außerhalb des Ortsnetzes einfach zu lange, sie erhalten den Besetztton.

Einzige Abhilfe: Nochmal probieren!

Hier nun unsere Telefonzeiten:

Mo, Di, Do	7.30 – 12.00 Uhr und 13.00 – 17.00 Uhr
Mi (Abendservice)	7.30 – 12.00 Uhr und 13.00 – 20.00 Uhr
Fr	7.30 – 12.00 Uhr und 13.30 – 16.00 Uhr

Technische Fragen stellen Sie bitte möglichst nur nachmittags, da hier die Berater verfügbar sind. Verlangen Sie gleich Herrn Köhler oder Herrn Pawlowitz.

Mittlerweile funktioniert unser Anrufbeantworter wieder einwandfrei; somit können Sie auch nachts anrufen und uns Nachrichten hinterlassen.

Unter der Nummer 0831-69330 erreichen Sie 24 Stunden unsere Datenbank, über die Sie auch Wünsche, Bestellungen oder Fragen einreichen können. Per Telex, merken Sie sich die Nummer 17 831 804=GRAF, per Teletex die 831 804=GRAF. Alles klar? . . . Dann auf Wiederhören!

Jetzt lieferbar - Jetzt lieferbar

Neue Version RL-BASIC 2.4

Verbesserte, erweiterte Version des RL-BASIC unter JADOS

Nunmehr ein Jahr ist RL-BASIC auf dem Markt und erfreut sich großer Beliebtheit. Doch es gibt nichts, was nicht noch besser werden könnte. RL-BASIC zeigt sich im neuen Gewand.

Zuerst für diejenigen unter Ihnen, die noch nicht wissen, was BASIC heißt, hier eine kurze Erläuterung: BASIC ist eine höhere Programmiersprache, die speziell für die Bedürfnisse von Anfängern entwickelt wurde, da sie sehr leicht zu erlernen ist.

BASIC hat im Laufe der Zeit einige Wandlungen durchgemacht, so daß heutzutage BASIC-Versionen auf dem Markt erhältlich sind, die sich in Punkte „strukturierter Programmierung“ hinter PASCAL nicht zu verstecken brauchen. Der allgemeine Vorzug von BASIC ist, daß es als Interpreter ausgelegt ist, d. h., der Benutzer kann interaktiv Befehle und Programme eingeben und erhält ohne Verzögerung (ohne zusätzlichen Übersetzungsschritt) ein Ergebnis.

Ein Beispiel hierzu bietet der PRINT-Befehl: Durch Eingabe von PRINT 237 * 3 (cr) erhält man sofort das Ergebnis 711, (Praktischer Tischrechner!) ohne vorhergehenden Übersetzungslauf, wie das z. B. bei PASCAL erforderlich wäre.

Man kann so Programme sofort starten und testen, nachdem eine Programmzeile eingegeben wurde. Bei Fehlern kann die entsprechende Programmzeile mit einem integrierten Editor (bei RL-BASIC Full-Screen-Editor) einfach geändert und das Programm sofort wieder gestartet werden.

Hierdurch ist eine schnellere Programm-entwicklung möglich, als dies durch Compilersprachen geschehen könnte.

Ein Nachteil, den jedoch alle Interpreter aufweisen, ist die geringere Arbeitsgeschwindigkeit, im Vergleich mit entsprechenden Übersetzern, mit denen Programme bearbeitet werden, da jeder Befehl zuerst in eine Reihe von Maschinen-

befehlen umgesetzt werden muß (Interpretiervorgang).

In der hier vorgestellten Version RL-BASIC 2.4 ist Geschwindigkeit jedoch kein Thema mehr. Programme unter RL-BASIC 2.4 sind nahezu so schnell wie Programme unter Turbo Pascal auf einem IBM/PC, auf alle Fälle jedoch schneller, als BASIC-Programme auf gängigen Heimcomputern. RL-BASIC wurde bereits in LOOP 8 vorgestellt, so daß ich hier nur auf die wesentlichen Neuerungen eingehen möchte.

1. wesentliche Neuerung:

Als Diskettenbetriebssystem liegt nun JADOS zu Grunde. RL-BASIC kann aber, wie die Vorgängerin auch, in EPROMs bezogen werden, ist somit unabhängig vom vorhandenen DOS einsetzbar.

JADOS wurde gewählt, da es sich als „das Standardbetriebssystem“ zum NDR-Computer herauskristallisiert hat und bereits andere Programme unter JADOS angeboten werden. Programme und Daten können auf Diskette abgelegt bzw. von dort geladen werden.

2. wesentliche Neuerung:

Dateien werden im Zusammenhang mit JADOS unterstützt. Es können bis zu 3 Diskettendateien und bis zu 4 Gerätedateien gleichzeitig bearbeitet werden. Als Gerätedateien stehen der Drucker (LST:), die serielle Schnittstelle (SER:), die Kassettenschnittstelle (CAS:) und der Bildschirm mit Tastatur (CON:) zur Verfügung, auf die geschrieben bzw. von denen gelesen werden kann. Eine Datei kann z. B. von der Diskette eingelesen und auf die serielle Schnittstelle (oder Drucker) ausgegeben werden. Dateien sind dabei nicht nur auf Text beschränkt. Es können auch einzelne Bytes verarbeitet werden, so daß z. B. einem Drucker Steuerzeichen oder Graphiken übermittelt werden können. Ein weiterer Vorzug des Dateikonzepts ist, daß RL-BASIC-Programme auch textuell auf Diskette abgelegt werden können, um sie eventuell mit einem Editor weiter zu bearbeiten. Auf diese Art und Weise ist es auch möglich, BASIC-Programme fremder Rechner per serieller Schnittstelle in den Rechner zu laden.

3. wesentliche Neuerung:

Einführung von speziellen Kontrollstrukturen zur strukturierten Programmierung. Ähnlich wie in PASCAL stehen in RL-BASIC alle wichtigen Kontrollstrukturen, die eine strukturierte Programmierung fördern sollen, dem Anwender zur Verfügung; hierzu zählen unter anderen: REPEAT .. UNTIL, WHILE .. WEND, FOR .. NEXT sowie IF .. ELSE .. END IF. Alle Konstrukte dürfen sich über mehr als eine Programmzeile erstrecken. Hier ein einfaches Beispiel:

```

10 REM Apfelmaennchen in RL-BASIC
20 REM R. Lobreyer 21.04.87
30 PRINT "Apfelmaennchen in RL-BASIC"
60 INPUT "Gebe den Bereich des Realteils ein (xa,xe)";xa,xe
70 INPUT "Gebe den Bereich des Imaginaerteils ein (ya,ye)";ya,ye
80 INPUT "Anzahl der Iterationen (i0..)";itmax%
90 PAGE 3,3: CLPG
100 inx = (xe - xa) / 512
110 iny = (ye - ya) / 512
120 FOR ix = xa TO xe STEP inx
130   FOR iy = ya TO ye STEP iny * 2
140     it% = 0
150     zr = 0
160     zi = 0
170     REPEAT
180       zrq = zr * zr
190       ziq = zi * zi
200       it% = it% + 1
210       zi = 2 * zr * zi + iy
220       zr = zrq - ziq + ix
230       UNTIL zrq + ziq > 4 or it% > itmax%
240       IF it% MOD 2 = 1 AND it% < > itmax% THEN
250         PSET INT ((ix - xa) / inx), INT ((iy - ya) / iny)
260       END IF
270     NEXT iy
280   NEXT ix
290 END

```

Zeichnet einen Ausschnitt aus der Mandelbrot-Menge („Apfelmännchen“). Damit die Programmstruktur sichtbar wird, wird beim Auslisten eines Programms, bzw. Bereichen davon, die Struktur durch Einrücken hervorgehoben. Dies sieht dann wie im obigen Beispiel aus!

Neu ist weiterhin, daß als Sprungzeile Marken, sog. Labels, verwendet werden dürfen. So kann man statt GOSUB 10000, GOSUB gibaus! eingeben, was sicherlich die Lesbarkeit des Programmes erhöhen dürfte. Variablenamen werden jetzt in allen Stellen unterschieden (früher maximal 5).

4. wesentliche Neuerung:

Die Arbeitsgeschwindigkeit hat sich erhöht. Im Vergleich zu RL-BASIC 2.0 sind nun rechenintensive Programme bis zu 3 mal schneller!

FOR i% = 1 to 10000 : a = i%/3 : NEXT benötigt z. B. **nur noch 17 Sekunden**. (Anstatt vormals 38 oder 55 Sekunden auf einem APPLE II unter APPLESOFT-BASIC, bzw. 52 Sekunden unter TURBO PASCAL und 62 Sekunden auf C64 unter Commodore BASIC.)

5. wesentliche Neuerung:

Die COL 256-Graphikkarte wird durch eine Reihe von BASIC-Befehlen unterstützt. Alle Graphikfunktionen, die bereits von RL-BASIC 2.0 bekannt sind, wie CIRCLE, ARC, LINE, CONNECT sowie Turtlegraphikbefehle und WRITE „.“, können ebenfalls aufgerufen werden. Selbst Schrift in verschiedenen Größen und Farben stellt dank des in RL-BASIC integrierten Zeichensatzgenerators kein

Problem dar. Neu gegenüber der GDP 64K-Graphik ist hier:

- Zeichnen mit 256 verschiedenen Farb-/Grautönen.
- Ausfüllen von beliebig geformten Flächen.
- Zurücklesen der Grauwerte/Farbwerte einzelner Graphikpunkte.

6. wesentliche Neuerung:

Folgende Baugruppen können unter RL-BASIC 2.4 mit eigenen Kommandos angesprochen werden:

Hardcopy/Maus, Serielle Schnittstelle, AD 8 * 16 Karte, AD 10 Karte, Sound-Baugruppe und die Uhrenbaugruppe.

So können mit den Funktionen MOUSEX, MOUSEY, MOUSEK ganz einfach die Mauskoordinaten und Tasten abgefragt werden, oder mit HARDCOPY ein Ausdruck des aktuellen Bildschirms, auch innerhalb eines Programms, gemacht werden. Mit TIMES\$ kann die Uhrzeit gesetzt bzw. zurückgelesen werden.

7. wesentliche Neuerung:

Alle Prozessoren der 68000er Familie werden unterstützt. Neu ist dabei, daß jetzt erstmals eine Version des Interpreters mit ausgeliefert wird, der den Prozessor 68020 mit FPU 68881 unterstützt. Insbesondere in Verbindung mit dem Gleitkommaprozessor 68881 läßt RL-BASIC in Punkto Rechengenauigkeit kaum noch Wünsche offen.: Alle Berechnungen werden intern 80 Bit breit ausgeführt, dies entspricht etwa der Genauigkeit von 16 Dezimalstellen!

Bleibt zum Schluß noch am Rande zu erwähnen, daß RL-BASIC 2.4 einen bild-

schirmorientierten Editor enthält, mit dem auf dem Bildschirm Programmzeilen kinderleicht eingegeben und abgeändert werden können, RL-BASIC die Graphikfähigkeiten des NDR-Klein-Computers

nutzt, im Lieferumfang ein ca. 160 seitiges, ausführliches, deutsches Handbuch enthalten ist und, daß sich alle Versionen des Interpreters (1 für 68008/68000/68020 + 1 für 68020 + FPU) auf

der ausgelieferten Diskette befinden. Und dies alles zu einem fairen Preis!

Preise und Lieferformen: Siehe unsere Anzeige auf der letzten Umschlagseite.

Endlich:

JADOS-TOOL-Diskette ist da!

Dank einer großen Zusendung von Anwenderprogrammen unter dem Betriebssystem JADOS für die CPUs der 680xx-Serie können wir Ihnen eine Programmsammlung von 76 ausgewählten Routinen bzw. Dokumentationen anbieten.

Zum Betrieb der Diskette wird JADOS benötigt, ein Teil der Programme verlangt RL-BASIC (.BAS-Files) oder PASCAL/S (.PAS-Files).

Die Programme dienen teilweise als Werkzeug im Umgang mit JADOS, teilweise nur als Anwenderprogramme mit Lerneffekt oder spielerischem Charakter. Hier eine Kurzbeschreibung für **einen Teil** der Programme:

● Aufruf des Editors mit automatischem Backup des Textes

– Jeder Text wird vor Bearbeitung als .BAK-File abgespeichert

● Komfortable Diskettenverwaltung unter JADOS

– Dateien selektieren
– Selektierte Dateien kopieren
– Selektierte Dateien löschen
– Dateien einzeln löschen oder umbenennen

● Ausdruck eines Text-Files mit Extras
– Automatischer LF nach 65 Zeilen
– Fehlerbehandlung

● Besserer Ausdruck eines Assemblerlistings

– Schmalschrift und amerikanischer Zeichensatz
– Zeit und Daten vom System für den Kopf

● Schnelle Eprom-Programmerroutine

– Eproms (2764, 2732 und 2716) schreiben und überprüfen
– momentan nur für CPU 68008

● Breakout-Spiel (extrem schnell)
– alle CPUs der 680xx-Serie
– Individuelle Einstellung
– Laden und Speichern unter JADOS
– High-Score

vom „Spiele-Spezialisten“ Ralf Dombrowski!

● Generationen-Wachstums-Spiel (LIFE)

– Wie alle Programme mit Source

● Mathematische Matrizen-Berechnung

– Für Insider
– Umfangreiches Programm in Assemblersprache

● Erzeugung von Lissajous-Figuren auf dem Bildschirm

– Lissajous-Figuren manuell in Amplitude, Frequenz und Phasenlage veränderbar
– Sehr lehrreich

● Bedienungsoberfläche für GOSI unter JADOS

– Bedienung des Editors
– Programm compilieren und starten
– Programm laden und speichern unter JADOS
– Inhaltsverzeichnis

● Grafische Darstellungen von mathematischen Funktionen

– Zeichnet beliebige mathematische Funktion
– Programm in BASIC und Assemblersprache

● FOURIER-Entwicklung am Bildschirm

– Fourierentwicklung für Sägezahn, Zickzack und Rechteck
– Sehr lehrreich

● Smartwatch-Uhr unter JADOS

– Uhr kann unter JADOS gesetzt und gelesen werden
– Uhrrountinen können in eigene Programme übernommen werden

● Kopierprogramm für JADOS

– Komfortables Kopierprogramm
– Bereits kopierte Programme werden durchgestrichen

● Hauptmenue (für PASCAL/S) ähnlich wie bei TURBO-Pascal

– RDK-Editor
– Laden und Speichern des Programmes
– Directory und Optionen
– Compilieren und Assemblieren

● Betrieb des EGRUND 0-3 auf der BANKBOOT-Baugruppe

– FLOBOOT-Programm aus LOOP 8/9 Seite 24 wird mit ins Grundprogramm EASS 0-3 übernommen, um es auf der BANKBOOT zu betreiben

– Grundprogramm wird automatisch auf Adresse \$E0000 in RAM kopiert

● 3-Dimensionales LABYRINTH-Spiel
– Für alle 680xx-CPU's
– Mit Joy-Stick Betrieb

● AD-Wandler-Programm
– Digitales Speicheroszilloskop
– Werte werden von AD10/1 übernommen

– Betrieb unter JADOS
– Für alle CPUs der 680xx-Serie

Beim Durchforschen der JADOS-TOOL-Diskette können Sie noch viele andere interessante Programme entdecken.

NDR-Computer mit IBM-Diskettenformat

Programm MCOPY ermöglicht das Lesen und Schreiben von (Standard, 360 KByte) IBM-Disketten mit dem NDR-Computer!

Benötigter Ausbau:

NDR-Computer, 680xx-CPU, JADOS oder CP/M68k. Ein Laufwerk mit 40 Spuren (oder ein umschaltbares 80 Spur/40 Spur)

Nachdem mit der in LOOP 13 vorgestellten Koppelung nun auch der IBM-Computer beim NDR-Rechner eingezogen ist, hier nun die erste, von vielen NDR-Benutzern lang erwartete Software-Lösung.

Sehr häufig wurde der Wunsch geäußert, IBM-Disketten auch vom NDR-Computer zu lesen oder zu beschreiben. Texte, die unter dem NDR-Rechner vielleicht zu Hause erstellt wurden, sollten im Betrieb auf einem IBM weiterverarbeitet werden. Programme in einer Hochsprache, z. B. Turbo Pascal oder BASIC sollen vom IBM auf den NDR-Rechner übernommen werden.

Dies ist nun alles möglich mit MCOPY, einer reinen Software-Lösung (zum sehr günstigen Preis!).

Es ist nur sinnvoll, lesbare Text-(ASCII-Dateien) zu übernehmen.

Die Übernahme von Maschinenprogrammen ist (leider) nicht möglich. Warum? Nun, diese Programme sind in einer Maschinensprache (darum heißen sie auch so...) geschrieben; die „Maschine“ des IBM-Rechners ist die CPU 8088 oder die dazu kompatible CPU 8086 oder 80286 (80386).

Da der NDR-Computer diese CPUs nicht unterstützt, laufen auch die Maschinenprogramme nicht.

Ein Programm einer Hochsprache (z. B. TURBO PASCAL) wird vom Compiler (dem „Turbo Pascal“) auf die jeweilige Maschinensprache übersetzt. Das Programm wird in der Quelle übernommen und vom Turbo Pascal, das auf dem NDR-Rechner läuft, neu (für den Z80, hoffentlich bald für den 680xx . .) übersetzt.

Interessant ist sicher auch die Übernahme von Daten, die z. B. unter dBasell auf dem NDR-Computer erfaßt worden sind. DBase bietet die Möglichkeit, Datensätze zur Übernahme in ASCII-Dateien zu schreiben (COPY-Befehl) und aus solchen Datensätzen zu lesen. Damit ist die Übergabe, z. B. zum IBM, mit dBasell oder Clipper problemlos möglich.

MCOPY ist menügeführt und sehr leicht zu bedienen. Beim Betrieb auf der RAM-Floppy geht der Kopiervorgang ungefähr so schnell wie bei einem MS-DOS-Rechner. Das Kopieren von Laufwerk zu Laufwerk geschieht clusterweise und ist relativ langsam.

Wie man das „Standard“ 80 Spur Laufwerk TEAC FD55F auf 40 Spuren umschalten kann, wird in TIPS und TRICKS beschrieben. Hierbei Achtung: Disketten bitte nur mit dem IBM formatieren!

Was kann MCOPY:

- Lesen von einer MS-DOS-Diskette
- Schreiben auf eine MS-DOS-Diskette
- Initialisierung einer MS-DOS-Diskette
- Kopieren MS-DOS zu CP/M 68k oder JADOS
- Kopieren CP/M 68k oder JADOS zu MS-DOS
- Zeichensatzanpassung IBM/NDR

Derzeit im Test: Tast3

Der Nachfolger der Tastatur 2



Durch den günstigen Preis der TAST 2 im Sonderangebot ist diese Tastatur nun ausverkauft und nicht mehr lieferbar.

Wir haben derzeit einen Nachfolgetyp im Test. Diese Tastatur wird in der gleichen Preisklasse wie die „alte“ TAST 2 liegen (ca. DM 220,-), hat jedoch wesentliche Vorteile. Der Aufbau ist ergonomischer,

und für den Anwender ist diese Tastatur bequemer. Zeichensätze z. B. lassen sich über das eingebaute EPROM leicht ändern.

Nach Abschluß des Tests und der Vertriebsfreigabe folgt eine detaillierte Beschreibung!

Vorankündigung:

Die ROA 256/1M

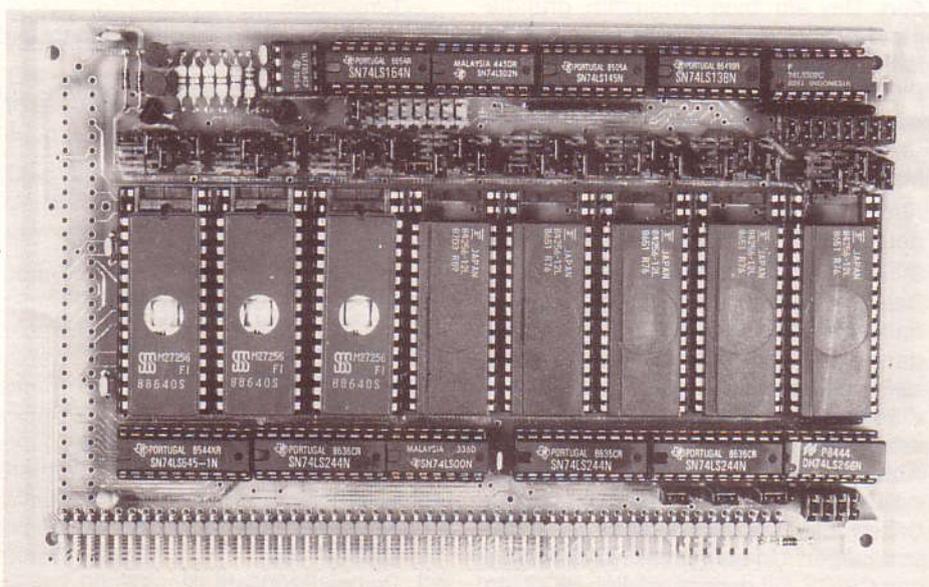
256 KByte statischer Speicher. Gemischt (EPROM/RAM) bestückbar. Batterie optional. NDR- und ECB-BUS. Lieferbar ab September 87.

Die ROA 256 ist eine von vielen NDR- und mc-CP/M-Anwendern gewünschte Baugruppe. Sie befindet sich derzeit in der Freigabe und wird ab September lieferbar sein.

Eingesetzt werden die jetzt relativ günstig erhältlichen statischen RAM-Speicher, die 32 K x 8 organisiert sind. Acht dieser Speicher finden auf der Baugruppe Platz. Die Beschaltung ist jedoch bereits für die 128 K x 8 organisierten Speicher ausgelegt; damit stellt die Baugruppe dann einen statischen Speicher von 1 MByte zur Verfügung.

Die 128 K x 8 organisierten Eproms sind schon, die statischen Speicher derzeit im Muster, ab 1988 wohl auch in Stückzahlen lieferbar.

Über Jumper (das sind steckbare Brücken, im Prinzip ein „billiger“ Schalter) kann zunächst einmal für jeden Speicher-Steckplatz individuell eingestellt werden, ob er mit einem EPROM oder einem RAM bestückt ist. Dies ist für die „Notstrom“versorgung wichtig, da ja EPROMs nicht mit Batteriespannung versorgt werden müssen. Über weitere Jumper wird die unterschiedliche Pinbele-



gung der verschiedenen Speichertypen angeleglichen. Über weitere individuelle Jumper kann jeder Speicherplatz getrennt ausgeblendet werden und von anderen, schon vorhandenen Baugruppen (z.B. ROA64) genutzt werden.

Weiter existieren „globale“, d. h. die gesamte Baugruppe betreffende Jumper. Natürlich kann der Adressbereich, in der die Baugruppe liegt, eingestellt werden. Weiter hat die ROA 256 eine integrierte WAIT-Erzeugung; sie kann zwischen 0

und 7 WAIT-Zyklen erzeugen und ist dadurch mit Speichern unterschiedlicher Zugriffsgeschwindigkeit bestückbar.

Optional kann auf der Baugruppe eine Batterie-Pufferung, verbunden mit einem elektronischen Spannungswächter untergebracht werden, die den Speicherinhalt auch nach dem Ausschalten puffert (wie bei der SBC3). Damit läßt sich die ROA 256 auch sehr gut als RAM-Floppy einsetzen; der Speicherinhalt kann dann eigentlich nur noch durch ein wild gewordenes Programm zerstört werden.

ROA 256 entspricht der GES-Norm; sie kann also mit dem NDR-BUS oder mit dem ECB-BUS ausgestattet werden. Durch die Möglichkeit, später auch größere Speicher einsetzen zu können, halten wir die ROA 256/1M (das 1 M steht für 1 MByte) für die Standard-Baugruppe der Zukunft.

Sie wird ab September lieferbar sein; die ROA 64 wird natürlich weiter im Programm bleiben!

Neu: Gehäuse 4 für den mc-CP/M-Computer

(und andere ECB- oder Europakarten-Systeme)

Das neue Gehäuse entspricht im Aussehen dem (IBM-ähnlichen) Gehäuse 2 für den NDR-Computer. Die Europakarten werden in einen 19" Einschub mit 10 Einschubplätzen untergebracht. Der Klappdeckel des Gehäuses ist nach hinten zu öffnen.

Das Gehäuse ist zum Einbau eines NE4, des IBM-kompatiblen Netzteiles, vorbereitet. Vorne können zwei oder vier Laufwerke 5 1/4", Slim-Line eingesetzt werden. Über ein Modul werden 3 1/2"-Laufwerke eingebaut (Zubehör).

Das Gehäuse wird komplett fertig mit Baugruppenträger, jedoch ohne Bus, Netzteil und Verkabelung geliefert. Eine komplett aufgebaute Version ist jedoch auch erhältlich.

CP/M 2.2 BIOS auf Diskette – BIOS mit Autostart-Möglichkeit

Auf der neuen BIOS-Diskette, die in allen unterstützten Formaten von 3 1/2" bis 8" lieferbar ist, befindet sich erst mal das Original-BIOS des „Standard“ CP/M 2.2. Das BIOS kann nur mit dem M80 Makro-Assembler übersetzt werden. Damit auch „nicht M80“-Besitzer zumindest das BIOS listen und damit patchen können, ist neben dem .ASM-File auch ein .HEX-File (im INTEL-HEX-Format) sowie ein .PRN-File (als lesbare Druck-Datei) untergebracht.

Weiter befindet sich auf der Diskette ein BIOS mit Autostart. Über ein Programm SETAUTO.COM (auch mit drauf) kann ein beliebiges Programm über eine SUBMIT-Datei gestartet werden. Siehe dazu auch den Autostart-Artikel in LOOP1 (als LOOP 11.TXT ebenfalls auf der DISC).

Weiter ist ein BIOS zum Betrieb des CP/M 2.2 mit einer Festplatte auf der Diskette

verfügbar. Formatierer und Installationsprogramm runden die Disk ab.

Ein wichtiges Werkzeug für alle Anwender, die ein eigenes BIOS schreiben wollen!

Für den mc-CP/M-Computer wird nun auch eine EPROM-Floppy (siehe LOOP 13a) sowie eine RAM-Floppy angeboten. Dafür ist ebenfalls eine eigene BIOS-Diskette verfügbar, die die Quellen des so angepaßten BIOS enthält.

BIOS PATCH Programm für NDR- und mc-CP/M- Computer

Laufwerksanpassung leicht gemacht

Das BIOS (BASIC I/O-System) ist der hardwareabhängige Teil des Betriebssystems CP/M 2.2 zum Rechner. Hierin werden besonders die bezeichnenden Werte für die Diskettenlaufwerke eingetragen; kurz, hier wird CP/M darüber informiert, wie viele Laufwerke es gibt, wie viele Spuren ein Laufwerk hat und wie viel Sektoren pro Spur untergebracht sind.

Die Zuweisung erfolgt über eine Anzahl von Tabellen, die wiederum von Makro-Befehlen (DISKDEF-Makro) erzeugt werden. Um ein Bios-Programm grundlegend zu ändern (die Quellen sind auf Diskette verfügbar, siehe Artikel „CP/M 2.2 BIOS auf Diskette in dieser LOOP), war bisher ein hoher Kenntnisstand über die „Innereien“ des Betriebssystems und ein Makro-Assembler M80 nötig.

Auch der „Austausch“ des alten zum neuen CBIOS war besonders für Einsteiger nicht ganz unproblematisch.

Das vorliegende Programmpaket beinhaltet ein spezielles CBIOS für die beiden genannten Computer, das bei der Bearbeitung eines fremden Diskettenformats automatisch geladen wird. Dieses CBIOS erlaubt es, fremde Diskettenformate in Abhängigkeit bestimmter Parameter ohne umfangreiche Änderungen zu bearbeiten. Ein Pascal-Programm dient zum vorläufigen Überschreiben des bisherigen CBIOS mit dem speziellen BIOS und

zur Erzeugung der nötigen Parameter. Dabei erzeugt das Programm nicht nur die Parameter für das spezielle CBIOS, sondern auch alle CP/M-Tabellen für alle vereinbarten Laufwerke. Das spezielle CBIOS mit den geänderten Kenndaten der Laufwerke bleibt bis zu einem Reset gültig.

Im Gegensatz zur ersten Version des Programmpakets gestattet diese Version die Speicherung der Kenndaten fremder Diskettenformate in einer Datei und ermöglicht damit die rasche Einstellung eines Fremdformats. Ebenso reduziert das Programm die Eingaben auf das absolute Minimum und verzichtet dabei auf spezielles Wissen des Benutzers über die Ansteuerung der Laufwerke. Weiterhin bestimmt das Programm automatisch die Charakteristik des Boot-Laufwerks. Dadurch gibt es auf der Diskette nicht mehr wie bisher drei Versionen des Programmes, sondern nur noch eine Version.

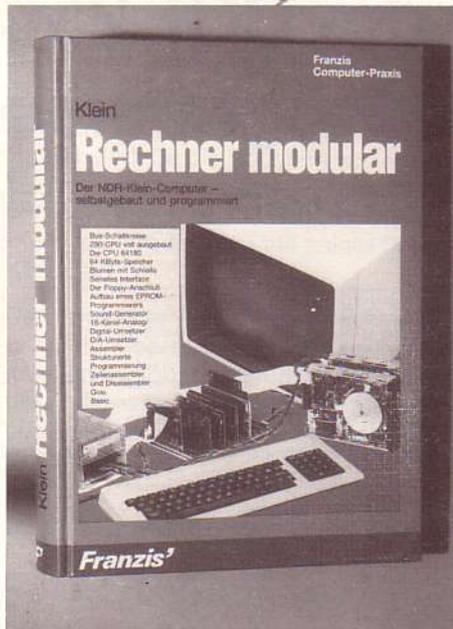
Folgende Programmfunktionen stehen zur Verfügung:

- (1) Bearbeiten von Fremddisketten
- (2) Definition eines neuen Diskettenformats
- (3) Änderung eines bestehenden Diskettenformats
- (4) Löschen eines bestehenden Diskettenformats
- (5) Ausgabe aller Diskettenformate auf Drucker
- (6) Ausgabe aller Diskettenformate am Bildschirm
- (7) Ende des Programms

Kennzahl der gewünschten Programmfunktion ? 6

Gespeicherte Diskettenformate:

- | | |
|-----------------------|---------------------|
| 1. mc-CP/M-Computer | SS 40 Tracks |
| 2. mc-CP/M-Computer | SS 77 Tracks |
| 3. mc-CP/M-Computer | DS 80 Tracks |
| 4. NDR-Klein-Computer | SS 77 Tracks |
| 5. NDR-Klein-Computer | DS 80 Tracks |
| 6. Osborne SD | SS 40 Tracks |
| 7. Osborne DD | SS 40 Tracks |
| 8. Televideo | (Handbuch beachten) |
| 9. QX-10 | DS 40 Tracks |
| 10. Kaypro II | SS 40 Tracks |
| 11. DEC VT-180 | SS 40 Tracks |
| 12. Philips P2000-12 | DS 80 Tracks |



Das neue Buch von Rolf-Dieter Klein für alle Z80 Anwender

Bestell-Nr. 10 991

DM 68,-

Die lang erwartete Neuauflage des NDR-„Standard“-Werkes von Rolf-Dieter Klein ist nun da. Das Buch heißt „Rechner modular“ und beinhaltet die wesentlichsten Teile des alten Buches sowie der beiden Sonderhefte zum NDR-Computer.

