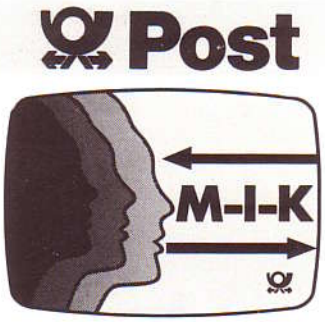
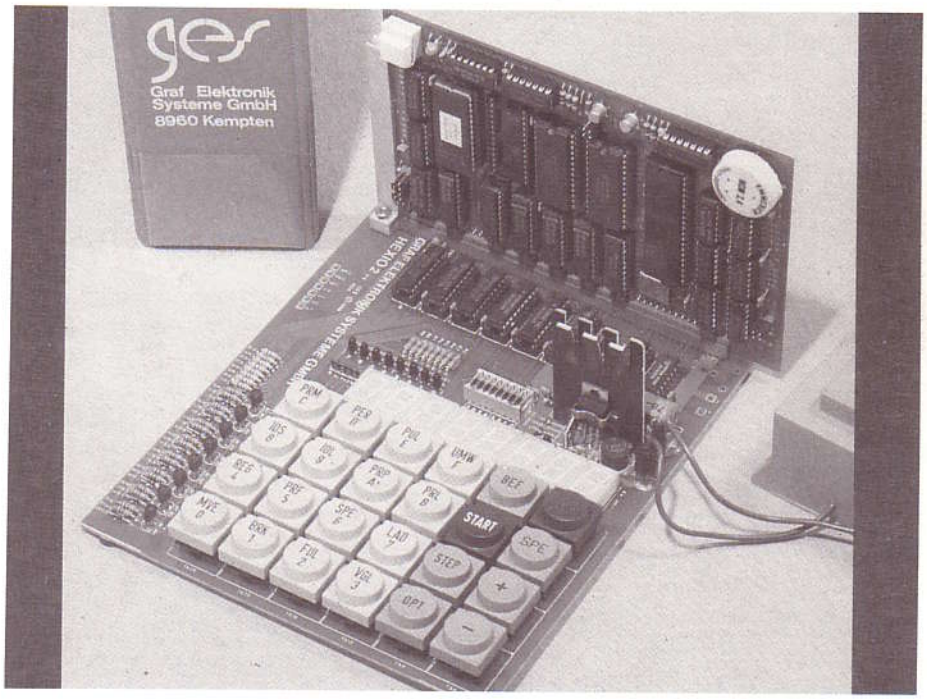
10.11.87
Uwe Koch
Frankenstraße 25
5880 LÜDENSCHEID
Tel. (0 23 51) 2 61 92

LOOP 15
3. JAHRGANG
Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter DM 3,–



Der NDR-Computer im Aufwind

Deutsche Bundespost setzt NDR-Computer ein



Bereits 1984 veröffentlichte die Bundesregierung eine Konzeption zur Förderung der Entwicklung der Mikroelektronik, der Informations- und Kommunikationstechniken (M-I-K). In dieser Konzeption werden unter anderem auch Forderungen an die betriebliche Bildungsarbeit im Zuge der Einführung der neuen Techniken gestellt.

Die Deutsche Bundespost ist als größtes Kommunikationsdienstleistungsunternehmen in der Bundesrepublik besonders gefordert. Deshalb wurde im Jahr 1986 ein umfassendes „Programm zur Förderung der Mikroelektronik, der Informations- und Kommunikationstechniken im Bereich der beruflichen Bildung der Deutschen Bundespost (Programm M-I-K)" gestartet.

Das Programm M-I-K verfolgt das Ziel, alle Mitarbeiterinnen und Mitarbeiter der Deutschen Bundespost auf den Technologiewandel vorzubereiten, die Akzeptanz im Umgang mit den neuen Kommunikationstechniken zu erhöhen und durch eine bessere Information eine höhere Motivation der Beschäftigten zu erreichen.

Der z. Z. sehr hohe (Nachhol)Bedarf an Fortbildung auf den Gebieten der M-I-K kann allein mit herkömmlichen Bildungsmaßnahmen nicht mehr bewältigt werden. So wird unter anderem als alternative Vermittlungsform die Wissensvermittlung durch Fernlehrgänge eingeführt.

Innerhalb dieses Fernlehrganges wird der NDR-Computer (modifiziertes Einsteigerpaket) verwendet.

Zur Zeit wird vom Bundespostministerium in Zusammenarbeit mit der Fachschule der deutschen Postgewerkschaft durchgeführt.

Besonders interessant an diesem Konzept ist, daß von der Deutschen Bundespost allen Teilnehmern des Lehrganges der NDR-Computer für die Dauer von drei Monaten kostenlos überlassen wird. Nach erfolgreichem Abschluß des Lehrganges kann der Teilnehmer den Computer dann entweder zurückgeben oder gegen einen geringen Kostenanteil behalten – die Deutsche Bundespost übernimmt dann einen großen Teil der Kosten.

In letzter Minute:

dBasell, Wordstar, Multiplan jetzt noch billiger:

Je Programm nur DM 99,–

Es ist kaum zu glauben: Die beliebte Standard-Software ist noch einmal um die Hälfte billiger geworden! Für den Preis von DM 99,– erhalten Sie das voll lauffähige Original-Produkt mit der kompletten Dokumentation! Lieferbar: Ab Lager!

Diese Möglichkeit der freiwilligen Weiterbildung erfreut sich sehr großer Beliebtheit – bereits wenige Wochen nach Bekanntgabe war die ursprünglich angepeilte Teilnehmerzahl von zunächst 1000 um ein Vielfaches überschritten.

Anfang Oktober wurden die Systeme und die Lehrgänge ausgeliefert.

Anmelden können sich die Mitarbeiterinnen und Mitarbeiter der Deutschen Bundespost bei der für den OPD-Bezirk zuständigen Bezirksfachschule. Die Lehrgangsteilnehmer sind aber nicht alleine gelassen – die laufende Betreuung erfolgt durch Tutoren, die von der Fachschule mit dieser Aufgabe betraut werden und auch während des Lehrganges Informationsveranstaltungen durchführen.

Der Einsatz des NDR-Computers bei der Deutschen Bundespost ist nicht nur ein kommerzieller Erfolg – wir sind unserem Ziel, mit Hilfe des modularen NDR-Computers etwas für die Weiterbildung im Bereich Mikroelektronik innerhalb der Bundesrepublik zu tun, einen gewaltigen Schritt näher gekommen.

Die Chance für alle unentdeckten Erfinder

Machen Sie mit beim 2. LOOP-Wettbewerb

Warengutscheine über 1000,–, 500,– und 100,– DM winken! Jeder Teilnehmer erhält eine Überraschung! Thema: Anwendungen mit dem Einsteigerpaket.

Endlich ist es soweit: Wir haben einen neuen *LOOP*-Wettbewerb ausgeschrieben. Diesmal ist das Thema klar begrenzt:

Anwendungen mit dem Einsteigerpaket

Das Einsteigerpaket findet, besonders in der Ausbildung, mehr und mehr Anwender. Wir suchen nun Anwendungen, die über die Ausbildung hinausgehen; oder – anders ausgedrückt: Was kann ich mit dem Einsteigerpaket eigentlich machen, nachdem ich begriffen habe, wie ein Computer funktioniert?

Hier ist besonders die sinnvolle Anwendung im Privatleben interessant. Einige Anregungen:

Wetterstation – die Werte können gespeichert und verglichen werden,

Geschwindigkeitsmessungen – z. B. für Trend-Geschwindigkeitsbestimmungen in verkehrsberuhigten Zonen,

Lichtorgel – mit programmierbaren Mustern, für den Partykeller,

durch Beleuchtung und Zufall – zeitgesteuerte Lampenschaltung zur Sicherung des Hauses während der Urlaubszeit,

Eisenbahnsteuerung für die Modelleisenbahn,

Alarmanlagensteuerung,

Vielfachtimer für das automatisierte Frühstück,

Telefonregister (mit Wähleinrichtung – für Nebenstellen),

Tinmer für Foto- und Filmstunde.

Man erkennt an diesen Anregungen: Zusatzhardware, wie Lichtschranken oder Leistungsrelais sind erlaubt. Hierbei sind allerdings folgende Voraussetzungen zu beachten:

Die Zusatzhardware muß auch von Anfängern leicht aufzubauen sein.

Sie muß (bei Ansteuerung höherer Spannungen) den VDE-Vorschriften genügen (Optokoppler, Sicherheit!).

Die Bauteile müssen gängig sein und im Elektronikhandel (z. B. Conrad) verfügbar sein.

Programme müssen für Anfänger geschrieben und dokumentiert sein. Programme sollen in Assembler- (mit Kommentaren), nicht in Maschinensprache vorliegen. Im Notfall genügt „Hand-Assemblierung".

Was müssen Sie einsenden?

- Die funktionstüchtige Zusatzhardware (**ohne** das Einsteigerpaket)
- Die Dokumentation
- Ein EPROM mit dem Programm

Einsendeschluß ist der 31. 12. 1987

Teilnehmen kann jeder, Mitarbeiter der GES ausgenommen.

Mit der Teilnahme gehen die Rechte, insbesondere zur Veröffentlichung, an GES über.

Und zuletzt: Die Preise!

1. **Preis: Warengutschein im Wert von DM 1000,–,**
2. Preis: Warengutschein im Wert von DM 500,–,
3. Preis: Warengutschein im Wert von DM 200,–,
4. bis 10. Preis: Warengutschein im Wert von DM 100,–.

Die Preise werden über eine Jury vergeben – der Rechtsweg ist ausgeschlossen!

Alle Teilnehmer erhalten einen „Teilnahmepreis" – also, dabeisein ist alles!

Ein Tip: Wir bewerten neben der Originalität und der praktischen Einsatzmöglichkeit auch die Qualität der Dokumentation und den didaktischen Inhalt. – Nun aber los! Wir freuen uns auf Ihre Einsendungen!

In eigener Sache

Stolz sind wir schon darauf, daß der NDR-Computer nun innerhalb der Deutschen Bundespost eingesetzt wird! Durch die dadurch erheblich verbreiterte Basis der Anwender ergeben sich sicher viele neue Impulse für das gesamte System.

Übrigens: Alle Mitarbeiter der Bundespost, die den Fernlehrgang absolvieren, sind automatisch *LOOP*-Abonnenten. Herzlich willkommen!

Zum zweiten Mal rufen wir nun zum *LOOP*-Wettbewerb auf. Bitte, machen Sie zahlreich mit. *Jeder* Einsender erhält einen Preis! Auch unsere „Cracks", die ganz oben schweben, haben sicherlich noch die eine oder andere interessante Lösung in der Schublade!

Auch in dieser *LOOP* können wir wieder viele Preisreduzierungen bekanntgeben. Besonders ist die nochmalige Preis-Halbierung der Multiplan, dBasell und Wordstar-Serie zu erwähnen – hier stimmen Preis und Leistung absolut überein.

Unser neuestes „Kind", das wir zusammen mit der Fachzeitschrift „mc" veröffentlichten, der „mc-modular-AT", hat eine unerwartet hohe Resonanz gefunden. Wir werden, nachdem die Artikelserie in der mc so gut angelaufen ist, auch in der *LOOP* einen „IBM-Teil" einrichten. Hier soll weniger über die Hardware als über IBM-kompatible Software berichtet werden. Es soll ab der nächsten *LOOP* losgehen: Autoren, Ihr seit gefordert!

Mit Einführung des mc-modular AT's haben wir unsere Produktlinie erweitert – keinesfalls werden bestehende Produkte, wie NDR-Computer und cm-CP/M-Computer darunter leiden!

Momentan laufen einige interessante Neuentwicklungen für das NDR-System. Rolf-Dieter Klein beschäftigt sich, wie schon in *LOOP* 14 angekündigt, intensiv mit dem Window-System; dabei werden vielleicht auch neue Baugruppen anfallen. An der Fachhochschule Kempten laufen einige Diplomarbeiten, die neue Baugruppen hervorbringen werden. Sobald wir Näheres wissen, werden wir Sie informieren.

Und nun die abschließende Bitte: Senden Sie uns *LOOP*-Artikel – über alles, was Ihnen Spaß macht und die *LOOP*-Leser interessiert!

Die Anschrift unseres Vertriebspartners in Berlin hat sich wie folgt geändert:
Firma
Korb Elektronik
Ahrweiler Straße 1 a
1000 Berlin 33
Telefon (030) 8211947

Aufruf an alle SOFTWARE-FREAKS!

Mit der neuen JADOS-TOOL-Diskette hat sich bestätigt, daß die Nachfrage nach TOOL-Disketten im Bereich der Software für den NDR-Computer groß ist.

Wir waren in der Lage, bei der JADOS-Disk aus einer großen Anzahl von eingesandten Programmen eine umfangreiche Diskette aus ca. 70 Beiträgen zusammenzustellen.

Da uns schon jetzt viele weitere Programmvorschläge vorliegen, sind wir entschlossen, weitere TOOL- und Spieldisketten zu veröffentlichen.

Momentan stehen folgende Produkte an:
- RLBasic-TOOL-Diskette
- Z80-Spiele-Diskette
- Z80Turbo-Pascal-Discette.

Bevor wir aber mit unserer Arbeit beginnen können, möchten wir Sie alle aufrufen, sich an der Zusammenstellung der NDR-Programme zu beteiligen.

Plündern Sie Ihre Schubladen . . .!

Es wäre wirklich schade um Ihre guten Ideen und Erfahrungen. Haben Sie keine Angst, daß Ihr Programm von uns nicht als ein solches angesehen wird. Unser Grundsatz ist: Jede Routine, und wenn sie nur eine Zeile umfaßt, wird von uns als Programm behandelt.

Nur wenn alle mitmachen, werden auch alle davon profitieren!

Bitte senden Sie Ihre Programme in folgender Form:
- Wenn möglich auf Diskette (Eprom),
- mit ausführlicher Beschreibung (Liesmich),
- nach Möglichkeit mit Listing.

Was wir Ihnen in jedem Fall versprechen können ist, daß Sie von uns in jedem Fall sofort eine Antwort bekommen und selbstverständlich einen leeren Datenträger zum Weitermachen erhalten.

Schreiben Sie uns auch dann, wenn Sie derzeit noch an einem Programm arbeiten oder demnächst damit beginnen möchten. Wir können dann durch gezielte Koordination evtl. Doppelentwicklungen umgehen und Ihnen mit unserer Erfahrung zur Seite stehen.

Wir freuen uns über jeden neuen Software-Partner und bedanken uns schon jetzt für Ihre geleisteten Aktivitäten.

Was „denkt" ein SCHACH-PROGRAMM

von Klaus Rumrich
Spessartstraße 3
6457 Maintal 2

Seit geraumer Zeit gibt es ein Schachprogramm für den NDR-Computer. Hier soll einmal beschrieben werden, wie Computer Schach spielen und was sie dabei „denken".

Anfang der fünfziger Jahre gab es die ersten Überlegungen von Shannon und Turing zu Schachprogrammen. Shannon schlug drei Typten von Programmen vor, die heute noch zur Einteilung von Schachprogrammen verwendet werden. Die Typen A und B basieren auf dem Minimax-Algorithmus, den wir gleich vorstellen wollen, während Tpy C ein zielorientiertes Programm sein sollte, das sich auf irgendeine Art und Weise Pläne ausdenkt. Inzwischen gibt es bereits Programme vom Typ C, die teilweise eine erstaunliche taktische Spielstärke haben (siehe z.B. [1]).

Der Minimax-Algorithmus

Wesentlich einfacher zu verstehen sind Programme vom Typ A oder B. Nehmen wir an, daß Programm soll sich in einer gegebenen Stellung für einen Zug entscheiden. Ein einfacher Algorithmus wäre, das Programm mit einem Zuggenerator jeden in dieser Stellung möglichen Zug erzeugen zu lassen und dann probeweise „in Gedanken" jeden dieser Züge einmal auf einem, für den Gegner unsichtbaren Schachbrett zu ziehen. Die daraus resultierenden Stellungen würden dann mit einer Stellungsbewertungsfunktion bewertet, deren Zahlenwert um so höher ist, je günstiger die Position des Rechners ist. Selbstverständlich sollte sich dann das Programm für den Zug entscheiden, dem der größte Wert der Bewertungsfunktion zugeordnet ist.

Das Problem dabei ist, eine geeignete Stellungsbewertungsfunktion zu finden, mit der so ein Programm gut spielen kann. Glücklicherweise zeigt sich, daß man auch mit vergleichsweise simplen Bewertungsfunktionen auskommt (darüber später mehr), wenn man den Algorithmus geringfügig abändert. Man läßt dabei das Programm nach jedem seiner Probezüge auch jeden möglichen Gegenzug ausführen und bewertet erst diese Stellungen. (Alle diese Operationen spielen sich nur innerhalb des Speichers ab und sind normalerweise für den Gegner unsichtbar.)

Diese Rechnungen lassen sich sehr anschaulich als Rechenbaum grafisch darstellen; ein Beispiel, in dem jeder Spieler nur drei Zugmöglichkeiten hat, zeigt Bild 1. In dem mit C bezeichneten Knoten ist der Computer am Zug, in den mit G bezeichneten Knoten der Gegner. Ein Probezug wird durch eine Kante von einem Knoten zu einem anderen Knoten dargestellt. Nach je einem Zug des Computers und des Gegners sind Endpositionen erreicht, deren Stellungsbewertung wir in Bild 1 angegeben haben. Für welchen Zug sollte sich das Programm in unserem Beispiel entscheiden?

Bei Zug 3 könnte nach dem Gegenzug 32 die Endposition mit der höchsten Stellungsbewertung von +8 erreicht werden. Aber dann hat man die Rechnung ohne den Gegner gemacht. Da eine Stellung, die für den Rechner günstig ist, logischerweise für den Gegner ungünstig ist, muß die Bewertungsfunktion für die gegnerischen Züge gewissermaßen umgekehrt interpretiert werden. Während also der Computer einen möglichst großen Wert – das Maximum – anstrebt, wird vom Gegner erwartet, daß er einen möglichst kleinen Wert zu erreichen sucht – das Minimum. Das Programm rechnet also stets mit einer optimalen Spielweise des Gegners.

Nach Zug 3 des Computers nehmen wir also an, daß der Gegner sich für die ihn günstigste Erwiderung entscheiden wird, also für Zug 33, der zur Stellungsbewertung –4 führt. Wir sehen also, daß in den G-Knoten angenommen wird, daß der Gegner sich für den Zug mit der niedrigsten Stellungsbewertung entscheidet. Diesen, vom Gegner erreichbaren Minimalwert schreiben wir uns nun jeweils an die G-Knoten (Bild 2).

Das bedeutet nun, daß der Gegner nach Zug 1 des Computers bei optimalem Spiel eine Stellung mit dem Wert +3 erreichen kann, nach Zug 2 des Computers den Wert +1 usw. Der Computer sollte jetzt natürlich von diesen Werten an den G-Knoten den größten auswählen, in unserem Beispiel also +3 am Knoten G-1. Der beste Zug des Computers, unter Berücksichtigung eines optimalen Gegenspielers, ist also Zug 1; er erreicht damit mindestens eine Stellung mit der Bewertung +3.

Dieses Verfahren heißt Minimax-Verfahren, weil in den verschiedenen Rechentiefen abwechselnd die Minimal- und Maximalwerte der Stellungsbewertungen ausgewählt werden.

Das Minimax-Verfahren kann natürlich auf größere Rechentiefen als zwei Züge ausgedehnt werden. Startend bei dem C-Knoten ganz links (der Wurzel des Baumes), müssen bei Durchlaufen des Baumes entlang der Kanten abwechselnd G- und C-Knoten folgen. Nach der gewünschten Rechentiefe folgen die Knoten mit den Endstellungen (die Blätter des Baumes). An diese Blätter werden die Stellungsbewertungen der Endstellungen geschrieben und dann, von den Blättern wieder rückwärts in Richtung auf die Wurzel laufend, an jeden Knoten die Minimal- oder Maximalwerte seiner Nachfolger (bei C-Knoten die Maximalwerte und bei G-Knoten die Minimalwerte). Hat man die auf die Wurzel folgenden G-Knoten bewertet, so entscheidet man sich für die Kante, die von der Wurzel ausgehend den größten Wert erreicht. Der dieser Kante entsprechende Zug ist (innerhalb der verwendeten Rechentiefe) der beste Zug.

Exponentielle Rechenzeit

Warum rechnet man nicht einfach mit dem Minimax-Verfahren bis an das Partieende? Dann käme man mit einer sehr einfachen Bewertungsfunktion aus, die nur entscheiden müßte, wer mattgesetzt worden ist. So ein Programm würde perfekt Schach spielen.

Die Anzahl der zu bewertenden Stellungen läßt sich grob abschätzen: Eine mittlere Schachpartie dauert 40 Züge (in Partien zählt je ein Zug von Weiß und von Schwarz zusammen als ein Zug, deshalb wird bei der Rechentiefe von Schachprogrammen oft von Halbzügen gesprochen), das sind 80 Halbzüge. In einer durchschnittlichen Stellung sind etwa 30 bis 40 verschiedene Züge möglich. Der Rechenbaum verzweigt sich also in jeder Stufe in etwa 30 Äste, jeder dieser Äste in 30 Unteräste usw. Dies wiederholt sich insgesamt 80 mal, woraus sich etwa 30^{80} = ca. 10^{120} Endstellungen an den Blättern des Baumes ergeben. Das entspricht einer 1 mit 120 Nullen! Kein noch so großer Rechner kann auch nur einen Bruchteil dieser gigantischen Zahl von Berechnungen durchführen. Da der Vorschlag, die Partie bis zu ihrem Ende vorauszuberechnen also nicht durchführbar ist, muß bereits bei einer kleineren Rechentiefe abgebrochen werden, wobei man dann allerdings nicht mehr mit einer so einfachen Bewertungsfunktion auskommt.

Allgemein gilt: Bei einem Verzweigungsfaktor v (in der Praxis etwa 30) und einer Rechentiefe d hat man v^d-Endstellungen zu bewerten. Schon bei einer Rechentiefe von nur vier Halbzügen sind das ungefähr 1.000.000 Stellungen. Mit jedem weiteren Halbzug verdreißigfacht sich diese Zahl.

Die Bewertungsfunktion

Bisher haben wir immer von der Bewertungsfunktion für die Endstellungen gesprochen, aber nicht gesagt, wie diese genau aussehen. Im günstigsten Fall wäre diese Funktion positiv für gewonnene Stellungen, 0 für unentschiedene Stellungen und negativ für verlorene Stellungen. Damit sich das Prgramm aber nicht nur mit gewonnenen Stellungen begnügt, sondern schließlich auch mattsetzt, sollte die Funktion auch angeben, in wieviel Zügen gegebenenfalls gewonnen werden kann und um so größere Werte ausgeben, je näher man dem Matt ist.

Tatsächlich würde mit einer solchen Bewertungsfunktion eine Rechentiefe von nur einem Zug ausreichen, um ein perfekt spielendes Schachprogramm zu erhalten.

Leider ist so eine Funktion, die für jede Stellung auf dem Schachbrett angibt, wer in wieviel Zügen gewinnen kann, nicht bekannt. Deshalb müssen hier Erfahrungswerte eingesetzt werden. Jeder weiß, daß ein Damenverlust in einer Schachpartie praktisch schon die Partie entscheidet. Es ist also bestimmt sinnvoll, den materiellen Vor- oder Nachteil in die Bewertungsfunktion aufzunehmen. Gängige Werte sind:

Bauer = 100, Springer oder Läufer = 300 bis 350, Turm = 400 bis 500, Dame = 800 bis 900, König = 10000 Einheiten.

Dabei kann es sinnvoll sein, zum Beispiel im Endspiel die Figurenwerte etwas zu verändern, also etwa den Wert eines Bauern anzuheben. Mit einer solchen, rein materiellen Bewertungsfunktion, spielt ein Schachprogramm noch miserabel. Eine Verbesserung wird durch Bewertung von Position und Beweglichkeit der Figuren sowie der Königssicherheit erreicht. Hierfür konkrete Werte anzugeben ist sehr schwierig, es helfen nur Experimente mit verschiedenen Werten und ein Mindestmaß an Schachwissen weiter. Die Positionsbewertung könnte folgende Situationen erkennen und bewerten: Besetzung der Zentrumsfelder, Besetzung offener Linien mit Türmen, Doppelbauern, isolierte Bauern, Freibauern, vorgerücke Bauern . . .

Shannon-A- und Shannon-B-Programme

Wir sprachen bereits von der Einteilung der Programme nach Shannon. Das oben beschriebene Minimax-Verfahren, bei dem bis zu einer vorgegebenen Rechentiefe alle möglichen Züge überprüft werden, ist vom Typ A. Wenn man von vornherein nur sinnvolle Züge zulassen würde, würde der Verzweigungsfaktor v wesentlich kleiner werden und damit auch die Rechenzeit. Programme, die so selektiv arbeiten, sind Shannon-B Programme.

Das Problem der Shannon-B Programme ist natürlich, die sinnlosen Züge auszusortieren. Man kann das so machen, daß in jeder Stellung zunächst alle möglichen Züge in einer Liste gesammelt werden und dann jedem Zug eine „Plausibilität" zugeordnet wird. Nur die plausibelsten Züge werden dann auch wirklich ausprobiert. Ähnlich wie bei der Stellungsbewertung hat man hier das Problem, eine gute Plausibilitätsfunktion zu finden. Beispielsweise sollte Schlagzügen, Schachgeboten und Angriffen auf ungedeckte gegnerische Figuren eine hohe Plausibilität zugeordnet werden. Dadurch, daß bei einem Shannon-B Programm nicht alle Züge durchprobiert werden, kann es allerdings passieren, daß gute Züge übersehen werden. Mit einer guten Plausibilitätsfunktion kann dieses Risiko kleingehalten, aber nicht ausgeschlossen werden.

Beschneiden des Rechenbaumes: Das Alpha-Beta-Verfahren

Es gibt ein einfaches Verfahren, welches erlaubt, einige Äste des Rechenbaumes aus der Betrachtung auszuschließen, was eine kleinere Rechenzeit ergibt, dabei aber das Endergebnis nicht ändert.

Um die Funktion zu verstehen, betrachten wir Bild 3. Es ist der bereits bekannte Beispielrechenbaum, allerdings fehlen einige Stellungsbewertungen. Zunächst verfolgen wir noch einmal den Minimax-Algorithmus in diesem Baum. Beginnend mit der Ausgangsstellung in C wird zunächst der Zug 1 ausprobiert und daran anschließend die drei Gegenzüge. Bei G-1 wird das Minimum der Stellungsbewertungen, also +3, eingetragen. Zu diesem Zeitpunkt ist also bereits bekannt, daß der Rechner eine Stellungsbewertung von +3 erreichen kann, damit er Zug 1 zieht. Trotzdem müssen auch die anderen Äste betrachtet werden, weil vielleicht noch ein größerer Wert erreichbar ist.

Als nächstes wird Zug 2 versucht, dann Zug 21 von G-2 aus, der zu einer Stellungsbewertung von +5 führt. Der nächste Gegenzug – Zug 22 – führt zur Stellungsbewertung +1. Jetzt kann aber schon darauf verzichtet werden, auch Zug 23 zu versuchen, weil wir jetzt schon wissen, daß der Gegner am Knoten G-2 bei optimalem Spiel höchstens die Bewertung +1 zuläßt, der Computer am Knoten C aber bereits eine Möglichkeit kennt, eine Bewertung von +3 zu erhalten. Gleichgültig, welchen Wert wir an das Blatt nach Zug 23 schreiben, mit Zug 2 von C aus kommen wir mit Sicherheit nicht über einen Wert von +1 hinaus, weil angenommen wird, daß der Gegner den

Wert zu minimieren sucht. Wenn die Stellungsbewertung an diesem Punkt aber ohnehin keinen Einfluß auf die Berechnung des besten Zuges hat, braucht der Knoten gar nicht aufgesucht zu werden. (Man weiß dann allerdings nicht genau, wie schlecht Zug 2 ist, was aber gewöhnlich auch uninteressant ist.)

Man macht sich leicht klar, daß bei Betrachtung des Zuges 3 von C aus, bereits Zug 31 von G-3 aus eine Widerlegung darstellt, da nun bereits feststeht, daß an G-3 höchstens der Wert −2 eingetragen wird und von C aus mit Zug 1 bereits ein besserer Zug bekannt ist. Von G-3 aus können also sogar zwei Züge eingespart werden.

In unserem Beispiel haben wir also drei von neun Endstellungen eingespart. Das hört sich zwar noch nicht wie eine gewaltige Einsparung an, aber bei einer größeren Rechentiefe würde anstatt eines Blattes noch ein kompletter Teilbaum folgen, der nicht mehr betrachtet zu werden braucht.

Wendet man das Verfahren in jeder Tiefe des Baumes erneut an, so werden insgesamt oft beträchtliche Teile des Baumes abgeschnitten und die Rechenzeit wird drastisch reduziert.

Hätten wir in unserem Beispiel den Baum statt von oben nach unten, von unten nach oben durchlaufen, so stellt man fest, daß nun der gesamte Baum durchlaufen werden muß und nichts eingespart werden kann, die Zahl der zu bewertenden Endstellungen also immer noch v^d beträgt. Wir sehen also, daß es beim Alpha-Beta-Verfahren auf die Reihenfolge ankommt, in der die möglichen Züge probiert und bewertet werden.

Der günstigste Fall tritt ein, wenn per Zufall immer zuerst der für die jeweilige Partei beste Zug ausprobiert wird. Bei optimaler Ausnutzung des Verfahrens erhält man dann nur noch etwa $2v^{d/2}$ Endstellungen, also eine enorme Einsparung gegenüber v^d ohne Verwendung von Alpha-Beta. Im Mittel wird man zwischen beiden Extremen liegen.

Das Alpha-Beta-Verfahren läßt sich sowohl bei Shannon-A als auch bei Shannon-B Programmen anwenden; man erhält bei der Anwendung immer den gleichen Zug wie ohne Anwendung von Alpha-Beta.

Einige Feinheiten

Um die Spielstärke von Schachprogrammen noch zu erhöhen, werden noch einige weitere Tricks benötigt. So ist es beispielsweise schlecht, grunsätzlich eine feste Rechentiefe zu verwenden. Bricht man nämlich die Rechnung mitten in einem Schlagabtausch ab, so können fehlerhafte Beurteilungen die Folge sein. Deshalb ist es besser, in solchen Fällen (ebenso z.B. bei Schachgeboten oder möglichen Bauernwandlungen) noch ein Stück weiter zu rechnen, bis eine stabile Stellung entstanden ist, in der alle Schlagabtäusche abgeschlossen sind.

Ein weiterer Punkt ist die Reihenfolge der Zugversuche. Da Schlagzüge oft Widerlegungen für schlechte Züge sind, ist es mit Blick auf das Alpha-Beta Verfahren günstig, zuerst Schlagzüge zu versuchen und erst danach auch andere Züge. Ähnlich wie bei Shannon-B Programmen kann es sogar sinnvoll sein, die Züge erst nach ihrer Plausibilität zu ordnen und die plausibelsten Züge zuerst zu verwenden. Eine andere Möglichkeit, das Alpha-Beta Verfahren mit möglichst guten Zügen zu beginnen, ist iterativ zu rechnen, d.h. man beginnt mit einer niedrigen Rechentiefe und verwendet die dabei gefundenen Züge als Anfangszüge in der höheren Rechentiefe, mit der Hoffnung, daß diese Züge nicht mehr ganz so schlecht wie zufällig gewählte Züge sind.

Mit der iterativen Berechnung läßt sich auch gut die selektive Suche in Shannon-B Programmen verbinden, da Züge, die sich bei der niedrigeren Rechentiefe als sehr schlecht erwiesen, später völlig weggelassen werden können. Hierbei muß man wieder etwas vorsichtig sein, weil sonst z.B. Opferzüge, die sich erst später als gut erweisen, ausgeschlossen werden.

SCHACH 2.9

Das für den NDR-Computer erhältliche Programm SCHACH 2.9 ist ein Shannon-A Programm mit einstellbarer Rechentiefe, wobei bei Erreichen der maximalen Tiefe Schachzüge und Schachs noch ein Stück weiterverfolgt werden können, um eine stabile Stellung zu erreichen. Es wird das Alpha-Beta Verfahren in Verbindung mit der iterativen Berechnung angewandt; Schlagzüge werden jeweils zuerst berücksichtigt um möglichst häufig Widerlegungen für die Anwendung von Alpha-Beta zu finden.

Die verwendeten Figurenwerte sind:
B = 100, S = 280, L = 300, T = 440, D = 800; im Endspiel erhalten die Bauern den Wert 120.

Es wird positiv bewertet:
Besetzung der Zentrumsfelder durch Bauern, Besetzung offener Linien durch Türme, Bauernvormarsch, im Endspiel der gegnerische König am Brettrand.

Es wird negativ bewertet:
Springer am Rand (bringt Kummer und Schand', alte Schachspielerweisheit), Entfernen der eigenen Bauern am Königsflügel, Verstellen der eigenen Zentrumsbauern durch eigene Figuren.

Literatur:

(1) Pitrat, J.: A Chess Combination Program Which Uses Plans Artificial Intelligence 8 (1977) 275 (Experimentalprogramm, Shannon-C)

(2) Newborn, M.: Computer Chess Academic Press, New York (1975) (OSTRICH, 6. Platz Computerschach-WM 1974, Shannon-B)

(3) Gillogly, J. J.: The Technology Chess Programm Artificial Intelligence 3 (1972) 145 (Shannon-A)

(4) Berliner, H. J.: A Chronology of Computer Chess and its Literature Artificial Intelligence 10 (1978) 201

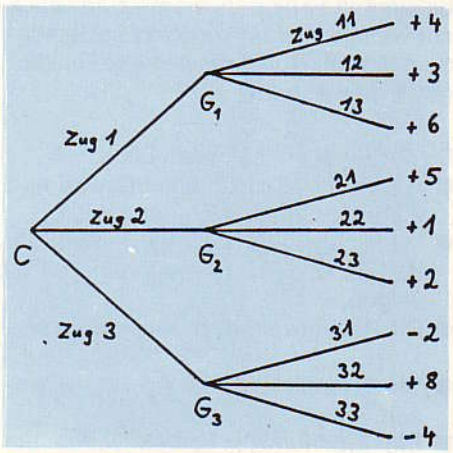


Bild 1: Ein Beispielrechenbaum der Tiefe 2. Die Stellungsbewertungen der Endpositionen sind an den Blättern angegeben.

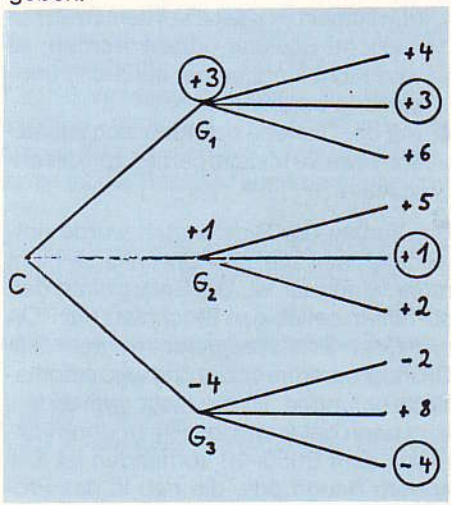


Bild 2: Das Programm rechnet mit optimalem Gegenspiel. Deshalb wird an den G-Knoten das Minimum der folgenden Knoten eingetragen. Die für den jeweiligen Spieler optimalen Werte sind markiert.

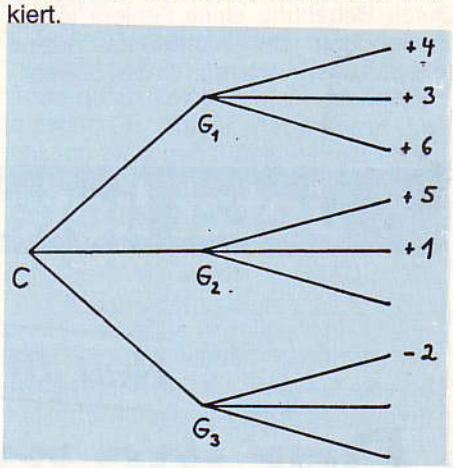


Bild 3: Mit Hilfe des Alpha-Beta-Verfahrens brauchen Teile des Baumes nicht betrachtet zu werden.

SCHACH 2.9 – eine neue Version

Nach einem halben Jahr wurde nun eine neue Version fällig, in der ich nicht nur alle bekannten Fehler (es waren zum Glück nur wenige leichte) korrigiert habe, sondern die ich auch um viele neue Funktionen erweitert habe.

Die Änderungen auf einen Blick:

- Außer der alten Uhrenkarte wird nun auch die Smartwatch unterstützt.
- Die Soundkarte wird verwendet, um das Ende der Zugberechnung anzuzeigen.
- Die Rechenzeit wurde wesentlich verkürzt.
- Die Endspielbehandlung ist verbessert.
- Die Eröffnungsbibliothek wurde um weitere Varianten erweitert.
- Während der Partie können die Seiten gewechselt werden.
- Bei Aufruf des Stellungseditors kann nun immer die zuletzt auf dem Brett befindliche Stellung editiert werden; es wird nicht mehr immer mit der Grundstellung begonnen.
- Auf der Diskette befinden sich zusätzlich etwa 20 Meisterpartien zum Nachspielen.