Dieses Buch ist ideal für den Einsteiger mit dem NDR-Computer geeignet. Besonders für Anfänger ist der erste Teil, der in die Digitaltechnik einführt, hervor-

Rechner modular

gend geeignet. In weiteren Kapiteln wird der Z80 grundlegend und ausführlich erläutert; danach folgt eine Einführung und die Beschreibung der GDP 64k. Im Kapitel 6 wird das Z80-Grundprogramm und das GOSI vorgestellt und die Bedienung an vielen Beispielen gezeigt.

Besonders die Programme, die in der Fernsehserie gezeigt wurden, können hier noch einmal in Ruhe durchgearbeitet werden.

Kapitel 7 beschäftigt sich mit der Peripherie; von IOE über CAS, SER über FLO zum EPROM-Programmierer, SOUND, A/D und D/A.

Kapitel 8 heißt schlicht „Software“; sehr ausführlich und mit vielen Beispielen wird die Z80 Software erläutert. Dies ist bei komplexeren Abläufen, wie Interrupt-Verarbeitung oder Blocktransfer sicher sehr von Vorteil.

Rolf-Dieter Klein bleibt nicht nur bei den Befehlen, sondern zeigt auch Beispiele der Assembler-Übersetzung von Hand mit einem Assembler, erklärt strukturierte Programmierung und bedingte Assemblierung.

Weiter werden Anwendungsbeispiele des Grundprogramms dargestellt; hier sieht man auch mal die inverse Darstellung mit der GDP 64k sowie ein bewegtes Dreieck.

GOSI wird mit vielen Beispielen gezeigt, anschließend von Dr. Hans Hehl das kleine 8 KByte BASIC mit einer ausführlichen Befehlsübersicht. Kleine Graphik-Programme, in BASIC geschrieben, runden dieses Kapitel ab.

Einen breiten Raum nimmt FLOMON, das Monitor-Programm für den CP/M-Ausbau, ein. Vielen Benutzern ist noch nicht klar, wie unter FLOMON der Graphik-Modus aufgerufen werden kann. RDK beschreibt hier schon die neuen Funktionen des FLOMON ab Version 4, die auch externe Terminals als Ein-/Ausgaben zuläßt.

Nach FLOMON wird das Standard-BIOS für den NDR-Rechner mit FLO2 beschrieben; ein BIOS-Listing ist abgedruckt. Den Abschluß bildet als Beispiel das komplette SCOP-Programm zum Abgleich der CAS-Baugruppe.

Das Buch ist nicht nur für Einsteiger, sondern sicher auch für Z80-Profis eine Fundgrube. Buch und Handbücher der Baugruppen ergänzen sich; im Buch können natürlich nicht die aktuellsten Versionen wie im Handbuch beschrieben sein, und in den Handbüchern fehlt die Hintergrundinformation, die das Buch auf 423 Seiten bietet.

Für Sie gelesen:

Helmut Ostermann
Maschinen- und Assemblersprache des MC 68000 – Eine Einführung mit vielen Beispielen für Amiga, NDR-Klein-Computer und Sinclair QL

CHIP-Wissen, Vogel-Verlag, Würzburg
 ISBN 3-8023-0174-9

Bestell-Nr. (GES) 11 003

DM 48,-

Dies ist ein Buch, auf das viele Einsteiger der 680xx-Linie (und viele „Umsteiger“) gewartet haben.

Der Autor setzt Grundkenntnisse der Assemblerprogrammierung voraus, um „... der Mehrzahl der Leser ein zügiges Vorwärtkommen zu ermöglichen.“ Durch Anhänge und Erklärungen kommen aber auch Einsteiger mit diesem Buch sehr gut zurecht. Die Beispiele werden meist für die drei im Untertitel erwähnten Rechnertypen – also auch den NDR-Computer – in der Praxis ausgetestet.

Zitieren wollen wir aus dem Kapitel „Zur

Rechnerauswahl“ den Punkt d) auf Seite 14, der sich auf den NDR-Computer bezieht:

„Selbstbaucomputer. Der Verfasser behauptet nicht, daß ihm der Bau ohne alle Probleme gelang. Er lernte dann aber im NDR-Computer einen Rechner kennen, der auf Prozessorebene mit beachtlichem Komfort große Datenmengen verwalten kann; das Ergebnis kam den Idealvorstellungen sehr nahe.“

Das tut uns natürlich besonders gut! Interessant ist auch, was der Autor über Computer mit graphikorientierten Betriebssystemen schreibt:

„Computer mit graphikorientierten Betriebssystemen (Benutzeroberfläche) sind für unser Vorhaben nicht besonders gut geeignet: Das Betriebssystem ist undurchsichtig, die Dokumentation auf der Prozessorebene oft dürrtig bis schlecht – oder teuer und hypertroph (man suche die Nadel im Heuhaufen); ...“

Nun zum Buch: Es beginnt mit einer Übersicht über die 68000-Familie, geht nach einer Einführung in die Adressierungsarten sehr schnell über zum speicherplatzabhängigen Programmieren (relativ Programme, ein Steckenpferd von RDK!), und erläutert mit vielen, leicht nachzuvollziehenden Beispielen die Befehle.

Nun gehts vom Condition Code Register über Schleifen zur Zeichenausgabe, Rotieren und Shiften, logischen Verknüpfungen bis zu anspruchsvolleren Programmen. Das Sieb des Eratosthenes wird ebenso gezeigt wie Bubble-Sort-Programme oder die Türme von Hanoi. Wichtig: Alle Programme sind im Quellcode (direkt vom NDR-Computer erstellt) im Buch abgedruckt und können direkt übernommen werden!

Den Ausblick bietet eine Raummühle für den NDR-Computer; das Listing dafür beträgt allein 27 Seiten!

Zusammenfassend: Das ideale Software-Buch für den 68000, das wir uns schon vor drei Jahren gewünscht hätten – jetzt ist es da!

Software-Unterstützung für die COL256 Universeller Treiber für 680xx-Programme

Die COL 256-Farbbaugruppe wird nun von zwei starken Software-Produkten unterstützt: Dem neuen RL-BASIC und dem COL 256-Treiber, den wir hier vorstellen wollen.

Der Treiber wurde bewußt auf Maschinenebene der 680xx-Prozessoren geschrieben, um auch hier den Benutzern die Möglichkeit zu geben, sehr schnelle Programme auf Basis dieser schnellen Baugruppe zu schreiben.

Er stammt von Kei Thomsen aus Hamburg, dessen Demos mit der COL 256 (Fraktal-Berechnung in Echtzeit und Farbe, springender Ball etc.) schon manchen Messebesucher auf der Hannover Messe zum ungläubigen Staunen gebracht haben.

Erstmals hat Kei auch ein sehr gutes Handbuch dazu geschrieben. Die kleinen Probleme mit der Orthographie werden durch klare und einfache Erläuterungen mehr als wettgemacht.

Was ist denn überhaupt ein „Treiber“? Kurz gesagt handelt es sich hierbei um eine sehr große Anzahl von Unterprogrammen. Jeder Programmierer, der ordentlich und modular programmiert, hat sich schon selbst eine kleine oder große-

C O L 2 5 6 Farbgrafik		Seite:16
SCROLLX Scrollen des Bildschirms in X-Richtung.		
<p>Mit diesem Befehl ist ein Scrollen des Bildschirms in X-Richtung möglich, ohne den Speicher verschieben zu müssen. Dadurch ist ein Scrollen nebenbei sehr einfach und es lassen sich Spiele recht schnell bewegen. Es müssen aber 2 Einschränkungen gemacht werden.</p> <p>1. Der Bildschirm lässt sich immer nur um 4 Bildpunkte verschieben, wodurch sich ein extrem feines Scrollen nur durch Speicherverschieben realisieren lässt. Das wird aber dadurch kompensiert, dass der Scrollbefehl sich die Anzahl der noch nicht gescrollten Punkte merkt und dann später ausführt, wenn wieder 4 Punkte zusammen sind.</p> <p>2. Zwischen dem linken und dem rechten Rand des Originalbild ergibt sich beim Scrollen ein versatz um 4 Bildpunkte nach unten. Wieder ein Problem welches sich nur durch Speicherverschieben realisieren lässt.</p>		
D0	Anzahl Bildpunkte um die gescrollt werden soll (Eine negative Zahl ergibt scrollen nach rechts)	
<p>START:</p> <pre> MOVE.W #0,D0 † Auflösung 256 † 256 JSR COLINIT MOVE.W #0,D1 MOVE.W #0,D2 JSR MOVETO † Setzen des Anfangspunktes auf 0,0 MOVE.B #03,D0 MOVE.W #255,D1 MOVE.W #255,D2 JSR DRAWTO † Linie ziehen LOOP: BSR SYNC † Vertikalen Synchron abwarten da sonst BEQ.S SYNC † zu schnell. MOVE.W #4,D0 JSR SCROLLX † Scrollen um je 4 Punkte BRA LOOP † Endlos wiederholen </pre>		

C O L 2 5 6 Farbgrafik		Seite:6
SETFDOT Setzen eines Punktes.		
<p>Dieser Befehl setzt einen Punkt in der Farbe D0 an die X und Y Position. Ein Punkt wird extrem schnell gesetzt, wodurch sich Bilder schnell aufbauen lassen. Als Ausgabe wird die Punktdresse in A0 geliefert. Dadurch lassen sich Punkte wiederholt sehr schnell setzen.</p>		
D0	Farbe des Punktes	>A0 Adresse des Punktes
D1	X-Position	
D2	Y-Position	
<p>START:</p> <pre> MOVE.W #0,D0 † Auflösung 256 † 256 JSR COLINIT LOOP: MOVE.W #255,D0 JSR @RND † Zufallszahl 0-255 MOVE.W D0,D1 † X-Position MOVE.W #255,D0 JSR @RND † Zufallszahl 0-255 MOVE.W D0,D2 † Y-Position MOVE.W #255,D0 JSR @RND † Farbe JSR SETFDOT † Punkt setzen BRA LOOP † Endlos wiederholen </pre>		

C O L 2 5 6 Farbgrafik		Seite:58
GETWINDOW Lesen eines Window vom Bildschirm.		
<p>Der GETWINDOW Befehl liest ein Window beliebiger Größe aus dem Bildschirmspeicher. Die X und Y Position wird in D1 und D2 bestimmt, die Größe in X und Y Richtung wird in D3 und D4 bestimmt und die Adresse des Windowspeichers, wo das Bild abgelegt wird, wird in A1 übergeben. Hierbei wird also der Bildschirminhalt ausgelesen und in den Windowspeicher geschrieben. Dieses ist sinnvoll, wenn man später diesen Bildabschnitt noch einmal wieder haben möchte, oder wenn der Bildteil sehr oft auf den Bildschirm gesetzt werden soll.</p>		
D1	X-Position	
D2	Y-Position	
D3	X-Größe	
D4	Y-Größe	
A1	Zeiger auf Windowspeicher	
<p>START:</p> <pre> MOVE.W #0,D0 JSR COLINIT MOVE.W #0,D1 MOVE.W #0,D2 JSR MOVETO MOVE.B #03,D0 MOVE.W #255,D1 MOVE.W #255,D2 JSR DRAWTO MOVE.W #128,D1 † Position 128,128 MOVE.W #128,D2 MOVE.W #10,D3 † Größe 10 † 10 MOVE.W #10,D4 LEA WINDOW1,A1 † Windowspeicher 1 JSR GETWINDOW RTS </pre> <p>WINDOW1: DS.B 10*10 † für 100 Byte frei halten</p>		

re Sammlung von solchen Unterprogrammen oder Prozeduren zusammengestellt.

Ein Treiber bietet nun eine solche Sammlung von Routinen, die ausgetestet und geprüft sind. Da die Routinen von einem fremden Programmierer geschrieben worden sind, steht und fällt die Qualität eines solchen Treibers mit seiner Dokumentation.

Der Anwender muß genau wissen, welche Datenübergabe und wo, zu den Routinen erfolgt, ob und wenn ja, welche Register diese Routinen verändern und ob und wenn ja, wie eine Rückgabe von Werten erfolgt.

Sehr schön zu wissen wäre auch noch, was die Routine eigentlich macht!

Hier nun ein dickes Lob an Kei: die Beschreibung ist wirklich – wie die Beispiele hier zeigen – sehr ordentlich gemacht. Vorbild war die Beschreibung des Treibers für die ACRTC-Baugruppe.

Jede Seite der Beschreibung ist in vier Abschnitte unterteilt.

Im ersten steht der Name der Routine und deren kurze Funktionsbeschreibung. Danach folgt ein größeres Bemerkungsfeld, das die Funktion deutlich erklärt. Im dritten Abschnitt werden die Register, in denen Daten übergeben werden bzw. die zerstört werden, aufgeführt. Im vierten Abschnitt folgt ein kleines Beispielpro-

gramm, das die Einbindung des Aufrufes in ein Anwenderprogramm (in 68000-Assembler) zeigt.

Der COL-Treiber stellt nun 88 Routinen zur Verfügung. Sie reichen von der Initialisierung der Baugruppe über das Löschen des Bildschirms, Setzen eines Punktes an X, Y, Zeichnen einer Linie, Scrollen des Bildschirms, Füllen einer Fläche mit einer Farbe bis zur Unterstützung von Bildschirm-Fenstern (Windows) und zum Austauschen von Bildschirmhalten.

Mit dem COL-Treiber lassen sich nun sehr schnell und effektiv Programme für die COL 256 erstellen; wichtig ist aber der neue, gesenkte Preis dieser Baugruppe.

SMART WATCH

Die neue Uhr für den NDR-Computer

Der Smart Watch-Baustein DS 1216 ist ein Baustein, mit dem es möglich ist, die Zeit in Hundertstel Sekunden, Sekunden, Minuten und Stunden sowie das Datum in Form des Wochentages, des Tages, Monats und Jahres abzuspeichern und jederzeit wieder abzurufen. Das Gehäuse des Smart Watch entspricht einer 28poligen IC-Fassung, die es möglich macht, einen CMOS-RAM aufzustecken. Ist der Uhrenbaustein nicht aktiviert, hat er nach außen keine Funktion; es wirkt lediglich der Speicherbaustein. Im „Sockelgehäuse“ ist eine Lithium-Batterie eingebaut,

die sowohl die Uhrenfunktion als auch die Speicherfunktion des aufgesteckten Speicherbausteins bei einem Spannungsausfall oder beim Abschalten des Systems gewährleistet. Wird der Smart Watch-Baustein durch eine 64 Bit lange Datenfolge aktiviert, ist der Zugriff zu dem aufgesteckten Speicherbaustein gesperrt.

Prinzipiell ist es möglich, den Smart Watch-Baustein auf jeden beliebigen Steckplatz der ROA einzustecken. Es sollte jedoch vermieden werden, ihn unter ein EPROM zu stecken, da dieses den eingebauten Akku des Uhrenbausteins sehr schnell entladen würde. Für die im Handbuch abgedruckten Programme ist es jedoch notwendig, daß sich der Smart

Watch-Baustein auf dem zweiten Steckplatz der ROA befindet, also auf Adresse 2000h. Es wäre jedoch auch möglich, ihn zum Beispiel auf den vierten Steckplatz zu setzen, es müßte jedoch dazu auch in den Programmen die Ansprungsadresse (im Programm als SCRATCHP definiert) auf 4000h geändert werden.

Bei dem Modula 2 Compiler existiert die Datei UHR.68k, die auf diesen Smart Watch-Baustein zugreift. Für diesen Anwendungsfall ist es gleichgültig, an welcher Stelle der Uhrenbaustein eingesetzt wird.

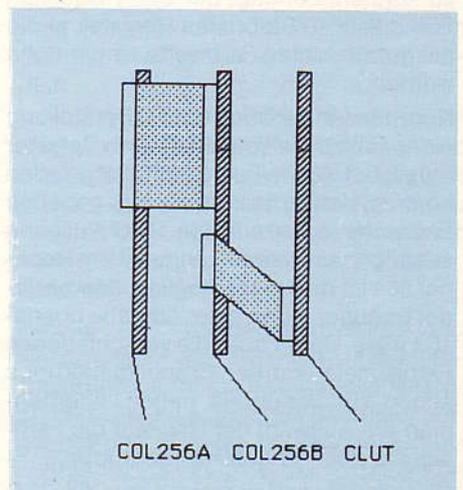
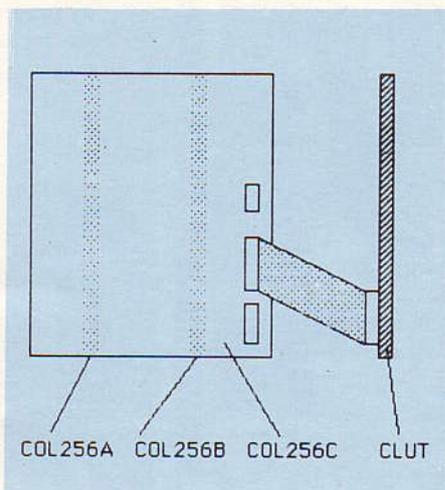
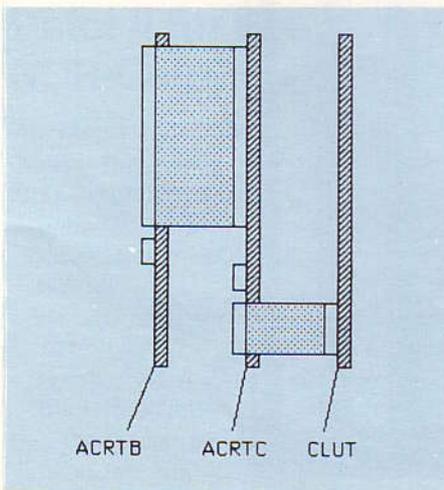
Die Programmierung der Smart Watch wird unter 680xx und JADOS im Artikel „Routinen für die Smart Watch“ vorgestellt.

CLUT – die Farbtabelle für NDR- und mc-CP/M-Computer ist da!

Die Baugruppe CLUT kann zusätzlich zu Farbgraphiksystemen als Farbtabelle eingesetzt werden. Die Baugruppe hat sowohl eine RDK- als auch eine ECB-Bus-Schnittstelle, so daß die CLUT sowohl bei den NDR-Farbgraphiksystemen

(ACRTC und COL256) sowie für sämtliche ECB-Bus-Farbgraphiksysteme, z. B. TERM-Farbe, eingesetzt werden kann.

Die Baugruppe wird sehr häufig bei den hochauflösenden Graphiksystemen des NDR-Computers verwendet. Deshalb soll



hier auf die Konfigurationsmöglichkeiten mit der ACRT und der COL256 eingegangen werden.

Technische Daten

- Colour Look up Table, Verwendeter Baustein: INMOS G 170-35
- Europakarte 100 * 160 mm doppelseitig, NDR-Bus
- Stromaufnahme: 300 mA, Betriebsspannung 5 V=
- Ausgang: RGB analog (IBM-Belegung oder 5 BNC-Buchsen)
- 256 auf 262144 möglichen Farben.

Prinzip der Farbtabelle (Colour Look up Table)

Durch eine Farbtabelle können Farbabstufungen, Farbnuancen und die Anzahl der insgesamt darstellbaren Farben fast beliebig erhöht werden. Allerdings ist die Anzahl der im Moment darstellbaren Farben nicht größer, als die, die an die Farbtabelle angeschlossenen Farbgraphik, d.h. es sind immer nur soviel Farbabstufungen gleichzeitig darstellbar, wie die Farbgraphik Farben darstellen kann.

Hierzu ein Beispiel: Die COL 256 kann 256 Farben darstellen. Wird die CLUT an die COL 256 angeschlossen, so sind weiterhin nur 256 Farben auf einem Bild darstellbar, allerdings sind die Farben aus einer Palette von 256k (262536) Farben wählbar.

Dabei sind die für ein bestimmtes Bild vorgesehenen Farben in ein Register des CLUT-Bausteins über den Datenbus einzutragen. Die Werte, die in diesen Registern stehen, entsprechen den Farbwerten für R, G und B (Rot, Grün und Blau). Beim hier verwendeten Baustein sind diese Register für R, G, B je 6 Bit breit. Diese 6 Bit werden über einen D/A-Wandler geführt und stehen dann als analoges Signal für RGB zur Verfügung. Aus den je 6 Bit für RGB (insgesamt 18 Bit) errechnet sich die gesamte mögliche Anzahl der Farben von $256k = 262536$.

Um jetzt 256 Farben auf einmal darstellen zu können, benötigt der CLUT-Baustein 256 dieser 18 Bit breiten Register, in die die gewünschten Farbwerte eingetragen werden.

Nun müssen natürlich bei der Darstellung eines Bildes die verschiedenen Register möglichst schnell umgeschaltet werden können, damit die vorher eingegebenen Farbwerte auch auf den RGB-Ausgang ausgegeben werden können. Vom Rechner aus ist dies nicht möglich, da wohl jeder Rechner mit so einer Aufgabe überlastet wäre. Da wir aber die verschiedenen Farbsignale von der Graphikbaugruppe schon zur Verfügung haben, übernehmen die Auswahl der Register die Farbsignale der Farbgraphikbaugruppe.

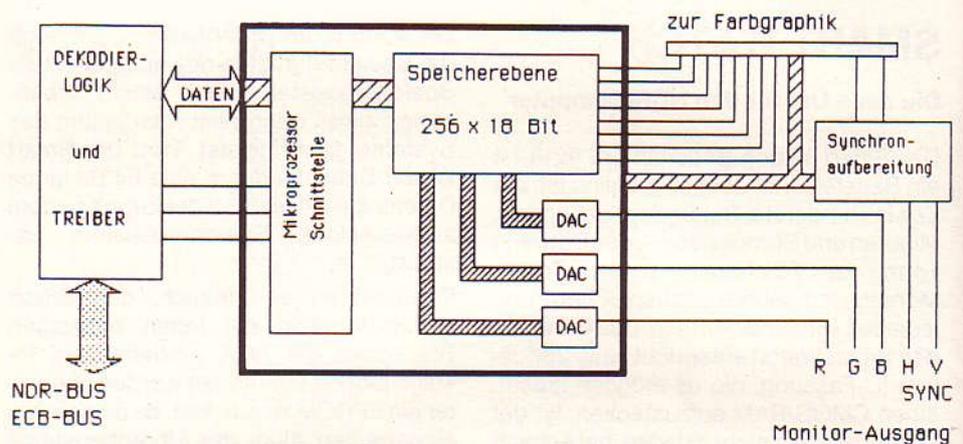
Hier wieder ein Beispiel mit der COL 256:

Die COL 256 kann 256 verschiedene Farben darstellen. Dabei werden die Farben folgendermaßen aufgegliedert. Für RGB sind jeweils 2 Bit vorgesehen, ebenfalls das Intensity-Signal, d. h. für einen Farbpunkt auf dem Monitor steht ein Datenwort von 8 Bit zur Verfügung (das entspricht 256 möglichen Farbwerten). Diese 8 Bit werden nun zum Adressieren der 256 Register verwendet. Am Einfachsten kann man sich das folgendermaßen vorstellen: Ändert sich auf einem Bild die Farbe, so wird das 8 Bit Datenwort verändert. Das hat zur Folge, daß durch dieses 8 Bit Datenwort bedingt, ein anderes 18 Bit Register des CLUT-Bausteines ausgewählt wird und sich dadurch auch am Ausgang der CLUT die Farbe ändert, falls in dem jetzt ausgewählten CLUT-Regi-

ster nicht derselbe Datenwert steht, wie im vorigen.

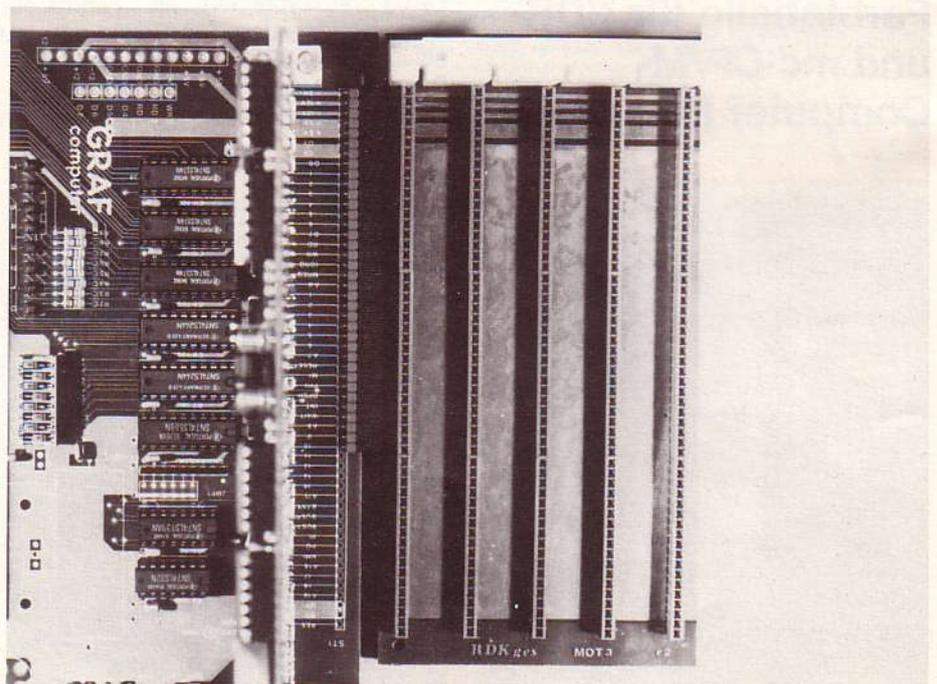
Fassen wir hier zusammen: Die CLUT kann immer nur so viele Farben gleichzeitig darstellen, wie die Farbgraphikbaugruppe darstellen kann. Mit der CLUT können aber aus 262536 Farben 256 ausgewählt und gleichzeitig dargestellt werden. Die Auswahl der CLUT-Register und damit der ausgewählten Farben übernimmt die Farbgraphikbaugruppe. Das untenstehende Blockschaltbild soll dies verdeutlichen.

Die CLUT macht den NDR-Computer mit der COL 256 oder ACRTC zu einem sehr hochleistungsfähigen Farbgraphiksystem. Die Programmierung der CLUT ist im Handbuch beschrieben und sehr einfach durchzuführen.



Buserweiterung jetzt problemlos möglich

Bisher war es wohl etwas schwierig, sich aus BUS2 einen BUS3 zu basteln. Man mußte durch verschiedene Schneid- und Lötarbeit die beiden Busse miteinander verbinden. Durch den von uns angebotenen BUS-Verbinder ist es jetzt problem-



los möglich, 2 Busse miteinander zu verbinden. Der BUS-Verbinder besteht aus drei abgewinkelten 18poligen Buchsenleisten (Präzisions 4 Finger-Kontakte) und drei 18poligen abgewinkelten Stiftleisten. Dadurch ist diese Verbindung

steckbar und kann jederzeit wieder gelöst werden. Besonders interessant ist diese Erweiterung für die HEXIO2. Da diese Baugruppe nur zwei Steckplätze zur Verfügung stellt, ist man hier sowieso knapp mit Ein/Ausgabe-Karten. Mit dem

Buserweiterungssatz kann man problemlos einen BUS2 an die HEXIO2 an-koppeln.

Zum Erweitern empfiehlt sich der altbewährte BUS2 (nicht BUS2A), da dieser problemlos anreihbar ist.

MLA 1 ist ein Macro-Link-Assembler zum vereinfachten Schreiben von größeren Programmen im 68000 Assembler-Code. Der Programmierer kann durch die Definierung von Macros sehr viel Arbeit sparen, da einmal definierte Macros immer wieder mit anderen Registern und Werten belegt werden können. Ein kleines Beispiel ist die Erstellung eines Menüs, bei dem viele Texte untereinander ausgegeben werden sollen. Normalerweise sieht ein Programm dann etwas unübersichtlich aus, da sich viele Zeilen immer wiederholen. Durch die Definition eines Macros ist das ganz anders. Hier das Beispiel des Menüs:

```
.GLOBL start      * Startadresse in Symboltabelle eintragen

.MACRO print      * Macro heißt print
(                * Macro beginnen
move.b  #&0,d0    * &0 bestimmt die Zeichengroesse
move.w  #&1,d1    * &1 bestimmt die X-Position
move.w  #&2,d2    * &2 bestimmt die Y-Position
lea  &3,a0        * &3 bestimmt die Textadresse
jsr  @write
)                * Macro beenden

start:
.print 32,20,220,text1
.print 21,10,180,text2
.print 21,10,160,text3
.print 21,10,140,text4
rts
```

Dieser Text ist doch schon wesentlich übersichtlicher, als wenn die 5 Befehle 4 mal hintereinander geschrieben werden.

MLA 1

Ein neues Werkzeug für den fortgeschrittenen Programmierer von Kei Thomsen

Von den Macros können bis zu 200 definiert werden, wodurch man sich sehr leicht eine große Macrobibliothek erstellen kann.

Der zweite Teil des MLA 1 besteht aus einem Linker. Mit dem Linker werden ein oder mehrere Texte beim Assemblieren zusammengesetzt. Das bedeutet, daß ein großes Programm nicht unbedingt aus einem Teil bestehen muß. Dadurch kön-

nen Programme in ihre grundlegenden Funktionen zerlegt werden. Eine Anwendung zum Beispiel ist das oben aufge-

führte Menü. Hierbei müssen ja noch die Texte definiert werden. Um das Programm übersichtlich zu gestalten, kann man dieses in einen zweiten Programmteil verbannen:

```
.GLOBL text1, text2, text3, text4
text1: dc.b 'Text 1',0
text2: dc.b 'Text 2',0
text3: dc.b 'Text 3',0
text4: dc.b 'Text 4',0
```

Im .GLOBL werden die Adressen definiert, welche dann vom ersten Programmteil übernommen werden. Dieses hat auch den ungemeinen Vorteil, daß Symbole, die man oft verwendet (wie etwa LOOP), in jedem Programmteil vorkommen können, ohne sich gegenseitig zu stören. Weiterhin braucht man sich auch um keine ORG-Adressen zu kümmern, da die Programmteile unmittelbar aneinander gesetzt werden, um somit das Programm kompakt zu halten. Dadurch können Unterprogramme so allgemein gehalten werden, wie irgend möglich. Dies können z. B. Wurzelberechnungen oder Graphikhilfen sein. Abschließend ist zu sagen, daß dieses Programm aus dem Problem eines viel zu langen Programmes entstanden ist. Hierbei mußten dann 12 Texte zu einem Programm zusammengesetzt werden, die zusammen eine Symboltabelle von 38 Kbyte ergeben hätten. Dieses konnte durch den Einsatz von 43 Macros und 177 globalen Adressen auf ein Minimum an Aufwand begrenzt werden.

Datei II für den NDR-Computer 68k

von Manfred Zehner

Dieses Datenbank- und Textverarbeitungsprogramm bietet:

- unbeschränkte Druckeranpassung (alle Befehle beliebiger Drucker sind nutzbar),
- nahezu unbeschränkter Daten- und Textumfang,
- kombinierbare Text-, Datenbank- und Steuerfunktionen.

Das Programm ist zur Zeit als JADOS-Version (ab 2.1), JOGIDOS-Version und auf EPROMs zum Preis von DM 96,90 lie-

ferbar. Zum Programm gehört ein 80seitiges Handbuch.

Für alle Eingaben - ob beliebiger Text, Datenbank-Daten oder Steuerbefehle - wird der gleiche Editor mit über 30 Funktionen benutzt, das vereinfacht die Eingabe.

Die Daten werden in sog. Datenblocks erfaßt, die über 65.000 Zeilen lang sein können. Größe und Anzahl der Datenblocks werden nur vom vorhandenen Arbeitsspeicher begrenzt. Die Zeilen sind zur leichten Orientierung durchnummeriert.

Bis zur Speichergrenze können z. B. beliebig viele Texte mit über 65.000 Zeilen Länge oder beliebig viele Datensätze mit über 65.000 Datenfeldern oder beliebig

viele Steuerblocks mit 65.000 Befehlen (z.B. für Druckertreiber) aufgebaut werden.

Das Programm ermöglicht es z. B., schnell einen Text (Brief) einzugeben und diesen mit Daten aus den Datenbanken (Adress-Datei, Textbaustein-Datei) zu verknüpfen und in der gewünschten Form (Briefbogen, Postkarte, Aufkleber, Liste) auszudrucken.

Neben der Textbausteinverarbeitung sind Serienbriefe, Tabellen- und Mehrspaltendruck möglich. Auch kleine Graphiken können schnell als Bitmuster aufgebaut und z. B. als Logos oder Unterschrift-Nachbildung bei der Textausgabe eingefügt werden.

Bei vorhandener Smart Watch kann auch Datum und Uhrzeit automatisch mit ausgegeben werden.

Darüberhinaus können Daten im NDR-Klein-Editorformat ein- und ausgelesen werden (schnelle Assembler-Text-Bearbeitung) und fremde Daten können übersetzt werden (EDATED-Daten sind damit weiterhin verwend- und ausbaubar).

Quell-Code für HEBAS

Basic-Interpreter ab CP/M 2.2 für CPU Z80 und HD 64180

von Dr. Hans Hehl

Vielen CP/M-Anwendern ist der BASIC-Interpreter HEBAS ein Begriff. Dieser Interpreter basiert auf einer Version des TDL-Basic namens Xitan von Neil Colvin und wurde umfangreich verbessert. So sind deutsche Fehlermeldungen und zusätzlich Befehle sowie ein ausführliches deutsches Handbuch vorhanden. Weiterhin werden keine illegalen Opcodes des Z80 mehr verwendet, eine wichtige Bedingung für die CPU HD 64180 oder neuere Z80-Nachfolger. HEBAS kann nun auch unter CP/M 3+ verwendet werden. In der NDR-Benutzerzeitschrift LOOP (siehe auch mc, Heft 5/1987, S. 151) ist eine ständige Rubrik „Tips und Tricks mit HEBAS“ vorhanden, in der auf Leserzuschriften eingegangen wird und Neuerungen für HEBAS gebracht werden. Ein UP-Date-Service für 10 DM beim Autor (Adresse siehe unten) ermöglicht den HEBAS-Kunden, neuere HEBAS-Versionen im Tausch (Einsenden der Original-Diskette) zu erhalten. In besonderen Fällen kann gegen Berechnung eine speziell

angepaßte HEBAS-Version erstellt werden.

Nun steht dem interessierten BASIC-Fan mit Kenntnissen in der Programmierung des Z80 der Quellcode auf Diskette und ausgedruckt als PRN-File zur Verfügung. Dieser besitzt einen Umfang von ca. 226 KByte und kann mit Standard-Textverarbeitungsprogrammen bearbeitet werden. Allerdings muß entsprechend Platz auf der Diskette sein, sonst dauern Such- und Austauschvorgänge sehr lange. Eine große RAM-Floppy (am besten 1 MByte) ist zu empfehlen.

Als Assembler können der bekannte M80 von Microsoft oder der schnellere SLR180 vom Elektronikladen Verwendung finden, letzterer erzeugt direkt ein COM-File ohne zusätzlichen LINK-Vorgang. Bei entsprechendem Bedarf wird auch eine Quell-Code-Version für den TDL-Assembler erstellt.

In der Quelle bestehen die Labels mit sinnvollen Abkürzungen aus 6 Bytes. Nun können Teile des Interpreters unmittelbar erweitert werden, ohne daß mit CALL an einen freien Platz gesprungen werden muß. Die Routinen sind mit der Beschränkung durch die Relativ-Sprünge des Z80 verschiebbar. Wie Bild 1 zeigt, sind die wesentlichen Teile der Quelle ausführlich kommentiert.

Eine wichtige Frage: Was kostet die Quelle?

Autor und die vertreibenden Firmen Elektronikladen aus Detmold und GES Graf aus Kempten sind der Auffassung, daß trotz des immensen Arbeitsaufwandes bei der Erstellung der kommentierten Quelle die folgenden äußerst günstigen Konditionen den Kundenwünschen Rechnung tragen.

Es gibt nun zwei Ringbücher, wobei der Text mit einem Laserdrucker erstellt wurde. Das erste enthält das neue, verbesserte und erweiterte Handbuch (ca. 80 Seiten, auf NDR-Rechner und mc-Computer wird jeweils getrennt hingewiesen) und die Diskette mit HEBAS.COM zum Preis von etwa 95 DM inkl. MWSt. je nach Diskettenformat.

Das zweite Ringbuch mit etwa 220 Seiten enthält das ausgedruckte PRN-File, also die Quelle des Interpreters mit den zusätzlichen Adressen bei den Labels, mit dem Kommentar und die Symbol-Tabelle. Auf der dazugehörigen Diskette sind die kommentierte Quelle und Symbol-Tabelle zum Gesamtpreis von 110 DM inkl. MWSt. (nur 80 Spur NDR-Format oder 8 Zoll IBM, aus Platzgründen ohne Symbol-Tabelle).

Nun kommt etwas sehr wichtiges. Das Ringbuch mit Quelle allein kann nur derjenige beziehen, der schon HEBAS gekauft hat. Dies kann durch Rechenkopie oder Einsenden der Original-Diskette glaubhaft gemacht werden.

Für Nichtbesitzer von HEBAS gilt bei Erwerb der Quelle der Gesamtpreis von 205 DM inkl. MWSt., da beide Ringbücher mit den Disketten bezogen werden müssen.

In absehbarer Zeit wird es dann eine Beschreibung der wichtigsten Abläufe im Interpreter als Ergänzung zum ersten Handbuch geben, damit ein Anfänger anhand der Quelle sich zurechtfinden kann.

Wartenburg, 26. 4. 1987

Dr. Hans Hehl, Lindenstraße 20

Hinweis: Die erwähnten Produkte sind ab August lieferbar; die von Dr. Hehl genannten Preise sind freibleibend.

```

*****
;*
;*      BASIC-INTERPRETER - QUELLENPROGRAMM
;*
;*      HEBAS.180      für mc-Computer
;*      bzw.          für NDR-Rechner (RDK)
;*
;*      Z-80-CPU, HD64180-CPU, CP/M 2.2, CP/M 3+
;*      VERS. 3.1+
;*
;*      (C) Dr. Hehl Hans   20.04.1987
;*      für
;*      Elektronik-Laden 4930 Detmold
;*      GES Graf         8060 Kempten
;*
;*      Assembler SLR180.COM bzw. M80.COM
;*
;*      URCODE: XITAN Z-80 DISK BASIC VERS. 1.06
;*              by NEIL COLVIN JUNE 1978 06/11/78
;*
*****
;STAND 24.4.87 REV. 1.1

```

```

.....
;BELL-Simulation, Tonausgabe:
RBELL:      ;RINGBELL
            ;Register sichern
            ;OFH = Steuerregister mc-C.
            ;NOP bei HD64180-CPU
            ;NOP " " "
            ;Tondauer
            LD   A,0FH
            OUT  (0F5H),A
            LD   HL,01FFF

```

```

TON1:
LD   A,0          ;Port aus
OUT  (0F4H),A    ; (mc-Computer:
;OUT  (30H),A    ; (NDR-RECHNER)
;OUT  (50H),A    ; (HD64180-CPU)

CALL  TONLP1
LD   A,0FH       ;Port ein
OUT  (0F4H),A   ; (mc-Computer)
;OUT  (30H),A   ; (NDR-RECHNER)
;OUT  (50H),A   ; (HD64180-CPU)

CALL  TONLP1    ;Frequenzschleife
DEC  HL        ;Zähler - 1
LD   A,H
CP   0         ;wenn 0, dann
JP   NZ,TON1   ;Ende
POP  BC        ;Register ok.
POP  DE
POP  HL
RET

EDITR:      ;Eingabe "nR" n Zeichen ersetzen (REPLACE)
LD   A,(HL)    ;Byte holen
OR   A        ;wenn 0,
RET  Z        ;dann fertig
CALL  BSCII01  ;Zeichen holen (BIOS)
CALL  CD2     ;Byte ausgeben (ggf. mit CR u.LF)
LD   (HL),A   ;neues Byte ablegen
INC  HL       ;Textzeiger + 1
INC  D
DJNZ EDITR    ;bis B = 0 (B = Parameter n)
RET

```

Bild 1: Ausschnitte aus dem Quellcode von HEBAS

Für Einsteiger Z80 SBC2

Motorsteuerung mit dem Einsteigerpaket

Die neu entwickelte Baugruppe ROB2 läßt vom Namen her vermuten, daß sie „nur“ zur Ansteuerung des Fischer Technik Roboters gedacht ist. Dies ist aber nur bedingt richtig; die Baugruppe ist sehr universell ausgelegt, um ganz allgemein etwas „Leistung“ mit dem Einsteigerpaket – und natürlich auch mit jeder weiteren Baustufe – zu steuern.

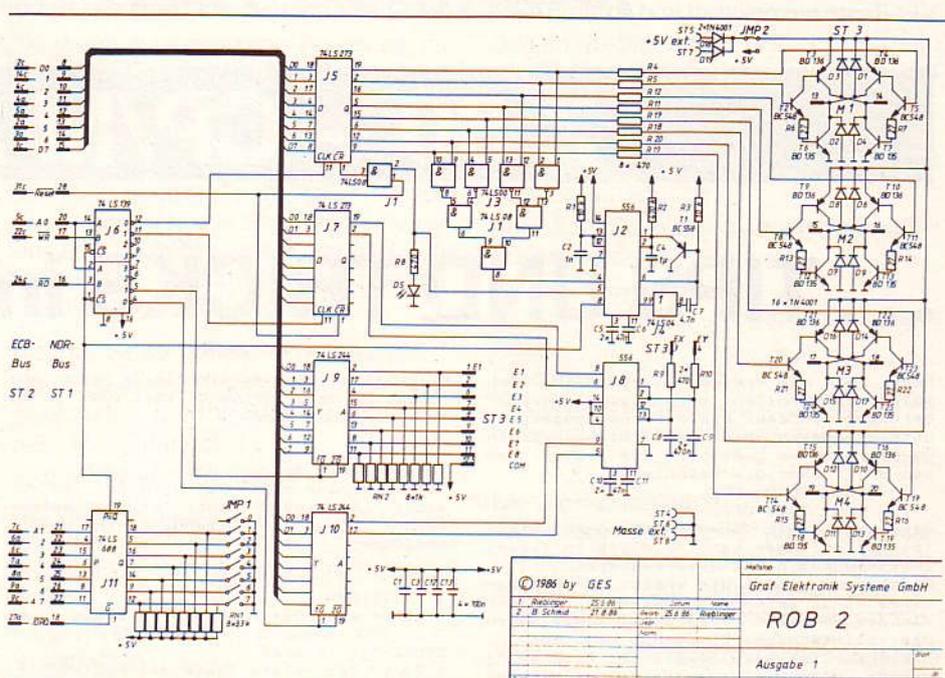
Die Baugruppe hat acht Leistungs-Ausgänge, die jeweils eine maximale Stromstärke von ca. 1 A zur Verfügung stellen können. Die Spannung für diese Leitungsausgänge wird nicht vom Bus genommen, sondern extern zugeführt; sie darf zwischen + 5 und + 12 Volt liegen.

Je zwei dieser Leistungsausgänge sind zu einer Brücke zusammengesetzt, um z.B. einen Gleichspannungsmotor vorwärts und rückwärts ansteuern zu können. Hier werden die fast unverwüstlichen Leistungstransistoren BD 135 und BD 136 eingesetzt.

Über eine recht aufwendige Verriegelungslogik wird unterbunden, daß bei falschen Datenübergaben Kurzschlüsse an den Ausgängen bzw. in der Endstufe entstehen können.

Neben den Leistungsausgängen verfügt die Baugruppe über acht digitale und zwei analoge Eingänge. Der Digitaleingang entspricht der IOE, die anstehenden Signale werden in ein 74LS244 eingetragene und auf den Bus gelegt.

Die Analog-Eingabe dient dazu, z.B. Rückmeldesignale über die Stellung eines Greifarmes in den Rechner einzulesen. Direkt an den Analogeingang kann je



ein Potentiometer angeschlossen werden.

Die Wandlung geschieht recht einfach über einen Timer 556. Das externe Poti bestimmt dessen Ablaufzeit, die direkt proportional der Poti-Stellung ist. Das analoge Spannungssignal wird also zu einem Zeitsignal gewandelt. Im Beispielprogramm im Handbuch wird gezeigt, wie man innerhalb eines Programmes ein solches Zeitsignal weiter verarbeiten kann.

Alle Ein/Ausgänge werden auf eine zweireihige, 20polige Stiftleiste geführt und können von dort mit einem Flachbandkabel abgenommen werden. Die Belegung

dieser Stiftleiste entspricht genau der Belegung der Fischer Technik Modellbaukästen aus dem Bereich „Computing“.

Fischer Technik wird im Herbst eine neue, interessante Baureihe mit einfachen Modellen vorstellen. Natürlich werden auch diese Modelle an die ROB2 anzuschließen sein.

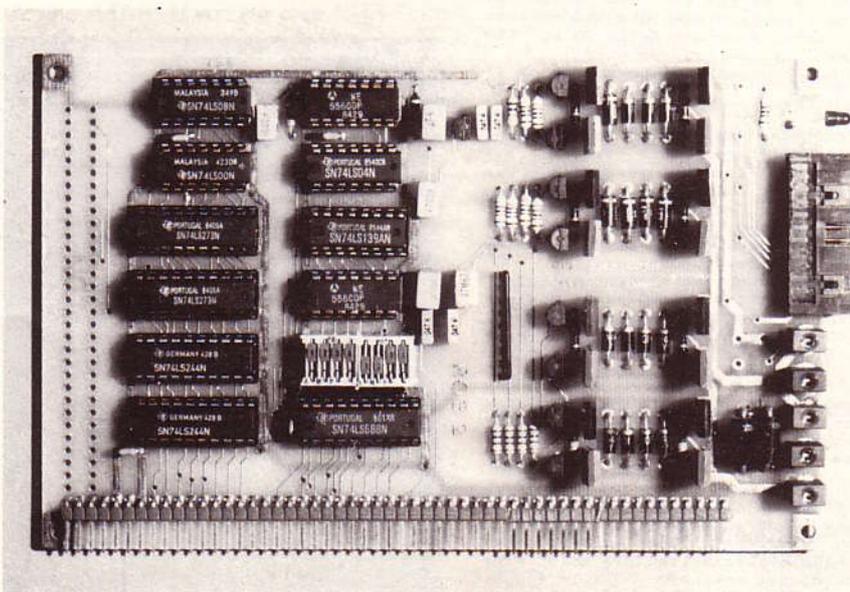
Generell eignet sich die ROB2 aber für viele Anwendungen; man könnte sie als Kombination einer IOE, einer „elektronischen“ Relais-Baugruppe REL sowie einer sehr einfachen A/D-Karte bezeichnen. Interessant ist es sicher, mit dieser Baugruppe das Einsteigerpaket zu erweitern. Ein Test der angeschlossenen Schaltungen ist über die Möglichkeiten „Eingabe lesen“ und „Eingabe setzen“ des Grundprogrammes leicht durchführbar.

Sollen „nur“ Motoren angesteuert werden, so bezieht man nur die Leiterplatte und bestückt eben nur den für die Motorsteuerung wichtigen Teil der Baugruppe.

Da die Baugruppe der „GES-Norm“ entspricht, kann sie auch in ECB-Systemen eingesetzt werden. Eine interessante Erweiterung auch für den mc-CP/M-Computer!

Mit der ROB2 läßt sich nun auch der große Fischer Technik Roboter (mit Greifer) bereits mit dem Einsteigerpaket antreiben.

Ein EPROM dafür ist ebenfalls verfügbar. Es heißt EROB für Trainings-Roboter



TURBO-Pascal und die ATARI-Maus für den NDR-Computer

von Christoph Köhler

Viele von Ihnen haben die Baugruppe HARDCOPY/MAUS und die ATARI-Maus gekauft oder wenigstens damit geliebäugelt und können in diesem Beitrag nähere Einzelheiten über die Anwendung derselben nachlesen.

Im Gegensatz zu käuflichen Home- und PC-Computern muß beim NDR-Computer das Programm für den bekannten Pfeil (z.B. GEM-Benutzeroberfläche) auf dem Bildschirm selbst geschrieben werden.

Nun, erst einmal stellt sich die Frage, welche Programmiersprache sich für den Einsatz der ATARI-Maus am besten eignet. Hierbei müssen zwei Fakten beachtet werden:

1. Die Geschwindigkeit des Anwenderprogrammes

Um die Maus schnell auf der Unterlage bewegen zu können, müssen ständig die Koordinaten abgefragt werden, um die Lage des Pfeiles am Bildschirm zu korrigieren.

2. Komfort beim Programmieren (z.B. TURBO-Pascal)

Eine gute Kombination ist die Programmierung in TURBO-Pascal, unter Zuhilfenahme von Teilen, die in Z80 Assembler-Sprache geschrieben sind.

Leser, die einer der beiden genannten Sprachen nicht mächtig sind, sollten jetzt bloß nicht aufhören zu lesen, da die Sprache TURBO-Pascal weiter unten vorgestellt wird und keine Vorkenntnisse in der Z80 Assembler-Sprache notwendig sind. Zunächst wird kurz die ATARI-Maus vorgestellt:

A: Die ATARI-Maus für den NDR-Computer

Durch Bewegen der Maus auf einer Unterlage, wandert z.B. ein Pfeil auf dem Bildschirm. Die beiden Tasten werden dazu benützt, um irgendwelche „bildschirmorientierte“ Vorgänge (z.B. Ende des Programmes) auszulösen.

Zum Betrieb wird zusätzlich die Baugruppe HARDCOPY/MAUS verwendet.

Um programmtechnisch die Koordinaten der Maus sowie die beiden Tasten abzufragen, kann man das später vorgestellte Programm (Hexdump) in allen Programmiersprachen einsetzen (z.B. BASIC, TURBO ... usw.)

B: Kleine Einführung in TURBO-Pascal

Dieser kleine Pascalüberblick ist nur für den TURBO-Neuling gedacht und kann vom „Insider“ übersprungen werden.

Die Programmiersprache Pascal ist von Professor Wirth entwickelt. Später entstanden verschiedene Varianten, eine der modernsten ist TURBO-Pascal.

Wie sich schon aus dem Namen schließen läßt, zeichnet sich TURBO-Pascal durch eine schnelle Programmentwicklung und -ausführungszeit aus. Nicht ohne Grund wurde es zur Programmiersprache des Jahres 1985 gewählt.

Aufbau eines TURBO-Programmes:

Die Programmstruktur besteht immer aus zwei Teilen, dem Vereinbarungsteil und dem Ausführungsteil.

1. Der Vereinbarungsteil

Der wohl wichtigste Satz für TURBO-Pascal müßte wohl heißen: Alle Bezeichner müssen vor ihrer Anwendung deklariert werden.

Es gibt verschiedene Bezeichner:

- Variablen
- Unterprogramme
- Konstante
- Datentypen

d.h., jede Variable muß vor Verwendung dem Programm „vorgestellt“ werden.

Im weiteren besteht der Vereinbarungsteil prinzipiell aus drei verschiedenen Anweisungen.

1.1 Die Programm-Anweisung:

Diese steht immer am Anfang und sieht z.B. so aus: PROGRAM Pfeil;

1.2 Die VAR-Anweisung:

Die Variablen können vom Typ Integer, Byte, Real, Char oder Boolean sein (siehe Handbuch) und ergeben etwa folgendes Bild:

```
VAR Vorname, Name      : Char;
    Alter, Größe       : Integer;
    Kontostand         : Real;
```

1.3 Die Procedure-Anweisung

Der Inhalt einer Prozedur (Unterprogramm) ist unerheblich, aber die Form (Syntax) ist vorgeschrieben:

```
PROCEDURE Name;
BEGIN hier steht der Inhalt der Prozedur
END;
Der Name kann frei gewählt werden.
```

2. Der Ausführungsteil

Der Ausführungsteil ist oft sehr kurz und kann eine beliebige Anzahl von Anweisungen und Unterprogramm-Aufrufen beinhalten. Die Form sieht folgendermaßen aus:

```
BEGIN Hier steht das eigentliche Programm (beliebige Kombination von Anweisungen)
END.
```

Beachten Sie den Punkt hinter dem „END“. Nach einem Unterprogramm wird der Befehl „END“ mit einem Semikolon „;“ abgeschlossen.

Das wars eigentlich schon! Zumindest was die Struktur betrifft!

Zum Schluß sollen noch einige Begriffe erklärt werden:

2.1 Die Anweisung

Es gibt grundsätzlich zwei verschiedene Arten:

Die Einzelanweisung ist quasi ein Befehl, z.B.: Größe = 176; oder ein Unterprogrammaufruf, z.B.: Name

Die Blockanweisung muß wiederum in BEGIN und END gekleidet sein, gefüllt von beliebigen Einzelanweisungen oder Blockanweisungen. Es kann beliebig verschachtelt werden, z.B.: BEGIN Hier können wieder beliebige Anweisungen stehen END;

2.2 Kontrollstrukturen

Wie in allen Programmiersprachen gibt es auch in TURBO-Pascal mindestens drei Arten von Kontrollstrukturen:

2.2.1 Bedingte Kontrollstruktur

Dazu gehört z.B. eine IF .. END-Anweisung

2.2.2 Wiederholende Kontrollstruktur

Hier kann die FOR .. DO, WHILE .. DO oder REPEAT .. UNTIL eingesetzt werden

2.2.3 Auswählende Kontrollstruktur

Es bietet sich eine CASE .. END-Anweisung

Eine nähere Beschreibung der Anweisungen würde den Rahmen der Strukturübersicht verlassen.

2.3 Vordefinierte Funktionen und Prozeduren

Vordefinierte Prozeduren können dem Handbuch entnommen werden und ersparen dem Anwender die Arbeit, immer wieder benötigte Unterprogramme wie CLSCR (Bildschirm löschen) selbst zu schreiben.

Vordefinierte Funktionen könnte man als Werkzeug bezeichnen, das von TURBO-Pascal gestellt wird. Die Funktion, z.B. Random, stellt dem Anwender eine Zufallszahl von beliebiger Größe (z.B. 1 .. 49 bei Lotto) für die Auswertung im Programm.

2.4 Der Kommentar

Überall im Programmtext kann ein Kom-

mentar eingefügt werden. Dieser beginnt mit (* und endet mit *). Z. B.:
(* Dies ist ein Kommentar *)

Die kleine Einführung in TURBO-Pascal kann selbstverständlich die sonst erhältliche Fachliteratur nicht ersetzen, sondern soll nur einen Überblick vermitteln.

C: Das Programmbeispiel:

Es gibt verschiedene Techniken, um ein Programm zu entwerfen; eine sehr bewährte Methode ist das Schreiben eines Struktogramms auf Papier.

Dies ermöglicht einen übersichtlichen und „computer-losen“ Entwurf eines Programmes, das auch noch lange nach dem Programmieren nachvollziehbar bleibt. In einer der nächsten Ausgaben

von LOOP können Sie eine Einführung in diese Technik lesen.

Es ist wirklich jedem zu empfehlen, sich mit TURBO anzufreunden, da es dem Anwender ein Software-Grundwissen für alle Programmiersprachen vermittelt, auch wenn dieser Beitrag für die Einsteiger unter Ihnen vielleicht etwas zu voll war, möchte ich Sie trotzdem dazu ermuntern, von Anfang an aufs richtige Pferd zu setzen.

Als guten Einstieg für die zukünftige Zusammenarbeit zwischen Ihnen, TURBO-Pascal und der ATARI-Maus bietet sich das neue Puzzlespiel an, das auf der Spiele-Diskette von GES untergebracht ist. Es wurde unter TURBO-Pascal geschrieben, wird mit Quelltext auf Diskette

ausgeliefert, demonstriert eine schnelle Mausebearbeitung und hilft schließlich beim Erstellen eigener Programme.

Zum Schluß wünsche ich Ihnen viel Spaß mit der neuen/alten Programmiersprache und verbleibe bis zum nächsten Artikel.

Literaturhinweise: TURBO-Handbuch, TURBO-TUTOR-Handbuch von BORLAND, Schritt für Schritt – Sonderheft, Seite 88/89, Turbo-Pascal von Markt & Technik ISBN 3-89090-150-6, Das Turbo Pascal Buch von SYBEX ISBN 3-88745-608-4

Hinweis: Der INLINE-Code in der Prozedur MAUS-Fragen entspricht dem Testprogramm im Hardcopy-Maus-Handbuch, Seite 37.

Listing of FLIEGE.PAS

```

1: PROGRAM Fliege;
2:
3: VAR   Y_neu, Y_alt, X_alt, X_neu       : INTEGER;
4:       Rechts, Links, Zoom             : BYTE;
5:       Pf_neu, Pf_neu                  : BOOLEAN;
6:
7:
8: PROCEDURE Maus_Fragen;
9: (* Dieses Unterprogramm kann immer verwendet werden und übergibt
10:  die Variablen X_neu, Y_neu fuer die Koordinaten und die Maustaster
11:  Rechts und Links als 0 oder 1 *)
12: BEGIN
13:   INLINE(%d3/%dB/%d3/%B);
14:   INLINE(%2a/Y_neu/%dB/%B/%f/%00/%2B/%04/%23/%3d/%20/%fc/%dB/%dB);
15:   INLINE(%f/%00/%2B/%04/%2b/%3d/%20/%fc/%22/Y_neu/%2a/X_neu/%dB/%dB);
16:   INLINE(%f/%00/%2B/%04/%23/%3d/%20/%fc/%dB/%B/%f/%00/%2B);
17:   INLINE(%04/%2b/%3d/%20/%fc/%22/X_neu/%dB/%B/%e/%B/%07/%32/Links);
18:   INLINE(Rechts/%dB/%B/%e/%0/%07/%07/%32/Links);
19: END;
20:
21: PROCEDURE Pfeil;
22: (* Hier wird der Pfeil (Fadenkreuz-Befehl) von Sonderheft 2.5. 08/89
23:  nur einmal definiert, um das Programm schnell zu machen. *)
24: BEGIN
25:   IF Pf_neu=TRUE THEN BEGIN
26:     WRITELN('WA', 1*HLS);
27:     WRITELN('WC', Zoom, ' 5');
28:     Pf_neu:=FALSE;
29:     Pf_neu:=TRUE;
30:   END;
31: END;
32:
33: PROCEDURE Fliege;
34: (* Siehe Pfeil *)
35: BEGIN

```

Page 1

```

36:   IF Pf_neu=TRUE THEN BEGIN
37:     WRITELN('WA', 1*HLS);
38:     Pf_neu:=FALSE;
39:     Pf_neu:=TRUE;
40:   END;
41: END;
42:
43:
44: (* Hauptprogramm
45:  Beim Aufruf erscheint am Bildschirm ein Pfeil der sich mit der
46:  Maus bewegen laesst, drueckt man die rechte Taste, waendelt sich
47:  dieser in eine 'Fliege' die auch. auch bewegt werden kann.
48:  Losgelassen arbeitet man wieder mit dem Pfeil.
49:  Diese kleine Routine kann in Ihren Programmen uebernommen werden. *)
50: BEGIN
51:   (* Voreinstellungen *)
52:   CLRSCLR;
53:   Zoom:=2;
54:   X_neu:=125;
55:   Y_neu:=150;
56:   Pf_neu:=TRUE;
57:
58:   (* Grafikmodus von FLUMON *)
59:   WRITELN(CHR(27),CHR(27), 'G', 'Y0');
60:
61:   (* Wiederholung bis linke Taste gedrueckt wird *)
62:   REPEAT
63:     X_alt:=X_neu;
64:     Y_alt:=Y_neu;
65:     Maus_Fragen;
66:     X_neu:=X_alt+(X_neu-X_alt)*4;
67:     IF Rechts=1 THEN Pfeil ELSE Fliege;
68:     WRITELN('F', X_neu, ' ', Y_neu, ' 0');
69:   UNTIL Links=0;
70:
71:   (* Zurueck zum Alphamodus *)
72:   WRITELN('A');
73: END;
74:
75:

```

Tips und Tricks bei HEBAS, Nr. 7

von Dr. Hans Hehl

1. Der Klammeraffe plottet nicht

Es ist erstaunlich, wo überall HEBAS eingesetzt wird. Ein Kunde verwendet den Interpreter auf dem Computersystem Moppel der Zeitschrift ELO. Allerdings bestand zunächst folgendes Problem:

Am Anfang jeder Bildschirmzeile erschienen drei Klammeraffen. Schuld ist die Tatsache, daß als Relikt an langsame Peripheriegeräte nach jedem ausgegebenen Carriage Return drei Nullen folgen. Dies ist im Handbuch unter dem Abschnitt Grundeinstellungen nachzulesen. Bei manchen Bildschirmtreibern bedeutet aber der Wert Null ein Steuerzeichen, nämlich Kontroll-Null. Dies erscheint als Klammeraffe.