Der Einbau der Smartwatch wurde notwendig, weil die alte Uhrenkarte nicht mehr verfügbar ist. Die Smartwatch darf auf einem beliebigen Steckplatz der ROA mit dem Grundprogramm hinter dem Grundprogramm sitzen und wird automatisch gefunden. Ist sie nicht vorhanden, wird noch getestet, ob die alte Uhrenkarte mit dem E050–16 vorhanden ist. Die andere Baugruppe, die neu in das Programm aufgenommen wurde, ist SOUND. Damit werden nun fehlerhafte Eingaben, das Ende einer Zugberechnung, Schachgebote und Matt akustisch angezeigt.

Durch Behebung eines Fehlers in der Verwendung des Alpha-Beta-Prinzips (die gängigen Verfahren in der Schachprogrammierung werden in einem späteren *LOOP*-Artikel dargestellt) konnte die Rechenzeit um bis zu 50% gegenüber Version 2.7 reduziert werden.

Drei neue Kommandos sind hinzugekommen: der Glockenton über die Soundkarte kann an- und abgeschaltet werden, während der Partie können die Seiten gewechselt werden und es kann bis an den Anfang der Partie zurückgegangen werden, um sie noch einmal nachzuspielen.

Eine häufige Kritik am Stellungseditor war, daß keine auf dem Brett gegebene Stellung editiert werden konnte. Ab Version 2.9 wird immer die Stellung, die sich zuletzt auf dem Brett befand, editiert. Das erleichtert das Verändern von vorhandenen Stellungen.

Als Besonderheit habe ich noch zwanzig Musterspiele von verschiedenen Großmeistern auf der Diskette untergebracht. Sie können genau wie selbst abgespeicherte Partien geladen und nachgespielt werden; besonders zum Lernen und Üben ist das zu empfehlen.

Ein weiterer Punkt ist die eingeschränkte Rechentiefe der alten Version 2.7. Dort konnte man nur bei Spielstufen unter 6.1 sicher sein, korrekte Ergebnisse zu erhalten. Jetzt habe ich diese Grenze auf 9.4 erhöht, was in der Praxis keine Einschränkung mehr bedeutet, da hier die Rechenzeit selbst in einfachen Stellungen schon enorm ist. Immerhin können in der Spielstufe 8.2 noch Schachprobleme mit Matt in fünf Zügen gelöst werden. Es wäre zwar leicht möglich, diese Grenze beliebig hoch zu treiben, das wäre aber Augenwischerei, da die Rechenzeit exponentiell mit der Rechentiefe anwächst und niemand jahrelang auf die Lösung eines Matts in 15 Zügen wartet, obwohl das Programm im Prinzip in einer Stufe 28.2 auch solche Probleme noch lösen könnte.

Neue Grundprogrammversionen geplant

Sehr geehrte Leser,
es ist wohl an der Zeit, daß das 68008-Grundprogramm und später auch das 68000- und 68020-Grundprogramm erweitert und verbessert wird. Ich hatte dies schon seit längerer Zeit vor und setzte nun mein Vorhaben in die Tat um.

Damit ich nun nicht an Ihren Wünschen vorbei arbeite, rufe ich hiermit alle *LOOP*-Leser auf, mir zu schreiben, was Sie am jetzigen Grundprogramm stört und was vielleicht noch eingebaut werden sollte.

Schicken Sie mir Ihre Vorschläge bitte direkt und nicht über GES zu. Außerdem bitte ich um Verständnis, daß ich Ihre Briefe nicht beantworten kann, da ich sonst nicht zum Programmieren komme. Es wird aber jeder Vorschlag sorgfältig von mir geprüft werden.

Nun eine kurze Aufstellung, was ich geplant habe:

1. Alle Routinen verbessern, soweit es sinnvoll und machbar ist,
2. Neue Routinen für HARDCOPY-, GDP-, COL-Karte,
3. Verbesserung der Menüsteuerung und der Menüpunkte (z.B. ANSEHEN),
4. Einbau eines besseren Editors,
5. Einbau neuer Unterprogramme,
6. Verbesserung der Druckeransteuerung (für Epson-Steuerzeichen).

Außerdem wird die neue Version wahrscheinlich auch als Quelltext auf einer JADOS-Diskette erhältlich sein, so daß jeder selber Verbesserungen oder Änderungen durchführen kann.

Wünsche, die darüber hinausgehen, werde ich natürlich auch berücksichtigen.

Ich bedanke mich schon einmal im Voraus für alle mir zugeschickten Vorschläge.

Ralph Dombrowski
Gr. Deichstraße 33, Postfach 1130,
2208 Glückstadt, Tel.: (04124) 2520
(Nach 18 Uhr für ausführliche Auskünfte)

Jetzt lieferbar - Jetzt lieferbar

Neu im Programm

Der mc modular-AT

Schon bei der in *LOOP* 13 vorgestellten Kopplung des NDR-Computers mit dem IBM setzten wir eine aktive CPU-Baugruppe ein.

Dieses Konzept haben wir in Zusammenarbeit mit der Zeitschrift „mc" weiterentwickelt – so entstand der „mc-modular-AT".

Zunächst: Es handelt sich um einen absolut kompatiblen, sehr schnellen Rechner der AT-Klasse.

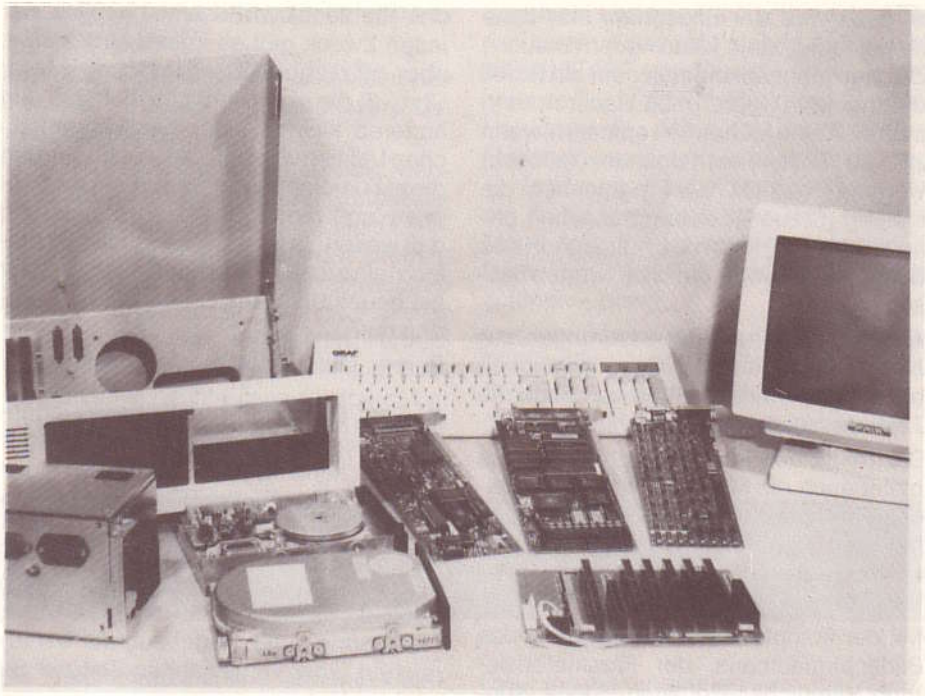
Die Zentraleinheit befindet sich dabei nicht auf einem Motherboard, sondern sie ist als ganz „normale", AT-Abmessungen entsprechende Steckkarte ausgeführt, die in einem nunmehr rein passiven Bus steckt.

Der kleine Nachteil (ein Bus-Platz weniger) wird durch den Vorteil eines modularen Systemaufbaues mehr als wettgemacht. So ist es später ohne Probleme möglich, z. B. eine andere CPU-Baugruppe mit der CPU 80386 einzusetzen.

Die Geschwindigkeit des mc-modular-AT ist an der oberen Grenze für kompatible Systeme angesiedelt; sie liegt mit einem NORTON-Faktor (der Norton-Faktor gibt an, um wieviel Mal das Gerät schneller als ein „normaler" PC mit 4.77 MHz ist) von 9,8 (ein Wait) bzw. 11.2 (no Wait) extrem gut. Die CPU taktet mit 10 MHz; bei Einsatz von schnellen (100 ns) PAMs kann durch Austausch des Quarzoszillators 12 MHz erzielt werden. Die gesamte CPU ist mit den modernen hochintegrierten Chips von Chips Technologies ausgestattet.

Der mc-modular-AT kann mit „normalen" 20 MB Festplatten oder auch mit schnelleren, 40 MB oder 80 MB Platten ausgerüstet werden.

In der „mc", ab Heft 9, erscheint eine Artikelserie über das System; wir werden von nun an auch in der *LOOP* eine Abteilung



„AT" einrichten und dort über Hard- und Software berichten. Über Zuschriften freuen wir uns!

Zur Systems zeigen wir, wie kompatibel das System ist; Tests können gerne durchgeführt werden. Neu: Bereits hier zeigen wir das neue Microsoft-Betriebssystem OS/2!

Am Festplatten-Floppy-Controller des Systems können zwei Festplatten und bis zu drei Floppy-Laufwerke angeschlossen werden; hierbei können Floppy-Laufwerke von 5¼" 360K, 5¼" 1.2 MB (AT-Standard), 3½" 1.2 MB und 3½" 720 KB Verwendung finden. Die letztgenannten entsprechen der Norm der neuen IBM Personal System/2-Familie; damit ist der Rechner auch dazu kompatibel.

Als Zubehör sind EGA- und VEGA-Baugruppen, Schnittstellen etc. erhältlich.

Window-Verwaltung unter CP/M 2.2

von Rüdiger Nahm

Software-Erweiterung für TERM1 und NDR-Z80 Computer bringt CP/M das Fensterln bei und bindet die Maus ein.

Endlich ist es soweit:
Nach vielen, auf sich allein gestellten Erweiterungsversuchen (schnelles Scrollen, Hardcopy, Maus, Uhr . . .), wurde jetzt mit dem vielen Kleinkram aufgeräumt und alles in ein neues Programmpaket integriert, das für CP/M völlig neue Möglichkeiten eröffnet, wie es sie sonst nur für MS-DOS gibt. Hier zunächst nur stichwortartig einige der integrierten Fähigkeiten:

- schnelleres Scrollen (bis 300 % bei TERM1, 200 % bei NDR)
- inverse Zeichen oder alternativ Unterstreichung von Zeichen
- beliebig viele Text-Fenster werden unterstützt
- Statuszeile (bleibt auch beim Löschen des Schirms erhalten!)
- Epson-Druckergrafik Emulation (u. a. BIT-IMAGE Mode -(ESC) K')
- automatisches Dunkelschalten des Bildschirms bei längerer Pause (ohne Hardwareveränderung)
- heller Hintergrund mit dunkler Schrift möglich
- zusätzliche TVI 912/925 Funktion
- Piezo-Summer anschließbar (TERM1)
- Echtzeituhr mit Alarmfunktion wird unterstützt
(TERM1: Software oder SMART-WATCH; NDR: SMART-WATCH)
- Fensterfunktion durch Maus auszulösen (NDR)
- PULLUP-Menüoberfläche für beliebige (Text-)Programme (WS, TURBO . . .) mit einfachsten Mitteln generierbar; Menüauswahl durch Maus (NDR)
- leicht erweiterbar (modularer Aufbau)
- ca. 300 KByte Quellcode, reichlich kommentiert
- Software-Lösung; keine Erweiterungen nötig

Ein schon lange bekanntes Problem war die langsame Bildschirmausgabe. Durch eine minimale Hardwareänderung, verbunden mit einer kompletten Neuorganisation des Bildschirmaufbaus, gelang es, die Ausgabegeschwindigkeit erheblich zu steigern.

- Bisher wurden immer zwei Seiten zugleich beschrieben und eine Seite wurde durch Zeichen mehrerer Blöcke gelöscht. Der Cursor wurde durch Umschalten zwischen 2 Seiten realisiert.

✶ Jetzt wird immer nur eine Seite aktualisiert. Das Löschen des Bildschirms erfolgt durch einen speziellen Befehl des GDP. Die RAM's werden im READ/MODIFY/WRITE gelöscht, daher kann eine Seite während eines Anzeigezyklus gleichzeitig gelöscht werden. Außerhalb des Anzeigefensters kann dann auf eine bereits fertig aufgebaute neue Seite umgeschaltet werden. Der Cursor wird direkt durch schreiben/löschen realisiert.

- Bisher wurde bei jedem Steuerzeichen sofort der Bildschirm verändert.

* Jetzt wird der Bildschirm erst dann aufgefrischt, wenn keine neuen (Steuer-) Zeichen mehr ankommen oder ein Scroll durchgeführt werden muß. Dadurch kann man sich eine Mehrarbeit ersparen, wenn z.B. 10 Zeilen nacheinander gelöscht werden. (Nachteil: Bei Programmen, die ständig (Steuer-)Zeichen ausgeben ohne zu scrollen, kann das Auffrischen des Bildschirms lange auf sich warten lassen.)

Alle diese kleinen Verbesserungen zusammen steigern die Geschwindigkeit bis auf das Dreifache, und das ohne die Taktrate zu erhöhen.

Die Taktik des kompletten Auffrischens bietet auch die Möglichkeit, inverse Zeichen darzustellen. Zuerst wird ein Block hell gezeichnet, auf dem dann durch Löschen der Punkte der Text geschrieben wird.

Mit der kompletten Umorganisation des Bildschirmaufbaus, der Parameterisierung sämtlicher Routinen (keine Konstanten mehr in den Unterprogrammen, z. B. ADD A,24) einher gingen noch einige weitere Verbesserungen.

- Es gibt jetzt die Möglichkeit, eine Statuszeile am unteren Bildschirmrand einzublenden, die auch beim Löschen des Bildschirms erhalten bleibt, und die vom Textmodus aus einfach geladen werden kann.
- Für Leute, die lieber dunkle Schrift auf hellem Hintergrund mögen, wurde auch diese Darstellungsform ermöglicht. Die Realisation ist ein einfaches XOR mit einem Flag, bevor der GDP auf Schreiben oder Löschen umgestellt wird.
- Weitere Funktionen des TVI-Terminals, zu dem Kompatibilität bestehen soll, wurden implementiert (z. B.: sende Zeile . . .).
- Bei längeren Arbeitspausen wird zur Schonung der Bildröhre der Bildschirm dunkel geschaltet. (Geht durch ein Kommando an den GDP- HIGH SPEED WRITE). Sobald wieder ein Zeichen eingegeben wird, erscheint das Bild wieder.

Ein großer Nachteil schrittweiser, getrennter Erweiterungen (RUCKSACK) ist das Problem des Zusammenspiels. Für jeden Zweck gibt es irgend eine kleine, aber nützliche Utility (Hardcopy, Maus, Uhr . . .), die sich allerdings nicht mit den anderen kleinen, aber genauso nützlichen Utilities verträgt. Deswegen wurden diese Funktionen von vornherein mit integriert und die Software so strukturiert, daß weitere Ergänzungen möglichst einfach einzubauen sind und nicht wieder ein neues Utility ins Leben gerufen werden muß.

Bereits integrierte Funktionen sind da

- Eine Echtzeituhr, die über Zeichen-ein-/-ausgabe abgefragt und gestellt werden kann. Diese Echtzeituhr kann über die SMART-WATCH oder bei der TERM1 auch rein durch Software realisiert werden. Die Uhr läßt sich auch bei Bedarf in den Bildschirm einblenden. Als besonderes Zuckerl ist hier gleich eine Alarmeinrichtung eingebaut, die bei Erreichen einer vorgewählten Zeit ein Signal auslöst.
- Die Möglichkeit, bei Bedarf eine Hardcopy des Bildschirms über einen Drucker auszugeben.
- Die Möglichkeit, die Maus in (fertige) Programme einzubeziehen. Doch dazu später genauer.
- Die Möglichkeit, den Bildschirm wie einen Drucker anzusteuern, wenn man ein gespeichertes Bild schnell aufbauen will. Man muß nicht selbst den Cursor steuern, es reicht, einfach eine BITMAP zu übergeben. Das Apfelmännchen wird z.B. in weniger als 10 sek. aufgebaut.

Bei der Neuimplementation der Bildschirmverwaltung wurde die Fähigkeit zur fensterorientierten Darstellung mit einbezogen. Es können beliebig viele ((= Speicherplatz) Fenster verwaltet werden. Ein Fenster ist ein Bildschirmausschnitt, der unabhängig von anderen Fenstern beschrieben werden kann. Das Kommando 'lösche Schirm' bezieht sich also nur auf das gerade aktive Fenster. Somit vereinfacht sich die Programmierung von anwenderfreundlicher Software ganz erheblich.

Beispiel:
Ein bildschirmorientiertes Programm (Editor . . .) kann Fehlermeldungen oder Testausgaben in ein anderes Fenster schreiben als die normalen, erwünschten Ausgaben. Es entsteht kein Kauderwelsch von Testausgaben, Fehlermeldungen und Steuerzeichen. Nebenbei können in einem anderen Fenster Hilfsmeldungen, etwa erlaubte Kommandos stehen.

Bisher hat sich aber jedes Anwenderprogramm selber um die entsprechende Bildschirmverwaltung kümmern müssen. Als Folge davon verzichtete man meist auf einen solchen Aufwand und nahm die Nachteile eines ungeteilten Bildschirms in Kauf.

Jetzt wird das Anwenderprogramm vom schlimmsten Organisationsaufwand befreit. Es kann direkt Kommandos ausgeben wie etwa

- öffne Fenster
- schließe Fenster
- verschiebe Fenster

und vieles andere mehr.

Die nachfolgenden Bemerkungen gelten momentan nur für den NDR-Computer.

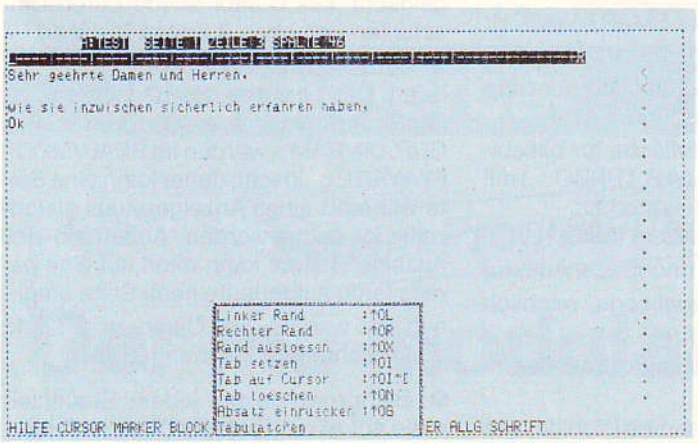
Da der Einbau in bestehende Programme große Probleme aufwirft, wurden noch weitere Möglichkeiten geschaffen, die Fenster zu benutzen.