Abhilfe bringt der Befehl `OPTION#0,"N",0` (0 = Null). Damit beim Kaltstart dies schon berücksichtigt ist, muß mit einem Debugger wie DDT.COM (siehe LOOP 5) der Wert 3 der Speicherstelle 2C3Eh auf Null gesetzt werden.

Mit dieser Änderung gibt es auch bei Plotter, die mit HEBAS angesteuert werden, keinen Fehler mehr.

2. Tastatur-Klick-klick-klick ...

Wer beim Drücken der Computertastatur nichts hört, weil ein „Tastatur-Klick“ fehlt, kann dies schnell ändern, wenn auf der IOE-Platine (wie in LOOP 3 beschrieben) ein Lautsprecher angeschlossen ist. Verwendet wird die Ringbell-Routine ab

Adresse 110h (siehe HEBAS-System-Routinen, LOOP 10, S. 17). Nach jedem Tastendruck wird zur Routine ab Adresse 330Dh gesprungen. Hier bauen wir eine Umleitung auf einen freien Speicherplatz im HEBAS ein, von dem wir zur Ringbell-Routine verzweigen. Folgende Bytes sind mit einem Debugger einzugeben:

ab Adresse 330B: D5 45
ab Adresse 45D5: CD 3F 37 F5 CD 10 01 F1 C9

Nach dem Abspeichern auf Diskette und nach dem Start von HEBAS kann die Frequenz (Speicherstelle 308D) und die Tondauer (Speicherstelle 280D) mit dem POKE-Befehl nach Wunsch angepaßt werden (allerdings auch beim RINGBELL-Befehl verändern). Geeignet ist z.B. der Wert 40 in beiden Speicherstellen.

Hat man geeignete Werte gefunden, können diese ebenfalls in den beiden Speicherstellen im HEBAS mittels Debugger abgelegt werden.

BIORHYTHMUS unter HEBAS

von Martin Bohlen, Kanalstraße 18, 2962 Ostgroefehn 1

Ich habe das Biorhythmus-Programm aus LOOP 7 modifiziert und erweitert.

1. Es läuft jetzt unter Hebas; 2. Die Graphikausgabe wurde verändert.

Die Graphik enthält bei der 50 % Marke eine X-Achse, damit sind die Schnittpunkte der Sinuskurven erkennbar. Diese Schnittpunkte sind besonders wichtig,

da an den Wechseltagen vom positiven zum negativen Bereich, bezogen auf die X-Achse, besondere Risiken bestehen, die zu Fehlverhalten führen können.

3. Letztlich enthält das Programm noch eine Hardcopy-Routine. Dieses Programm entspricht im wesentlichen dem, des im mc-Heft Nr. 11/85 veröffentlichten Hardcopyprogramms.

```

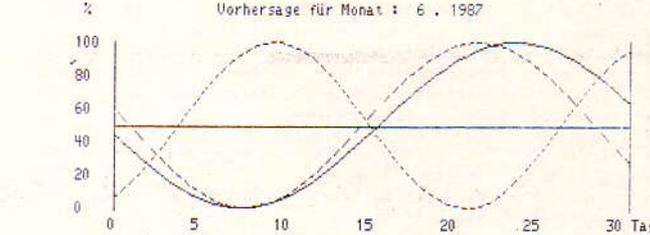
10 REM ***** Dieses Programm läuft unter HEBAS *****
20 REM ***** Biorhythmusprogramm mit Hardcopyausgabe *****
30 GS=CHR$(27)+CHR$(27)+"G" :REM Flomon-Graphik initialisieren
40 GOSUB 1070 :REM Sprung zum Vorspann
50 GOSUB 880 :REM Sprung zu den Datenzeilen
60 TS = BYTES(#0)
70 IF TS = "G" THEN 80 ELSE 60
80 PRINT CHR$(27)+"z"+"1" :REM Deutscher Zeichensatz
90 REM BIORHYTHMUS
100 REM EINGABE UND VORHERSAGE DATUM
110 CLOSE#0 :REM Bildschirm löschen
120 PRINT "Biorhythmus"
130 PRINT:PRINT
140 INPUT "GEBURTS TAG :";A
150 INPUT "GEBURTS MONAT:";B
160 INPUT "GEBURTS JAHR :";C
170 GOTO 190
180 CLOSE#0
190 PRINT
200 INPUT "VORHERSAGE TAG:";D
210 INPUT "VORHERSAGE MONAT:";E
220 INPUT "VORHERSAGE JAHR :";F
230 G=A+B*30.4+F*365.25
240 H=D+E*30.4+F*365.25-G
250 :
260 REM AUSGABE DER ERRECHNETEN BIORHYTHMUS-WERTE
270 CLOSE#0
280 PRINT TAB(25);"**** Biorhythmus ****"
290 PRINT "I=23;GOSUB 680
300 PRINT TAB(5);"Geburts.Dat. :";A;" :";B;" :";C;TAB(45);"Körperl.:";L;"% " --
--
310 I=28;GOSUB 680
320 PRINT TAB(5);"Vorhers.Dat. :";D;" :";E;" :";F;TAB(45);"Seelisch:";L;"% " --
--
330 I = 33;GOSUB 680
340 PRINT TAB(45);"Geistig :";L;"% " -----
350 PRINT
360 PRINT "0 = Ende 1 = Neues Geb.-Datum 2 = Neues Vorh.-Datum 3 = Graphik, d
ann mit
370 PRINT "H = H.Copy M = MENU"
380 PRINT
390 GOSUB 820
400 GOTO 670
410 REM AUSGABE DER GRAPHIK
420 :
430 PRINT TAB(9);"%":TAB(25);"Vorhersage für Monat : ";E;" :";F
440 FOR M=100 TO 0 STEP-20
450 PRINT:PRINT TAB(7);M
460 NEXT
470 PRINT TAB(11);0;TAB(21);5;TAB(31);10;TAB(41);15;TAB(51);20;TAB(61);25;TAB(71)
);30;TAB(75);"Tag"
480 PRINT GES :REM Graphik ein
490 PRINT "PO" :REM Seite Null
500 PRINT "YO" :REM Zeichnen M = MOVETO , D = DRAW
:
510 PRINT "M76 140 D76 38" :REM Seitenumschaltung aus
520 PRINT "M76 89 D448 89" :REM Zeichnen M = MOVETO , D = DRAW
530 PRINT "M448 140 D448 38"
540 GOSUB 800
550 I = 23
560 PRINT "G2 01" :REM Gepunktete Linie zeichnen
570 GOSUB 710
580 I = 28
590 PRINT "G2 02" :REM Gestrichelte Linie zeichnen
600 GOSUB 710
610 I = 33
620 PRINT "G2 00" :REM Durchgezogene Linie-zeichnen
630 GOSUB 710
640 CALL &000
650 PRINT "A" :REM Sprung zur Hardcopyroutine
660 GOSUB 820 :REM Graphik aus
670 CLOSE#0:END
680 K=SIN((H-INT(H/11)*1)/1*2*3.14) :REM Errechnung des Biorhythmus
690 L=50+INT(50*K+.4)
700 RETURN
710 H=E*30.4+F*365.25-G :REM Graphische Darstellung der Kurve
720 GOSUB 680
    
```

```

730 PRINT "M".76.L+40
740 FOR X=1 TO 62
750 H=X/2+*30.4+F*365.25-G
760 GOSUB 680
770 PRINT "D".X+76.L+40
780 NEXT:RETURN
790 REM TAGESMARKIERUNG
800 FOR N=45 TO 135 STEP 10
810 NEXT:RETURN
820 TS=BYTES(#0) :REM Taturabfrage
830 IF TS = "1" THEN 110
840 IF TS = "2" THEN 180
850 IF TS = "3" THEN 420
860 RETURN
870 END
880 FOR H=6000 TO 660D4 :REM Laden des H-Copy Maschinenprogramms
890 READ HC :REM ab Adresse 6000H
900 POKE H,HC
910 NEXT H
920 DATA &CD,&03,&F0,&FE,&48,&6A,&11,&60,&FE,&4D,&CA,&10,&80,&C3,&00,&60,&60
930 DATA &C9,&21,&600,&60,&39,&22,&CE,&60,&31,&6E,&61,&21,&8D,&02,&22,&6D
940 DATA &60,&6C,&64,&66,&60,&3E,&43,&F5,&6D,&79,&60,&6C,&64,&60,&6C,&6F,&60
950 DATA &F1,&3D,&C2,&26,&60,&3E,&60,&60,&6D,&3E,&68,&6D,&63,&6A,&5D,&3,&6B,&2A
960 DATA &CE,&60,&F9,&C3,&00,&60,&62,&1,&5D,&60,&6C,&64,&60,&62,&1,&68,&60,&64E
970 DATA &E5,&6C,&60,&6F,&F0,&E1,&23,&3E,&FF,&8E,&6E,&C2,&64,&F,&60,&6C9,&61B,&640,&6D
980 DATA 80A,80A,80A,80A,81B,833,818,6FF,620,620,620,620,620,620,620,620,620,620
990 DATA &20,&20,&20,&20,&20,&81B,&64B,&600,&601,&6FF,&605,&609,&60E,&600,&63E,&6FE,&6D3
1000 DATA &8B,&63E,&6FF,&6D3,&68A,&62A,&6D0,&660,&62B,&622,&6D0,&660,&67C,&62F,&6D3,&68B
1010 DATA &7D,&62F,&621,&6D2,&660,&6D3,&689,&6DB,&68D,&6DB,&60A,&607,&6D2,&699,&660,&607
1020 DATA &7E,&617,&677,&6DB,&68D,&623,&60D,&6C2,&699,&660,&605,&6C2,&67B,&660,&6C9,&621
1030 DATA &6D2,&660,&60E,&600,&67E,&62F,&6E5,&6C5,&64F,&6CD,&60F,&6F0,&6C1,&6E1,&623,&60D
1040 DATA &C2,&684,&660,&60E,&60D,&6CD,&60F,&6F0,&60E,&60A,&6CD,&60F,&6F0,&6C9,&600,&600
1050 DATA &600,&602,&68E,&600,&6FF
1060 RETURN
1070 PRINT GES :REM Graphik ein
1080 PRINT "ZY0" :REM Bildschirm löschen, Seitennum. aus
1090 PRINT "G2 4" :REM Schritt schrag
1100 PRINT "M150 200" :REM Schrittgröße X * 6, Y * 4
1110 PRINT "G3 64" :REM Zeichen schreiben
1120 PRINT "Bm a b o" :REM Schriftgröße
1130 PRINT "MO 100 G3 $7A"
1140 PRINT "BBiorhythmus"
1150 PRINT "G2 0 G3 $21 M10 50 BBitte U druecken dann geht's gleich weiter!!!!
1160 PRINT "G3 $11 A" :REM Schriftgröße normal und Graphik aus
1170 RETURN
    
```

**** Biorhythmus ****
 Geburts.Dat. : 20 . 5 . 1940 Körperl.: 41 % - - - - -
 Vorhers.Dat. : 16 . 6 . 1987 Seelisch: 63 % - - - - -
 Geistig : 51 % -----

0 = Ende 1 = Neues Geb.-Datum 2 = Neues Vorh.-Datum 3 = Graphik, dann mit H = H.Copy M = MENU



FÜR 68000-EINSTEIGER Routinen für die SMART WATCH

von Klaus Janßen
 Seit einiger Zeit wird für den NDR-Klein-Computer ein neuer Uhrenbaustein angeboten. Es handelt sich hier um das EC DS1216 von Dallas Semiconductor. Der Baustein sieht aus wie eine dicke 28polige IC-Fassung und wird einfach unter einen RAM-Baustein von 8 KByte gesteckt. Das IC verfügt über eine Uhrzeit- und Datumfunktion und puffert das RAM-

IC, so daß dessen Daten auch nach dem Abschalten des Rechners erhalten bleiben. Hierfür sorgt eine eingebaute Lithiumbatterie, die laut Hersteller mindestens 10 Jahre hält. Eine eingebaute Logik regelt den Zugriff zu den Uhrenfunktionen. Normalerweise leitet das IC alle Zugriffe auf den RAM-Baustein weiter. Erkennt die Logik jedoch ein bestimmtes Bitmuster auf dem nie-

derwertigsten Datenbit, so aktiviert sie den Zugriff auf die Uhrenregister. Dieses Bitmuster umfaßt 64 Bits. Die Wahrscheinlichkeit, daß dieses Bitmuster zufällig auftritt, liegt unter 10⁻¹⁹. Die Uhr umfaßt 8 Register, die immer komplett gelesen oder beschrieben werden müssen. Die Informationen über Datum und Uhrzeit sind BCD-codiert. Der Aufbau der Register ist wie folgt:

Mit OSC=0 wird die Uhr gestartet, mit OSC=1 angehalten. Mit RES=0 bewirkt eine log. Null an Pin 1 den Abbruch des Datentransfers, mit RES=1 wird der Pegel an Pin 1 ignoriert. Einige Bits in den Registern sind permanent auf 0.

Im folgenden Listing wird das IC DS1216 softwaremäßig unterstützt. Die Routinen SWACCESS, SWREAD und SWWRITE besorgen den direkten Zugriff auf die Uhr. Die Routinen GETUHR2 und SETUHR2 wandeln das Datenformat der Uhr so um, daß es kompatibel zu den Grundprogramm-routinen GETUHR und SETUHR ist. DATE und TIME geben Datum und Uhrzeit auf die C02-Schnittstelle aus. Dabei wird die Uhrzeit im Format hh:mm:ss (Stunden, Minuten und Sekunden) und das Datum im Format wo, dd.mm.yy (Wochentag, Tag, Monat und Jahr) ausgegeben. Die Routine DATIM gibt sowohl Datum als auch Uhrzeit aus. Stellen kann man die Uhr mit der Routine SDATIM. Dazu muß vorher der String BSPDATE auf die aktuelle Zeit eingestellt werden. Das Format entspricht dem der Grundprogramm-routine SETUHR.

Im Beispiellisting wird die Uhr auf der Speicheradresse \$EE000 erwartet. Wer die Uhr auf einer anderen Adresse einsetzen möchte, muß diese mit EQU vereinbaren.

Register	Bit 7	Bit 0
0	1/10 sec	1/100 sec
1	0 10 sec	Sekunden
2	0 10 min	Minuten
3	0 0 10 h	Stunden
4	0 0 OSC RES 0	Wochentag
5	0 0 10 Tage	Tage
6	10 Monate	Monate
7	10 Jahre	Jahre

Wer sich das Abtippen des Listings ersparen möchte, kann es gegen Einsendung einer formatierten Diskette (5 1/4" oder 3 1/2") und 10,- in Briefmarken oder

auf V-Scheck direkt von mir beziehen (unter JADOS):

Klaus Janßen,
Hanninxweg 74, 4150 Krefeld 1.

```

*****
* Ansprechen des Uhrenbausteins DS1216
*-----
* Routinen:
* date -> Datum in Wochentag, den dd.mm.yy
*         ausgeben
* datim -> Datum und Uhrzeit ausgeben
* getuhr2 -> Uhr lesen wie im Grundprogramm
* schreibe -> Textstring ausgeben
* sdatim -> Datum und Uhrzeit setzen
* setuhr2 -> Uhr setzen wie im Grundprogramm
* swaccess -> Baustein ansprechen
* swread -> Register auslesen
* swwrite -> Register setzen
* time -> Uhrzeit in hh:mm:ss ausgeben
*
* Datum: 04.04.1987
* Autor: Klaus Janssen
*-----
* Adresse im Speicherraum, ueber die der
* Baustein angesprochen wird.
* Muss bei Bedarf angepasst werden
swad EQU $EE000 * Uhradresse

* Bibliothekskopf
kopf: dc.l $55AA0180
      dc.b 'DS1216'
      dc.l datim-kopf
      dc.l pende-kopf
      dc.b 1,0,0,0
      dc.l 0,0

*-----
date: * Datum ausgeben
* Format: Wochentag, den dd.mm.yy
*-----
movem.l d0/d7/a0-a2/a6,-(a7) * Reg. retten
lea buffer(pc),a0 * Puffer adressieren
bsr getuhr2 * Uhrzeit lesen
movea.l a0,a1 * Uhrpuffer in A1
lea outbuf(pc),a0 * Ausgabepuffer in A0
    
```

```

cir d0
move.b 5(a1),d0 * Wochentag holen
subq.b #1,d0 * und als Index aufbereiten
asl #2,d0 *
lea wotab(pc),a2 * Wochentagstabelle adressieren
movel 0(a2,d0),(a0)+ * und in Ausgabepuffer
move.b 2(a1),d0 * Datum holen
moveq #41,d7 * in ASCII wandeln
trap #1
move.b #',(a0)+ * Trennzeichen
move.b 3(a1),d0 * Monat holen
moveq #41,d7 * in ASCII wandeln
trap #1
move.b #',(a0)+ * Trennzeichen
move.b 4(a1),d0 * Jahr holen
moveq #41,d7 * in ASCII wandeln
trap #1
clr.b (a0) * mit 0 terminieren
lea outbuf(pc),a0 * Datum ausgeben
bsr schreibe
movem.l (a7)+,d0/d7/a0-a2/a6 * Reg. wiederherstellen
rts * date

*-----
datim: * Datum und Uhrzeit ausgeben
*-----
movem.l a0,-(a7) * Reg. retten
bsr date * Datum anzeigen
lea space(pc),a0 * einige Leerzeichen lassen
bsr schreibe
bsr time * Uhrzeit ausgeben
movem.l (a7)+,a0 * Reg. wiederherstellen
rts * datim

*-----
getuhr2: * Liefert Zeit und Datum
* wie die Routine getuhr im Grundprogramm
* in A0: Zeiger auf Buffer
* ===Format des Buffers in BCD codiert===
* Stunden Minuten Datum Monat
* Jahr Wochentag Sekunde
*-----
    
```

```

movem.l d0-d1/a0-a2,-(a7) * Reg. retten
move.l a0,d1 * A0 retten
lea xlatbuf(pc),a0 * Puffer adressieren
bsr swread * Uhrregister lesen
and.b #$0F,4(a0) * Modus ausblenden
movea.l a0,a1 * Uhrregister in A1
movea.l d1,a0 * Ablagebuffer in A0
lea xlattab(pc),a2 * Tabelle in A2
move #6,d1 * Schleifenzaehler
clr d0 * D0 loeschen

guhrlp:
move.b 0(a2,d1.w),d0 * Tabellenwert holen
move.b 0(a1,d0.w),0(a0,d1.w) * umkopieren
dbf d1,guhrlp *
movem.l (a7)+,d0-d1/a0-a2 * Reg. wiederherstellen
rts * getuhr2

xlatbuf: ds.b 8 * Puffer fuer Uhr
xlattab: dc.b 3,2,5,6,7,4,1 * Uebersetzungstabelle
ds 0

*-----
schreibe: * Textstring ausgeben
* in A0: Zeiger auf Text
* Text muss mit 0 terminiert sein
*-----
movem.l d0/d7/a0/a6,-(a7) * Reg. retten
schrlp:
move.b (a0)+,d0 * Textzeichen in D0
beq.s schrend * Textende
moveq #33,d7 * C02
trap #1 * in Grundprogramm
bra.s schrlp
schrend:
movem.l (a7)+,d0/d7/a0/a6 * Reg. wiederherstellen
rts * schreibe

*-----
sdatim: * Zeit und Datum einstellen
* Die Zeit- und Datumangabe muss der
* tatsaechlichen Zeit angepasst werden
*-----
lea bspdate(pc),a0 * Beispieldatum holen
bsr setuhr2 * und einstellen
rts * sdatim

*-----
setuhr2: * Setzt Zeit und Datum
* wie die Routine setuhr im Grundprogramm
* in A0: Zeiger auf Buffer
* ===Format des Buffers in BCD codiert===
* Stunden Minuten Datum Monat
* Jahr Wochentag Sekunde
*-----
movem.l d0-d1/a0-a2,-(a7) * Reg. retten
movea.l a0,a1 * Ablagebuffer in A1
lea xlatbuf(pc),a0 * Puffer fuer Uhrregister in A0
lea xlattab(pc),a2 * Tabelle in A2
move #6,d1 * Schleifenzaehler
clr d0 * D0 loeschen

suhrlp:
move.b 0(a2,d1.w),d0 * Tabellenwert holen
move.b 0(a1,d1.w),0(a0,d0.w) * umkopieren
dbf d1,suhrlp *
clr.b 0(a0) * 1/100 sec auf 0
and.b #$7F,1(a0) * unbenutzte Bits auf 0 setzen
and.b #$7F,2(a0) *
and.b #$3F,3(a0) * 24-Stunden Modus
and.b #$0F,4(a0) * Oszillator an
or.b #$10,4(a0) * Resetpin ignorieren
and.b #$3F,5(a0) * unbenutzte Bits auf 0 setzen
and.b #$1F,6(a0) *
bsr swwrite * Uhrregister setzen
movem.l (a7)+,d0-d1/a0-a2 * Reg. wiederherstellen
rts * setuhr2

*-----
pattern: dc.b $C5,$3A,$A3,$5C,$C5,$3A,$A3,$5C
* Suchmuster
*-----
swaccess: * Zugang ueber Suchmuster
*-----
movem.l d0-d2/a0-a2,-(a7) * Reg. retten
movea.l #swad,a1 * Uhradresse in A1
move.b (a1),d0 * Dummy Read
lea pattern(pc),a0 * Suchmuster adressieren
move #7,d1 * Bytecounter

swacl:
clr d0
move.b (a0)+,d0 * Suchmusterbyte holen
move #7,d2 * Bitcounter
    
```

```

swac2:
move.b d0,(a1) * Bit D0 senden
ror.b #1,d0 * Naechstes Bit holen
dbf d2,swac2 * ENDE Bit-Loop
dbf d1,swacl * ENDE Byte-Loop
movem.l (a7)+,d0-d2/a0-a2 * Reg. wiederherstellen
rts * swaccess

*-----
swread: * Uhrenregister lesen
* in A0: Zeiger auf Buffer
* ===Format des Buffers in BCD codiert===
* 1/100sec Sekunde Minuten Stunden
* Wochentag Datum Monat Jahr
*-----
movem.l d0-d5/a0-a2,-(a7) * Reg. retten
movea.l #swad,a1 * Uhradresse in A1
move.b (a1),d3 * Ramzelle retten
bsr swaccess * Uhr ansprechen
move #7,d1 * Bytecounter

swrd1:
clr d0
move #7,d2 * Bitcounter

swrd2:
move.b (a1),d5 * Bit D00 holen
and.b #1,d5 * Bit D01-D07 ausblenden
or.b d5,d0 *
ror.b #1,d0 * Naechstes Bit
dbf d2,swrd2 * ENDE Bit-Loop
move.b d0,(a0)+ * Byte in Buffer ablegen
dbf d1,swrd1 * ENDE Byte-Loop
move.b d3,(a1) * Ramzelle wiederherstellen
movem.l (a7)+,d0-d5/a0-a2 * Reg. wiederherstellen
rts * swread

*-----
swwrite: * Uhrenregister setzen
* in A0: Zeiger auf Buffer
* ===Format des Buffers in BCD codiert===
* 1/100sec Sekunde Minuten Stunden
* Wochentag Datum Monat Jahr
*-----
movem.l d0-d3/a0-a2,-(a7) * Reg. retten
movea.l #swad,a1 * Uhradresse in A1
move.b (a1),d3 * Ramzelle retten
bsr swaccess * Uhr ansprechen
move #7,d1 * Bytecounter

swwr1:
clr d0
move.b (a0)+,d0 * Byte holen
move #7,d2 * Bitcounter

swwr2:
move.b d0,(a1) * Bit D0 senden
ror.b #1,d0 * Naechstes Bit
dbf d2,swwr2 * ENDE Bit-Loop

dbf d1,swwr1 * ENDE Byte-Loop
move.b d3,(a1) * Ramzelle wiederherstellen
movem.l (a7)+,d0-d3/a0-a2 * Reg. wiederherstellen
rts * swwrite

*-----
time: * Uhrzeit ausgeben
* Format: hh:mm:ss
*-----
movem.l d0/d7/a0/a1/a6,-(a7) * Reg. retten
lea buffer(pc),a0 * Puffer adressieren
bsr getuhr2 * Uhrzeit lesen
movea.l a0,a1 * Uhrpuffer in A1
lea outbuf(pc),a0 * Ausgabepuffer in A0
move.b 0(a1),d0 * Stunden holen
moveq #41,d7 * in ASCII wandeln
trap #1
move.b #'',(a0)+ * Trennzeichen
move.b 1(a1),d0 * Minuten holen
moveq #41,d7 * in ASCII wandeln
trap #1
move.b #'',(a0)+ * Trennzeichen
move.b 6(a1),d0 * Sekunden holen
moveq #41,d7 * in ASCII wandeln
trap #1
clr.b (a0) * mit 0 terminieren
lea outbuf(pc),a0 * Uhrzeit ausgeben
bsr schreibe
movem.l (a7)+,d0/d7/a0/a1/a6 * Reg. wiederherstellen
rts * time

*-----
* Texte
wotab: dc.b 'Mo, Di, Mi, Do, Fr, Sa, So,
space: dc.b ' ',0
    
```

* Beispieldatum: Sa, 04.04.87 16:09
 bspdate: dc.b \$16,\$09,\$04,\$04,\$07,\$06,\$00
 ds 0

* Variablenbereich
 outbuf: ds.b 22 * Puffer fuer Textausgabe
 buffer: ds.b 8 * Puffer fuer Uhrenregister

* Letztes Label
 pende:

Apfelmännchen mit der COL 256

von Kei Thomsen

Zum Programm:

Das Programm entspricht dem aus der LOOP 5, Seite 16, von Rolf-Dieter Klein. Es wurde nur einiges an der Geschwindigkeit verändert und die Ausgabe auf die COL 256 eingebracht. Weiterhin ist ein Factor zur Farbverfälschung (apfelror) mit eingebracht. Dieser Factor verändert die Farbe durch verschieben der Bits um diese Anzahl „apfelror“. Dadurch ergeben sich schönere Farbabstufungen innerhalb des Bildes.

Es wird mit einer Auflösung von 256 * 256 Punkten in 256 Farben gearbeitet. Tests haben ergeben, daß das Bild bei höheren

Auflösungen mit weniger Farben (16) nicht mehr so gut aussieht. Als weitere Werte sollte man die folgenden Werte probieren:

```
move.1 #-250,pmin
move.1 #100,pmax
move.1 #850,qmin
move.1 #1100,qmax
move.b #02,apfelror
```

Apfelmännchen in Farbe

Ein besonderer Reiz Apfelmännchen darzustellen, ist das Darstellen von Ausschnitten in Farbe. Dieses ist mit der COL 256 recht einfach und ergibt sehr hübsche und bunte Bilder. Da die COL 256 insgesamt 256 Farben darstellen kann, ergibt sich aus einigen Bildern eine ganz besondere Farbenpracht. Mit dem Programm ist es möglich, Apfelmännchen bis zu einer bestimmten Tiefe zu errechnen.

Dieses hängt von der Genauigkeit der Arithmetik ab. Da hier nur Integer-Werte benutzt werden, ist die Genauigkeit sehr schnell verbraucht. Es existieren bei mir aber schon Versionen, die mit einer selbstgeschriebenen Fixed-Point Arithmetik arbeiten. Weiterhin wird seit kurzem auch der 68881 Floating-Point-Coprozessor des 68020 unterstützt. Dieses ergibt eine extreme Geschwindigkeitssteigerung gegenüber eigener Arithmetik. Die Berechnungen können nun auch bis zu einer fast unendlichen Tiefe berechnet werden. Bei diesen sehr tiefen Berechnungen kommt die COL 256 voll zur Geltung, da das gesamte Farbenspektrum ausgenutzt wird.

Diese Programme sind allerdings sehr lang, können aber direkt bei mir bezogen werden.