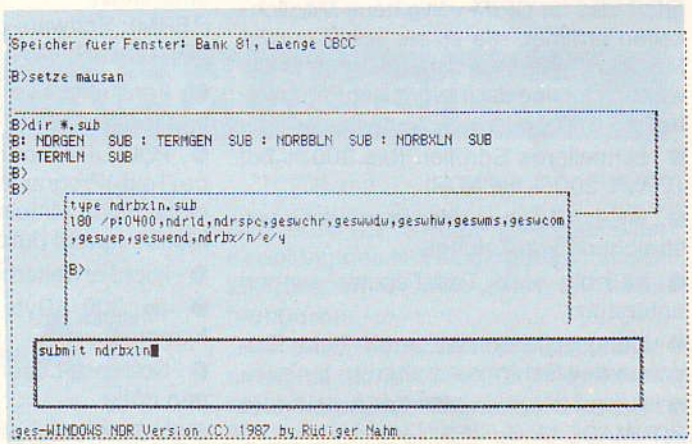
Die Maus, die bisher ein Schattendasein in vereinzelten Grafikprogrammen und Utilities fristete, wird voll in das System eingebunden. Das heißt im Klartext:

- Mit der Maus können Fenster geöffnet, geschlossen, verschoben, vergrößert, verkleinert werden, auch ohne daß das Anwenderprogramm diesen Vorgang expliziert unterstützt.

Beispiel:
DIR A:
(neues Fenster aufmachen)
PIP B:=A:PROG1
PIP B:=A:PROG2
. . .

Das Inhaltsverzeichnis von A bleibt die ganze Zeit über sichtbar und braucht nicht ausgedruckt zu werden, wenn man sich nicht alles merken kann.





● In Textverarbeitungsprogrammen wie etwa WS, TURBO . . . kann **der Cursor über die Maus gesetzt werden** – gewünschte Position anfahren – KLICK – Cursor steht – langwieriges Drücken von Cursortasten entfällt.

● Die Maus kann eine Hardcopy auslösen.

Wie oft hat ein CP/M Benutzer schon neidisch zu seinen Kollengen mit PC, ATARI, MAC oder AMIGA geblickt, nur weil es dort so hübsche PULL-DOWN bzw. PULL-UP Menüs gibt? Und nur er hatte sowas nicht!

Aber: Ab jetzt PULL-UP Menüs auf dem NDR-Computer!

Der neue Terminaltreiber beinhaltet alle Funktionen, die für ein solches Menü benötigt werden.

Wird die linke Maustaste gedrückt, während das Fadenkreuz auf der Statuszeile steht, dann wird ermittelt, wie viele Leerzeichen links vom Fadenkreuz stehen. Aus dieser Zahl wird errechnet, welches Fenster zu öffnen ist. Kann auf dieses Fenster erfolgreich zugegriffen werden, dann kann mit der Maus in diesem Fenster die gewünschte Zeile anklicken. Ein vordefinierter String wird an den Rechner geschickt und das Fenster geschlossen. Auf diese Weise läßt sich zum Beispiel WORDSTAR vollständig über Menü steuern. (Unterstreichen ein/aus . . . siehe Hardcopy 2). Mit dem mitgelieferten Menügenerator und einem beliebigen Texteditor kann auf einfachste Weise für ein beliebiges Programm ein solches Menü erstellt oder verändert werden. – Dem wäre eigentlich nichts mehr hinzuzufügen – oder?

Und am Schluß noch eine Bemerkung: Die Quellen (ca. 300 KByte) werden als SOURCE-CODE mitgeliefert!

Das Programm ist ab Lager lieferbar!

Schneller 12 Bit AD- und DA-Wandler jetzt lieferbar

Die Baugruppe AD/DA r3 mit dem 12-Bit AD-Wandler AM6112 und den zwei 12-Bit DA-Wandlern AM6012 ist ab sofort lieferbar. Vor allem der AD-Wandler zeichnet sich mit 6 usec durch eine schnelle Wandelzeit aus. Nähere Einzelheiten zu dieser Baugruppe können Sie in den nächsten Absätzen sehen.

Grundsätzliches zur Baugruppe AD/DA

Mit der Verbreitung der Mikroelektronik kommt den Schnittstellen zwischen analogen und digitalen Signalen immer größere Bedeutung zu, und die Umsetzung von analogen in digitale Meßgrößen erfolgt deshalb in einem frühen Stadium.

Daß digitalisierte Daten aus vielen Gründen vorteilhafter als analoge Signale übertragen werden können, ist seit langem bekannt. Es mangelte an Realisierungsmöglichkeiten mit vertretbarem Aufwand und Kosten. Inzwischen hat die Halbleitertechnik so gewaltige Fortschritte gemacht, daß eine schnelle 12-Bit AD- oder DA-Wandlung mit niedrigem, äußerem Beschaltungsaufwand und geringen Kosten durchgeführt werden kann, denn neuere AD- und DA-Wandler können gegenüber früheren Versionen, in Standardarbeitsgängen hergestellt werden, ohne daß danach jeder Baustein individuell abgeglichen werden muß (Laser-Trimming).

Für die 16-Kanal ADU- und 2-Kanal DAU-Karte wurde der 12-Bit AD-Wandler AM6112 und der 12-Bit DA-Wandler AM6012 verwendet, die später beschrieben werden.

Wie setzt man die AD/DA ein?

Die Schaltung der AD/DA wurde sehr einfach gehalten. Es wurde bei ihrer Entwicklung Wert darauf gelegt, daß sie mit allen Prozessortypen, für die Karten für den NDR-Computer existieren oder für die Zukunft geplant sind, arbeiten kann.

Die Schaltung der Karte ist zwar sehr einfach, sie bietet aber je nach Geschwindigkeitsanforderung und Anwendung entsprechende Möglichkeiten, den AD-Wandler zu starten und die gewandelten Werte einzulesen. Auf der analogen Seite stehen verschiedene festlegbare Eingangs- und Ausgangsspannungsbereiche zur Verfügung.

Technische Daten:

Versorgungs-spannung:	+ 5 V, + 12 V, – 12 V
Bus-Format:	NDR-Klein-Bus oder ECB-Bus
Größe der Leiterplatte:	Europaformat
Auflösung und Genauigkeit:	12-Bit Analog-Digital und 12-Bit Digital-Analog
Linearitätsfehler:	± 1/2 LBS
Analoge Einlesekanäle:	16
Analoge Ausgabekanäle:	2
Einlese-spannungen:	0 bis 10 V, – 5V bis + 5 V
Ausgabe-spannungen:	0 bis 5 V, 0 bis 10 V, – 5 V bis + 5 V, – 10 V bis + 10 V
Umsetzzeiten:	AD-Wandler max. 6.25 Mikrosekunden DA-Wandler ca. 600 Nanosekunden

Prinzipielle Funktionen der Baugruppe AD/DA

Die I/O-Grundadresse der Baugruppe wird über einen Adressvergleicher angewählt. Diese und 7 folgende Adressen sprechen den AD-Wandler, das Statusregister, das Kontrollregister und die DA-Wandler an. Über diese Adressen kann eine AD-Wandlung gestartet, die gewandelten Werte eingelesen oder Spannungen über die beiden DA-Kanäle ausgegeben werden. Die Status- und die Kontrollregisteradresse sind dem AD-Wandler zugeordnet. Mit dem Kontrollregister wird einer von 16 Einlesekanälen eingestellt und die Betriebsart des AD-Wandlers festgelegt, während der Inhalt des Statusregisters dessen Zustand anzeigt.

Die Eingangsspannung des AD-Wandlers wird während des Umsetzvorganges über einen Sample & Hold-Verstärker konstant gehalten, da deren Änderung das Umsetzergebnis verfälschen würde. Den DA-Wandlern können 12-Bit nicht gleichzeitig übergeben werden, deshalb werden die niederwertigen 8-Bit zwischengespeichert und zusammen mit den höherwertigen 4-Bit gleichzeitig übergeben. Die Übergabe der 12-Bit an die DA-Wandler erfolgt über 2 Adressen pro Kanal. Da die DA-Wandler Stromausgänge haben, sind Operationsverstärker zur Strom-Spannungswandlung nachgeschaltet worden.

Der AD-Wandler AM6112

Der AM6112 ist ein AD-Wandler, der von Mikroprozessoren direkt angesteuert werden kann. Er enthält eine Referenzspannungsquelle, einen DA-Wandler, einen Komparator, ein „Sukzessive Approximationsregister", Buffer mit Tri State-Ausgang und eine umfassende Steuerlogik für die direkte Kommunikation mit verschiedenen Mikroprozessoren. Der AM6112 kann eine 12-Bit Umsetzung in 6 Mikrosekunden vollbringen und Eingangsspannungen von 0 bis 10 V oder – 5 bis 5 V ohne externe Schaltungskomponenten verarbeiten. In Verbindung mit Mikroprozessoren sind 4 verschiedene Betriebsarten möglich, die per Software programmiert werden können.

Der AM6112 arbeitet nach dem Prinzip der sukzessiven Approximation. Bild 3–1 zeigt das Blockschaltbild des AM6112. Die Referenzspannungsquelle liefert eine genaue und stabile Spannung für den 12-Bit DAU. Der Ausgangsstrom des DAU wird mit Hilfe eines Komparators mit dem Eingangsstrom (Spannung an RIN) verglichen. Der DAU wird über ein Sukzessive Approximation Register (SAR) betrie-

ben, das durch den Ausgang des Komparators gesteuert wird.

Beim Start einer Umsetzung werden die Bits des SAR auf 1 gesetzt. Dadurch liegt der maximale Ausgangsstrom am DAU an. Die Umsetzung beginnt mit dem MSB und endet nach der Abfrage des LSB. Das jeweilige Bit wird in der Umsetzphase auf 0 gesetzt und der Ausgangsstrom des DAU mit dem Eingangsstrom verglichen. Ist der Ausgangsstrom größer als der Eingangsstrom, verbleibt das Bit im Zustand 0. Falls aber der Ausgangsstrom kleiner als der Eingangsstrom ist, wird dieses Bit auf 1 zurückgesetzt.

In einem DAU nach diesem Umsetzverfahren ist die Umsetzgeschwindigkeit abhängig von der Taktfrequenz und der Anzahl der Bits. Ein Bit wird jeweils während einer Taktperiode geprüft, und deshalb benötigt ein N-Bit DAU N Taktzyklen für eine Umsetzung. Der AM6112 benötigt 12.5 Taktzyklen. Mit einer Taktfrequenz von 2.OMHz dauert eine Umsetzung 6.25 Mikrosekunden.

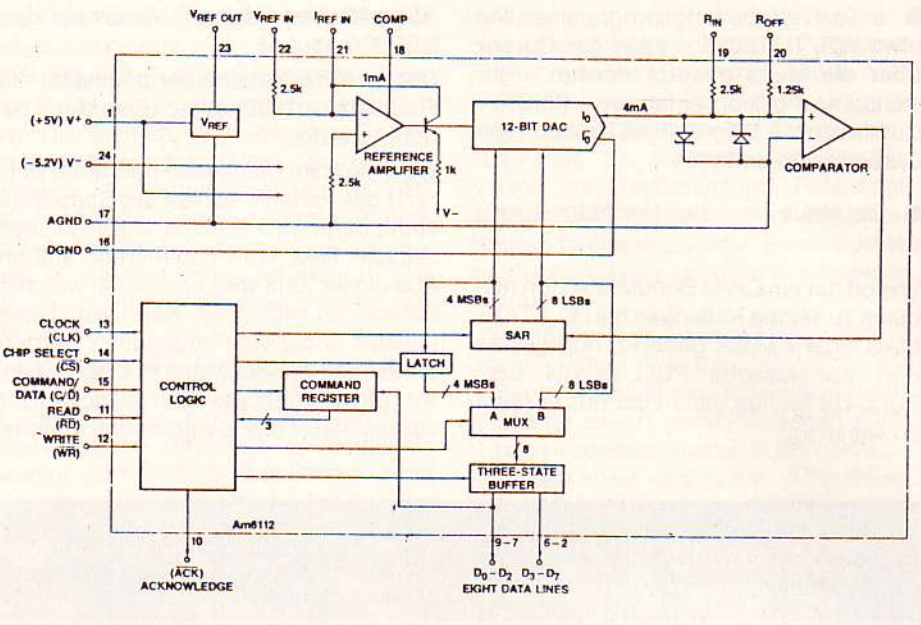


Bild 3–1: Blockschaltbild des AM6112

Der DA-Wandler AM6012

In einem DAU wird der digital gekennzeichnete Meßwert nach der Vorschrift eines Codes in eine physikalische Größe (z.B. elektrische Spannung) umgesetzt.

Bild 3–2 zeigt eine vereinfachte Darstellung eines 12-Bit R-2R DA-Wandlers. Die Referenzquelle bildet über das Bewertungsnetzwerk dual gewichtete Teilströme, die entsprechend der angelegten Dualzahl mittels Bitschalter zum Summenstrom IOUT summiert werden. Mit einem Operationsverstärker als Strom-Spannungs-Wandler wird der Strom in die analoge Ausgangsspannung umgewandelt. Die Struktur des AM6012 ist wegen Toleranzproblemen des 2R-Widerstandes, der den größten Summenstrom (MSB) bestimmt, etwas anders, denn dieser Strom muß eine Genauigkeit von ±1/2 LSB haben, um die Linearitätsanforderung an einen DA-Wandler zu erfüllen.

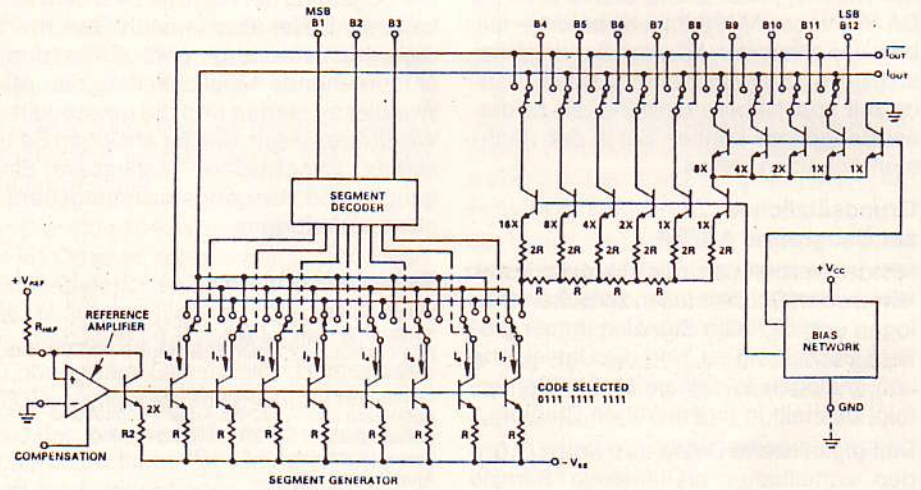


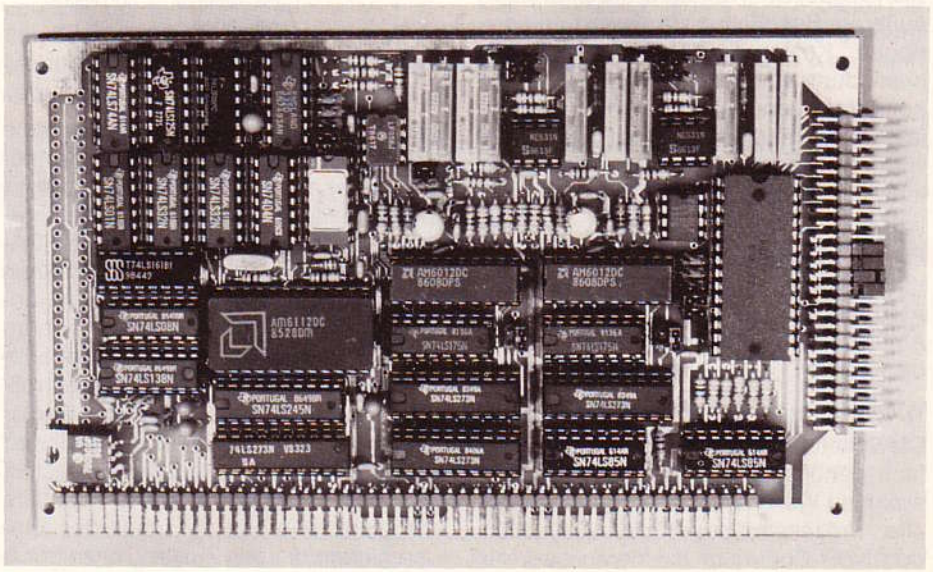
Bild 3–2:
Struktur eines 12-Bit DA-Wandlers

Solche Widerstände kann man mit Standardmethoden nicht herstellen. Die 4096 möglichen Ausgangspegel werden deshalb beim AM6012 in 8 Gruppen zu 512 Schritten unterteilt. Dadurch können Widerstände mit 8facher niedriger Toleranz verwendet werden.

Preise:

10736	AD/DA Handbuch	**20,00 DM**
10735	AD/DA Leiterplatte	**249,00 DM**
10734	AD/DA Bausatz	**998,00 DM**
10733	AD/DA Fertiggerät	**1198,00 DM**

Im Lieferumfang des Bausatzes und des Fertiggerätes ist eine Diskette enthalten, die ein Abgleichprogramm für den AD- und DA-Teil enthält. Außerdem ist ein kleines Demo-Oszilloskop Programm als COM-File auf dieser Diskette, das Sie unter CP/M 2.2 direkt starten können.



20 MByte Winchester für NDR- und mc-Computer

Nachdem die 10 MByte Festplatte jetzt nicht mehr lieferbar ist, wird von uns, wie schon angekündigt, die 20 MByte Festplatte vertrieben.

Zum Lieferumfang dieser Festplatte:
- OWL-Festplattee 20 MByte mit integriertem Controller,
- 2 Schrauben zur Montage der Festplatte ins GEH1, GEH2, GEH3 oder GEH4 (mit Zollgewinde),
- 1 Diskette 5¼", 80 Spuren, auf der sich verschiedene Utilities für die Festplatte befinden (siehe unten).

Achtung: Das Verbindungskabel (Art.-Nr. 10304) von der Winchester zur SASI ist nicht im Lieferumfang enthalten. Bitte einzeln bestellen.

Prinzipielles

Die 20 MByte Festplatte zieht natürlich eine BIOS-Änderung nach sich, um die 20 MByte Speicher verwalten zu können. Für das CP/M68k ist das neue BIOS auf der mitgelieferten Diskette enthalten. Das Einbinden des neuen BIOS ist ebenfalls ausführlich auf dem „LIESMICH.TXT" (diese Datei befindet sich ebenfalls auf der Diskette) erklärt.

Für den Z80 ist das neue BIOS noch nicht fertig. Sie können aber in der Zwischenzeit mit dem alten BIOS arbeiten, allerdings nur mit 8 MByte (da CP/M 2.2 pro Laufwerk nur 8 MByte verwalten kann; deshalb ist hier die BIOS-Änderung etwas umfangreicher).