Rolf-D.Klein 68020/68881 Assembler 5.0 (C) 1985, Seite 1

```

JAI000          # DARSTELLUNG VON APFELMÄNNCHEN
JAI000          # RDK 20.9.85 (LOOP 5 SEITE 16)
JAI000          # VERSION FUER COL 256 AM 27.5.1986 VON KEI THOMSEN
JAI000          #
JAI000 00000000  PMIN: DC.L 0
JAI004 00000000  PMAX: DC.L 0
JAI008 00000000  QMIN: DC.L 0
JAI00C 00000000  QMAX: DC.L 0
JAI010 00000000  QMAXQMIN: DC.L 0
JAI014 00000000  PMAXPMIN: DC.L 0
JAI018          #
JAI019 00          APFELROR: DC.B 0
JAI019 00          DS 0
JAI01A          #
= 00F42400        M EQU 16#1000#1000
= 00000932        ANZAHL EQU 50          # BERECHNUNGSTIEFE (MAXTIEFE)
= 00000000        ENDFARBE EQU #00       # ENDFARBE (FARBE BEI MAXTIEFE)
JAI01A          #
JAI01A 6100 0100  START:
JAI01A          BSR COLINIT          # INITIALISIEREN DER GRAFIK
JAI01E          #
JAI01E 23FC FFFF736 MOVE.L #-2250,PMIN # GRENZWERTE FESTLEGEN
JAI024 003A1C00
JAI028 23FC 000002EE MOVE.L #750,PMAX
JAI02E 003A1C04
JAI032 23FC FFFF7A24 MOVE.L #-1500,QMIN
JAI038 003A1C08
JAI03C 23FC 000005DC MOVE.L #1500,QMAX
JAI042 003A1C0C
JAI046 13FC 0000        MOVE.B #00,APFELROR # FARBVERFÄLSCHUNG
JAI04A 003A1C18
JAI04E          #
JAI04E 23F9 003A1C04 MOVE.L PMAX,PMAXPMIN # P-SCHRITT BERECHNEN
JAI054 003A1C14
JAI058 2039 003A1C00 MOVE.L PMIN,DO
JAI05E 91B9 003A1C14 SUB.L DO,PMAXPMIN
JAI064          #
JAI064 23F9 003A1C0C MOVE.L QMAX,QMAXQMIN # Q-SCHRITT BERECHNEN
JAI06A 003A1C10
JAI06E 2039 003A1C08 MOVE.L QMIN,DO
JAI074 91B9 003A1C10 SUB.L DO,QMAXQMIN
JAI07A          #
JAI07A 4281          CLR.L D1          # X-COUNTER AUF 0
JAI07C          YLOOP:
JAI07D 4282          CLR.L D2          # Y-COUNTER AUF 0
JAI07E          YLOOP:
JAI07E 2639 003A1C14 MOVE.L PMAXPMIN,D3
JAI084 C7C1          MULS D1,D3
JAI086 87FC 00FF        DIVS #255,D3
JAI08A D679 003A1C02 ADD PMIN+2,D3
    
```

```

JAI090 2839 003A1C10 MOVE.L QMAXQMIN,D4
JAI096 C9C2          MULS D2,D4
JAI098 89FC 00FF        DIVS #255,D4
JAI09C D879 003A1C0A ADD QMIN+2,D4
JAI0A2 4285          CLR.L D5
JAI0A4 4286          CLR.L D6
JAI0A6 4287          CLR.L D7
JAI0AA 4BE7 6000        MOVEM.L D1-D2,-(A7)
JAI0AC          ITERAT:
JAI0AC 3205          MOVE D5,D1
JAI0AE C3C1          MULS D1,D1
JAI0B0 2096          MOVE D5,DO
JAI0B2 C1C0          MULS D0,DO
    
```

Rolf-D.Klein 68020/68881 Assembler 5.0 (C) 1985, Seite 2

```

JAI0B4 9280          SUB.L D0,D1
JAI0B6 93FC 03EB        DIVS #1000,D1
JAI0BA D243          ADD D3,D1
JAI0BC 3405          MOVE D5,D2
JAI0BE C5C6          MULS D5,D2
JAI0C0 85FC 01F4        DIVS #500,D2
JAI0C4 D444          ADD D4,D2
JAI0C6 3A01          MOVE D1,D5
JAI0C8 3C02          MOVE D2,D6
JAI0CA 5247          ADDD #1,D7
JAI0CC C3C1          MULS D1,D1
JAI0CE C5C2          MULS D2,D2
JAI0D0 D282          ADD.L D2,D1
JAI0D2 0C81 00F42400  CMP.L #M,D1
JAI0D8 6C0A          BGE.S AUSGABE
JAI0DA 0C47 0032        CMP #ANZAHL,D7
JAI0DE 6794          BEQ.S AUSGABE
JAI0E0 6000 FFCA        BRA ITERAT
JAI0E4          AUSGABE:
JAI0E4 4CFF 0006        MOVEM.L (A7)+,D1-D2
JAI0E8 6606          BNE.S OUTPUT # HAENGT VON * CMP #ANZAHL,D7 * AB
JAI0EA 103C 0000        MOVE.B #ENDFARBE,DO # WENN TIEFE = MAXTIEFE (ANZAHL) DANN
JAI0EE 600A          BRA.S OUTPUT # FARBE = ENDFARBE
JAI0F0          OUTPUT:
JAI0F0 1007          MOVE.B D7,DO # TIEFE ALS FARBE DARSTELLEN
JAI0F2 1E39 003A1C18  MOVE.B APFELROR,D7 # FARBVERFÄLSCHUNG
JAI0F8 EE38          ROR.B D7,DO
JAI0FA          OUTPUT1:
JAI0FA 48A7 6000        MOVEM.W D1-D2,-(A7)
JAI0FE 6100 009C        BSR SETFOOT # PUNKT SETZEN
JAI102 4C9F 0006        MOVEM.W (A7)+,D1-D2
JAI106          #
JAI106 5242          ADDD #1,D2 # Y UM 1 VERGROESSERN
JAI108 0C42 0100        CMP #256,D2
JAI10C 6D00 FF70        BLT YLOOP
    
```

```

JA1D10 5241      ADD #1,D1          # X UM 1 VERGROESSERN
JA1D12 0C41 0100  CMP #256,D1
JA1D16 6D00 FF64  BLT XLOOP
JA1D1A
JA1D1A 4E75      RTS
JA1D1C
JA1D1C          # COL 256 UNTERPROGRAMM
JA1D1C
= 00000001      CPU EDU 1          # 68008 = 1 68000 = 2 68020 = 4
JA1D1C
= FFFFFFFAC    CRT EQU $FFFFFFAC*CPU # IO ADRESSEN DER COL 256
= FFFFFFFAD    CRTD EQU $FFFFFFAD*CPU
= FFFFFFFAE    CRTB EQU $FFFFFFAE*CPU
JA1D1C
= 000DC000     COLRAM EQU $DC000    # RAM ADRESSE DER COL 256
JA1D1C
JA1D1C
JA1D1C          COLINIT:          # INITIALISIEREN DER GRAFIK
JA1D1C          # UND LOESCHEN DER BILDSCHIRM
JA1D1C 41FA 0020  LEA TAB(PC),A0
JA1D20 4241      CLR D1
JA1D22 363C 000E  MOVE.W #LENGTH-1,D3
JA1D24
JA1D24 LPP:
JA1D26 13C1 FFFFFFFAC  MOVE.B D1,CRT
JA1D2C 13DB FFFFFFFAD  MOVE.B (A0)+,CRTD
JA1D32 5241      ADDQ.W #1,D1
JA1D34 51CF FFF0     DERA D3,LPP
    
```

Rolf-D.Klein 68020/68881 Assembler 5.0 (C) 1985, Seite 3

```

JA1D38
JA1D38 6100 0014    BSR CLR
JA1D3C
JA1D3C 4E75      RTS
JA1D3E
JA1D3E          TAB:
JA1D3E 6F 40 50 07 4D  DC.B 111,64,80,7,77,0,64,70,0,3,0,0,0,0,0
JA1D43 00 40 46 00 03
JA1D48 00 00 00 00 00
= 0000000F      LENGTH EQU 1-TAB
JA1D4D
JA1D4D 00        DS 0
JA1D4E
JA1D4E          CLR:
JA1D4E 4280      CLR.L D0
    
```

```

JA1D50 13FC 0080    MOVE.B #80,CRTB
JA1D54 FFFFFFFAE
JA1D58 6100,002E    BSR CLR
JA1D5C 13FC 0081    MOVE.B #81,CRTB
JA1D60 FFFFFFFAE
JA1D64 6100 0022    BSR CLR
JA1D68 13FC 0082    MOVE.B #82,CRTB
JA1D6C FFFFFFFAE
JA1D70 6100 0016    BSR CLR
JA1D74 13FC 0083    MOVE.B #83,CRTB
JA1D78 FFFFFFFAE
JA1D7C 6100 000A    BSR CLR
JA1D80 4239 FFFFFFFAE  CLR.B CRTB
JA1D84 4E75      RTS
JA1D88
JA1D88          CLR:
JA1D88 41F9 000DC000  LEA COLRAM*CPU,A0 # GEHT AUCH SCHNELLER, DAFUER
JA1D8E 363C JFFF    MOVE.W #54000-1,D3 # ABER ALLGEMEIN!
JA1D92
JA1D92 4210      CLR.B (A0)
JA1D94 5288      ADDQ.L #CPU,A0
JA1D96 51CB FFFA  DBRA D3,CLRLOOP
JA1D9A 4E75      RTS
JA1D9C
JA1D9C          SETFOOT:
JA1D9C 3F02      MOVE.W D2,-(A7) # PUNKT BERECHNEN
JA1D9E 4602      NOT.B D2
JA1DA0 EC5A      ROR.W #5,D2
JA1DA2 0092 0080  OR.B #80,D2
JA1DA6 11C2 FFAE  MOVE.B D2,CRTB.S
JA1DAA E44A      LSR.W #2,D2
JA1DAC 1401      MOVE.B D1,D2
JA1DAE
JA1DAE          # LSL.L #1,D2 # NUR FUER 68000
JA1DAE          # LSL.L #2,D2 # NUR FUER 68020
JA1DAE
JA1DAE 41F9 000DC000  LEA COLRAM*CPU,A0
JA1DB4 D1C2      ADDQ.L D2,A0
JA1DB6 341F      MOVE.W (A7)+,D2
JA1DB8 1080      MOVE.B D0,(A0) # PUNKT DARSTELLEN
JA1DBA 4238 FFAE  CLR.B CRTB.S
JA1DBE 4E75      RTS
JA1DC0
JA1CA2 Ende-Symboltabelle
    
```

68000 — YOGIDOS UND JADOS, RL-BASIC

Funktionenplot mit RL-BASIC

PLOT65 zeichnet (fast) beliebige Funktionen

Von Uwe Koch, Lüdenscheid

Benötigter Rechner: 680xx-System mit RL-BASIC

Voranmerkung der Redaktion: Mit diesem Programm zeigt Uwe Koch einige der vielen Möglichkeiten, die im NDR-Computerschulmann und von vielen Anwendern (noch) nicht eingesetzt worden sind.

Programme wie diese sollen als Anwendungsprogramme diese Möglichkeiten aufzeigen und Sie, lieber Leser, dazu ermuntern, auch selbst nach weiteren Anwendungen zu suchen.

Das Programm ist noch mit der alten Version von RL-BASIC geschrieben worden; daher finden Sie noch nicht die Strukturbefehle, die Herr Koch sicher angewendet hätte!

Bevor Sie das Listing abtippen: Das Programm ist mit auf der JADOS-TOOL-Dis-

kette enthalten! Dies nützt natürlich nur den Lesern etwas, die neben RL-BASIC auch JADOS im Einsatz haben!

Das Programm heißt „Plot 68“ und dient zur grafischen Darstellung von mathematischen Funktionen. Dieses Programm nutzt vor allem die grafischen Möglichkeiten des NDR-Rechners. So wird z.B. das Bildfenster mit der Funktionszeichnung durch geschicktes Umschalten der Bildseiten einzeln gelöscht und das Koordinatenfenster wird mit dem Grafikprozessor aus gepunkteten und gestrichelten Linien erzeugt. Die Eingaben erfolgen ebenfalls ausschließlich über INKEYS-Befehle, weil dadurch einerseits einzelne Zeichen ausgewählt werden können (andere werden ignoriert) und andererseits die Ausgabe auf dem Bildschirm nicht auf den Seiten 0 und 1 stattfindet. Die Bildausgabe findet bei diesem Programm nur auf den Seiten 2 und 3 statt. Das Bild besteht aus sechs „Fenstern“, die den Bildschirm in zusammengehörende Einheiten teilen. Neben dem Programmnamen werden die Fenster zur Anzeige und Eingabe der Graphdaten und zur Anzeige von Programmierungen

benötigt. Das größte Fenster ist zur Darstellung der Funktion vorgesehen. In der untersten „Statuszeile“ werden die Programmierungen ausgegeben. Die drei anderen Felder dienen zur Eingabe, wobei man bei der Eingabe von (SPACE) in das jeweils nächste Feld springen kann, die Eingabe von (CR) übernimmt den alten Wert. Neben den Anfangs- und Endwerten für die X- und Y-Achse muß bei 'Raster' mit 'j' oder 'n' geantwortet werden, je nachdem, ob ein Koordinatengitter eingezeichnet werden soll oder nicht. Bei 'Anzahl' ist die Anzahl der Graphen einzugeben, die übereinander in ein Bild gezeichnet werden sollen. Möglich ist hier '1' bis '9', andere Eingaben werden ignoriert. Bei 'Punkt' muß mit 'j' oder 'n' geantwortet werden. Bei 'j' wird die Funktionszeichnung aus einzelnen Punkten aufgebaut, bei 'n' werden die Punkte mit Linien verbunden.

Die wichtigste Eingabe ist dann die Eingabe der Funktion. Hierbei muß die darzustellende Funktion in der üblichen Basic-Notation eingegeben werden, aber nur der Teil rechts vom Gleichheitszeichen. Beispiel:

Die Funktion $y = \sin(ax)$ soll dargestellt werden

Einzugeben ist: $\sin(K*y)$

Es dürfen die Funktionen +, -, *, /, ^, (,), sin, cos, tan, sqr, exp, log, abs, int und sgn verwendet werden, auch geschachtelt. Als Funktionswert wird immer 'x' benutzt und eine Konstante 'k' ist möglich. Die so eingegebene Funktion wird von dem Basic-Programm in die entsprechenden Basic-Token umgewandelt und in das Programm „eingepoked“. Es handelt sich also um ein selbstmodifizierendes Basic-Programm. Da die Umwandlung je nach Umfang der Funktion schon mehrere Sekunden benötigt, habe ich auf eine zusätzliche Syntax-Kontrolle verzichtet. Daher sollte die Funktion möglichst fehlerfrei eingegeben werden. Die Umwandlung geschieht in dem Unterprogramm ab Zeile 18000. Das Prinzip dieser Umwandlung funktioniert mit kleinen Änderungen auch bei anderen Basic-Interpretern.

Es werden nur die eindimensionalen Felder BEFEHL\$, BEFL und TKN benutzt, die die Schreibweise des Befehls, die Länge und das dazugehörige Token enthalten. Für andere Interpreter braucht nur das Feld mit den Token geändert werden. Außerdem muß bei Interpretern, die aus-

schließlich einstellige Token verwenden die Zeile 18150 ersatzlos gestrichen werden.

In Zeile 18220 wird der String, der die fertigen Token enthält (TKN\$), auf genau 30 Zeichen durch Leerzeichen verlängert. Dies ist nötig, damit die Zeile, in die dieser String „gepoked“ wird, eine konstante Länge behält. Sonst würde der Interpreter die nächste Zeile nicht mehr finden und wahrscheinlich abstürzen. Daher ist es auch wichtig, bei der Eingabe des Programms in der Zeile 20020 exakt 30 (beliebige) Zeichen hinter 'DEF FN F(X)==' einzugeben. Die Zeichen werden beim Programmablauf überschrieben, aber sie erzeugen bei der Eingabe den richtigen Abstand der Zeile 20030. Die Zeile 20020 muß im RAM von dem Basic-Programm gefunden werden können, um die Token einsetzen zu können. Da bei RL-BASIC das Programm-RAM jedoch beliebig festgelegt werden kann, muß das Programm die Zeile suchen. Als Kennung dafür dient die Zeile 20010 mit den drei '000'. Diese Bytekombination wird in Zeile 18230 - 18260 gesucht. Um das Programm zu beschleunigen, kann man den Anfangswert für die Suche in Zeile 18230 möglichst dicht vor die Zeile setzen. Dabei muß die echte RAM-Adresse als Dezimalzahl ein-

gesetzt werden. Bei einem Arbeitsbereich ab \$2400 liegt dieser Wert bei etwa \$18300. In Zeile 18280 wird dann der Tokenstring hinter das '='-Zeichen der DEF-Anweisung „gepoked“, dazu ist in Offset von 16 nötig. Dies ist der Abstand zwischen dem ersten 'C' und der ersten freien Stelle hinter dem '=' in Zeile 20020. Dieser Abstand entsteht durch die interne Kodierung der Basic-Zeilen.

Die Einsetzung der Funktion in die Programmzeile ist zwar etwas umständlich und langsam, jedoch ist es die einzige Möglichkeit, denn die Version, die in mc 10/86, Seite 74 beschrieben wurde, ist wohl nur in MBASIC möglich und setzt auch einen Ausbau mit Floppy-Laufwerk voraus. Meine Version läuft auch mit der EPROM-Version von RL-BASIC. Für den Z80 mit RDK-Basic 1.5 habe ich ein ähnliches Programm, das bis auf die „Fenster-technik“ (wegen fehlender WRITE-Befehle) die gleiche Funktion hat. Dabei wird für die Statuszeile der WRITE-Befehl imitiert, das ist für mehrere WRITES aber zu langsam. Z80-Anfänger, die diese Version haben wollen, teilen mir das bitte mit.

Uwe Koch,
Frankenstraße 25, 5880 Lüdenscheid,
Telefon: (/2351) 2 6192

```

1 REM * Funktionsplot-Programm für den NKC 680xx
2 REM * (C) Uwe Koch 10/86
3 REM * Version 1.1
4 REM
5 REM * Dieses Programm zeichnet eine beliebige mathematische Funktion.
6 REM * Das Koordinatensystem kann beliebig gewählt werden und ein
7 REM * Koordinatengitter kann wahlweise eingezeichnet werden.
8 REM * Die Funktion wird als String eingegeben und von dem Programm
9 REM * selbstständig in Token zerlegt und in das Programm eingetragen.
10 REM * Dabei wird aus Zeitgründen keine Syntaxprüfung vorgenommen.
11 REM * sodass die Funktion in der Basic-Notation korrekt eingegeben
12 REM * werden muss. Bei allen Eingaben kann mit <CR> der alte Wert
13 REM * uebernommen werden und mit <SPACE> kann zum naechsten Eingabefeld
14 REM * weitergesprungen werden. Falsche Eingaben werden ignoriert oder
15 REM * mit einer Fehlermeldung neu erbeten.
16 REM * Der Wert fuer die Variable PROG$ in Zeile 18130 bestimmt den
17 REM * Anfangswert fuer die Suche nach den drei '$', die den Beginn
18 REM * der DEF FN F(X) = -Zeile markieren. Bei der Eingabe der DEF-Zeile
19 REM * muessen exakt !! 30 Zeichen hinter dem '='-Zeichen eingegeben
20 REM * werden, damit der Anfang der naechsten Zeile korrekt ist.
21 REM * Z.B. 20020 DEF FN F(X)=.....<CR>
22 REM
23 REM * Bei Fragen oder Verbesserungsvorschlaegen bitte melden bei
24 REM * Uwe Koch
25 REM * Frankenstrasse 25
26 REM * 5880 Luedenscheid
27 REM * 02351/26192
28 REM
29 REM * Die Bewegung des Zeigers bei (A)nalyse geschieht wie bei WordStar
30 REM * durch a,s,d,f. Eine Feineinstellung ist mit '+' und '-' moeglich
31 REM * Mit 'e' wird die Routine wieder verlassen, mit 'x' kann der X-Wert
32 REM * mit voller Genauigkeit angezeigt werden, ebenso F(X) mit 'y' und
33 REM * F'(X) mit 'z', alle anderen Tasten fuehren zur gerundeten Anzeige.
40 REM
50 REM * *** Initialisierung ***
60 X1 = -1: X1$ = STR$(X1): X2 = 1: X2$ = STR$(X2)
70 Y1 = -1: Y1$ = STR$(Y1): Y2 = 1: Y2$ = STR$(Y2)
80 RASTER$ = "n": ANZAHL = 1: ANZAHL$ = STR$(ANZAHL): PUNKTES = "j"
90 K = 1: K$ = STR$(K): FUNKTES = "Titelbild": FUNKALTS = ""
100 VIN = 1
130 FOR SEITE = 2 TO 3: PAGE SEITE,0: CLPG : NEXT SEITE
140 GOSUB 11000: REM * Maske
150 GOSUB 13000: REM * Titelbild
160 STATUS$ = "Funktionsplot"
170 GOSUB 12000: REM * Statuszeile
180 GOSUB 17000: FOR I = 1 TO 1000: NEXT I
190 GOSUB 11000: REM * Bild loeschen
200 REM * *** Hauptprogramm ***
210 GOSUB 8000: REM * Anzeige der Graphdaten
220 GOSUB 9000: REM * Eingabe der Graphdaten
230 IF FUNKTES < > FUNKALTS THEN GOSUB 18000
240 GOSUB 1000: REM * Graph zeichnen
250 ANZAHL = ANZAHL - 1: ANZAHL$ = STR$(ANZAHL)
260 IF ANZAHL > 0 THEN 210
270 STATUS$ = "[E]nde [N]ochmal [A]nalyse": GOSUB 12000
280 AS = INKEY$: IF AS = "" THEN 280
290 IF AS = "E" OR AS = "e" THEN CLS : END
300 IF AS = "N" OR AS = "n" THEN ANZAHL = ANZAHL + 1: GOTO 210
310 IF AS = "A" OR AS = "a" THEN GOSUB 2000
320 GOTO 260
330 :
340 :
1000 REM * *** Zeichenprogramm ***
1010 PI = 3.1415926
1020 GOSUB 20000: REM * Funktionsdefinition
1030 XM = X2 - X1
1040 YM = Y2 - Y1
1050 XF = 1
1060 IF XM <= 2 THEN XF = 10
1070 IF XM > 20 THEN XF = 0.1
1080 IF XM > 200 THEN XF = 0.01
1090 IF XM > 2000 THEN XF = 0.001

```

```

1100 X1 = INT (X1 * XF + 0.5) / XF
1110 X2 = INT (X2 * XF) / XF
1120 YF = 1
1130 IF YM <= 2 THEN YF = 10
1140 IF YM > 20 THEN YF = 0.1
1150 IF YM > 200 THEN YF = 0.01
1160 IF YM > 2000 THEN YF = 0.001
1170 Y1 = INT (Y1 * YF + 0.5) / YF
1180 Y2 = INT (Y2 * YF) / YF
1190 XM = X2 - X1: YM = Y2 - Y1
1200 IF XM = 0 OR YM = 0 THEN GOSUB 5000
1210 Y0 = INT (400 * ABS (Y1) / YM) + 106
1220 GR = 390 / YM
1230 DX = XM / 390
1240 GOSUB 7000: REM * Koordinatenkreuz
1250 STATUS$ = "Funktion wird gezeichnet": GOSUB 12000
1260 X = X1: XA = 10: YA = Y0
1270 FOR XI = 10 TO 400
1280 Y = FN F(X)
1290 X = X + DX
1300 Y = INT (Y * GR + Y0)
1310 IF Y < 110 THEN 1370
1320 IF Y > 500 THEN 1390
1330 IF R = 1 THEN STATUS$ = "Funktion wird gezeichnet": GOSUB 12000
1340 PAGE 2,2: R = 0
1350 IF PUNKTES = "j" THEN PSSET XI,Y: ELSE LINE (XA,YA) - (XI,Y)
1360 XA = X1: YA = Y: NEXT XI: GOTO 1430
1370 STATUS$ = "Graph unterhalb des Bildbereiches"
1380 GOTO 1400
1390 STATUS$ = "Graph oberhalb des Bildbereiches"
1400 IF R = 0 THEN GOSUB 12000
1410 R = 1
1420 GOTO 1360
1430 RETURN
1440 :
1450 :
2000 REM * *** Analyse ***
2010 STATUS$ = "Analyse": GOSUB 12000
2020 X = X1: XPA = 20
2030 AS = INKEY$: IF AS = "" THEN 2030
2040 IF AS = "a" THEN X = X - DX * 10
2050 IF AS = "s" THEN X = X - DX
2060 IF AS = "d" THEN X = X - DX / 10
2070 IF AS = "+" THEN X = X + DX / 10
2080 IF AS = "d" THEN X = X + DX
2090 IF AS = "f" THEN X = X + DX * 10
2100 IF AS = "x" THEN GOSUB 2500
2110 IF AS = "y" THEN GOSUB 2520
2120 IF AS = "z" THEN GOSUB 2540
2130 IF AS = "e" THEN 2600
2140 IF X < X1 THEN X = X1
2150 IF X > X2 THEN X = X2
2160 XP = 10 + 390 * (X - X1) / XM
2170 GOSUB 16000: REM * Pfeil
2180 Y = FN F(X): YY = (FN F(X) - FN F(X + DX)) / DX
2190 XS = INT (X * 1000) / 1000: YS = INT (Y * 1000) / 1000
2200 YYS = INT (YY * 1000) / 1000
2210 STATUS$ = "x=" + STR$(XS) + " f(x)=" + STR$(YS)
2220 STATUS$ = "STATUS=" + " f'(x)=" + STR$(YYS): GOSUB 12000
2230 GOTO 2030
2240 :
2500 STATUS$ = "x = " + STR$(X): GOSUB 12000
2510 GOTO 2550
2520 STATUS$ = "f(x) = " + STR$(Y): GOSUB 12000
2530 GOTO 2550
2540 STATUS$ = "f'(x) = " + STR$(YY): GOSUB 12000
2550 AS = INKEY$: IF AS = "" THEN 2550
2560 RETURN
2570 :
2600 REM * Analyse beenden
2610 COLOR = 0

```

68 000 - YOGIDOS UND JADOS, RL-BASIC

```

2620 MOVETO XP,95: TURN TO 270: MOVE 15
2630 MOVETO XP,95: TURN TO 225: MOVE 7
2640 MOVETO XP,95: TURN TO 315: MOVE 7
2650 RETURN
2700 STATUS$ = "f'(x) = " + STR$(YY): GOSUB 12000
2710 RETURN
2720 :
2730 :
5000 REM " *** Fehlerhafte Eingabe ***
5010 STATUS$ = "Fehlerhafte Eingabe"
5020 GOSUB 12000
5030 FOR I = 0 TO 100
5040 SOUND $64,0,0,0,0,0,SFE,$10,0,0,0,$A,0,0,0
5050 NEXT I
5060 STATUS$ = "Graph-Daten-Eingabe": GOSUB 12000
5070 RETURN
5080 :
5090 :
7000 REM " *** Koordinatenkreuz ***
7010 STATUS$ = "Koordinatenkreuz wird gezeichnet": GOSUB 12000: REM " Status
7020 LINE (10,Y0) - (400,Y0)
7030 XS = 390 / XM / XF
7040 X = XI
7050 FOR I = 10 TO 400 STEP XS
7060 IF ABS(X) < ABS(DX) THEN GOSUB 7500: REM " Y-Achse
7070 LINE (I,Y0 - 8) - (I,Y0 + 8)
7080 IF RASTERS < > "j" THEN 7120
7090 POKE $FFFFF72,1: REM LINIE GEPUNKTET
7100 LINE (I,110) - (I,500)
7110 POKE $FFFFF72,0: REM LINIE NORMAL
7120 X = X + DX * XS
7130 NEXT I
7140 RETURN
7150 :
7160 :
7500 REM " *** Y-Achse ***
7510 LINE (I,110) - (I,500)
7520 DY = 390 / YM / YF
7530 FOR I = 110 TO 500 STEP DY
7540 IF ABS(J - Y0) < DY / 2 THEN 7570
7550 IF RASTERS = "j" THEN GOSUB 7900: REM " Y-Raster
7560 LINE (I - 4,J) - (I + 4,J)
7570 NEXT J
7580 RETURN
7590 :
7600 :
7900 REM " *** Y-Raster ***
7910 POKE $FFFFF72,2: REM " Linie gepunktet
7920 LINE (10,J) - (400,J)

7930 POKE $FFFFF72,0: REM " Linie normal
7940 RETURN
7950 :
7960 :
8000 REM " *** Anzeige der Graph-Daten ***
8010 WRITE 450,372,$11,X1$
8020 WRITE 450,352,$11,X2$
8030 WRITE 450,292,$11,Y1$
8040 WRITE 450,272,$11,Y2$
8050 WRITE 468,232,$11,RASTER$
8060 WRITE 468,192,$11,ANZAHL$
8070 WRITE 468,152,$11,PUNKT$
8080 WRITE 145,60,$21,FUNKT$
8090 WRITE 145,40,$21,KS
8100 RETURN
8110 :
8120 :
9000 REM " *** Eingabe der Graph-Daten ***
9010 STATUS$ = "Graph-Daten-Eingabe": GOSUB 12000: REM " Statuszeile
9020 XIN = 450:YIN = 372:GRIN = 17:LIN = 5:INAS$ = "-":INES$ = "9": GOSUB 15000
9030 IF INXS$ = "" THEN 9230
9040 IF INXS$ = "" THEN 9060
9050 X1 = VAL(IN$) * VIN:X1$ = STR$(X1)
9060 XIN = 450:YIN = 352:GRIN = 17:LIN = 5:INAS$ = "-":INES$ = "9": GOSUB 15000
9070 IF INXS$ = "" THEN 9230
9080 IF INXS$ = "" THEN 9100
9090 X2 = VAL(IN$) * VIN:X2$ = STR$(X2)
9100 IF X2 > X1 THEN 9130
9110 GOSUB 5000: REM " Eingabe-Fehler
9120 GOTO 9020
9130 XIN = 450:YIN = 292:GRIN = 17:LIN = 5:INAS$ = "-":INES$ = "9": GOSUB 15000
9140 IF INXS$ = "" THEN 9230
9150 IF INXS$ = "" THEN 9170
9160 Y1 = VAL(IN$) * VIN:Y1$ = STR$(Y1)
9170 XIN = 450:YIN = 272:GRIN = 17:LIN = 5:INAS$ = "-":INES$ = "9": GOSUB 15000
9180 IF INXS$ = "" OR INXS$ = "" THEN 9200
9190 Y2 = VAL(IN$) * VIN:Y2$ = STR$(Y2)
9200 IF Y2 > Y1 THEN 9230
9210 GOSUB 5000: REM " Eingabe-Fehler
9220 GOTO 9130
9230 XIN = 468:YIN = 232:GRIN = 17:LIN = 1:INAS$ = "j":INES$ = "n": GOSUB 15000
9240 IF INXS$ = "" THEN 9360
9250 IF INXS$ = "" THEN 9300
9260 RASTERS = IN$
9270 IF RASTERS = "j" OR RASTERS = "n" THEN 9300
9280 GOSUB 5000: REM " Eingabe-Fehler
9290 GOTO 9160
9300 XIN = 468:YIN = 192:GRIN = 17:LIN = 1:INAS$ = "1":INES$ = "9": GOSUB 15000
9310 IF INXS$ = "" OR INXS$ = "" THEN 9330
9320 ANZAHL = VAL(IN$):ANZAHL$ = STR$(ANZAHL):ANZAHLT = ANZAHL
9330 XIN = 468:YIN = 152:GRIN = 17:LIN = 1:INAS$ = "j":INES$ = "n": GOSUB 15000
9340 IF INXS$ = "" OR INXS$ = "" THEN 9360
9350 IF IN$ = "j" THEN PUNKT$ = IN$: ELSE PUNKT$ = "n"
9360 XIN = 145:YIN = 60:GRIN = 33:LIN = 20:INAS$ = "(":INES$ = "n": GOSUB 15000
9370 IF INXS$ = "" THEN 9020
9380 IF INXS$ = "" AND FUNALTS < > "" THEN 9400
9390 IF IN$ < > "0" THEN PUNKT$ = IN$: ELSE GOTO 9360
9400 XIN = 145:YIN = 40:GRIN = 33:LIN = 5:INAS$ = "-":INES$ = "9": GOSUB 15000
9410 IF INXS$ = "" OR INXS$ = "" THEN 9430
9420 K = VAL(IN$) * VIN:KS = STR$(K)
9430 RETURN
9440 :
9450 :
10000 REM " *** Bildschirmmaske ***