Auf der Diskette befindet sich außerdem je ein Festplattenformatierer für CP/M 2.2 und für CP/M68k.

Zum Anschluß der Festplatte benötigen Sie weiter noch die SASI-Schnittstelle, die als Bausatz oder Fertiggerät lieferbar ist. (Für NDR-Computer: SASI Bausatz 10378, SASI Fertiggerät 10379 und für mc-Computer: FLOSASI Bausatz 10477, dto. Fertiggerät 10478.)

Außerdem empfehlen wir bei Betrieb mit der Festplatte nur das Netzteil NE3 für den NDR-Computer oder NE4 für den mc-Computer. Die Netzteile NE1 und NE2 sind aufgrund ihrer Belastbarkeit bei +12 V (nur 1 A) nicht geeignet.

Technische Daten:

Speicherkapazität (formatiert):	17,5 MByte
Zylinder:	612
Sektor Größe:	256 Bytes
Sektoren pro Spur:	32
Bytes pro Spur:	8192
Anzahl der Köpfe:	4
Spur-Dichte:	740 TPI
Aufzeichnungsverfahren:	MFM
Daten Transfer Rate:	5 MBit/sec
Zugriffszeit:	
Zugriffszeit von Spur zu Spur:	16 msec
durchschnittl. Zugriffszeit:	75 msec
maximale Zugriffszeit:	120 msec
Stromverbrauch +5 V:	typ. 1.3 A max. 1.6 A
Stromverbrauch +12 V:	typ. 0.7 A max. 2.5 A
Maße (mm):	203,2 x 146 x 41,4

10979 20 MByte Festplatte mit integriertem Controller **1995,00 DM**

ROA 256/1M –

die neue schnelle Speicherbaugruppe für den NDR- und mc-Computer

Nachdem die ROA 256/1M bereits in der letzten *LOOP* (Nr. 14) angekündigt wurde, können wir jetzt, nachdem sie lieferbar ist, noch etwas näher auf die Baugruppe eingehen. Die wichtigsten Daten der Baugruppe:

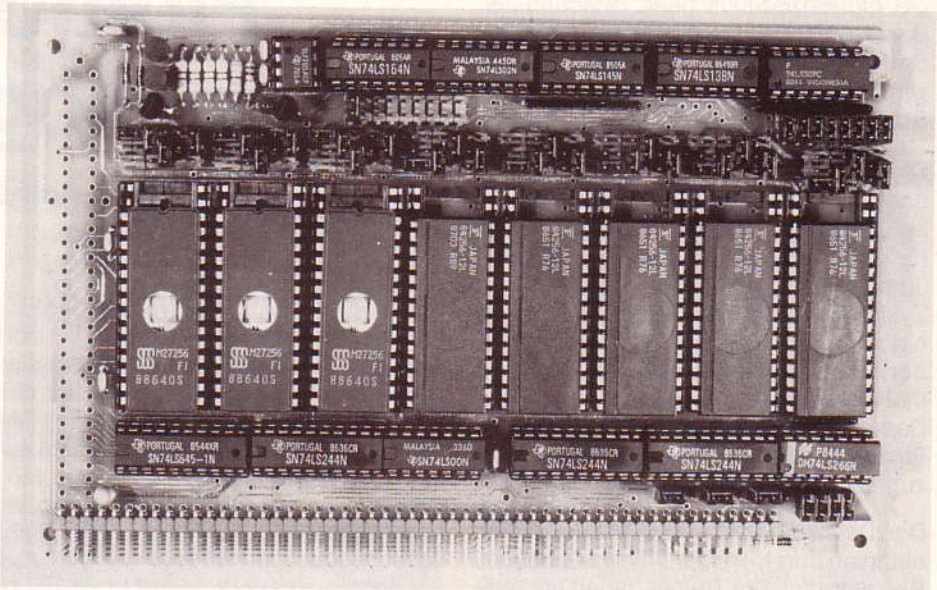
Technische Daten

Spannung:	+5 V
Stromaufnahme:	abhängig von den gesteckten Speichern, max. 500 mA, typ.: 200 – 300 mA
Leiterplattengröße:	Europakarte 100 x 160 mm
Bussysteme:	NDR-Bus 54polig oder ECB-Bus
Einsetzbare Speicher:	
EPROM:	32k x 8 (27256) 128k x 8 (27101)
RAM:	32k x 8 LP (LOW-Power) 128k x 8 LP (LOW-Power)
Batterie:	Nennspannung 3,6 V Lebensdauer 4 – 5 Jahre
Einstellbare WAIT-STATES:	0 (kein WAIT-STATE) bis 8
Ausblendlogik:	jeder Speicherbaustein kann ausgeblendet werden

Daten-, Adress- und Steuerbus voll gepuffert

Beschreibung des Blockschaltbildes und des Schaltungsprinzips

Der eigentliche Sinn und Kern der Schaltung sind die Speicher. Hier können statische RAM und EPROMs mit einem internen Speicheraufbau von 32k x 8 und 128k x 8 verwendet werden. Speicher mit diesem Aufbau stellen pro Adresse ein Byte zur Verfügung, was die Schaltung doch sehr vereinfacht. Jeder Speicherplatz ist



entweder mit einem EPROM oder mit einem statischen RAM bestückbar. Allerdings ist keine gemischte Bestückung von Speichern verschiedener Größen möglich, also entweder nur statische RAMs und EPROMS 32k x 8, oder statische RAMs oder EPROMs 128k x 8.

Zusätzlich wird auf der ROA 256/1M eine Schaltung zur Pufferung von statischen RAMs zur Verfügung gestellt. Die zusätzlichen Bauelemente und die Lithium-Batterie für die Batteriepufferung sind in der Standardversion der Baugruppe nicht enthalten. Diese Teile sind als Option erhältlich. Die Batteriepufferung besteht aus einer Abschaltelogik und der Lithium-Batterie. Die Abschaltelogik hat die Aufgabe, die statischen RAMs in den „Stand by Mode" zu überführen, sobald die Versorgungsspannung der Baugruppe unter 4,75 V absinkt. Der Kern dieser Schaltung ist ein intelligenter Spannungswächter von TI. Die Umschaltung in den „Stand by mode" wird mit Hilfe des Blockes RAM-Select bei Batteriepufferung durchgeführt. Dabei wird diesem Block bei Abfall der Versorgungsspannung unter 4,75 V die Versorgungsspannung abgeschaltet (auch Masse wird getrennt). Dadurch hängen die Ausgänge des Blockes, die die Speicher aktivieren, in der Luft und werden durch die Pull-up-Widerstände, die mit der Batterie verbunden sind, auf HIGH gezogen. Dies führt die statischen RAMs in den Stand-by-mode.

Wird die Batteriepufferung nicht verwendet, wird der RAM und ROM-Select von der gleichnamigen Logik durchgeführt. Diese Logik wählt je nach Adresse einen der Speicherbausteine aus. Abhängig von der Größe der verwendeten Speicher ist diese Logik durch Jumper (Brücken) einstellbar.

Werden Speicher von der Größe 128k x 8 verwendet, wird der gesamte, auf dem NDR-Computer adressierbare Speicher, auf der ROA 256/1M angesprochen (Board-Select). Dies ist vor allem dann ärgerlich, wenn man noch eine oder mehrere alte ROAs hat, die man weiter verwenden will. Aus diesem Grund wurde eine Ausblendlogik vorgesehen, mit der jeder Speicherbaustein der Baugruppe ausgeblendet werden kann. Dadurch wird für diesen Baustein der Board-Select unterbunden und es kann auf eine andere Speicherkarte (ROA 64k) zugegriffen werden.

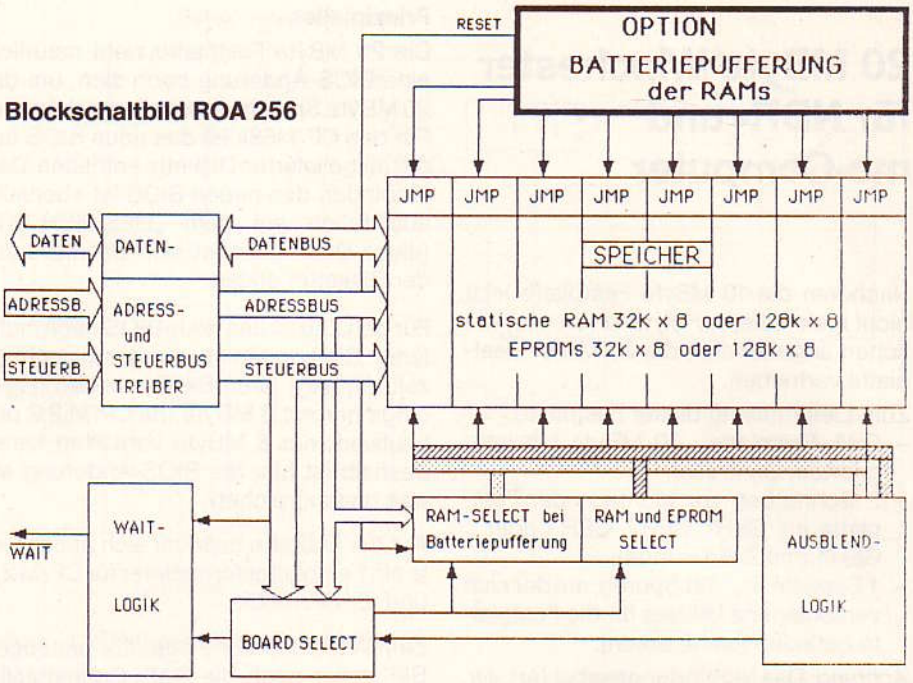
Die Board-Select-Logik stellt fest, wie der Name schon sagt, ob die Baugruppe angesprochen wird. Diese Logik ist natürlich davon abhängig, welche Größe von Speichern verwendet wird. Werden 32k x 8 statische RAMs oder EPROMs verwendet, gibt es die Möglichkeit, zwischen vier Basisadressen (je vier Bänke) zu wählen. Bei Verwendung von 128k x 8 Speichern wird die ganze Baugruppe bei einem Speicherzugriff ausgewählt.

Die WAIT-Logik dient dazu, den Prozessor bei einem Speicherzugriff zu bremsen, wenn die Zugriffszeit der Speicher für den Prozessor zu langsam ist. Über Jumper können 8 verschiedene WAIT-Zyklen eingestellt werden. Dabei errechnet sich die Dauer des WAIT-Signals aus der Anzahl der eingestellten WAIT-Zyklen mal der Periodendauer des CPU-Taktes. Dadurch kann die Geschwindigkeit des Prozessors (CPU) ideal an die Speicherzugriffszeit angepaßt werden.

Der Daten-, Adress- und Steuerbus ist natürlich durch Treiber physikalisch vom Bus getrennt. Die Treiber haben die Aufgabe, die Signale zu verstärken, um somit den Bus nicht zusätzlich zu belasten. Der Datenbustreiber ist bidirektional, da die Daten sowohl auf die Baugruppe geschrieben, als auch von der Baugruppe gelesen werden.

Die Baugruppe ROA 256/1M wurde nach GES-Norm entwickelt und stellt demnach außer dem NDR-Bus auch den ECB-Bus zur Verfügung. Leider existiert im Programm des mc-CP/M Computers im Moment noch keine Baugruppe für den mc-Computer, die eine „Page-Logik" zur Verfügung stellt, welche zum Betrieb mit der ROA 256 benötigt wird. Natürlich kann die Baugruppe bei allen ECB-Systemen, die der Anschlußbelegung entsprechen, eingesetzt werden.

Blockschaltbild ROA 256



Flomon 4.2 – neue Möglichkeiten

von Rolf-Dieter Klein

Ab sofort kann eine neue Version des Flomon geliefert werden. Mit dieser Version wird der NDR-Klein-Computer noch flexibler. So ist es nun möglich, statt der GDP ein Terminal anzuschließen. Die Adresse für die SER-Baugruppe liegt auf 0F4H, somit ist mit ZEAT auch der Modem-Betrieb ohne Probleme möglich, es ist jedoch eine zweite SER erforderlich. Die Adresse 0F0H für das Modem blieb erhalten. Der Betrieb des ZEAT-Systems kann ebenfalls über das Terminal erfolgen; jedoch geht das Graphikbild in der Startmeldung verloren, was aber nicht weiter stört. Wer ein Terminal anschließen, aber seine parallele Tastatur behalten möchte, kann dies natürlich auch, ebenso kann jetzt die GDP und eine serielle Tastatur verwendet werden. Für den Terminal-Betrieb bzw. für eine serielle Tastatur stehen Baudraten von 50 bis 19200 zur Verfügung. Die Steprate der Laufwerke kann zwischen langsam und schnell (3ms) umgeschaltet werden. Die GDP kann auch zur Druckausgabe verwendet werden. Statt der CAS kann über RDR und PUN auch die SER verwendet werden. Es sind dann die Sprungadressen auf 0F1D7H von CD 022C nach DC 0261 und 0F1EF von CD 0235 nach DC 0261 zu patchen. Die Scrollroutinen für die GDP wurden schneller gemacht. Es wird nur noch eine Bildschirmseite verwendet. Da bei jedem Scroll das Videosignal abgeschaltet wird, flimmert der Bildschirm bei jedem Scroll, dies läßt sich leider nicht verhindern. Es wird dadurch die Scrollgeschwindigkeit erheblich erhöht.

Zur Umlenkung der Ein- und Ausgaben wird der Schalter auf der KEY-Baugruppe verwendet. Hier die einzelnen Schalter und deren Bedeutung:

Schalter 7:
1 = Consolenausgabe umleiten an SER (Terminal), GDP nur für Graphik
0 = Consolenausgabe an GDP

Schalter 6:
1 = Consoleneingabe über SER (Terminal oder serielle Tastatur), Consolenstatus über SER abfragen
0 = Consoleneingabe über KEY, Consolenstatus ebenfalls über KEY abfragen

Schalter 5:
1 = Langsamste Steprate
0 = Schnellste Steprate

Schalter 4:
1 = Druckerausgabe an GDP
0 = Druckerausgabe an CENT
Schalter 3 – 0: Zur Baudrateneinstellung

```
Baudraten:

SCHALTER:   3 2 1 0
            0 0 0 0 =   9600
            0 0 0 1 =     50
            0 0 1 0 =     75
            0 0 1 1 =    109,92
            0 1 0 0 =    134,58
            0 1 0 1 =    150
            0 1 1 0 =    300
            0 1 1 1 =    600
            1 0 0 0 =   1200
            1 0 0 1 =   1800
            1 0 1 0 =   2400
            1 0 1 1 =   3600
            1 1 0 0 =   4800
            1 1 0 1 =   7200
            1 1 1 0 =   9600
            1 1 1 1 =  19200
```

Noch ein Hinweis: Schalter = 0 entspricht Schalter ON, da der Kontakt gegen Masse geschlossen wird.

Wenn die Schalter 7 und 6 = 1 und Schalter 4 = 0 sind, wird die GDP nicht initialisiert. Will man sie trotzdem verwenden, sind vorher folgende Routinen in dieser Reihenfolge aufzurufen: PENDOWN, CLRALL.

Die Schalterstellungen auf der KEY-Baugruppe werden nur nach einem RESET eingelesen und an einer Speicherstelle abgelegt. Es ist somit sinnlos, während des Betriebs die Schalterstellungen auf der Baugruppe zu verändern. Muß während des Betriebes die Konfiguration geändert werden, so ist dazu ein Patch auf die Speicherstelle 0F033 nötig.

Komfortabler Disassembler für die 68000-Serie unter JADOS

NEU!

Jetzt stellt sich Ihnen vielleicht die Frage, was ein Disassembler sein soll? Nun, dieses Programm ist das Gegenstück zum Assembler; er stellt aus dem Hexadezimalcode, dem Programm, wieder einen lesbaren Text her, so daß es möglich ist, Programme, zu denen man keinen Quelltext hat, zurückzuübersetzten und nach Wunsch zu ändern.

Für den NDR-Computer gibt es ja bekanntlich schon ein paar Disassembler. Dieser nun verfügbare kann aber wesentlich mehr als die bisherigen Programme. Er kann nicht nur 68008-Befehle übersetzen, sondern auch die 68010/68020- und sogar die FPU-Befehle. Dabei stehen auch die neuen 68020-Adressierungsarten zur Verfügung, die recht umfangreich sind. Es ist damit möglich, Programme von Assemblern, die diese Befehle bereits kennnen, zu übersetzen und an den Grundprogramm-Assembler anzupassen.

Aber dieses sind nicht alle Neuerungen des Programms:

Die Ausgabe kann mit einem Anfangsmenü, ebenso wie die CPU ausgewählt werden. Dabei kann auf den Bildschirm oder einen Drucker ausgegeben werden. Außerdem kann der Assemblertext auch im Speicher abgelegt werden, wodurch die Möglichkeit besteht, den übersetzten Code mit einem Editor zu bearbeiten, zu ändern und mit einem Assembler wieder zu übersetzen.

Die CPU kann ausgewählt werden, damit ein Programm, das für den 68000 geschrieben wurde, auch richtig übersetzt wird und IO-Marken erkannt werden, da diese ja mal zwei genommen werden.

Weiterhin kann der Disassembler eigene Marken setzen, selber welche definieren, Symbole aus der Symboltabelle übernehmen und IO-Adressen bezeichnen. Er arbeitet mit dem JADOS-Betriebssystem zusammen und kann auch den Editor direkt auf eine JADOS-Diskette abspeichern oder ein Programm zum Übersetzen von der Diskette laden.

Die RDK-Befehle aus dem Grundprogramm wie §schreite, §drehe usw. werden auf Wunsch erkannt und zurückübersetzt, wodurch Programme auf TRAPs umgearbeitet werden können.

Die oben genannten Funktionen stellen eine Auswahl dar und geben natürlich nur einen kleinen Überblick über die Leistungsfähigkeit des Programms, das sehr schnell bei der Bearbeitung ist. Es werden z. B. 32 KByte ohne Marken in 15 Sekunden in den Speicher übersetzt (68008 CPU). Die Bedienung ist durch mehrere Menüs auf einem Bildschirm sehr leicht und übersichtlich zu bedienen.

Das Programm benötigt mindestens 18 KByte freien Speicherplatz. Es wird natürlich für die Definition von Marken oder für die Symboltabelle noch Platz benötigt. Außerdem kann ja auch in den Editor übersetzt werden, was auch noch Speicher verbraucht. Die Beschreibung zu dem Disassembler hat einen Umfang von 22 KByte.

Aufruf zur 23. Wettbewerbsrunde Jugend forscht 1988

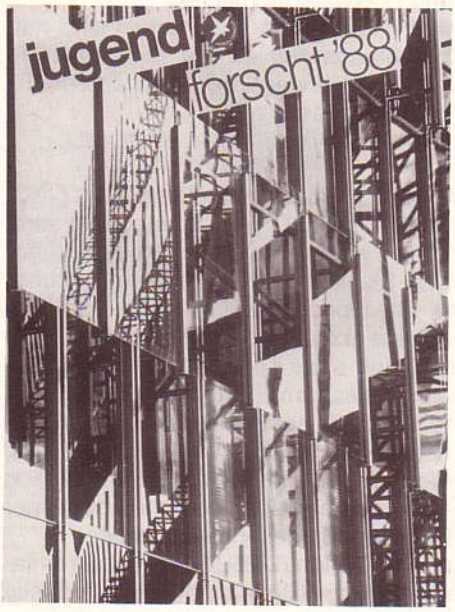
Guckt mal hinter die Fassade!

- Wieso reflektiert Glas?
- Was ist das – Licht?
- Wie kann ich Sonnenlicht nutzen?

Das sind Fragen, mit denen eine Jugend forscht-Arbeit anfangen könnte: suchen, was dahinter steckt, neugierig fragen nach dem, was nicht gleich sichtbar ist, neue Lösungen für Altbekanntes finden.

Wenn Ihr am 31. Dezember 1987 noch keine 22 Jahre alt seid, könnt Ihr Euch in den Fachgebieten Biologie, Chemie, Geo- und Raumwissenschaften, Mathematik/Informatik, Physik, Technik oder beim Sonderpreis Arbeitswelt mit einem selbstgewählten Thema anmelden.

Anmeldeschluß für die 23. Runde ist der 30. November 1987. Alles weitere erfahrt Ihr bei der Stiftung Jugend forscht e.V., Notkestraße 31, 2000 Hamburg 52.



Endlich:

HEBAS nun auch auf EPROM für SBC3

EHEBAS, ein komfortables EPROM-BASIC von Dr. Hehl

Endlich ist es soweit: Für die CPU-Baugruppe SBC3 gibt es jetzt den bekannten BASIC-Interpreter HEBAS als 16 KByte-EPROM-Version in zwei Eproms 2764. Ausgehend von dem Gedanken, daß der Einsteiger bei einem modularen System möglichst viel erweitern kann, ohne daß Baugruppen dann nicht mehr benötigt werden, wurde nun der BASIC-Interpreter HEBAS für die Baugruppe SBC3 angepaßt.

Das eingegebene Programm (max. 13 KByte lang) bleibt im Speicher (zwei 8 K RAM-Bausteine), da dieser mit einem Akku gepuffert ist. Als zusätzliches Speichermedium kann der preiswerte Kassetten-Recorder verwendet werden. Bei

einem späteren Ausbau auf Diskettenbetrieb mit dem Betriebssystem CP/M 2.2 können die Programme von Kassette weiter verwendet werden. Die Eproms mit EHEBAS kommen auf eine Speicherkarte ROA64 und stehen auch bei anderen Betriebssystemen (Z80-Prozessor oder Nachfolger) zur Verfügung. Sogar das Handbuch zum BASIC-Interpreter wird weiter verwendet. Das lästige Umlernen bei verschiedenen BASIC-Interpretern entfällt, da die Befehle und die Befehls-Schlüsselwörter (Token) von EHEBAS und dem großen HEBAS gleich sind.

EHEBAS besitzt fast alle Befehle vom Disketten-HEBAS außer den Disketten-Befehlen, d.h. auch alle komfortablen BASIC-Befehle wie z. B. RENUMBER, COPY, REPLACE, TRACE, FIND, PRINT USING. Weiterhin sind ausführliche deutsche Fehlermeldungen vorhanden. Programmzeilen können mit dem EDIT-Befehl geändert werden. Es gibt die Graphik-Befehle MOVETO, DRAWTO, PAGE und CLR sowie die Kassettenbefehle LOADC und SAVEC, mit denen ein Programm auf Kassette abgelegt wird. Die Drucker-Schnittstelle ist als parallele Centronics-Schnittstelle programmiert.

Später könnte der Einsteiger unter zusätzlicher Verwendung der Baugruppe COL256 mit einem erweiterten EHEBAS (dann zwei Eproms 27128 mit je 16 KByte) auch in Farbe programmieren, ohne daß ein Disketten-Betriebssystem vorhanden sein muß.

Endlich:

JADOS, CP/M68K, SYSTEMA:

Der neue PET-Texteditor ist da!

Wie auch bei anderen Rechnern – wer je einmal mit dem „EDLIN" beim IBM arbeiten mußte, weiß das – fristet auch der Text-Editor beim NDR-Computer ein Schattendasein. Der von Rolf-Dieter Klein ins Grundprogramm integrierte Editor war nur zur Eingabe einfacher und kurzer Texte gedacht; sehr schnell kam man mit ihm an die Grenzen der Möglichkeiten.

Die Z80-Anwender konnten problemlos auf den preiswert angebotenen WordStar umsteigen; für CPU 68xxx-Benutzer gibt es dieses Programm jedoch leider nicht. Ist ab jetzt auch nicht mehr nötig, denn der neue PET-Editor kann mehr als WordStar!

Einige herausragende Möglichkeiten, die alle mit der „normalen", ungeänderten GDP-Baugruppe funktionieren:

Problemlose inverse Darstellung

Zeilenlänge max. 2048 Zeichen

Extrem hohe Scrollgeschwindigkeit; dabei Vertikal- **und** Horizontal-Scroll möglich

Blockverarbeitung:
Markierte Blöcke werden invers dargestellt
Blöcke können horizontal **und** vertikal verschoben werden

Makrodefinitionen sind möglich, d. h. immer wieder kehrende Kommandos sind als Makro zu definieren

Wohldokumentierte Benutzeranweisung sowie Hilfe-Bildschirme.

Der PET-Texteditor zum Preis von DM 99,– ist eine Anschaffung, die sich lohnt! Der Editor funktioniert mit JADOS, CP/M68K sowie Systema-Dos 5.0. Man muß das Programm gesehen haben und damit arbeiten! Bitte denken Sie auch daran: Solche Produkte sind nur dann für den Programm-Autor lohnend, wenn er auch seine (wohlverdiente) Lizenzgebühr erhält. Beim Preis von DM 99,– lohnt sich kopieren nicht mehr!

Update Modula-2 Compiler Version V2.30

Unser Modula-2 Compiler für den NDR-Klein-Computer unter CP/M 68K liegt jetzt in einer neuen Version vor. Die Version 2.30 bietet gegenüber den alten Versionen erhebliche Vorteile, die u.a. das Arbeiten mit großen Datenmengen bedeutend erleichtern.

Die Update enthält im wesentlichen die folgenden Änderungen gegenüber den älteren Versionen:

- Die 32K-Beschränkung für Datenstrukturen entfällt, d. h. Felder können nun beliebig groß vereinbart werten.
- Die Typen INTEGER, CARDINAL, LONGINT und LONGCARD sind alle untereinander zuweisungskompatibel.
- INC und DEC sind auch für LONGINT und LONGCARD zulässig.
- In konstanten Ausdrücken sind nun alle Standardfunktionen (einschließlich Typanpassungen) mit konstantem Ergebnis zulässig, z. B.

```
CONST c = MAX (INTEGER) - TSIZE (REAL);
      dif = CHR (ORD ("a") - ORD ("A"));
```

- Die Typwandelfunktionen LONG, SHORT, TRUNC, usw. können nun auch ihren Ergebnistyp als Parametertyp haben. Sie liefern dann die leere Leistung.
- Die Option /ERROR in der Kommandozeile erzeugt eine Datei M2ERRORS.LOG mit einer Kopie der Ausgaben auf dem Bildschirm. Existiert die Datei M2ERRORS.LOG schon, dann werden die neuen Ausgaben hinten angefügt. Auf diese Weise kann man sich ein Logging über mehrere Übersetzungen aus einer Kommandodatei erstellen lassen.
- Die Option /DEBUG erzeugt als einfache Debug-Hilfe in der Assemblerquelle Kommentarzeilen mit Quell-Zeilennummer und PC-Offset zum Modulanfang.
- Der Pointertest (Test auf NIL vor Dereferenzierung) ist nun realisiert.
- Der Compiler besitzt nur noch 3 statt 4 Läufe.
- Im Modul Text aus M2LIB wurden geändert:
  CondRead geht nun auch für "AUX:"
- Im Modul SimpleIO aus M2LIB wurden geändert:
  WriteInt und WriteCard haben nun LONGINT und LONGCARD als Parametertypen.
  Die Prozeduren WriteLongNum und ReadLongNum wurden ergänzt.
- Das Modul ReaLLO wurde ergänzt
- Das Modul MathLib wurde auf REAL umgestellt und für LONGREAL wurde das Modul MAthLibLong ergänzt.
- Ein Utility-programm M2Cross zur Ausgabe von modulübergreifenden Referenzen wird mitgeliefert.
- Neu überarbeitete Bedienungsanleitung.

SoftDesign

CP/M-Z80 Emulator für den mc 68000

Übersicht:

- **Echtzeit-Emulation aller Z80 Maschinenbefehle** im gesamten 64K Adressbereich.
- Eine CP/M-80 Version 2.2 kompatible Systemumgebung und Benutzerschnittstelle sind implementiert, **alle CP/M-80 kompatiblen Programme** können ausgeführt werden.
- Durchschnittliche CP/M-80 Programme laufen unter dem Emulatoren mit der Geschwindigkeit eines 2 MHz Z80-Systems (auf einem 8 MHz CP/M-68K-System).
- Durch den Emulator verhält sich **jedes CP/M-68K-System** wie ein CP/M-80-System ohne jede Hardware- oder Software-Änderung. Der Emulator läuft unter Version 1.1, 1.2 und 1.3 von CP/M-68K.
- Disketten sind zwischen CP/M-68K und dem emulierten CP/M-80 **voll austauschbar.** Eine unter CP/M-80 geschriebene Diskette kann unter CP/M-68K gelesen und geschrieben werden und umgekehrt.
- Die Emulation des Z80 ist auf jedem mc 68000 System lauffähig. Die CP/M-80 Emulation kann an andere Betriebssysteme als CP/M-68K angepaßt werden.
- Z80 Ein-/Ausgabe-Maschinenbefehle werden durch eine Schnittstelle zu hardwareabhängigen Ein-/Ausgabe Treibern unterstützt.
- Grundprogramm (68008) ab $E0000!

Einfache Ansteuerung durch REL1-Programm:

Relais-Baugruppe am Einsteigerpaket

Dieses Programm weist jedem Relais auf der Baugruppe REL eine Taste von 0 bis 7 auf der HEXI/O zu. Durch Betätigen einer Taste zwischen 0 und 7 zieht das bezügliche Relais an, bei erneutem Druck auf die gleiche Taste fällt das Relais wieder ab.

Auf der REL-Baugruppe muß durch die DIL-Schalter die Portadresse 30h eingestellt sein.

Programm:

Mit der Betriebssystemroutine HOLETASTE wird die Tastatur abgefragt. Die Routine TONUM wandelt dann den Tastaturcode in die Werte um, die auch auf der Tastatur angeschrieben sind.

In dem Unterprogramm MASKE wird ein Byte erzeugt, das an der Stelle des Tastaturwertes ein Bit gesetzt hat. Wurde z.B. die Taste „3" gedrückt, so wird das Bitmuster "00001000" erzeugt. Dies ist notwendig, um zu prüfen, ob das Relais angeschaltet ist oder nicht.

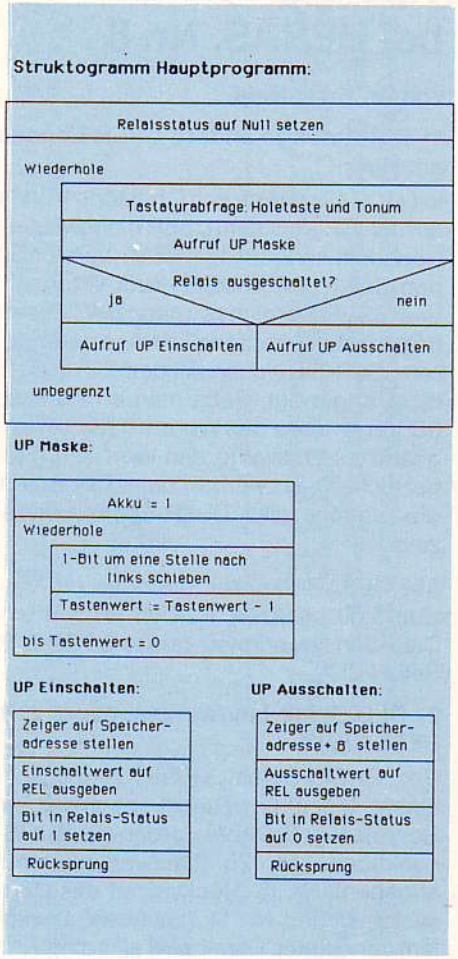
Dieses Bit-Muster wird dann mit dem Relais-Status UND-verknüpft. Im Relais-Status-Register ist ein Bit gesetzt, wenn das zugehörige Relais angezogen ist, und Null, wenn das Relais aus ist. Durch die UND-Verknüpfung mit der Maske ergibt sich im Akku eine Null, wenn das angewählte Relais abgefallen ist, ansonsten ein von Null verschiedener Wert. Je nach dem Ergebnis dieser Operation wird entweder in die Einschalt- oder Ausschaltroutine verzweigt.

In dem Unterprogramm EINSCHALT wird zuerst die Adresse der Ausgabedaten in das HL-Register geladen. Zu diesem wird nun der Tastenwert hinzuaddiert und das Byte, das auf dieser Adresse gefunden wird, auf die REL-Baugruppe gegeben. Dadurch wird das Relais angeschaltet.

Nachdem durch die ODER-Verknüpfung mit der Maske das Bit des soeben eingeschaltetenen Relais auf 1 gesetzt wurde, erfolgt der Rücksprung in das Hauptprogramm.

Das Unterprogramm AUSSCHALT ist ähnlich aufgebaut. Hier wird das entsprechende Byte an die REL-Karte ausgegeben, das das Relais veranlaßt auszuschalten. Mit EXKLUSIV-ODER wird der Relaisstatus mit der Maske so verknüpft, daß in das entsprechende Bit eine Null geschrieben wird. Dies signalisiert den Zustand des ausgeschalteten Relais.

Nach dem Rücksprung ins Hauptprogramm geht das Programm wieder auf den Anfang und es kann erneut eine Taste gedrückt werden.




```
MACRO-80 3.43   27-Jul-81      PAGE   1

                        .z80
                        ;*****************************************
                        ;*  Ansteuerung der REL-Baugruppe        *
                        ;*  Rel1   V 1.0                         *
                        ;*  (C)  Michael Giulini   22.09.1987    *
                        ;*****************************************

                                org 2100h
0030                            rel equ 030h
000C                            holetaste equ 000ch
000F                            tonum equ 000Fh

2100'  16 00            start:  ld d,00h            ; D-Register löschen
2102'  21 2140'                 ld hl,ablage        ; Adresse für Tabelle

2105'  CD 000C          eingabe: call holetaste     ; Tastaturabfrage
2108'  CD 000F                  call tonum          ; Umwandlung des Tastaturcodes
210B'  4F                       ld c,a              ; Tastaturwert nach Reg. C
210C'  CD 211A'                 call maske          ; Aufruf Up Maske
210F'  5F                       ld e,a              ; Bit-Maske nach Reg. E
2110'  A2                       and d               ; Prüfen, ob Relais aus
2111'  CA 2124'                 jp z, einschalt     ; wenn ja, Sprung zu Einschaltroutine
2114'  C4 2133'                 call nz, ausschalt  ; sonst Aufruf Ausschaltroutine
2117'  C3 2105'                 jp eingabe          ; Schleifenende

                        ;Unterprogramme


211A'  41               maske:  ld b,c              ; Tastaturwert nach Reg. B
211B'  3E 01                    ld a,01h            ; 01h in Akku laden
211D'  CB 07            schieben: rlc a             ; bit um 1 Stelle nach links schieben
211F'  05                       dec b               ; Zähler erniedrigen
2120'  C2 211D'                 jp nz, schieben     ; Schleifenende, wenn B=0
2123'  C9                       ret                 ;Rücksprung ins Hauptprogramm


2124'  06 00            einschalt: ld b,0h          ; B-Reg. löschen
2126'  21 2140'                 ld hl,ablage        ; Adresse von Tabelle nach Reg. HL
2129'  09                       add hl,bc           ; Adresse um Tastaturwert erhöhen
212A'  7E                       ld a,(hl)           ; Ausgabewert aus Tabelle nach A
212B'  D3 30                    out (rel),a         ; Wert an Relais-Karte ausgeben
212D'  7A                       ld a,d              ; Relais-Status in Akku laden
212E'  B3                       or e                ; Relais-Status auf 1 setzen
212F'  57                       ld d,a              ; Relais-Status in Reg. D ablegen
2130'  C3 2105'                 jp eingabe          ; Sprung ins Hauptprogramm


2133'  06 00            ausschalt: ld b,0h          ; Reg. B löschen
2135'  21 2148'                 ld hl,ablage+8      ; Adresse Aus-Tabelle nach Reg. HL
2138'  09                       add hl,bc           ; Tastaturwert zu Adresse addieren
2139'  7E                       ld a,(hl)           ; Ausgabewert von Tabelle nach Reg. A

MACRO-80 3.43   27-Jul-81      PAGE   1-1

213A'  D3 30                    out (rel),a         ; Wert an REL-Karte ausgeben
213C'  7A                       ld a,d              ; Relais-Status in Akku laden
213D'  AB                       xor e               ; Relais-Status auf 0 setzen
213E'  57                       ld d,a              ; Relais-Status in Reg. D ablegen
213F'  C9                       ret                 ; Rücksprung ins Hauptprogramm

2140'  01 03 05 07      ablage: db 01h,03h,05h,07h,09h,0bh,0dh,0fh,00h,02h,04h,06h,08h,0ah
2144'  09 0B 0D 0F
2148'  00 02 04 06
214C'  08 0A
214E'  0C 0E                    db 0ch,0eh

                                end

MACRO-80 3.43   27-Jul-81      PAGE   S

Macros:

Symbols:
2140'  ABLAGE      2133'  AUSSCHALT   2105'  EINGABE
2124'  EINSCHALT   000C   HOLETASTE   211A'  MASKE
0030   REL         211D'  SCHIEBEN    2100'  START
000F   TONUM

No Fatal error(s)
```

PASCAL und BASIC

Tips und Tricks bei HEBAS, Nr. 8

von Dr. Hans Hehl

1) DIR-Einträge andere USER-Ebenen anzeigen

In *LOOP* 13a wurde von Christoph Köhler auf den CP/M-Befehl USER hingewiesen. Nachdem die aktuelle HEBAS-Version 3.1 den Befehl CALL USER zum Wechseln des USER-Bereiches unter BASIC enthält, werden Directory-Einträge bei anderen USER-Bereichen mit dem DIR-Befehl nicht angezeigt. Setzt man an Adresse 35B0h anstelle des Wertes 0 (für voreingestelltes Laufwerk) den Wert 3Fh (Fragezeichen), so werden beim DIR-Befehl die Einträge aller USER-Ebenen angezeigt.