```

```

10230 :
10240 :
11000 REM " *** Bild loeschen ***
11010 FOR SEITE = 2 TO 3: PAGE SEITE,5 - SEITE
11020 CLPG
11030 GOSUB 10000
11040 GOSUB 8000
11050 NEXT SEITE
11060 PAGE 2,2
11070 RETURN
11080 :
11090 :
12000 REM " *** Statuszeile ***
12010 XST = (512 - LEN(STATUS$) * 12) / 2
12020 FOR SEITE = 2 TO 3: PAGE SEITE,2
12030 WRITE 5,6,$21,"
12040 WRITE 7,6,$21,"
12050 WRITE ASST,6,$21,STATUS$
12060 NEXT SEITE
12070 PAGE 2,2
12075 AS = INKEYS
12080 RETURN
12090 :
12100 :
13000 REM " *** Titelbild ***
13010 LINE (10,300) - (390,300)
13030 LINE (200,110) - (200,490)
13040 CONNECT (10,300) - (90,250) - (150,400) - (230,450)
13050 CONNECT (230,450) - (280,300) - (330,250) - (390,300)
13060 RETURN
13070 :
13080 :
15000 REM " *** Spezial-Input ***
15010 PAGE 2,2:INXS$ = INKEYS
15020 WRITE XIN,YIN,GRIN,CHR$(127) + SPACES$(LIN - 1)
15030 IN$ = "" : VIN = 1
15040 INXS$ = INKEYS
15050 IF INXS$ = CHR$(13) THEN 15300
15060 IF INXS$ = CHR$(8) THEN 15200
15070 IF INXS$ = "" THEN 15400
15080 IF INXS$ < INAS$ OR INXS$ > INAS$ THEN 15040

15090 IN$ = IN$ + INXS$
15100 WRITE XIN,YIN,GRIN,IN$ + CHR$(127) + "
15110 LIN = LIN - 1: IF LIN > 0 THEN 15040
15120 GOTO 15300
15130 :
15200 IF LEN(IN$) > 0 THEN IN$ = LEFT$(IN$,LEN(IN$) - 1)
15210 LIN = LIN + 1
15220 WRITE XIN,YIN,GRIN,IN$ + CHR$(127) + "
15230 GOTO 15040
15240 :
15300 PAGE 2,2: WRITE XIN,YIN,GRIN,IN$ + "
15310 PAGE 3,2: WRITE XIN,YIN,GRIN,IN$ + "
15320 IF IN$ = "" THEN IN$ = "0":INXS$ = "" : GOSUB 15500
15330 IF LEFT$(IN$,1) < > "-" THEN RETURN
15340 IN$ = RIGHT$(IN$,LEN(IN$) - 1):VIN = - 1
15350 RETURN
15360 :
15400 GOSUB 15500: REM " Daten-Anzeige
15410 RETURN
15420 :
15500 PAGE 2,2: GOSUB 8000: REM " Daten-Anzeige
15510 PAGE 3,2: GOSUB 8000: REM " Daten-Anzeige
15520 RETURN
15530 :
15540 :
16000 REM " *** Pfeil ***
16010 COLOR = 0
16020 MOVETO XPA,95: TURN TO 270: MOVE 15
16030 MOVETO XPA,95: TURN TO 225: MOVE 7
16035 MOVETO XPA,95: TURN TO 315: MOVE 7
16040 COLOR = 1
16050 MOVETO XP,95: TURN TO 270: MOVE 15
16060 MOVETO XP,95: TURN TO 225: MOVE 7
16065 MOVETO XP,95: TURN TO 315: MOVE 7
16070 XPA = XP
16080 RETURN
16090 :
16100 :
17000 REM " *** Vorbereitung ***
17010 DIM BEFEL$(20),BEFL(20),TKN(20)
17020 FOR I% = 0 TO 15
17030 READ BEFEL$(I%): READ BEFL(I%): READ TKN(I%)
17040 NEXT I%
17050 DATA "SIN",3,137,"COS",3,140,"TAN",3,141,"SQR",3,135
17060 DATA "EXP",3,139,"LOG",3,138,"+",1,242,"-",1,243
17070 DATA "ABS",3,134,"INT",3,133,"SGN",3,132
17080 DATA "*",1,244,"/",1,245,"",1,246
17090 DATA "<=",1,241,">=",1,239
17100 RETURN
17110 :
17120 :
17130 :
18000 REM " *** Ablegen der Funktion ***
18010 STATUS$ = "***** Moment bitte *****": GOSUB 12000
18020 FKT$ = ""
18050 FOR I% = 1 TO LEN(FUNKT$)
18060 HILFSS$ = MID$(FUNKT$,I,1)
18070 IF ASC(HILFSS$) > 95 THEN HILFSS$ = CHR$(ASC(HILFSS$) - 32)
18080 FKT$ = FKT$ + HILFSS$
18090 NEXT I%
18100 I% = 1:TKN$ = "" : HILFSS$ = ""
18110 FUNKT$ = FKT$
18120 J% = 0
18130 MERKER% = 0:HILFSS$ = MID$(FUNKT$,I%,BEFL(J%))

18140 IF HILFSS$ < > BEFEL$(J%) THEN 18170
18150 IF BEFL(J%) > 1 THEN TKN$ = TKN$ + CHR$(255)
18160 TKN$ = TKN$ + CHR$(TKN(J%)):MERKER% = 1: GOTO 18200
18170 IF J% < 14 THEN J% = J% + 1: GOTO 18130
18190 TKN$ = TKN$ + HILFSS$
18200 I% = I% + 1 + MERKER% * BEFL(J%) - MERKER%
18210 IF I% < = LEN(FUNKT$) THEN 18120
18220 IF LEN(TKN$) < 30 THEN TKN$ = TKN$ + " ": GOTO 18220
18230 PROG$ = "8000"
18240 IF PEEK(PROG%) < > 64 THEN PROG$ = PROG$ + 1: GOTO 18240
18250 IF PEEK(PROG$ + 1) = 64 AND PEEK(PROG$ + 2) = 64 THEN 18270
18260 PROG$ = PROG$ + 1: GOTO 18240
18270 FOR I% = 1 TO LEN(TKN$)
18280 ADRESSE% = PROG$ + 16 + I%
18290 POKE ADRESSE%,ASC(MID$(TKN$,I,1))
18300 NEXT I%
18310 FUNALTS = FUNKT$
18320 RETURN
18330 :
18340 :
20000 REM " *** Funktionsdefinition ***
20010 REM $$$
20020 DEF FN F(X) = SIN(X) * EXP(-K * X)
20030 RETURN

```

Lehrgänge zum NDR-Computer

Wenn Sie sich intensiv mit der Hard- und Software des NDR-Computers beschäftigen wollen, helfen Ihnen diese Kurse weiter:



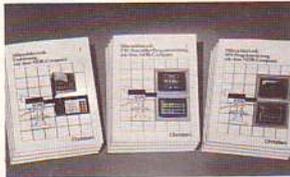
Kompakt-Kurs Elektronik

Ca. 200 Seiten im Format A4 mit Experimentiermaterial, Tonbandkassette und Flip-chart zur Einführung in die Grundlagen der Elektrotechnik und Halbleiterphysik. Der Kompakt-Kurs ist ideal für alle, die sich nicht nur mit dem Nachbauen von Schaltungen zufrieden geben, sondern auch selbst Schaltungen ändern, ergänzen oder entwerfen wollen.



Kompakt-Kurs BASIC

Ca. 200 Seiten im Format A4 mit Abschlußtest. Der Lehrgang behandelt alle gängigen BASIC-Anweisungen und führt Sie anhand von zahlreichen Beispielprogrammen in die ersten Schritte der BASIC-Programmierung ein.



Kompakt-Kurs Mikroelektronik – Einführung

Ca. 240 Seiten Lehrmaterial im Format A4 mit Abschlußtest. Der Lehrgang zeigt anhand des NDR-Einsteigerpakets, wie Sie Ihren Computer in Maschinensprache programmieren. Jeder Befehl wird erläutert und anhand von Beispielen lernen Sie den Einsatz der Maschinensprache des Z80 lernen.



Kompakt-Kurs Z80-Assembler-Programmierung

Ca. 240 Seiten Lehrmaterial im Format A4 mit Abschlußtest. Wenn Sie sich in die Assemblersprache einarbeiten wollen, ist dies der richtige Kurs für Sie. Als Hardware-Grundlage dient das ZEAT-Betriebssystem.

Christiani

Bitte Bestellschein abtrennen und einsenden an:
Dr.-Ing. P. Christiani GmbH, Postfach 35000, 7750 Konstanz



Bestellung · Information

Senden Sie mir gleich über die Christiani Kurse ausführliches, kostenloses Informationsmaterial wie Lehrpläne, Probeseiten und Preisliste.

Name, Vorname

Straße, Nummer

PLZ, Ort

35 519

68 000 - YOGIDOS UND JADOS, RL-BASIC

Sonderzeichen auf der GDP

Ralph Dombrowski
Gr. Deichstraße 33
2208 Glückstadt

Sonderzeichen auf der GDP

Im Grundprogramm ist eine Möglichkeit vorhanden, Sonderzeichen auf der GDP-Karte auszugeben. Über diese Möglichkeit möchte ich einmal näher berichten. Jedes Zeichen, das der GDP-Prozessor EF 9366 ausgibt, besteht aus einer 5*8 Matrix. Das Zeichen A sieht z.B. so aus:

```
.XXX.
X...X
X...X
X...X
```

X bedeutet, daß der Punkt gezeichnet wird. Nun hat die GDP-Karte nur die Zeichen gespeichert, die im ASCII-Standard definiert sind. Wer andere Zeichen ausgeben möchte, muß sich anders behelfen. Dazu ist im Grundprogramm die Routine PROGZGE (Programmierbarer Zeichengenerator) vorhanden. Es ist durch ihn möglich, eine eigene 5*8 Punktmatrix aufzustellen und auszugeben. Ein gesetztes Bit bedeutet, daß der Punkt gezeichnet wird. Unser A würde dort so aussehen:

```
dc.b *00111110
dc.b *00001001
dc.b *00001001
dc.b *00001001
dc.b *00111110
dc.b *00000000
```

Es ist hierbei darauf zu achten, daß die Matrix um 90 Grad gedreht ist. In so einer Matrix kann man nun alle gewünschten Zeichen ablegen und ausgeben. Abstand zum nächsten Zeichen, deshalb 6*8

Allerdings ist die Grundprogrammroutine sehr langsam, weswegen ich hier eine schnellere Routine vorstellen möchte. Nun ist es jedem selbst überlassen, eigene Zeichen zu erfinden und eine WRITE-Routine zu schreiben, die diese Zeichen erkennt und ausgibt, wodurch man in eigenen Texten gewünschte Zeichen automatisch ausgeben lassen kann. Als Anwendung könnten zum Beispiel physikalische oder mathematische Programme diese Routine verwenden, um griechische Buchstaben oder Symbole auszugeben. Dabei könnte so verfahren werden, daß bei den Buchstaben, die im EF 9366 gespeichert sind, Bit 7 nicht gesetzt ist, wie es auch im Ascii-Standard vorgesehen ist, während bei allen selbst definierten Zeichen Bit 7 gesetzt ist. Es soll nicht verschwiegen werden, daß die Ausgaben mit dieser Routine auch nicht superschnell ablaufen, deshalb sollte im Regelfall der GDP-Prozessor direkt angesprochen werden.

An die Routine müssen folgende Daten übergeben werden:
Der GDP-Schreibkopf muß an der gewünschten Stelle stehen und die Schriftgröße muß eingestellt sein. Außerdem muß im Register a0 die Anfangsadresse der Punktmatrix stehen. Damit sind alle Bedingungen erfüllt.

Beispiel:

```
clr d1
moveq #100,d2
moveq #1,moveto,d7
```

* GDP positionieren

```
trap #1
lea tabelle,a0
clr.b gdp3.w
moveq #13-1,d0
loop:
add.b #$11,gdp3.w
bcr progzge
dbra d0,loop
rts
```

- * Adresse des Zeichens nach a0
- * Schriftgröße auf Null
- * 13 verschiedene Größen
- * Neue Schriftgröße
- * Zeichen ausgeben

Ich wünsche viel Spaß beim Ausprobieren der Routine.

```
cpu equ 1
gdp0 equ $ff70*cpu.w
gdp1 equ $ff73*cpu.w
gdp5 equ $ff75*cpu.w
gdp8 equ $ff78*cpu.w
gdp9 equ $ff79*cpu.w
gdp11 equ $ff7b*cpu.w
```

- * Auswahl der CPU (1,2,4)

```
progzge:
moveml d0-d7/a0,-(a7)
trap #1
move.b gdp3.w,d4
cmp.b #$11,d4
beq prog1zge
move.b gdp5.w,-(a7)
move.b d4,d3
lsl #4,d3
and #f,d3
subq #1,d3
and #f,d3
move.b d3,gdp5.w
and #f,d4
subq #1,d4
and #f,d4
moveq #5-1,d6
progzlp0:
move.b (a0)+,d0
moveq #8-1,d5
progzlp1:
move d4,-(a7)
progzlp2:
btst.b d5,d0
beq.s progz1
btst.b #2,gdp0.w
beq.s *-6
```

- * Schreibposition laden
- * Schriftgröße laden
- * Bei Schriftgröße \$11 extra schnelle Routine
- * Vergrößerung in X-Richtung
- * Breite 6 Punkte
- * Höhe 8 Punkte
- * Bit testen
- * Null, dann nicht zeichnen
- * Warten bis GDP fertig

```

move.b #10,gdp0.w      * Linie zeichnen
btst.b #2,gdp0.w      * Warten bis GDP fertig
beq.s #-6
move.b gdp8.w,d7
lsl #8,d7
move.b gdp9.w,d7
sub d3,d7
move.b d7,gdp9.w
lsr #8,d7
move.b d7,gdp8.w
proglz:
  addq.b #1,gdp11.w    * Y-Position erhöhen
dbr4 d4,proglzlp2
  move (a7)+,d4
dbr4 d5,proglzlp1
  add d3,d1
  addq #1,d1
  moveq #!moveto,d7    * Nächste Reihe
  trap #1
dbr4 d6,proglzlp0
  add d3,d1
  addq #1,d1
  moveq #!moveto,d7    * Positionieren für nächstes Zeichen
  trap #1
  move.b (a7)+,gdp5.w
  movem.l (a7)+,d0-d7/a0
rts

```

```

proglzge:
  moveq #5-1,d6
  proglzge:
  move.b (a0)+,d0
  moveq #8-1,d5
  proglzge:
  btst.b d5,d0
  beq.s proglz3g
  move.b #80,gdp0.w
  proglzge:
  btst.b #2,gdp0.w
  beq.s proglz1
  proglz3g:
  addq.b #1,gdp11.w
  dbr4 d5,proglz2g
  addq #1,d1
  moveq #!moveto,d7
  trap #1
  dbr4 d6,proglzge
  addq #1,d1
  moveq #!moveto,d7
  trap #1
  movem.l (a7)+,d0-d7/a0
  rts

```

CP/M 68 K, C UND MODULA

UNERA – im „c“-Programm

von Werner Jäger

Off ist es mir beim Arbeiten mit CP/M-68k passiert, daß ich unvorsichtiger Weise eine falsche Datei gelöscht habe. Meist

benötigte ich dann einen immensen Aufwand an Zeit und Arbeit, um diesen, in Sekunden erfolgten Fehler wieder zu beheben. Dabei wäre alles so einfach, wenn es ein Kommando „unera dateiname“ geben würde. Diese Idee aufgreifend machte ich mich an die Arbeit. Das Ergebnis meiner Bemü-

hungen liegt diesem Schreiben in Form eines Programmlistings bei. Ich nehme an, daß auch andere Anwender schon des öfteren dankbar für so ein kleines Kommando gewesen wären. Deshalb würde ich vorschlagen, daß Sie es in der *LOOP* abdrucken oder mit auf die Tool-Diskette packen.

```

unera.c
-----
* Programm zum Wiederherstellen eines gelöschten, aber noch nicht
* ueberschriebenen Directoryeintrags.
*
* Benötigt ein Assemblerinterface welches folgende Routinen zur
* Verfüugung stellt: readdr um Directory in den Speicher einzu-
* lesen. Return Speicheradresse des Directory
* oder Null wenn Fehler aufgetreten ist.
*
* writedir evtl. geändertes Directory vom Speicher
* zurueckschreiben. Return Speicheradresse des
* Directory bzw. Null wenn Fehler aufgetreten ist.
*
* Ausserdem muss ein Label diranz exportiert werden,
* in dem nach Aufruf von writedir die Anzahl der
* maximal verfügbaren Directoryeintraege steht.
*
* Aufruf: unera [d:] Dateiname.Text]
*
* Es werden alle Eintraege auf USER 0 gesetzt.
* Bei Aufruf ohne Laufwerksangabe wird Laufwerk a selektiert.
*
* 16.02.1987 Jaeger Werner
*/

#include "stdio.h"

#define MAXENTRY 511      /* Es koennen max. sovielen Eintraege */
                        /* eingelesen werden (Speicherplatz) */

struct entry {
  char user;             /* user-number oder 0xe5 wenn gelöscht */
  char name[8];         /* Dateiname */
  char t1, t2, t3;      /* Extension und Zugriffsberechtigung */
  char link;            /* Fortsetzungsnummer */
  char reserve;
  char nr_of_rec;      /* Anzahl der Records 0x80 wenn Fortsetzung */
  char blocknr[16];    /* Zugehoerige Blocknummern */
};

main(argc, argv) int argc; char *argv[]; {
  extern int diranz;    /* Anzahl der Directory-Eintraege */
  /* identisch mit DRM in Disk Parametr Block */
  /* wird von readir versorgt */

  struct entry *dir, *readdr(), *writedir();

  BOOLEAN linknr[256]; /* Fortsetzungsnummer eines Eintrags */
  struct entry * address[MAXENTRY]; /* Speicher fuer Adressen der betroffenen */
  /* Eintraege */

  register int i, j;    /* Laufvariable */
  register int len, lwnr; /* Laenge Dateiname; Laufwerksnummer 0-15 */
  int begin, ende;     /* merker fuer Fortsetzungsnummern */
  static char filename[] = " ";

  if (argc == 2) {
    if ((len=strlen(++argv)) > 14) error("Dateiname zu lang");
    else {
      /* Dateibezeichner auf Directoryformat bringen */
      if (**argv[1] == ':') { /* optionale Laufwerksangabe */
        lwnr = (*argv)[0] - 'a';
        i = 2;
      }
    }
  }

```

```

  else {
    lwnr = 0;          /* default Laufwerk = A: */
    i = 0;
  }
  for (j=0; j<8; j++) /* Nanen kopieren */
    if (**argv[i] == ':') (*argv)[i] = 0;
    break;
    else filename[j] = toupper((*argv)[i++]);
  i++;
  for (j=0; j<len; j++) /* Extension anhaengen falls da */
    filename[j] = toupper((*argv)[i++]);
  } else error("usage: unera [d:] filename.Text");

  if ((dir = readdr(lwnr)) == NULL) error("Fehler beim Lesen des Directory");

  if (diranz > MAXENTRY) error("Directory zu gross");
  for (i=0; i<255; linknr[i++] = FALSE); /* kein Eintrag noch da */
  j=0; /* Anzahl der gefundenen Eintraege */

  /* Suchschleife */

  for (i=0; i<diranz; i++) {
    if (cmp(dir, filename) == 0) {
      if (!linknr[dir->link]) error("Mehrdeutig"); /* Fortsetz.nr schon da */
      else {
        linknr[dir->link] = TRUE;
        address[j++] = dir; /* Adresse des Eintrags notieren */
      }
    }
    dir++;
  }
  if (j==0) error("Eintrag nicht gefunden");
  for (i=0; i<255; i++)
    if (!linknr[i]) break;
  begin = i;
  if (i != 0) error("Dateianfang fehlt");

  for (i=begin; i<255; i++)
    if (!linknr[i]) break;
  ende = i;

  for (i=ende; i<255; i++) /* Test ob alle Einr. da */
    if (!linknr[i]) error("Lueckenhaft");

  for (i=0; i<j; address[i++]->user = 0); /* jetzt restaurieren USER 0 */

  if (writedir() == NULL) error("Fatalerr. beim Rueckschreiben des Directory");

  int cmp(a, b)
  struct entry * a;
  char *b;
  {
    register int i;

    for (i=0; i<11; i++)
      if (a->name[i] != b[i])
        return (-i);
    return 0;
  }

  int toupper(c) register char c; {
    if (c<'a') return c;
    return (c + 'A' - 'a');
  }

  error(s) char *s; {
    putwkw=?text(i);
  }

```

```

.....
*
*          dirutil.s
*          =====
* Assemblerinterface zu C-Programmen die mit dem Diskettendirectory arbeiten.
* Achtung vor Aufruf von writedir muss immer ein fehlerfreier Aufruf von
* readdir erfolgen, sonst wird Diskettendirectory zerstört
*
* Zwei Funktionen sind hier definiert:
*
* Directory in Speicher einlesen und Adresse in da zurückgeben.
* struct entry *readdir(dr) int dr; / 0 - 15 /
*
* Directory zurückschreiben.
* struct entry *writedir();
*
* Wenn ein Nullzeiger zurückgegeben wird, liegt ein Fehler vor
*
* Zur Einbindung in C-Programm
* 16.02.87 Jaeger Werner
.....

```

```

.globl _readdir
.globl _writedir
.globl _diranz

```

```

reboot      equ 0
seldsk     equ 14
cdbp       equ 31
setdna     equ 12
printstr   equ 9
settrack   equ 10
setsect    equ 11

```

```

.text
_readdir:
    link a6,#0
    move.l d0-a7,-(a7)
    move.w #0(a6),dr
    move.w dr,d1
    move.w #seldsk,d6
    trap #2
    move.l #dbp,d1
    move.w #cdbp,d0
    trap #2
    move.l d1,a0
    move.w #a01,sektanz
    move.w #B(a0),_diranz
    move.w #e(a0),off
    move.w #e(a0),spur
    move.w #1,sektor
    move.l #diradr,d1
    move.l _diranz,d6
    ext.l d6
    divu #4,d6
    addq #1,d6
loop:
    move.l d1,-(a7)
    move #setdna,d0
    trap #3
    move.w spur,d1
    move.w #settrack,d0
    trap #3
    move.w sektor,d1
    move.w #setsect,d0
    trap #3
    move.w sektor,d0
    cap.w sektanz,d0
    blo samtrack
    add #1,spur
    move #1,sektor
    bra endi
samtrack:
    add #1,sektor
endi:
    move #13,d0
    trap #2
    move.l (a7)+,d1
    tst d0

```

```

* stack frame
* Laufwerk
* Laufwerksnummer
* Laufwerk waehlen
* B005
* Adresse Diskparameter
* B005
* Anzahl der log. 128 Byte Sektoren merken
* Anzahl der Directory-Eintraege merken
* Spur ab der das Directory beginnt
* Spurzahler initialisieren
* Sektorzaehler initialisieren
* Speicherbeginn fuer Directory initialisieren
* Schleifenzaehler initialisieren
* 32 Byte pro Eintrag (128 / 32 = 4)
* Directory von Diskette einlesen
* DMA - Setzen
* BIOS
* Spurnummer setzen
* BIOS
* Sektornummer setzen
* BIOS
* neue Spur- und Sektornummer ermitteln
* Sektor lesen
* BIOS
* Fehler ?

```

```

    bne error
    add.l #12B,d1
    dbeg d6,loop
    move.l (a7)+,d0-a7
    move.l #diradr,d0
    unlk a6
    rts

error:
    move.l (a7)+,d0-a7
    move.l #0,d0
    unlk a6
    rts

_writedir:
    link a6,#0
    move.l d0-a7,-(a7)
    move.w dr,d1
    move #seldsk,d0
    trap #2
    move.l _diranz,d6
    ext.l d6
    divu #4,d6
    addq #1,d6
    move off,spur
    move #1,sektor
    move.l #diradr,d1
lp:
    move.l d1,-(a7)
    move #setdna,d0
    trap #3
    move.w spur,d1
    move.w #settrack,d0
    trap #3
    move.w sektor,d1
    move.w #setsect,d0
    trap #3
    move.w sektor,d0
    cap.w sektanz,d0
    blo samtrack
    add #1,spur
    move #1,sektor
    bra endi
samtrack:
    add #1,sektor
endi:
    move #14,d0
    move #1,d1
    trap #3
    move.l (a7)+,d1
    tst d0
    bne error1
    add.l #12B,d1
    dbeg d6,lp
    move.l (a7)+,d0-a7
    move.l #diradr,d0
    unlk a6
    rts

error1:
    move.l (a7)+,d0-a7
    move.l #0,d0
    unlk a6
    rts

.data
.even

dr: dc.w 0
_diranz: dc.w 0
off: dc.w 0
spur: dc.w 0
sektor: dc.w 0
sektanz: dc.w 0

.bss

dpb: ds.b 1B
diradr: ds.b #4000

```

```

* dann dorthin
* neue Adresse
* bis fertig

* Laufwerk waehlen
* B005

* Schleifenzaehler initialisieren
* Spurzahler initialisieren
* Sektorzaehler initialisieren
* Speicherbeginn

* DMA - Setzen
* BIOS

* Spurnummer setzen
* BIOS

* Sektornummer setzen
* BIOS

* neue Spur- und Sektornummer ermitteln

* Sektor schreiben
* Directoryspur
* BIOS

* Fehler ?
* dann dorthin
* neue Adresse
* bis fertig

```

TIPS + TRICKS — TIPS + TRICKS

Betrieb des Z80 als Coprozessor im 68008 NDR-Computer

von Holger Lech,
Dahlmannsweg 240, 4390 Gladbeck

Ziel dieser Entwicklung war es, die Z80 CPU weiterhin ohne Einschränkungen benutzen zu können, ohne auf den Hauptbetrieb des 68008 verzichten zu müssen. Ausgehend von der Tatsache, daß beide Prozessoren für den Betrieb einer DMA-Einheit ausgelegt sind, wurde

die Z80 CPU-Karte so modifiziert, daß dem Z80 Prozessor bei laufender 68008-Karte ein DMA-Buszugriff vorgetäuscht wird, der einen Buszugriff seinerseits verhindert. Als Sicherheit wurde ihm zusätzlich das Taktsignal gesperrt. Durch einfachen Wechsel des BUSRQ-Signals ist es nun möglich, den jeweils laufenden Prozessor anzuhalten und auf den Coprozessor umzuschalten. Nach jedem Signalwechsel ist ein RESET beider Prozessoren notwendig! Um beide Bootprogramme wechselweise zur Verfügung zu haben, mußte auch die Bank-Boot-Karte verändert werden. Abhängig vom

BUSRQ-Pegel wurde nun das jeweils benötigte EPROM selektiert und die Adressen A16 – A19 entweder gesperrt oder getrieben. Die Umschaltung kann entweder durch eine Umschaltung von Hand (Schalter gegen OV) oder durch einen I/O-Port durchgeführt werden. Dazu könnte ein Menüpunkt im 68008-Grundprogramm vorgesehen werden. Die Änderungen der CPU Z80 könnten auch analog dazu an der CPU 68k durchgeführt werden, um den 68008 als Coprozessor zu betreiben. Die aufgeführten Änderungen beeinträchtigen die Funktionsfähigkeit der einzelnen Prozessoren nach

meinen Erfahrungen bisher nicht. Die Änderungen im Einzelnen:

1. Verknüpfung des BUSRQ-Signals über ein Oder-Gatter (74LS32) mit dem Takt-Signal vom IC 1 Pin 10.
2. Invertieren des BUSRQ-Signals über ein Inverter (74LS04) und Zuführen auf den BUSRQ-Eingang der CPU.
3. Zuführen des BUSRQ-Signals auf Pin 19 der Bustreiber (74LS245); unterbrechen der Verbindung zu IC 2 Pin 3.
4. Unterbrechen der Busverbindung der Signale \bar{I} Ext, RESET Ext, BUSAK sowie Entfernen des Widerstandes R5.

Änderungen der Bank-Boot-Karte:

1. Invertieren des BUSRQ-Signals über einen Inverter (74LS04) und Verknüp-

fen mit dem CS-Signal des Decodierers (74LS138) Pin 1 & IC 10 und Zuführen auf Pin 20 des Boot 68 EPROMs IC 1.

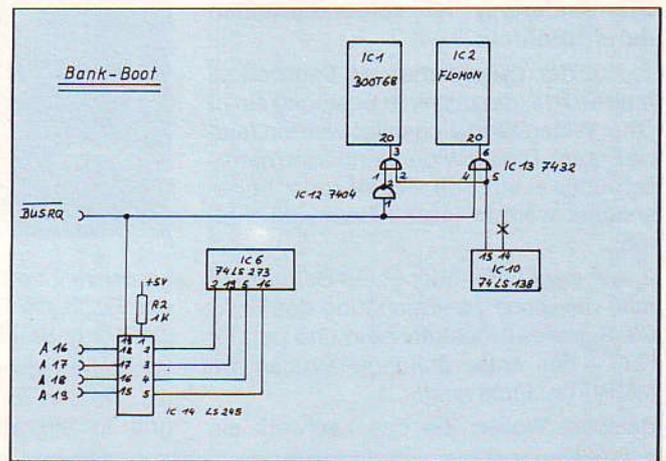
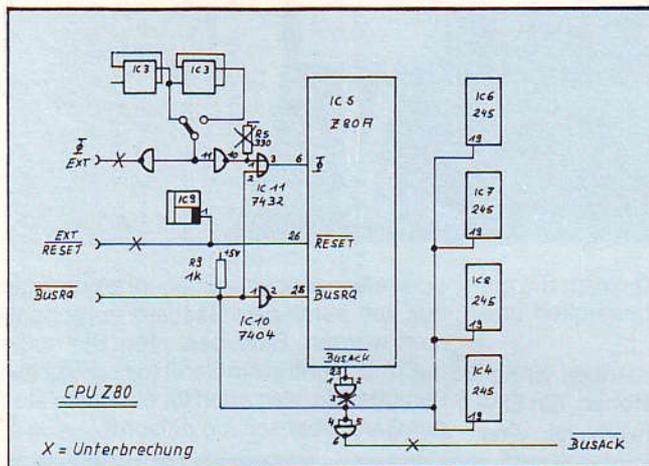
2. Verknüpfen des BUSRQ-Signals über ein Oder-Gatter (74LS32) mit dem CS-Signal des Decodierers IC 10 Pin 15 sowie Zuführen auf Pin 20 des Flomon EPROMs. Die Verbindung zu Pin 14 IC 10 muß unterbrochen werden.
3. Zuführen der Adressen A16 - A19 auf die Eingänge eines Bustreibers (74LS245), Verbinden der Ausgänge mit dem Adressbus. (Eingänge: Pin 2, 3, 4, 5. Ausgänge Pin 18, 17, 16, 15)
4. Verbinden des BUSRQ-Signals mit Pin 19 des Bustreibers sowie Beschaltung

des Pin 1 mit einem 1kOhm-Widerstand gegen + 5 V.