Alle HEBAS-Besitzer, die noch die Version 1.00 besitzen, werden an den UP-Date-Service erinnert (siehe *LOOP* 8/9, Seite 22).

2) RESET mit Laufwerksangabe: z. B. RESET (B)

Der BASIC-Befehl RESET ermöglicht einen Diskettenwechsel nach einem Schreibzugriff. Dabei werden die BDOS-Funktionen Nr. 25 (Laufwerksnummer angeben), Nr. 13 (Rücksetzen des Disksystems) und Nr. 14 (Laufwerk anwählen) verwendet. Damit wird allerdings immer das zuvor unter CP/M angewählte Laufwerk wieder eingestellt, ein Wechsel ist nicht möglich. Nähere Einzelheiten dazu sind im Handbuch HEBAS-Quelle für die kommentierte Quelldatei von HEBAS enthalten.

Der neue RESET-Befehl ermöglicht nun die Angabe eines Laufwerks in Klammern, z.B RESET(B). Damit kann man auch den Befehl DIR ohne umständliche, zusätzliche Angaben für dieses Laufwerk verwenden. Beim BYE-Befehl startet CP/M auf dem angesprochenen Laufwerk.

Platz für die Erweiterung bietet der Bereich von 33F1h bis 340Ch, der der IO-Byte-Einstellung dient und beim NDR- und mc-Rechner nicht verwendet wird. In HEBAS sind ab Adresse 33F1h sowie bei 3612h die in Bild 1 enthaltenen Befehle einzugeben. Aus Platzgründen muß eine Überprüfung, ob das angegebene Laufwerk sinnvoll ist, unterbleiben. Der Leser möge hier gerne Verbesserungen durchführen. Wer die HEBAS-Quelle besitzt, kann natürlich die Änderung an anderer Stelle einfügen.

3) 64 KByte-EPROM-Floppy am NDR-Rechner für HEBAS

Mit einer ROA64-Speicherkarte und einer Änderung der RAM-Floppy-Routine im BIOS und einer Definition eines Laufwerkes F können Programme neben der RAM-Floppy in EPROMs abgelegt werden, z.B. HEBAS. Auf der ROA-Karte werden ab Adresse 0 Eproms vom Type 2764 verwendet und ein statischer Baustein auf dem ganz rechten Sockel für den Kopiervorgang auf Bank 0. Das Eprom 0 muß das Directory für das jeweilige Programm enthalten, ab Adresse 800 beginnt das Programm. Für HEBAS werden drei Eproms 2764 benötigt. BIOS und Eproms können beim Autor H. Hehl erworben werden. Damit können dann auch die „Karten-Eproms" verwendet werden, die wie Scheckkarten aussehen und mittels Adapter dann bequem und schnell bei der Eprom-Floppy auswechselbar sind. Eventuell wird eine spezielle Platine mit diesem Adapter entwickelt.

4) Leserreportage

Wer arbeitet mit HEBAS und will uns darüber berichten? Zuschriften dazu bitte an Dr. Hans Hehl,
Lindenstraße 20 · 8059 Wartenberg.
Wir würden uns über eine Resonanz freuen.

```
33F1  4F          LD C,A          ;LAUFWERK RETTEN
33F2  7E          LD A,(HL)       ;BYTE NACH BEFEHL
33F3  FE 28       CP 28           ;FOLGT DAS ZEICHEN '('
33F5  20 0C       JR NZ,3403      ;SONST ALTES LAUFWERK
33F7  23          INC HL          ;NACHSTES BYTE
33F8  7E          LD A,(HL)
33F9  FE 40       CP 40           ;NUR BYTE > 40 ERLAUBT
33FB  DA 3F 20    JP C,203F       ;SONST SYNTAX-FEHLER
33FE  D6 41       SUB 41          ;A = 41H = LAUFWERK 0
3400  23          INC HL
3401  23          INC HL          ;ZEIGER AUF NACHSTEN BEFEHL
3402  4F          LD C,A          ;NEUER LAUFWERKCODE
3403  3E 0E       LD A,0E         ;BDOS-FUNKTION NR. 14
3405  C9          RET             ;ZURUCK NACH 3615

3612 CD F1 33     CALL 33F1       ;UNTERPROGRAMM AUFRUFEN
```

Bild Nr. 1: Änderungen in HEBAS.COM für RESET mit Laufwerksangabe

FÜR 68000-EINSTEIGER

Lottozahlengenerator

mit dem Einsteigerpaket

Wenn Sie allwöchentlich das Problem haben, sich für Ihren Lotto-Tip zu entscheiden, dann kann Ihnen das Programm LOTTO helfen. Dieses Programm ermittelt mit Hilfe eines sogenannten Zufallszahlengenerators eine mögliche Tipfolge. Es kann natürlich nicht das richtige Ergebnis vorhersagen! Erwarten Sie also nicht sofort "Sechs Richtige" !

Wie kann der der Mikroprozessor "Zufallszahlen" erzeugen, er kann doch keinen Würfel werfen!? In der Tat ist der Begriff "Zufall" im Zusammenhang mit Computern umstritten. Schauen wir uns den erwähnten Würfel einmal genauer an: Wenn Sie alle Umstände genau kennen würden, Gewicht, genaue Form und Oberfläche des Würfels, evtl. Luftbewegungen, Richtung, Stärke und Drehimpuls des Wurfes etc., könnten Sie dann nicht das Ergebnis des Würfelns vorausberechnen? Genauso ist es bei computererzeugten Zufallszahlen. Wie fast alles in und um einen Computer ließen sich diese Zahlen vorhersagen, nur wäre das praktisch viel zu kompliziert und bei der Methode, die wir verwenden wollen, praktisch unmöglich.

Also noch einmal: Wie kann der Mikroprozessor Quasi-Zufallszahlen erzeugen?

Der Mikroprozessor Z80 bietet hier eine recht einfache Möglichkeit. Es gibt einen Befehl ld a,r (lade a mit dem Inhalt von r), wobei a der Akkumulator ist und r das "Refresh"-Register. Refresh heißt Auffrischen und bezeichnet bei speziellen Speicherbausteinen (dynamische RAMs) einen Vorgang, der regelmäßig wiederholt werden muß, um nicht den Speicherinhalt zu verlieren. Im Refresh-Register r des Z80 steht eine Zahl zwischen 0 und 127, die bei diesen Speicherbausteinen angibt, welcher Teil des Speichers als nächstes aufgefrischt werden muß. Da so ein Refresh sehr häufig durchgeführt werden muß, wird das Register r sehr schnell weitergezählt, so daß jedesmal, wenn wir einen Befehl ld a,r verwenden, im Akku eine andere Zahl ankommt.

Das Programm LOTTO benutzt genau diese Methode. Das Unterprogramm random dient hier dazu, eine Zufallszahl zwischen 1 und 38 zu erzeugen. Dazu wird zunächst der Akku mit dem Wert des Refresh Registers geladen, damit haben wir eine Zufallszahl zwischen 0 und 127. Nun wird von dieser Zahl solange 38 abgezogen, bis sie kleiner als 38 ist. Damit haben wir eine Zahl zwischen 0 und 37. Jetzt wird noch eine Eins zum Akkuinhalt addiert, das nennt man inkrementieren (englisch mit "c" statt "k", deshalb "inc"). Wir wollen die Zahl aber im Registerpaar HL haben, um sie einfach mit Hilfe des HEXMON-Unterprogrammes PrtDez (Print Dezimal) ausgeben zu können. Also laden wir das Register l mit dem Inhalt des Akkus und setzen das Register h auf Null. Soweit, so gut!
-Aber... Damit das LOTTO- Programm brauchbar wird, ist entprechend den Regeln die Einschränkung erforderlich, daß jede der Zahlenkugeln nur einmal gezogen werden darf. Eigentlich muß sich der Computer nur merken, welche Zahlen er bereits gezogen hat. Dafür gibt es

Preisliste **Artikelgruppe 1**

Stand: 10.10.87

Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM
10000	428.00	10058	11.99	10122	10.00	10185	10.00	10244	50.00
10001	388.00	10059	11.99	10123	25.00	10186	20.00	*10245	
10002	528.00	10060	248.00	10124	59.00	10187	38.00	10246	50.00
10003	488.00	10061	20.00	10125	129.00	10188	38.00	10247	10.00
10004	97.99	10062	798.00	10126	10.00	10189	30.00	10248	20.00
10005	147.99	10064	798.00	10127	20.00	10190	144.00	10249	147.99
10006	348.00	10067	198.00	10128	38.00	10191	208.00	10250	55.01
10007	448.00	10068	268.00	10129	38.00	10192	10.00	*10251	
10008	498.00	10069	10.00	10130	38.00	10193	99.00	10252	378.00
10009	598.00	10070	40.00	10131	38.00	10195	50.00	10253	10.00
10010	585.00	10071	97.99	10132	97.99	10196	120.00	*10254	
10011	698.00	10072	158.00	10133	45.00	10197	95.00	10255	20.00
10012	1015.00	10073	30.00	10134	268.00	10198	15.00	10256	15.00
10013	1268.00	10074	39.00	10135	97.99	10199	50.00	10257	15.00
10014	348.00	10075	69.00	10136	147.99	10200	50.00	*10258	
10015	388.00	10077	10.00	10137	10.00	10201	50.00	10259	199.00
10016	1198.00	10078	38.00	10138	39.00	10202	35.00	10260	15.00
10017	1348.00	10079	59.00	10139	29.00	10203	30.00	10261	249.00
10018	1398.00	10080	35.00	10140	38.00	10204	78.00	10262	298.00
10019	1698.00	*10081		10141	8.00	10205	65.00	10263	10.00
10020	2238.00	10082	48.00	10142	98.00	10206	45.00	10264	30.00
10021	2298.00	10084	69.00	10143	128.00	10207	20.00	10265	1298.00
10022	2779.00	10085	94.00	10144	38.00	10208	30.00	10266	169.01
10023	3098.00	10086	10.00	10145	38.00	10209	90.00	10267	285.00
10024	2668.00	10087	20.00	10146	38.00	10210	338.00	10268	130.01
10025	2148.00	10088	35.00	10147	38.00	10211	20.00	10269	275.00
10026	2198.00	10089	109.00	10150	298.00	10212	65.00	10270	10.00
10027	4468.00	10090	20.00	10154	398.00	10213	95.00	10274	20.00
10028	4998.00	10092	39.00	10155	10.00	10214	50.00	*10275	
10029	5289.00	10093	20.00	*10156		10215	50.00	*10276	
10030	360.00	10094	59.00	10157	97.99	10216	50.00	10277	145.00
10031	480.00	10095	89.00	10158	248.00	10217	50.00	*10278	
10032	468.00	10096	3.80	10159	199.00	10218	6.00	*10279	
10033	698.00	10097	2.50	10160	248.00	10219	30.00	10280	97.99
10034	1225.00	10098	12.49	10161	248.00	10220	30.00	10281	97.99
10035	1618.00	10099	4.80	10162	248.00	10221	30.00	10282	97.99
10036	1118.00	10100	5.20	10163	399.00	10222	40.00	10283	97.99
10037	1448.01	10101	12.20	*10164		10223	35.00	10284	89.00
10038	2358.00	10102	8.99	10165	45.00	10224	35.00	10285	129.00
10039	2048.00	10103	8.99	10166	268.99	10225	248.00	10286	20.00
10040	3548.00	10104	24.80	10167	349.00	10226	348.00	10287	15.00
10041	4269.00	10105	19.00	10168	15.00	10227	20.00	10288	49.90
10042	4548.00	10106	6.90	10169	49.50	10228	50.00	10289	74.00
10043	3138.00	10107	69.00	10170	178.00	10229	30.00	10290	10.00
10044	3598.00	10108	49.00	10171	298.00	10230	30.00	10291	25.00
10045	3898.00	10109	97.00	10172	398.00	10231	50.00	10292	140.00
10046	2600.00	10110	65.00	10173	528.00	10232	50.00	10293	11.40
10047	6198.00	10111	30.00	10174	20.00	10233	30.00	10294	8.99
10048	7048.00	10112	159.00	10175	60.00	10234	30.00	10295	11.40
10049	7298.00	*10113		10176	999.00	10235	29.00	10296	19.00
10050	48.01	*10114		10177	20.00	10236	25.00	10297	11.40
10051	8.99	*10115		10178	439.00	10237	90.00	10298	44.00
10052	8.99	10116	169.01	10179	129.00	10238	90.00	10299	6.00
10053	8.99	*10117		10180	195.00	10239	1140.00	10300	38.00
10054	8.99	10118	24.00	10181	10.00	10240	20.00	10301	70.00
10055	8.99	10119	10.00	10182	30.00	10241	90.00	10302	48.01
10056	8.99	10120	40.00	10183	59.00	10242	35.00	10303	48.01
10057	11.99	10121	60.00	10184	65.99	10243	30.00	10304	35.00

GRAF ELEKTRONIK SYSTEME GMBH
Magnusstraße 13 · Postfach 1610 · 8960 Kempten (Allgäu) · Telefon: (08 31) 62 11
Teletex: 831804=GRAF · Telex: 17 831 804=GRAF · Datentelefon: (08 31) 6 93 30

GRAF computer

Bitte umseitig bestellen, abschneiden
und an uns absenden.
Paßt für Fensterkuvert!
Absender und Lieferform bitte nicht vergessen.

GRAF computer

Graf Elektronik Systeme GmbH
Postfach 1610

8960 Kempten

Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM
10305	69.00	10363	74.00	10421	348.00	10484	240.00	10542	20.00
10306	5.65	10364	198.00	10422	448.00	10485	298.00	10543	99.00
10307	20.00	10365	268.00	10423	29.00	10486	10.00	10544	97.99
10308	8.00	10366	20.00	10424	18.00	10487	49.50	10545	59.00
10309	12.00	10367	40.00	10425	12.00	10488	40.00	10546	1598.00
10310	49.00	10368	9.50	10426	12.00	10489	49.00	10547	600.00
10311	20.00	10369	45.00	10427	62.81	10490	25.00	*10548	
10312	30.00	10370	89.00	10428	29.50	10491	20.00	10549	39.00
10313	64.00	10371	15.00	10429	309.00	10492	20.00	10550	39.00
10314	119.00	10372	49.00	10434	78.00	10493	25.00	10551	49.00
10315	10.00	10373	94.00	10436	295.99	10494	20.00	10552	439.00
10316	20.00	10374	10.00	10437	118.00	10495	39.00	10553	638.00
10317	10.00	10375	30.00	10438	248.00	10496	39.00	10554	638.00
10318	20.00	10376	15.00	10439	295.99	10497	39.00	10555	39.00
10319	10.00	10377	298.00	10440	145.00	10498	598.00	10556	39.00
*10320		10378	89.00	10441	89.00	10499	198.00	*10557	
10321	6.00	10379	129.00	10442	99.00	10500	298.00	10558	439.00
*10322		10380	10.00	10443	1995.00	10501	65.00	10559	399.00
10323	59.00	10381	39.50	10444	11.99	10502	20.00	10560	439.00
10324	59.00	10382	59.00	*10445		10503	55.01	10561	400.00
*10325		10383	99.00	10446	11.99	10504	75.00	10562	50.00
10326	30.00	10384	10.00	10447	9.95	10505	20.00	10563	38.00
10327	199.00	10385	20.00	10448	49.00	10506	39.90	10564	185.00
10328	199.00	10386	169.01	10449	129.00	10507	76.00	10565	250.00
10329	1299.00	10387	258.00	10450	195.00	10508	120.00	10566	290.00
10330	785.00	10388	10.00	10451	9.80	10509	10.00	10567	290.00
10331	99.00	10389	50.00	10452	58.00	10510	38.00	10568	439.00
10332	59.00	10390	4.80	*10453		10511	147.99	10569	439.00
10333	79.00	10391	3.00	10454	398.00	10512	120.00	10570	147.00
10334	20.00	10392	7.50	10455	278.00	10513	20.00	10571	399.00
10335	225.00	10393	8.00	10456	378.00	10514	20.00	10573	39.00
10336	298.00	10394	99.00	10457	20.00	10515	198.00	10575	39.00
10337	598.00	10395	189.00	10458	49.00	10516	248.00	*10576	
10338	39.00	10396	10.00	10459	348.00	10517	20.00	*10577	
10339	99.00	10397	30.00	10460	448.00	10518	69.00	10578	8.60
10340	68.00	*10398		10461	20.00	10519	97.99	10579	79.00
10341	97.99	10399	28.00	10462	49.00	10520	26.00	10580	58.00
10342	20.00	10400	25.00	10463	348.00	10521	245.00	10581	42.00
10343	30.00	10401	198.00	10464	450.00	10522	97.99	10582	59.00
10344	50.00	10402	258.00	10465	20.00	10523	245.00	10583	48.00
10345	10.00	10403	10.00	10466	50.00	10524	20.00	10584	48.00
10346	15.00	10404	39.50	10467	20.00	10525	147.99	10585	58.00
10347	149.00	10405	1.50	10468	450.00	10526	198.00	10586	48.00
10348	99.00	10406	3.00	10469	97.00	10527	68.99	10587	38.00
10349	179.00	10407	4.50	10470	248.00	10528	20.00	10588	78.00
10350	10.00	10408	3.90	10471	154.99	10529	598.00	10589	48.00
10351	20.00	10409	3.90	10472	250.00	10530	298.00	10590	48.00
10352	19.00	10410	99.00	10473	16.01	10531	398.00	10591	68.00
10353	40.00	10411	159.00	10474	20.00	10532	398.00	10592	68.00
10354	10.00	10412	175.00	10475	298.00	10533	498.00	10593	59.00
*10355		10413	56.00	10476	20.00	10534	958.00	10594	39.00
10356	30.40	10414	82.65	10477	398.00	10535	1028.00	10595	38.00
10357	8.90	10415	97.99	10478	528.00	10536	248.00	10596	38.00
10358	298.00	10416	97.99	10479	95.00	10537	20.00	10597	35.99
10359	97.99	10417	97.99	10480	69.00	10538	99.00	10598	58.00
10360	10.00	10418	10.00	10481	15.00	10539	97.99	10599	58.00
10361	50.00	10419	299.00	10482	330.00	10540	20.00	10600	54.00
10362	175.00	10420	10.00	10483	4500.00	10541	20.00	*10601	

BESTELLKARTE

Ich / Wir bestelle(n) unter Anerkennung Ihrer Geschäfts- und Lieferungsbedingungen folgende Artikel:

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10834	GES-Katalog	10,-
	10061	LOOP-Abo	20,-

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

Datum

Anschrift:

Lieferform: ☐ Nachnahme ☐ V
☐ Bankeinzug

Bankeinzug: Hiermit ermächtige ich die Fir
GmbH, den Rechnungsbetrag für die auf dies
angegebenen Bestellungen von meinem Ko

BLZ _____ Konto-Nr. _____

Bank: _____

abzubuchen. Falls mein Konto die erfor
Deckung nicht aufweist, besteht seitens de
führenden Kreditinstitutes keine Verpflicht
Einlösung.