Ergänzung:

Wird nach Einschalten des Systems auf den Z80 umgeschaltet, so muß zunächst der 68008 einen RESET erhalten, damit alle anderen Karten zurückgesetzt werden (GDP, Bankboot, FLO). Wird aus dem Flomon-Menü auf die Bank E0000 gesprungen, dann erfolgt der RESET des Z80 nur auf dieser Bank, d.h. man gelangt nur ins Grundprogramm. Will man ins Flomon-Menü zurück, ist der RESET des 68008 erforderlich.

Hinweis der Redaktion: Wir freuen uns über diesen Vorschlag und geben ihn ungeprüft an unsere Leser weiter.



Transfer C 64 - NDR

von Ivo Hasler, Berghaldenstr. 22, CH-8330 Pfäffikon

Da ich seit einigen Jahren einen Commodore 64 besitze und auf diesem schon etliche Programme geschrieben habe, wollte ich natürlich auch ein paar von die-

sen Programmen auf dem NDR-Computer laufen lassen. Darum habe ich B-TRANS (Name des Programms) geschrieben, welches Basic-Programme des C-64 auf den NDR-Computer (8 k-Basic, RDK, evtl. auch andere) transferiert.

Das Programm ist auf dem C-64 geschrieben und wird auch auf diesem gestartet. Das Prinzip ist ganz einfach: Man steckt die Tastatur des NDR-Computers aus und benützt den C-64 als neue Tastatur.

B-TRANS

Beschreibung:

Das Programm wird auf dem C-64 geladen und gestartet. In Zeile 20 wird die Routine für das Ausgeben eines Zeichens, auf mein Programm umgebogen. B-TRANS springt normalerweise die Betriebssystemroutine an, ausser die Ausgabe erfolge auf Geräteadresse 5, dann wird das Zeichen über den Userport auf die Baugruppe KEY geführt. So simuliert der C-64 sozusagen den Programmierer, der das Basic-Programm abtippt. Vorgehen: Zuerst folgende Verbindung zwischen C-64 und NDR-Computer vornehmen.

C-64	Userport	Baugruppe	KEY (NDR)
Pin	Name	Name	
C	PB0		1.Bit
D	PB1		2.Bit
E	PB2		3.Bit
F	PB3		4.Bit
H	PB4		5.Bit
J	PB5		6.Bit
K	PB6		7.Bit
L	PB7		Strobe
N	GND		GND

NDR-Computer einschalten und 8K-Basic starten. B-Trans auf C-64 starten. Zu übertragendes Programm laden. Danach mit 'OPEN 5,5' Kanal auf Gerät 5 (NDR-Computer) eröffnen, mit 'CMD 5' normale Ausgabe auf Gerät 5 leiten. Mit 'LIST' Transaktion starten. Das Basic-Programm wird nun auf dem NDR-Computer angezeigt und eingelesen. Die Daten gehen per Userport auf die Baugruppe KEY. Natürlich können auch Befehle "von Hand" mit 'PRINT#5,"befehl"' übertragen werden.

ACHTUNG:

Es gibt etliche Befehle die auf dem C-64 vorhanden sind und im 8K-Basic nicht! Auch darf beim 8K-Basic nach 'GOSUB x' kein Befehl mehr folgen, beim C-64 jedoch schon. Ebenso existiert beim C-64 'PI', beim 8K-Basic aber nicht.

Ob B-TRANS auch C-64 Programme zu z.B. HEBAS überträgt weiss ich nicht, ich könnte es mir aber gut vorstellen.

Man könnte sich vorstellen, die Zeichen von KEY selbst im NDR-Computer abzufangen, dann hätte man ein komfortables Datenübertragungsprogramm.

PROGRAMM:

Als Basic-Lader:

```

10 Fori=49152to49239:reada:pokei,a:next
20 poke806,16:poke807,192
32000 data72,152,72,160,0,136,208,253,202,208,248,104,168,104,96,234,72,165,154
32001 data201,5,240,4,104,76,202,241,169,3,133,154,169,255,141,3,221,104,9,128
32002 data141,1,221,41,127,141,1,221,72,138,72,162,16,32,0,192,104,170,104,201
32003 data13,208,16,72,138,72,162,0,32,0,192,162,0,32,0,192,104,170,104,32,202
32004 data241,72,169,5,133,154,104,96
ready.
    
```

Als Maschinenprogramm:

```

c000 pha
c001 tya
c002 pha
c003 ldy #500
c005 dey
c006 bne $c005
c008 dex
c009 bne $c003
c00b pla
c00c tay
c00d pla
c00e rts
c00f nop
c010 pha
c011 lda $9a
c013 cmp #505
c015 beq $c01b
c017 pla

c018 jmp $f1ca
c01b lda #503
c01d sta $9a
c01f lda #fff
c021 sta $dd03
c024 pla
c025 ora #580
c027 sta $dd01
c02a and #57f
c02c sta $dd01
c02f pha
c030 txa
c031 pha
c032 ldx #510
c034 jsr $c000
c037 pla
c038 tax
c039 pla

c03a cmp #504
c03c bne $c04e
c03e pha
c03f txa
c040 pha
c041 ldx #500
c043 jsr $c000
c046 ldx #500
c048 jsr $c000
c04b pla
c04c tax
c04d pla
c04e jsr $f1ca
c051 pha
c052 lda #505
c054 sta $9a
c056 pla
c057 rts
    
```

Umstellen eines 80-Spur-Laufwerkes (TEAC FD55FV-03-U) auf 40 Spuren

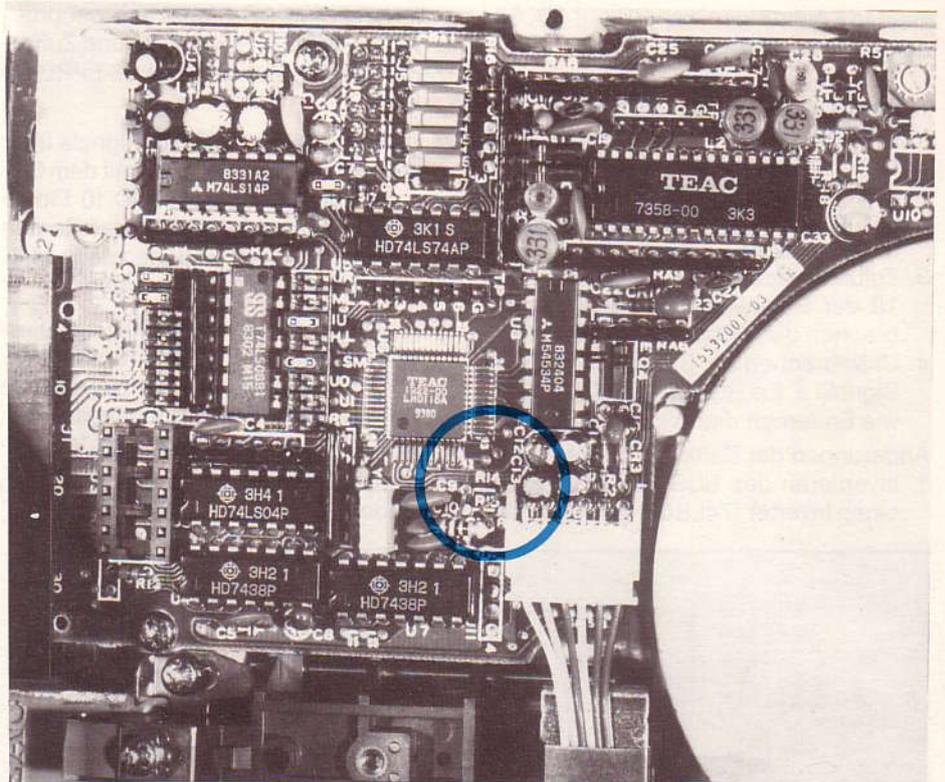
Das TEAC-Laufwerk FD 55 F-03-U kann durch einen kleinen Eingriff auf 40 Spuren umgestellt werden. Diese Änderung funktioniert in 90% aller Fälle und reicht in den meisten Fällen aus. Eventuell auftretende Fehler sind z. B., daß von einem anderen 40-Spur-Laufwerk beschriebene Disketten nicht gelesen werden können. Diese Änderung kann nicht bei TEAC-Laufwerken vom Typ FD 55 FV-13-U durchgeführt werden.

Die Änderung ist folgendermaßen durchzuführen:

1. Auf der Leiterplatte des Laufwerkes muß für R14 (derzeit nicht bestückt) ein 10 Ohm Widerstand eingelötet werden (siehe Photo). Dieser Widerstand kann natürlich über einen Schalter zu- oder abgeschaltet werden (40/80 Spur umschalten).

2. Auf der FLO2- oder FLO3-Baugruppe muß die Diode zur Erzeugung des READY-Signales eingelötet sein und bei der FLO3 die entsprechende Brücke an (JMP1) bestückt sein.

Beispiel: Wollen Sie das Laufwerk als



Laufwerk 2 betreiben, so müssen Sie auf der FLO3 die Diode D2 bestücken und JMP1/2 gebrückt haben.

Wichtiger Hinweis: Dieser Artikel ist nur als „Tip und Trick“ zu verstehen. Ein Eingriff in ein Floppy-Laufwerk ist eine

schwerwiegende Angelegenheit und darf nur von versierten Bastlern vorgenommen werden. Bitte beachten Sie auch, daß bei Eingriff in ein Gerät (das gilt für alle Fertigeräte, also auch für ein Radio etc.) der Garantiespruch erlischt!

FRÜHJAHRESANGEBOT

GES GmbH
Magnusstraße 13
8960 Kempten

2.0. MAI 1987

Bielefeld, den 14.05.1987

Erliebigt:

Sehr geehrte LOOP-Redaktion,

der LOOP Nr. 13a entnahm ich die Ankündigung des JADOS-Update v2.1. Dazu würde ich Ihnen gerne meine Meinung mitteilen.

Das JADOS-Problem

Das Betriebssystem JADOS hat sich inzwischen auf den NDR 680xx'ern als Standard durchgesetzt. Seine Beliebtheit liegt zum einen in der einfachen, komfortablen Bedienung begründet, aber auch der günstige Preis spielt sicherlich eine Rolle. Daß JADOS vollständig maschinenabhängig, und damit entsprechend schnell ist, dürfte die meisten Benutzer in keiner Weise stören. Da der NKC aber in erster Linie ein Bastler- und Hobbyprogrammierer-Rechner ist, fällt das fehlende Angebot von höheren Programmiersprachen unangenehm auf.

Als ich 1984 anfang einen NKC zu bauen, war das der einzige Rechner mit ROM-residentem PASCAL, alle Konkurrenten boten nur BASIC. Inzwischen ist mein Rechner etwas gewachsen, u.a. habe ich auch Diskettenlaufwerke angeschafft und die PASCAL-ROM's gegen zusätzliche RAM's getauscht. Wenn ich nun in höheren Sprachen programmieren möchte, habe ich (bisher) unter JADOS nur (das hervorragende) RL-Basic als Interpreter zur Verfügung. Deshalb sind ausnahmslos alle für JADOS kommerziell angebotenen Programme in Assembler geschrieben. Ich selbst programmiere nicht (mehr) in Assembler, und Basic-Programme sind nur zusammen mit dem Interpreter lauffähig, d.h. kommerziell praktisch nicht verwertbar (mal abgesehen von den vielrätigen didaktischen Gründen, die eigentlich jeden von der Basic-Programmierung abhalten sollten).

Diese Situation ist meiner Meinung nach untragbar. Ein Beispiel mag das Veranschaulichen: Ich habe Ralph Dombrowski vorgeschlagen, einige Details seines Disketten-Monitors zu verändern, u.a. andere Diskettenformate (EUMEL) zu erlauben, bzw. nicht-EPSON Drucker zu unterstützen. Er hat dies ablehnen müssen, denn das Programm ist vollständig in Assembler geschrieben und die notwendigen Änderungen wären zu umfangreich. Dies liegt natürlich auch etwas am Programmierstil, ist aber vor allem ein Assembler-Problem.

Warum gibt es aber bisher keine Compiler für höhere Programmiersprachen unter JADOS ? Das kann doch nicht nur an der relativ geringen Verbreitung unseres Computers und am hohen Aufwand einer Compiler-Anpassung liegen ? Die Firma pl hat doch bereits ihren Modula-2 Compiler an CP/M-68K für den NKC angepasst, warum gibt es keine JADOS-Version ?

Schauen wir uns zunächst die Aufgaben eines Betriebssystems im allgemeinen und die Lösung durch JADOS im speziellen an:

Ein Betriebssystem soll die Ressourcen eines Computersystems verwalten und dem Benutzer eine Oberfläche zur Verfügung stellen, die es ihm ermöglicht, dem System seine Wünsche mitzuteilen.

Zunächst der zweite Punkt: JADOS bietet eine einfach zu bedienende und dabei sehr komfortable Benutzeroberfläche. Obwohl in der letzten Zeit immer mehr grafisch orientierte, mausgesteuerte Oberflächen verwendet werden, ist es für den eingearbeiteten Benutzer oft einfacher, ein Kommando über die Tastatur einzugeben. Im Übrigen ist das einzige (mir bekannte) professionellere Betriebssystem für den NKC, CP/M-68K in dieser Hinsicht

(Vllleicht fragen Sie sich zunächst, ob das überhaupt, realistisch betrachtet, vorkommt? Ja, z.B. der p1 Modula-2 Compiler für CP/M-68K benötigt allein auf Diskette 184 kByte, im Speicher noch mehr!)

Was soll also der JADOS-Linker tun, wenn ein Programm mit den aus der Library benötigten Moduln diese magische Grenze überschreitet? - Den Bindevorgang mit einer Fehlermeldung abzubrechen wäre unbefriedigend. Also muß Abhilfe geschaffen werden. Der erfahrene Assembler-Programmierer kennt natürlich sofort mehrere Lösungsmöglichkeiten:

1. Geschickte Anordnung der Moduln. Wenn die einzelnen Moduln in sich relokativ sind, müssen sie nur so sortiert werden, daß alle externen Adressen nicht weiter als 32 kByte entfernt zu liegen kommen.
2. Verschränkte Sprungketten. Reicht 1) nicht zur Lösung aller externen Adressen, müssen die verbleibenden 'langen' Sprünge in mehrere kurze zerteilt werden. Dabei muß evtl. ein zusätzlicher Sprung in ein anderes Modul eingefügt werden.

Noch einmal das obige Beispiel:

```

text1:  dc.b '.....',0
main:   lea text1(pc),a0
        bsr _prints1
        .
        * Hier steht weiterer Code (32 kByte
    )

```

```

        bra lprints
        _prints1: bra _prints
lprints: .
        * Der hier unterbrochen werden muß
        * um den Sprung weiterzureichen.
        * Hier geht's dann weiter

```

Diese Lösung ist aber nur auf Sprünge anwendbar, für den lea Befehl ist mir kein einfaches Verfahren eingefallen (das gilt natürlich auch für adda etc.).

Was hier beim Binden geschehen muß, geht weit über die üblichen Aufgaben eines Linkers hinaus. Eine Lösung scheint nicht unmöglich, sie liegt aber auch nicht auf der Hand. Sollte sie gelingen bedeutet das für JADOS einen großen Sprung nach vorne. Alle, die Interesse an solchen Aufgaben der Systemprogrammierung haben, seien also hiermit aufgerufen: bastelt einen JADOS-Linker und veröffentliche das Format der object-files möglichst bald in der LOOP. Nach diesem ersten Schritt wird es dann hoffentlich nicht mehr lange dauern, bis eine ganze Flut von Compilern für JADOS auftaucht. Nach diesem zweiten Schritt werden wir uns dann gar nicht mehr vor Anwenderprogrammen retten können - ein Software-Schlaraffenland öffnet sich für den NKO.

Zum Schluß noch ein kurzer Blick auf andere Speicherverwaltungssysteme: Auf dem 680xx gibt es noch zwei Konzepte: die hardwaremäßige Adressumrechnung per MMU (Memory Management Unit) oder die Anpassung aller absoluten Adressen beim Laden des Programms in den Speicher. Der erste Ansatz ist wahrscheinlich der schnellste und teuerste, während der zweite entscheidende Nachteile hat: die Programme werden auf Diskette mit einer abgemagerten Symboltabelle gespeichert und brauchen deshalb viel Platz, außerdem dauert das Laden des Programms deutlich länger. Andere Prozessoren bieten eine weitere Möglichkeit: die Segmentierung. Beim Intel 8086 z.B. werden alle Adressen mit 16 Bit angegeben, um absolute Adressen zu erhalten, addiert der Prozessor zur Laufzeit den Inhalt eines Segmentregisters. Um so ein Programm zu relokieren läßt man einfach eine andere Basisadresse in das entsprechende Segmentregister (per Systemaufruf). Dies ist aber auch der einzige Vorteil einer segmentierten Adressierung, der größte Nachteil liegt auf der Hand: Datenobjekte können maximal 64 kByte groß werden (eine zweifarbigte Grafik aus 1024 x 1024 Punkten benötigt aber schon 128 kByte).

Schlußbemerkung:

Der 680xx und JADOS bringen uns einige Probleme bei längeren Programmen. Das ist aber kein Grund den Prozessor oder das System schlecht zu machen, vielmehr sollte wir in die Hände spucken und das optimale aus den Gegebenheiten herausholen. Ich hoffe bald von einer Lösung der oben beschriebenen Probleme zu hören.

wesentlich weniger komfortabel.

Betrachten wir nun den ersten Teil, die Verwaltung von Ressourcen. Als Ressourcen bezeichnet man üblicherweise Rechenzeit, Speicherplatz und Ein-/Ausgabemedien. Da JADOS ein sogenanntes single tasking, single user System ist (immer nur eine Aufgabe und ein Benutzer), fällt die Verwaltung von Rechenzeit weg. Die Ein-/Ausgabemedien, z.B. Drucker, Bildschirm, Tastatur und Disk, werden im üblichen Umfang unterstützt, es wäre in der Tat nicht sehr aufwendig, die Bibliotheksmodule der Modula-2 Bibliothek für JADOS zu implementieren.

Was bleibt ist die Verwaltung des Speicherplatzes. Hier geht Klaus Janßen einen sehr eleganten Weg: um die Speicherplatzverwaltung so flexibel wie möglich zu machen, müssen alle Programme *relokativ* sein, also an jeder beliebigen Adresse im Speicher lauffähig sein. Das ermöglicht den optionalen Einsatz der Bankboot-Karte auf 68008 und 68000, der verschiedene Ladeadressen erfordert und die enge Zusammenarbeit mit dem Grundprogramm (das auch relokativ ist). Die jetzt von Klaus Janßen vorgestellte Art einer RAM-Disk ist wohl die schnellste denkbare auf einem 680xx ohne MMU (Speicherverwaltungs-Chip).

Aber genau hier liegt das JADOS-Problem:

Compiler für höhere Programmiersprachen erzeugen üblicherweise keinen vollständigen relokativen Code, was aber durchaus möglich wäre. Z.B. könnten vom p1 Modula erzeugte Programme von Hand sehr leicht relokativ gemacht werden (bzw. der Compiler könnte mit geringen Änderungen automatisch relokativen Code erzeugen), wenn da nicht die Sache mit den Bibliotheken wäre. Üblicherweise werden Standardfunktionen höherer Programmiersprachen komplett übersetzt in sogenannten Libraries ausgeliefert. Der Compiler erzeugt dann nur einen entsprechenden Aufruf:

```

text1:  dc.b 'Guten Tag und viel Erfolg mit BASIC.',13,10,0
_main:  movea.l #text1,-(sp)
        jsr  _prints
        adda.l #4,sp
        rts

```

Hier ist `_prints` der Name einer Bibliotheksroutine zum Ausdrucken von Strings. Damit das Programm nun lauffähig wird, muß aus der Bibliothek die entsprechende Routine mit dem erzeugten Programm verbunden werden, wobei dann erst die endgültige Adresse für den `jsr`-Befehl ermittelt werden kann. Diesen Vorgang nennt man binden oder linken.

Zum Lieferumfang von CP/M-68K gehört neben dem C-Compiler auch ein Assembler, der Code in linkfähiger Form (sogenannte object-files) erzeugt, außerdem zwei (!) Linker (Programme, die object-files und Libraries "binden") und ein Programm zum erstellen eigener Libraries. Für JADOS existieren derartige Programme bisher noch nicht.

Um diesen Mangel abzustellen, müßte sich jetzt irgendein gewitzter Systemprogrammierer hinsetzen und zunächst ein object-file Format definieren, um dann einen Library-Manager zu programmieren, mit dem Bibliotheken aus einzelnen object-files zusammengestellt werden können. Woher diese object-files kommen kann dabei zunächst ignoriert werden, evtl. schreibt jemand auch noch einen Assembler, der den Maschinencode im entsprechenden Format ausgibt. Die anspruchsvollste Aufgabe ist aber der Linker, denn der muß nicht nur alle nötigen object-files aus einer Bibliothek mit anderen object-files zusammenkopieren können um daraus ein 68K-File zu machen, er muß auch die in den verschiedenen Modulen definierten Adressen, die in anderen Modulen angesprochen werden, eintragen.

Und hier stoßen wir auf das eigentliche JADOS-Problem! Solange ein Programm kürzer als 32 kByte ist, kann der Motorola 680xx alle Adressen relativ zum aktuellen PC adressieren, so daß diese Programme relokativ sind, was von JADOS gefordert wird. Der Linker muß in diesem Fall praktisch nur eine Symboltabelle aufbauen und dann alle objects zusammenkopieren, wobei er nötige externe Adressen seiner Symboltabelle entnehmen kann.

Was aber, wenn ein Programm länger als 32 kByte wird?

Die Lösung des JADOS-Problems

von Klaus Janßen

Ich glaube, Herr Husemann hat mit seinem Artikel sehr vielen NKC-Anwendern aus der Seele gesprochen, so auch mir. Wenn es gelingt, die angesprochenen Probleme zu lösen, und gute Hilfsmittel zur Softwareentwicklung bereitzustellen, dann könnte der NKC mit seiner ausgezeichneten Hardware den entscheidenden Schritt nach vorne tun.

Vor der Entwicklung eines JADOS-Linkers muß jedoch zuerst das Problem gelöst werden, Programme, die größer als 32 KByte sind, relokativ zu machen. Herr Husemann hat zwei Beispiele als Lösungsansatz gebracht, die aber beide nicht praktikabel sind. Durch geschickte Anordnung der Module kann man zwar die Grenze von 32 KByte durchbrechen, aber je größer das Programm wird, desto schwieriger wird es, eine geeignete Anordnung zu finden; bis hin zur Unmöglichkeit! Die Lösung mit den Sprungketten ist ebenfalls nicht praktikabel, da sie erstens Lautzeit kostet und zweitens, was viel schlimmer ist, den Programmierer dazu zwingt, solche Sprungmarken in seine Module einzubauen. Dabei kann es vorkommen, daß Sprünge weitergereicht werden müssen, die mit dem Modul überhaupt nichts zu tun haben. Noch schlimmer wird es, wenn man seine Module um Sprungketten erweitern muß, nur weil man eine zusätzliche Library einbinden will.

Eine dritte Möglichkeit wäre die Verwendung von TRAPS. Durch die Traps ist es möglich, jede Sprungadresse im Adreßbereich des Prozessors zu erreichen, ohne die Relokativität zu verletzen. Nur durch die Traps ist es z.B. möglich, die Routinen des Grundprogramms (TRAP #1) und des JADOS (TRAP #6) zu benutzen und dennoch relokative Programme zu erstellen. Solange es nur einen Compiler und 2-3 Libraries gibt, wäre der Trap-Mechanismus durchaus geeignet, das JADOS-Problem zu lösen. Sobald aber die Zahl der Compiler und der Libraries wächst, wird es immer schwieriger, die Traps zu verwalten, da sich dann alle Beteiligten genauestens absprechen müßten, um nicht Doppelbelegungen zu produzieren. Außerdem verfügt der 600xx nur über 16 Traps, im Zusammenspiel mit dem Grundprogramm sogar nur über vier frei verfügbare. Die organisatorischen Probleme wären also viel zu groß.

Nach intensivem Nachdenken ist mir dann ein praktisches Lösungsverfahren eingefallen, das im Zusammenspiel mit dem (noch zu entwickelnden) Linker zu relokativen Programmen führt, die beliebig großen Code besitzen dürfen. Hierbei sind die vom Linker produzierten Programme voll verschiebbar. Eine Anpassung absoluter Adressen beim Laden in den Speicher ist nicht erforderlich, da es keine absoluten Adressen gibt!

Voraussetzung für die Lösung ist die Beschränkung der maximalen Modulgröße auf 32 KByte. Dadurch ist sichergestellt, daß alle modulinternen Sprungmarken mit relativen Sprüngen erreicht werden können. Diese Beschränkung ist völlig unkritisch und fördert außerdem das Konzept der Modularisierung. Übersetzungsprogramme hätten die Aufgabe, die 32 KByte-Grenze zu überwachen und gegebenenfalls mit einer Fehlermeldung abzubreaken. Alle externen Sprungmarken müssen am Modulanfang deklariert werden. Im Beispiel von Herrn Husemann z.B. mit

```
EXTERNAL PROCEDURE _prints
```

Ein Übersetzungsprogramm müßte nun aus dieser Externdeklaration folgenden kleinen Programcode generieren:

```
_prints:      lea    -2(pc),a6
              adda.l #xxxxxxx,a6
              jmp
```

Im Modul selbst wird _prints mit bsr _prints aufgerufen, also mit einem relativen Sprung. Dieser führt zur Marke _prints. Dort wird der aktuelle Programmzähler an der Adresse _prints berechnet (lea -2(pc),a6). Anschließend wird auf diesen Wert ein Offset addiert und dann registerindirekt gesprungen. Der Linker hat nun die Aufgabe, an die mit xxxxxx markierte Stelle die Adreßdifferenz zwischen dem Hilfslabel _prints und dem Beginn der eigentlichen _prints-Prozedur einzusetzen. Die Adresse kennt der Linker, wenn _prints in dem Modul, wo es realisiert ist, als global kenntlich gemacht ist; also z.B. mit

```
PUBLIC PROCEDURE _prints
```

Dieses Verfahren müßte für alle externen Sprungmarken durchgeführt werden. Jede Externdeklaration erfordert also einige Bytes Programcode. Die Vorteile des Verfahrens sind:

1. Die Anzahl der "Fixupadressen" für den Linker sind relativ gering. Für jede Externdeklaration muß nur ein Fixup durchgeführt werden.
2. Da der Linker Adreßdifferenzen einsetzt, bleibt das Programm voll relokativ.
3. Der Zeitverlust durch die Berechnung der Sprungadresse ist im allgemeinen geringer als bei den verschränkten Sprungketten und vor allen Dingen immer konstant und nicht von der Sprungdistanz abhängig.
4. Der Programmierer wird durch keine Formalismen belastet. Er muß nur alle externen Prozeduren am Modulanfang deklarieren und darauf achten, Prozeduren nur mit bsr aufzurufen.
5. Das vom Linker erzeugte Programm ist voll relokativ und muß beim Laden in den Speicher nicht angepaßt werden.

Mit der aufgezeigten Lösung wäre das JADOS-Problem also behoben. Was noch bleibt, wäre ein entsprechendes Verfahren für die Daten. Hier halte ich es für akzeptabel, den Datenbereich auf 64 KByte zu begrenzen. Dann können alle Daten relativ zu einem global vereinbarten Adreßregister adressiert werden, z.B. mit A4. Größere Datenstrukturen können dann mit Hilfe einer dynamischen Speicherplatzverwaltung angelegt werden, bei der die Daten zur Laufzeit angelegt werden und im Datenbereich lediglich einen Zeiger der Länge 4 Bytes beanspruchen.

Was bleibt jetzt noch zu tun ?

Als Entwickler des JADOS werde ich nun ein Format für Objectfiles definieren und den JADOS-Linker entwickeln. Unterstützung durch Rat und Tat sind mir dabei sehr willkommen! Irgendein gewitzter Programmierer müßte dann anschließend einen Assembler entwickeln oder anpassen, der Programme im Objectfileformat erstellt und Extern- und Publicdeklarationen unterstützt. Der Assembler sollte aufwärtskompatibel zum RDK-Assembler sein und Makros unterstützen. Gleichzeitig sind die Compilerbauer aufgerufen, ihre Compiler anzupassen. Dann ist Herrn Husemanns (und unser aller) Traum in Erfüllung gegangen.

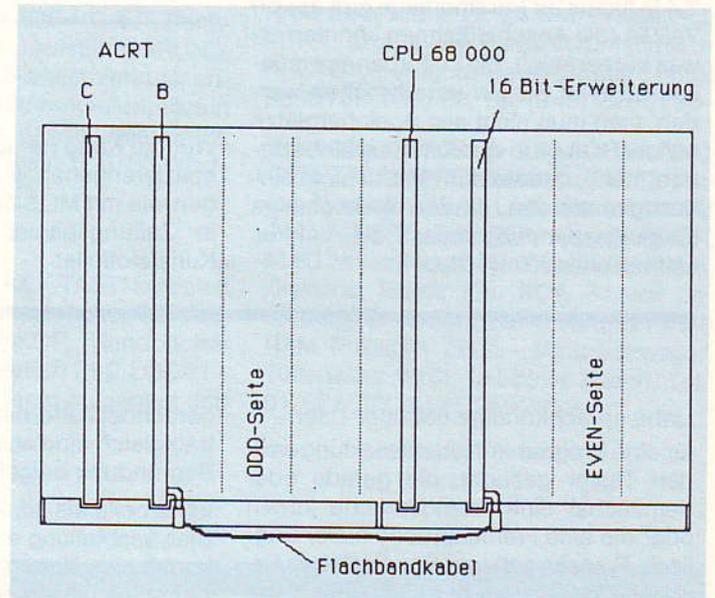
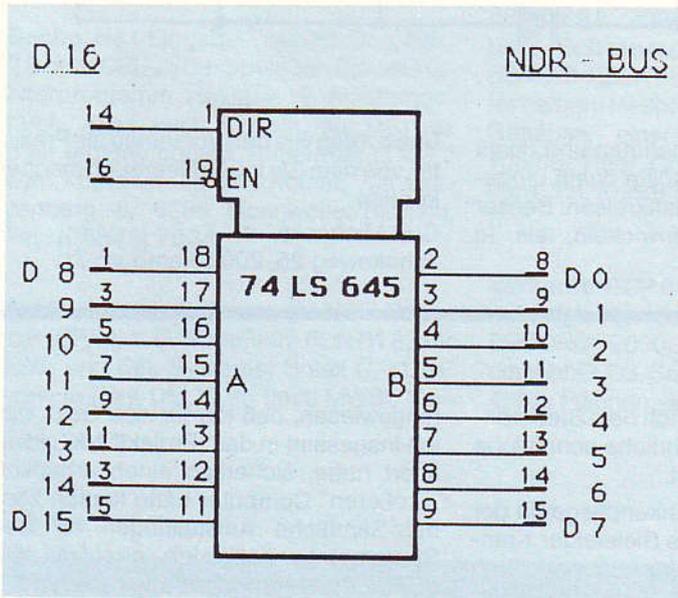
In die Hände haben wir nun gespußt; laßt uns also zur Tat schreiten !!!

Selbstbau der 16 Bit-Erweiterung für die ACRT

Um der ACRT-Baugruppe auch die Möglichkeit auf einen gleichzeitigen Zugriff

auf 16 Bit zu geben, hier nun eine kleine Ergänzungsschaltung, die über ein Flachbandkabel mit der ACRT-BASIS verbunden ist. Natürlich ist diese Schaltung nur bei den Prozessorkarten CPU 68000 und CPU 68020 sinnvoll. Verbunden mit dieser Zusatzschaltung ist ebenfalls eine Software-Änderung, da die 16

Bit nicht mehr wie bisher hintereinander geholt werden müssen. Die Software-Änderung sowie der Aufbau der Schaltung sind hierbei jedoch selbst auszuführen. Wichtig ist noch zu sagen, daß die ACRT auf der Odd-Seite des Prozessors gesteckt sein muß; die Zusatzschaltung auf der Even-Seite.



Briefe - Kontakte - Kleinanzeigen

Die LOOP gefällt mir zwar sehr gut, jedoch vermisse ich in letzter Zeit die Leserbriefe. Auch wäre es nicht schlecht, wenn ein wenig mehr auf die Hardware eingegangen würde, da in letzter Zeit zunehmend eine Spezialisierung auf die Software stattfindet. Zum Beispiel würden mich die Unterschiede der von GES und Elektronikladen angebotenen Karten interessieren. Als Beispiel seien hier die Floppy-Controller FLO3 und FDC erwähnt.

Es wäre auch sehr begrüßenswert, wenn einmal die Control-Bytes der verschiedenen Anwenderprogramme genannt und ihre Auswirkung auf den Programmablauf beschrieben würden.

Genug der Kritik, anschließend noch ein kleines Programm zur Ausgabe von Texten über die CO2-Routine. Die Startadresse des Textes wird wie gewohnt in A0 übergeben. Alle Control-Codes der CO2-Routine können in den Text mit eingefügt werden.

```

TEXTPT:
MOVE.B (A)+,DO
BEQ ENDTXT
JSR §CO2
BRA TEXTPT
ENDTXT:
RTS
    
```

Ich wäre erfreut, wenn auch mehr solche kleinen Programme veröffentlicht würden, die beim Programmwurf doch sehr hilfreich sind.

Zudem suche ich Kontakt zu Programmierern im Raum Kassel, kann jedoch auch ein größerer Umkreis sein, dann jedoch nur Briefkontakt. Meine Adresse:

Kai Uwe Bachmann
Mozartstraße 5, 3507 Baunatal I,
Telefon: (0561) 495091

Sehr geehrte LOOP-Redaktion!

Zunächst möchte ich mich einmal bedanken, für den Artikel in der letzten LOOP-Zeitschrift (Textbearbeitung), Seite 10. Das Programm läuft auf meinem Matrixdrucker „P6“ von NEC ausgezeichnet. Nun habe ich als Computer-Anfänger, und nicht mehr der Jüngste, mir hier in Hamburg bei „GES“ für meinen 68008-Computer einen Bausatz „Hardcopy“ gekauft.

Herr Michael Milzsch war so freundlich, mir gleich ein Programm für die Hardcopy mitzugeben. Dieses Programm läuft jedoch nicht richtig. In der Hoffnung dieses Programm an meinen „P6“ angepaßt zu bekommen verbleibe ich.

Herbert Oschkinat,
Horner Landstr. 418, 2000 Hamburg 74,
Telefon: (040) 7325431

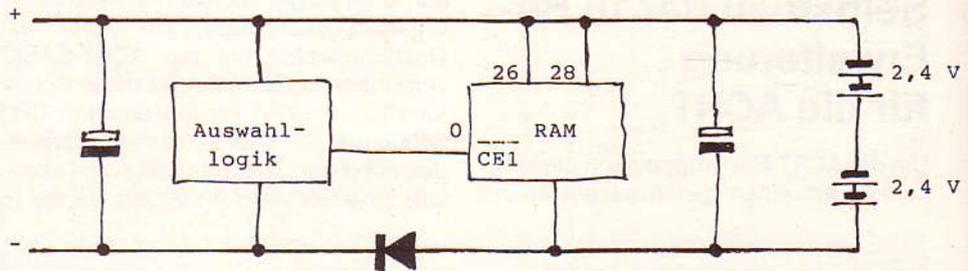
PS.: Da sich offensichtlich die preiswerten 24-Nadeldrucker immer mehr durchsetzen, wäre es ganz schön, wenn das Hardcopy-Programm auch an diese 24 Nadeln angepaßt wäre.

Antwort LOOP: Wer kann Herrn Oschkinat helfen?

Ein akkugepufferter RAM-Speicher, wie es ihn bei Taschenrechnern und handheld-Computern gibt, ist schon eine attraktive Sache. Wer allerdings versucht hat, dies mal eben auch für den NDR-Computer zu verwirklichen, wird seinen Ärger damit gehabt haben: Nach dem Einschalten gibt es immer wieder einzelne, für den jeweiligen RAM-Typ typische Fehler! Ich habe lange daran rumgeknoelt, bis ich schließlich auf den Gedanken gekommen bin, daß die RAMs das rabiate Einschaltverhalten des stabilisierten Netzteils übernehmen; gewissermaßen fallen dabei immer wieder ein paar Kegel um! Statt nun ein Netzteil einzusetzen, das die Spannung sanft hochfährt und den Prozessor entsprechend warten zu

lassen, habe ich den Spannungssprung auf eine andere Weise „widerlegt“: Ich habe zwei Akkus mit insgesamt vier Ni-Cd-Zellen auf die Karte gesetzt! Die Schaltung wird dadurch noch einfacher:

Die Diode *muß* ein Schottky-Typ sein; sie *muß* in der negativen Zuleitung liegen, damit CE1 (Pin 20) auf VDD gehalten wird. 2,4 V-Akkus für Printmontage gibt es von VARTA (die Anschlußfahnen könnten etwas kürzer sein!). Die Leiterbahnen müssen an zwei Stellen unterbrochen werden; man muß nicht alle Speicherplätze auf der ROA 64 in die Pufferung einbeziehen; insbesondere darf die Uhr nicht einbezogen werden, da der niederohmige Eingangsspannungsteiler die Akkus schnell entladen würde.



Wer die Karte herausnehmen und damit spazieren gehen will, sollte damit umgehen wie mit MOS-Schaltkreisen: Besser in Zeitungspapier einwickeln, als in Kunststoffolie!

Betrachten Sie den Vorschlag als Praxistipp, von dem Sie freien Gebrauch machen können.

Curt Michaelis,
Jahnkeweg 25, 2000 Hamburg 71

Liebe sprachkundige 68000er-User, für eine Programm-Neuentwicklung werden Tester gesucht, die gerade oder demnächst eine Fremdsprache lernen oder die eine Fremdsprache *außer* Englisch, Französisch und Spanisch beherrschen (für „Vielzeichen-Sprachen“, wie Chinesisch und Japanisch, besteht – noch – kein Interesse).

Gesucht sind jedoch Leute, die sich z.B. mit Dänisch, Schwedisch, Norwegisch, Italienisch, Griechisch, Russisch, Hebräisch, Arabisch, Türkisch usw. befassen.

Haben Sie solche Sprachen bereits mit dem FRAGE/ANTWORT-TRAINER erfaßt, dann melden Sie sich bitte auf jeden Fall. Geben Sie bitte an, welches DOS Sie einsetzen und – falls Sie die Sprache gerade lernen – mit welchen Lehrbüchern Sie arbeiten. Ideal ist auch, wenn Ihre Muttersprache „ausländisch“ ist.

Bitte schreiben Sie an:

Manfred Zehner,
Schwenckestraße 2, 2000 Hamburg 20

Hier: Heft 10/S. 27 und Heft 12/S. 23

Stichwort: Computer und Steuer

Sehr geehrte Damen und Herren, nach Rückkehr aus dem Osterurlaub enthielt der Poststapel u. a. auch den Steuerbescheid für das Jahr 1986. Und siehe da, sämtliche Kosten, die ich als Werbungskosten geltend gemacht hatte, waren anerkannt. Daher möchte ich die beiden o.e. Artikel um eine weitere Variante erweitern.

Computer und Finanzamt: Um die Frage von Herrn Simon in LOOP 10/S. 27 und die Überschrift in LOOP 12/S. 23 zu beantworten: Ja, es geht!

Der Finanzbeamte fragte recht erstaunt, wieso ich als Krankenpfleger einen Computer absetzen wolle. Da ich mit der Frage

gerechnet hatte, hatte ich dem Steuerantrag gleich eine ausführliche schriftliche Begründung beigefügt.

Ich arbeite als lfd. Krankenpfleger in der Dialyseabteilung eines Bielefelder Krankenhauses. Neben meinen pflegerischen Aufgaben bin ich auch zuständig für Reparaturen, Wartung und Sicherheit der Dialysegeräte. Da diese Tätigkeiten recht wenig mit der eines Krankenpflegers gemein haben, mußte ich mir verständlicherweise die hierzu nötigen Kenntnisse selbst aneignen. Für die älteren Geräte reichten elektronische Grundkenntnisse, die ich mit dem ebenfalls steuerlich anerkannten „Kompaktkurs Elektronik“ des Instituts Christiani aufgefrischt habe. Die Dialysegeräte der neuen Generation sind dem Stand der Technik entsprechend mit Mikrorechnern ausgestattet. Und wie eignet man sich diese Kenntnisse eben besser an, als durch den schrittweisen Bau eines eigenen Computers. Damit nun, wenn die Struktur eines solchen Gerätes einem annähernd klar geworden ist, dieses nicht nutzlos in der Ecke steht, liegt es nahe, den Rechner zu einem, wenn auch etwas veralteten, aber dennoch praxisnahen Computer auszubauen. Mittels dBASE II wurden Übersichten für die tägliche Arbeit erstellt, ja sogar ein Patientenverwaltungsprogramm wurde begonnen. Mit Wordstar 3.0 werden ein Teil des Schriftverkehrs auf der Station oder Manuskripte für den Unterricht in der Krankenpflegeschule erstellt. Mein Aufgabengebiet habe ich mir vom Arbeitgeber ausführlich beschreiben lassen, desgleichen habe ich mir vom Hersteller der Dialysegeräte die Notwendigkeit von umfangreichen Kenntnissen in Elektronik, Digitaltechnik und Mikroprozessortechnik bescheinigen lassen, die zur Teilnahme an Technikerseminaren vorausgesetzt werden. Schließlich habe ich ausdrücklich darauf

hingewiesen, daß ich für das Geld, das ich insgesamt in das „Projekt“ NKC investiert hatte, sicherlich einen erheblich „größeren“ Computer hätte kaufen können. Sämtliche Aufstellungen für den Steuerantrag habe ich ebenfalls mit dBASE II erstellt und alles, fein säuberlich mit dem ebenfalls abgesetzten Drucker ausgedruckt, beim Finanzamt vorgelegt.

Die Gesamtaufwendungen habe ich auf 4 Jahre verteilt; das erste Viertel ist anerkannt, also dürfte der Rest auch keine Probleme mehr bereiten.

Peter Schwarck,
Reyener Straße 67, 4905 Spenge,
Telefon: (05225) 17 69

Kleinanzeigen

TEXTEDITOR NDR 68k: 2 Dat. gleichztg. editierbar (Fenster) Wordstar komp. arbeitet sehr schnell! Progr./Dokum-Modus, Blocksatz, progr. Druckertreiber ..., CPM 68k, 79 DM, JADOS i. V. **SPURPUFFER-BIOS:** Floppyzugr. unter CPM 68k 5 x schneller! INFO 1,20 DM, H. Gohs, Buchenring 42 b, 7513 Stutensee, Tel. (0721) 683129

NDR 68000/8: JADOS-Entwickler verk.

- starkes Strategiespiel REVERSI 1.1, jetzt auch maugesteuert! DM 39,-
- menügesteuerten DISASSEMBLER, mit Hexdump und Abspeichern auf Datei DM 50,-
- Diskettenmonitor mit Sektoren-Editor, Suchfunktion und Dateimodus DM 35,-
- Assemblerquellmodule für: Ansteuerung von Maus/Fadenkreuz DM 18,-, Ansteuerung der Smartwatch DS1216 DM 18,- + Porto 3,50 bei V-Scheck, 5,- DM bei NN, auf Disk 3 1/2" oder 5 1/4", unter JADOS, Janßen, Hannixweg 74, 4150 Krefeld 1

NDR-CP/M-PLUS-KOMPLETTSYSTEM im Plexiglasgeh., HW: 128 KB, 2 x FD à 780 KB, DIN-Tast., Monitor, Centr., V. 24, Promer, SW: CP/M-3 + Doku., Basic, SLR-Ass, Multiplan, Bios-Quellprog., Schneider-CPU-kompatibel. Preis VB, Tel.: (0911) 880008, G. Jörger, Kemptener Str. 64, 8500 Nürnberg 60

Suche: Hex-Eingabe-Tastatur, Bus, SBC 2,3 oder CPU Z80 + Speicher, Eprom Monitorprogramm Hexmon + Anleitungsbuch. Alles auch einzeln, als Bausatz oder fertig aufgebaut. Außerdem LOOP 1 zum kopieren. Jürgen Römer, Am Butzenberg 8, 6696 Nonnweiler/Bierfeld, Tel.: (06873) 7934

Public-Domain-Software für NDR-Rechner (CP/M) z. B. Sprachen: FORTH 83 für Z80- und 68k-Systeme; Small C, XLISP usw. je Disk DM 19,95 (incl. MWSt, Ver-

sand gegen Vorkasse o. per Nachnahme). Außerdem Anwendungen, Utilities, Spiele: Bestellung oder Info bei: Roland Steyer, Softwareversand, Kappenbergweg 2, 6200 Wiesbaden 15

Wegen Zeitmangel **zu verkaufen:** CPU Z80, KEY, große Tastatur, CAS, 2 x ROA 64 mit 12 x 8 k, GDP 64k, IOE + CENT, Bankboot, Schaltnetzteil, Prufstift, EGRUND, EBASIC, EZEAT und Assemblerkurs, unter halbem Neupreis zu verkaufen, alles in Gehäuse, einschl. Kassettenlaufwerk. Thomas Heer, Goethestr. 11, 8700 Würzburg, Tel. (0931) 51636

Verkaufe: GDP 64k, KEY, TAST1 komplett mit Originalgehäuse und -kabel, EBASIC, EGRUND 2000, ESKOP, Handbücher einschl. FLO3, Sonderheft 1+2, LOOP1 - 4; alle Platinen sind fertig aufgebaut und

funktionsfähig, Verhandlungsbasis: DM 350,-, einzeln: ca. 1/2 Bausatzpreis, Josef Eisele, Tel. (0807) 8254

Kombikarte ROA 512/128 Neu lieferbar: Speicherkarte wahlweise zu bestücken mit 16 32 x 8 RAMs (=512 kB) oder 16 8 k x 8 RAMs, EPROMS (= 128 k), Preise: Leerplat. 45 DM, fert. best. ohne ICs 120 DM, RAM 32 k x 8 z. Zt. 24 DM, Preise incl. MWSt. zzgl. 4 DM Porto/Verp. HEINZ + KRUTOF, August-Bebel-Straße 123, 2050 Hamburg 80, Tel. (040) 7206196

Zu verkaufen fertig aufgebauter NKC: CPM-Z80, IBM-PC-Gehäuse, TAST2, Philips Monitor, 2 TEAC Laufwerke, RAM 64/256, AD 8/16, DA, SER, IOE, PROMER, M 80 Assembler, Turbo Pascal, Tool Disk, DiskDoc. Ersatz ICs, ROA 64 voll bestückt, ROA 64 mit EGRUND und 1 k x 8 RAM. Preis SFr. 2800,-. Marc Schwager, Tüffenwies 19/31, CH-8064 Zürich, Tel.: 01/624563, G: 01/2562259

Speziell für LOOP-Leser

Matrix-Printer BX 100



Technische Daten:

Punktmatrix-Drucker mit 100 Zeichen/sec. · Friktions- u. Traktortrieb
Bidirektionaldruck mit Druckwegoptimierung · Inkremental-Druck
Proportional-Schrift · Unidirektionaldruck bei Graphik
Papiertransport vorwärts/rückwärts · Kursiv-Schrift
8 Nationale Zeichensätze · Programmierbarer Zeilenabstand
Text-Matrix: 9 x 9 Dots · Bit Image Mode (1 : 1 Graphik!)
Graphic-Matrix: 8 x 480 bzw. 576 Dots · 8 x 640 bzw. 720 Dots
8 x 960 bzw. 1920 Dots · 9 x 480 / 960 Dots

Programmierbare Seitenlänge · Skip over perforation
Programmierbarer linker Rand / rechter Rand
Vertikaler Tabulator · Horizontaler Tabulator
Papierbreite: 4" bis 10" (114,3 mm – 254 mm)
1 Original + 2 Durchschläge · Leistung ca. 80 Watt
Lebensdauer Druckkopf ca. 100 000 000 Zeichen
Lebensdauer Farbband ca. 2 500 000 Zeichen

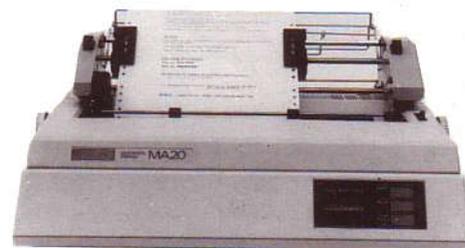
LOOP-Leser-Aktion

inkl. 14 % MWSt.

Versand per UPS-NN – frei Haus **DM 798,-**

Typenraddrucker PETAL-MA 20

Korrespondenzqualität!
Jetzt zum erschwinglichen Preis.
Anschließbar an alle Mikrocomputer!



LOOP-Leser-Aktion:

inkl. 14 % MWSt.

Versand per UPS-NN – Frei Haus **DM 798,-**

Optional: Traktor DM 349,- inkl. MWSt.

**mirwald
electronic**

Computer-Peripherie

Fasanenstraße 8b · 8025 Unterhaching/München
Tel. (089) 6111224 und 6112040 · Telex 5213476

Büro Frankfurt:

Adalbertstr. 15 · 6000 Frankfurt 90 · Tel. (069) 703538

Wir stellen vor:

Neue Produkte für NDR- und mc-CP/M-Computer. Vom Microsoft-Entwicklungspaket für DM 248,- bis zum BASIC-Interpreter für den 68020.

ROA256/1M

Die neue Standard-RAM-ROM-Baugruppe für den NDR- und mc-CP/M-Computer. Bestückbar gemischt mit RAM (32 KByte) oder EPROM, NDR-Bus und ECB-Bus. Zusatzoption: Akkupufferung, dadurch ideal als RAM-Floppy. Vorbereitet für 128 KByte RAMs bzw. EPROMs, Speicherkapazität dann 1 MByte (statisch).

Best.-Nr.	Bezeichnung	Preis DM
11028	Handbuch	20,00
11029	Leiterplatte	40,00
11026	Bausatz ohne Akku-Pufferung	98,00
11027	Fertigerät ohne Akku	158,00
11030	Bausatz mit Akkupufferung	148,00
11031	Fertigerät mit Akkupufferung	198,00

Hinweis: Alle Geräte werden ohne Speicher geliefert!

Passende Speicher:

11068	RAM32K x 8, 84256, Fuj. 120 ns	28,00
11067	EPROM 27256 32K x 8	14,90

Hinweis: Speicherpreise können von Tag zu Tag schwanken!

Color Look Up Tabelle CLUT

Universelle Color Look Up Tabelle. NDR-Bus und ECB-Bus. Eingang: RGB digital, z.B. TERM, ACRTC oder COL256. Ausgang: RGB-Analog. Buchsen: IBM-Belegung oder 5 mal BNC. Farben: 256 gleichzeitig aus 262144.

Best.-Nr.	Bezeichnung	Preis DM
10716	Handbuch	20,00
10715	Leiterplatte	50,00
10714	Bausatz, komplett	398,00
10713	Fertigerät	488,00

ROB2

Univers. Leistungs-Karte f. NDR- u. ECB-Systeme
Acht Leistungsstufen, 5 ... 12 V, 1 A. Acht Digital-Eingänge. Zwei Analog-Eingänge. Besonders geeignet zur Ansteuerung von Gleichstrom-Motoren. Schnittstellen-Belegung kompatibel zu Fischer-Technic „computing“-Modellen.

Best.-Nr.	Bezeichnung	Preis DM
10632	Handbuch	20,00
10783	Leiterplatte	45,00
10631	Bausatz	228,00
10629	Fertigerät	298,00

Programme zur Roboter-Steuerung mit dem Einsteigerpaket

Best.-Nr.	Bezeichnung	Preis DM
10925	Handbuch mit Quellisting	20,00
10924	EPROM für SBC3 (2764)	40,00
10923	EPROM für SBC2 (2732)	40,00

Eproms jeweils inkl. Handbuch!

Smart Watch

Der Uhrenbaustein für den NDR-Computer. Wird einfach auf RAM-Einbauplatz gesteckt. Akku-Pufferung, auch f. RAM 1/100 sec bis Jahr.

Best.-Nr.	Bezeichnung	Preis DM
10702	Smart Watch (jetzt billiger)	98,00

BUS-Steckerverbindersatz

Ermöglicht das problemlose Verbinden zweier Busse bzw. eines Busses an die HEXIO2.

Best.-Nr.	Bezeichnung	Preis DM
11018	Satz	25,00

Das neue Buch zum Z80-Teil des NDR-Computers Rolf-Dieter Klein: „Rechner modular“, Franzis-Verlag, Best.-Nr. 10991 ... DM 68,00

Das Buch deckt die gesamte Information des Z80-Teiles des NDR-Systems ab, von den Anfängen der Digital-Elektronik über die Baugruppen bis zur Software und Z80-Programmierung.

Microsoft BASIC- u. Assembler Entwicklungspaket:

MBASIC-Interpreter, BASCOM-Compiler, M80 Makro-Assembler und TOOLS für DM 248,00!

Auf der Diskette befinden sich:

MBASIC	BASIC-Interpreter, V 5.21	L80	Linking Loader
BASCOM	BASIC-Compiler, V 5.4	CRF80	Cross Reference
M80	Makro-Assembler	LIB	Library Manager

... und die komplette Original-Dokumentation wird im Ringbuch (ca. 1,25 kg) mitgeliefert.

Best.-Nr.	Bezeichnung	Preis DM
11059	Microsoft-Entwicklungstools 3 1/2" 80 Spuren	248,00
11058	5 1/4" 80 Spuren	248,00

RL-BASIC V 2.4

Das mächtigste BASIC für den NDR-Computer. BASIC-Interpreter unter allen 68000-CPUs. Unterstützt 68020 u. Floating Point Unit, damit extrem schnell! Läuft allein sowie unter JADOS. Unterstützt Diskettenzugriff sowie Devices. Strukturierte Programmierung durch Befehle wie REPEAT ... UNTIL, WHILE ... WEND möglich. Sprungziele als Marken definierbar (GOSUB gibaus!). Kostengünstiges UPDATE möglich.

Best.-Nr.	Bezeichnung	Preis DM
-----------	-------------	----------

RL-BASIC, Vollprodukt, auf Diskette (benötigt JADOS):

10680	RL-BASIC auf 3 1/2" Diskette	159,00
10679	RL-BASIC auf 5 1/4" Diskette	159,00

RL-BASIC, Vollprodukt, auf EPROM (ohne JADOS):

10676	EPROMS für 68008	159,00
10677	EPROMS für 68000	159,00
10678	EPROMS für 68020	159,00

UPDATES

entsprechend dem Vollprodukt, werden bei Vorlage der Rechnungsnummer oder der Seriennummer des Originalproduktes ausgeliefert. Ein Zurücksenden der Originaldiskette oder Eprom ist nicht mehr nötig!

Best.-Nr.	Bezeichnung	Preis DM
11035	Diskette 3 1/2"	50,00
11036	Diskette 5 1/4"	50,00
11032	EPROMS für 68008	75,00
11033	EPROMS für 68000	75,00
11034	EPROMS für 68020	75,00

JADOS TOOL DISC

JADOS ist ein komfortables Disketten-Betriebssystem, das von vielen NDR-Benutzern mit 680xx-Prozessoren eingesetzt wird. In der ersten TOOL-Disc sind 76 Programme enthalten, die von der technischen Anwendung (Unterstützung A/D-Wandler als Oszilloskop) bis zum Spiel (3-D-Labyrinth) reichen.

Auszug aus den Programmen: Disk-Verwaltung, Text-Formatierer, Eprom-Programmierer, Tennis-Spiel (extrem schnell), Matrizen-Berechnung, Lissajous-Figuren-Erzeugung, Funktionen-Darstellung, Fourier-Entwicklung.

Best.-Nr.	Bezeichnung	Preis DM
11056	Diskette 3 1/2"	69,00
11055	Diskette 5 1/4" 80 Spuren	69,00

Neues Gehäuse für den mc-CP/M-Computer (... und andere Europakarten-Systeme).

PC-ähnliches Gehäuse für 10 Europakarten. Vorbereitet zum Einbau von zwei oder vier 5 1/4" oder 3 1/2" Laufwerken sowie eines (IBM-kompatiblen) Netztes.

Best.-Nr.	Beschreibung	Preis DM
-----------	--------------	----------

11041	Handbuch zum Gehäuse 4	10,00
10981	Gehäuse 4, Fertigerät, mit 19" Rahmen für 10 Europakarten, ohne Netzteil, Bus und Verdrahtung	285,00

Zubehör:

11039	NE4, Schaltnetzteil, IBM-komp. + 5V 15A, -5V, 1A, +12V 5A, -12V 1A, kompl. mit Befestigungsmaterial	198,00
10847	ECB-BUS-Leiterplatte, 10 Plätze	59,00
10836	Bausatz	149,00
10857	Fertigerät, geprüft	198,00
10789	3 1/2" Laufwerksmodul, ermöglicht Einbau eines 3 1/2" Laufwerkes	50,00
11040	Kompletthaus, mit eingebautem und verdrahtetem Netzteil 11039 sowie BUS 10857, geprüft	698,00

BIOS und BIOS-Patch

für NDR- und mc-CP/M-computer

Best.-Nr.	Bezeichnung	Preis DM
11022	Diskette 3 1/2"	49,00
11023	Diskette 5 1/4" 80 Spuren	49,00
11024	Diskette 5 1/4" 40 Spuren	49,00
11025	Diskette 8"	49,00

BIOS-Diskette nur für mc-CP/M-Computer

unterstützt RAM-Floppy und EPROM-Floppy sowie gemischten Betrieb 5 1/4" / 8".

Best.-Nr.	Bezeichnung	Preis DM
10965	Diskette 3 1/2"	49,00
10966	Diskette 5 1/4" 80 Spuren	49,00
10945	Diskette 5 1/4" 40 Spuren	49,00
10946	Diskette 8"	49,00

BIOS-Patch-Programm

für NDR- und mc-CP/M-Computer, CP/M 2.2

Ermöglicht einfachstes Selbst-Generieren eines BIOS und damit Anpassung an beliebige Laufwerk-Formate. Formate können abgespeichert werden.

Best.-Nr.	Bezeichnung	Preis DM
11057	BIOS-PATCH, 3 1/2" 80 Spuren	95,00
10869	BIOS-PATCH, 5 1/4" 80 Spuren	95,00

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

Menügesteuertes CAD, 400 TTL-Symbole Bauteile-Bibliothek, Zeichenfläche bis 6 x 6 Meter, Auflösung bis 1/100 mm, gleichzeitiger Zugriff auf 5 Bibliotheken / 1000 Symbole. Neuer, günstiger Preis.

Best.-Nr.	Bezeichnung	Preis DM
10725	3 1/2" 80 Spuren	398,00
10724	5 1/4" 80 Spuren	398,00

MCOPI

Liest und schreibt IBM-Disketten mit dem NDR-Computer unter JADOS oder CP/M68K.

Best.-Nr.	Bezeichnung	Preis DM
11047	Für JADOS, 5 1/4" 80 Spuren	49,00
11049	Für JADOS, 3 1/2" 80 Spuren	49,00
11046	Für CP/M68K, 5 1/4" 80 Spuren	49,00
11048	Für CP/M68K, 3 1/2" 80	49,00

COL-Treiber für 680xx-Anwender

Der COL-Treiber unterstützt mit 80 Routinen die universelle Farb-Graphikbaugruppe COL 256 des NDR-Computers. Die Möglichkeiten der Treiberprogramme reichen vom Initialisieren über den Hardware-Scroll bis zur Erstellung und Verwaltung von Windows.

Best.-Nr.	Bezeichnung	Preis DM
10706	3 1/2" 80 Spuren	95,00
10705	5 1/4" 80 Spuren	95,00

Makro-Link-Assembler MLA1

Ermöglicht das Erstellen und Einbinden von Makros unter 680xx-Assembler-Programme.

Best.-Nr.	Bezeichnung	Preis DM
11060	MLA1, Disk und Handbuch	98,00

DATEI2 Die neue Dateiverwaltung unter JADOS

Best.-Nr.	Bezeichnung	Preis DM
11062	5 1/4" 80 Spuren	96,90
11061	3 1/2" 80 Spuren	96,90

Neue Preise „alter“ Produkte:

Best.-Nr.	Bezeichnung	Preis DM
10062	ACRTC-BASIS Fertigerät	798,00
10064	ACRTC-Farberweiterung	798,00
10157	COL256P, 3 Platinen	98,00
10150	COL256 Bausatz	298,00
10154	COL256 Fertigerät	398,00
10401	Bausatz Sprachausgabe	198,00
10402	Fertigerät Sprache	258,00
10609	CP/M-Z80 Emulator-Programm für CP/M68K, 5 1/4" 80 Spuren	298,00
11065	dto., 3 1/2" 80 Spuren	298,00

GRAF ELEKTRONIK SYSTEME GMBH

Magnusstraße 13 • Postfach 1610 • 89 60 Kempten (Allgäu) • Telefon: (08 31) 6211

Teletex: 831804=GRAF • Telex: 17831804=GRAF • Datentelefon: (08 31) 69330

GRAF[®] computer