Datum Unterschrift

Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM	Art. Nr.	Bruttopreis DM
10892	49.00	10950	130.01	11008	2390.00	11066	398.00	11111	50.00
10893	12.00	10951	186.00	11009	25.00	11067	14.90	*11112	9.50
10894	39.00	10952	35.00	11010	298.00	11068	28.00	11113	14.00
10895	39.00	10953	45.00	11011	6.00	11069	74.00	11114	8.00
10896	50.00	*10954		11012	430.00	*11070	399.00	*11115	Anfrage
10897	298.00	10955	97.99	11013	325.00	11071	258.00	11116	1198.00
10898	298.00	10956	1155.00	11014	138.00	11072	220.00	*11117	138.00
10899	298.00	10957	215.00	11015	598.00	11073	299.00	11118	1198.00
10900	298.00	10958	393.00	11016	248.00	11074	30.00	11119	678.30
10901	498.00	10959	652.00	11017	159.00	11075	147.99	11120	1849.00
10902	498.00	10960	198.00	11018	25.00	11076	2748.00	11121	2.50
10903	498.00	10961	1098.00	*11019		11077	3748.00	*11122	40.00
10904	498.00	10962	1180.00	*11020	49.00	11078	4498.00	*11123	
10905	598.00	10963	59.00	*11021	49.00	11079	20.00	*11124	
10906	598.00	10964	698.00	11022	49.00	11080	49.00	*11125	20.00
10907	598.00	10965	49.00	11023	49.00	11081	49.00		
10908	598.00	10966	49.00	11024	49.00	11082	49.00		
10909	94.00	10967	50.00	11025	49.00	11083	3.00		
10910	20.00	10968	3.00	11026	97.99	11084	378.00		
10911	1028.00	10969	20.00	11027	158.00	11085	248.00		
10912	48.01	10970	40.00	11028	20.00	11086	159.00		
10913	3.00	10971	55.01	11029	40.00	11087	159.00		
10914	2.05	10972	15.00	11030	147.99	11088	3876.91		
10915	20.00	10973	4.10	11031	198.00	11089	3542.00		
*10916		10974	6.50	11032	75.00	11090	68.00		
10917	147.99	10975	40.00	11033	75.00	11091	96.90		
10918	158.00	10976	80.01	11034	75.00	11092	96.90		
10919	258.00	*10977		11035	50.00	11093	97.99		
10920	258.00	10978	610.00	11036	50.00	*11094			
10921	358.01	10979	1995.00	*11037		11095	75.00		
10922	358.01	10980	410.97	11038	79.00	11096	20.00		
10923	40.00	10981	285.00	11039	198.00	11097	159.60		
10924	40.00	10982	75.00	11040	698.00	11098	960.01		
10925	20.00	10983	21.00	11041	10.00	11099	1085.00		
10926	97.00	10984	33.00	11042	298.00	11100	20.00		
10927	97.00	10985	199.00	11043	467.40	11101	49.00		
10928	159.00	10986	249.00	11044	1347.48	11102	49.00		
10929	159.00	10987	199.00	11045	2198.00	11103	97.99		
10930	248.00	10988	249.00	11046	49.00	11104	20.00		
10931	249.00	10989	89.00	11047	49.00	11105	12.00		
10932	73.01	10990	40.99	11048	49.00	11106	39.00		
10933	97.99	10991	68.00	11049	49.00	11107	15.00		
10934	24.80	10992	147.99	11050	10.00	11108	15.00		
10935	1190.00	10993	50.00	11051	60.00	11109	65.00		
10936	75.00	10994	109.00	11052	60.00	11110	65.00		
10937	125.00	10995	55.01	11053	60.00				
10938	179.99	10996	6793.01	*11054	498.00				
10939	20.00	10997	6823.90	11055	69.00				
10940	125.00	10998	6823.90	11056	69.00				
10941	179.99	10999	4028.66	11057	95.00				
10942	398.00	11000	6356.93	11058	248.00				
10943	358.01	11001	20.00	11059	248.00				
10944	358.01	11002	3.00	11060	97.99				
10945	49.00	11003	48.00	11061	96.90				
10946	49.00	11004	152.00	11062	96.90				
10947	40.00	11005	190.00	11063	110.00				
10948	55.01	11006	152.00	11064	110.00				
10949	75.00	11007	348.00	11065	298.00				

Artikelgruppe 2

20000	1117.20
20001	20.52
20002	38.76
20003	59.28
20004	79.80
20005	100.32
20006	20.52
20007	38.76
20008	59.28
20009	79.80
20010	100.32
20011	558.60
20012	433.20
*20013	
20014	4463.10
20015	998.00
20016	1054.00
20017	1998.00
20018	2054.00
*20019	
20020	114.00
20021	2054.00

* Die mit * gekennzeichneten Art.-Nr. **ohne** Preisangabe sind nicht mehr im Lieferprogramm.

Die mit * gekennzeichneten Art.-Nr. **mit** Preisangabe sind noch nicht lieferbar.

GRAF® computer

GRAF ELEKTRONIK SYSTEME GMBH

Magnusstraße 13 · Postfach 1610 · 8960 Kempten (Allgäu) · Telefon: (08 31) 62 11
Telefax: 831804=GRAF · Telex: 17 831 804=GRAF · Datentelefon: (08 31) 69 33 0

BESTELLKARTE

Ich / Wir bestelle(n) unter Anerkennung Ihrer Geschäfts- und Lieferungsbedingungen folgende Artikel:

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10834	GES-Katalog	10,-
	10061	LOOP-Abo	20,-

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

Datum

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vor
☐ Bankeinzug

Bankeinzug: Hiermit ermächtige ich die Firm GmbH, den Rechnungsbetrag für die auf diese angegebenen Bestellungen von meinem Kon

BLZ _____ Konto-Nr. _____

Bank: _____

abzubuchen. Falls mein Konto die erforde Deckung nicht aufweist, besteht seitens des führenden Kreditinstitutes keine Verpflichtu Einlösung.

Datum Unterschrift

mehrere Möglichkeiten, er könnte zum Beispiel eine Liste führen, in der alle Kugeln stehen, die noch nicht gezogen wurden. Oder er benutzt eine Tabelle, in der für jede Kugel steht, ob sie schon gezogen wurde. Nach diesem lezteren Verfahren arbeitet das Programm LOTTO.

Da für diese Methode bereits zu Beginn des Programmablaufes Vorbereitungen getroffen werden müssen, kehren wir also zurück an den Anfang des Programmes und erklären alles schön der Reihe nach (wie es sich schließlich für eine anständige Dokumentation gehört). Beim Start des Programms muß zunächst die oben erwähnte Tabelle erstellt werden. Das geschieht in der Schleife clrLoop. Zunächst wird aber die Anzahl der Kugeln in das Register b geladen, das wieder als Schleifenzähler dient. In das Registerpaar HL kommt die letzte Adresse, die noch zur Tabelle gehört, denn in clrLoop wird die Tabelle von hinten nach vorne erzeugt. Dazu wird in den Speicherplatz, dessen Adresse in HL steht, der Inhalt von b geschrieben. Dazu dient der Befehl ld (hl),b. HL enthält hier eine Adresse, man nennt das dann Zeiger (oder neudeutsch "pointer"), HL zeigt auf die Stelle in der Tabelle, an die die nächste Zahl kommt. Da die Tabelle von hohen zu niedrigen Adressen erstellt wird, muß HL nun um eins vermindert (dekrementiert) werden, das übernimmt der Befehl dec hl. Danach sorgt der djnz Befehl wieder dafür, daß die Schleife noch einmal durchlaufen wird, außer wenn b bereits den Wert Null erreicht hat. Anschließend steht an der Adresse store+1 eine 1, bei store+2 steht eine 2, bei store+10 steht eine 10 (0A sedezimal) u.s.w. Später soll überall dort, wo eine Kugel gezogen wurde eine Null stehen, damit das Unterprogramm random erkennen kann, ob die Zufallszahl, die es ermittelt hat noch vorhanden ist, oder ob eine andere ermittelt werden muß.

Nach dem die Tabelle eingerichtet ist folgt das Hauptprogramm. In einer Scheife, die maximal 7 mal durchlaufen wird, ruft das Hauptprogramm zunächst das Unterprogramm PRINT auf, das den Text "Lotto" zur Anzeige bereitstellt. Danach wird im Unterprogramm random die Zufallszahl generiert und auf Zulässigkeit hin überprüft. Wie wird's gemacht?

Dazu wird die Anfangsadresse der Tabelle, also die Adresse store, in das Registerpaar DE geladen (DE zeigt auf die Tabelle, HL ist der Index in diese Tabelle, also die laufende Nummer des Eintrags). Addiert man nun HL und DE, so erhält man die Adresse des Bytes, in dem eigentlich die Zufallszahl stehen müßte, das geschieht mit dem Befehl add hl,de.

HL zeigt jetzt also auf das fragliche Byte in der Tabelle. Dann vergleicht das Programm dieses Byte (also das Byte, dessen Adresse in HL steht) mit dem Akku (in dem immer noch die Zufallszahl steht): cp (hl). Ist der Unterschied Null, dann war das fragliche Byte dasselbe wie die Zufallszahl im Akku, das bedeutet, daß diese Kugel noch nicht gezogen wurde. Andernfalls, also wenn der Unterschied nicht Null ist, wurde sie schon einmal gezogen und der Befehl jr nz,random (Jump Relative if Not Zero) springt noch einmal an den Anfang des Unterprogramms, um eine neue Zahl zu erzeugen. (Das ist nicht unbedingt geschickt, aber da das Refresh-Register sich auch für den Mikroprozessor zu schnell ändert, kommt beim nächsten Versuch wahrscheinlich eine andere Zahl heraus. Praktisch merkt man nachher nicht, daß das Programm für einige Zahlen länger braucht als für andere.) Wenn nun (evtl. nach mehreren Versuchen) eine Zahl gezogen wurde, die noch nicht vergeben ist, muß diese Zahl jetzt als benutzt markiert werden. Dazu dient der Befehl ld (hl),0. Er lädt das Byte, dessen Adresse in HL steht mit dem Wert 0. Da das Hauptprogramm als Ergebnis des Unterprogramms eine Zahl im Registerpaar HL erwartet, nicht die Adresse, die jetzt noch dort steht, muß von HL wieder die Anfangsadresse der Tabelle, die immer noch in DE steht, abgezogen werden, dann kann die Rückkehr ins Hauptprogramm erfolgen. Leider gibt es den Befehl sub hl,de beim Z80 nicht, man darf hier aber auch sbc hl,de (SuBtract with Carry) verwenden, da an dieser Stelle im Programm kein Übertrag (Carry) vorliegen kann.

Abschliessend wird ins' Hauptprogramm zurückgesprungen (mit dem Befehl ret). Nach dem Befehl call HoleTaste folgt jetzt ein cp 57H. Warum das?
Nun, das Unterprogramm HoleTaste des HEXMON wartet bis eine Taste gedrückt wird und legt dann den Code dieser Taste im Akkumulator ab. Der Befehl cp (compare) vergleicht diesen Wert mit dem Code der CR Taste (57 sedezimal). Wurde die CR-Taste gedrückt, dann ist der Unterschied zum Vergleichswert Null. Deshalb springt der Befehl jr nz,ENDE (Jump Relative if Not Zero) zur Adresse ENDE wenn eine andere Taste als CR gedrückt wurde. Man kann dieses Programm also durch drücken einer beliebigen Taste abbrechen oder mit CR weitermachen.

Starten Sie dieses Programm mit dem START-Befehl, so erscheint die Schrift "Lotto" und die erste gezogene Zahl. Jetzt können Sie sich diese Zahl notieren und CR drücken um die nächste Zahl zu bekommen, oder Sie drücken irgendeine andere Taste, dann erlischt die Anzeige und nach einem kurzem Moment erscheint das HALLO-1.1. Das passiert auch, wenn Sie alle sieben Zahlen gezogen haben.

Lassen Sie sich jetzt einmal alle Zahlen anzeigen, das heißt Sie starten das Programm und drücken immer wieder CR bis das HALLO-1.1 erscheint. Nun können Sie sich die vom Programm angelegte Tabelle des Programms "angucken". Drücken Sie dazu die Taste SPE und geben als Adresse den sedezimalen Wert von store (=80D9 sedezimal) an. Dies ist der Eintrag für die Kugel Null, da es aber keine Null gibt, steht hier nichts sinnvolles. Also drücken Sie + oder CR um zum nächsten Speicherplatz zu kommen. Dort erscheint nun 01 oder 00, je nachdem ob die Zahl 1 gezogen wurde oder nicht. Schreiben Sie diesen und alle folgenden Speicherplätze auf, so erhalten Sie zum Beispiel die Folge:

01 02 03 04 05 06 07 00 00 0A 0B 0C 00 0E 0F 10 11 12 13 14 15 16 17 18 19 00 1B 1C 1D 1E 1F 00 00 22 00 24 25 26 und hier ist die Tabelle zu Ende.
Da überall dort, wo eine Null steht eine Zahl gezogen wurde, muß das Programm folgende Zahlen ermittelt haben (sedezimal): 08 09 0D 1A 20 21 23.
Tatsächlich hatte es diese Zahlen in folgender Reihenfolge ausgegeben:
9 26 33 35 32 8 13 (dezimal).

Vielleicht benutzen Sie den hier vorgestellten Zufallszahlengenerator einmal als Grundlage für ein selbstentworfenes, kleines Würfelspiel ?

```
                ; Programm  LOTTO-1     1.0
                ;
                ; Dies Programm ermittelt per Zuffalszahlengenerator die Lottozahlen für
                ; die nächste Ziehung, leider arbeitet der Algorithmus noch nicht fehlerfrei,
                ; so daß die "Sechs Richtigen" nicht garantiert werden können.
                .Z80
                Ziehungen  EQU  6+1     ; Wieviel Kugeln werden gezogen
                Kugeln     EQU  38   ; aus wieviel Kugeln wird gezogen

                HoleTaste  EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                Print      EQU  0015H  ; Kopiert einen Text in die Anzeige
                PrtDez     EQU  0021H  ; Druckt eine Dezimalzahl
                Anzeige    EQU  8000H  ; Adresse des Anzeige-Speichers

                store      EQU  80FFH-Kugeln  ; Freier RAM-Speicher für eine Tabelle der Zahlen

                           ORG 1000H

                LOTTO:

                ; Zunächst wird eine Tabelle angelegt, in der steht, welche Zahlen noch nicht
                ; gezogen wurden, es werden praktisch alle Kugeln in die Ziehungsmaschine
                ; gefüllt. Diese Tabelle beginnt bei der Adresse store, sie muß im RAM stehen.

1000  06 26                ld b,Kugeln     ; wieviel Kugeln gibt es
1002  21 FF 80             ld hl,store+Kugeln  ; letztes Byte der Tabelle für noch vorhandene
                clrLoop:                   ; Zahlen, die hier angelegt wird
1005  70                   ld (hl),b       ; Zahl eintragen
1006  2B                   dec hl          ; Adresse herunterzählen
1007  10 FC                djnz clrLoop    ; für jede Kugel wiederholen

                ; Jetzt werden die Ziehungen durchgeführt.

1009  06 07                ld b,Ziehungen  ; dazu muß die Anzahl in Register b stehen
                loop:
100B  C5                   push bc         ; Zähler für die Wiederholung retten
100C  21 43 10             ld hl,text      ; Adresse des Textes in das Registerpaar HL laden
100F  CD 15 00             call Print      ; Unterprogramm zur Textausgabe aufrufen
1012  CD 29 10             call random     ; Unterprogramm zum würfeln einer Zahl
1015  DD 21 07 80          ld ix,Anzeige+7 ; Hier kommt die Zahl hin
1019  CD 21 00             call PrtDez     ; Gewürfelte Zahl ausgeben
101C  CD 0C 00             call HoleTaste  ; Auf Tastendruck warten
                                           ; Hier steht im Akku der Tastencode
101F  FE 57                cp 57H          ; Taste CR gedrückt ?
1021  20 03                jr  nz,ENDE     ; wenn nicht Programm beenden
1023  C1                   pop bc          ; Schleifenzähler wiederherstellen
1024  10 E5                djnz loop       ; b herunterzählen und springen falls nicht null
                ENDE:
1026  C3 00 00             jp  0000H       ; Zurück in den HEXMON springen,
                                           ; nach kurzer Zeit erscheint wieder das HALLO-1.1

                ; Soweit das Hauptprogramm,
                ; jetzt kommt nur noch das Unterprogramm zum Ziehen einer Zahl.

                random:                    ; Würfel eine Zahl von 1 bis Kugeln
1029  ED 5F                ld a,r          ; lade den Refresh-Wert in den Akku, dieser Wert ist
                                           ; zufällig genug
102B  FE 26     modulo:    cp Kugeln       ; Ist der Wert kleiner als Kugeln ?
102D  38 04                jr c,endrandom  ; wenn ja fertig
102F  D6 26                sub Kugeln      ; sonst Kugeln vom Wert abziehen
1031  18 FB                jr modulo       ; und noch einmal testen
                endrandom:
1033  3C                   inc a           ; der Wert liegt zwischen 0 und Kugeln-1
                                           ; er soll aber zwischen 1 und Kugeln liegen, also
                                           ; wird er um eins erhöht (inkrementiert)
1034  6F                   ld l,a          ; das Ergebnis soll im Registerpaar HL stehen
1035  26 00                ld h,00h        ; das obere Byte ist null
1037  11 D9 80             ld de,store     ; Anfangsadresse der Tabelle in Registerpaar de laden
103A  19                   add hl,de       ; Adresse in der Tabelle berechnen
103B  BE                   cp (hl)         ; Vergleich die Zahl in der Tabelle mit der Zahl im
                                           ; Akkumulator, dort steht immer noch die gezogene
                                           ; Kugel, in der Tabelle steht entweder Null, oder
                                           ; auch die gezogene Kugel
103C  20 EB                jr  nz,random   ; standen verschiedene Zahlen in Akku und Tabelle,
                                           ; dann wurde diese Zahl schon einmal gezogen.

                ; Jetzt wurde eine neue Zahl gezogen

103E  36 00                ld (hl),0       ; Diese Zahl löschen, damit sie nicht noch einmal
                                           ; gezogen werden kann
1040  ED 52                sbc hl,de       ; Adresse der Tabelle wieder abziehen, damit die
                                           ; gezogene Zahl in HL steht
1042  C9                   ret             ; zurück ins Hauptprogramm

                text:
1043  C7 A3 87 87 A3 FF FF FF   DB  0C7h, 0a3h, 87h, 87h, 0a3h, 0ffh, 0ffh, 0ffh
                                           ; hier steht  L o t t o
                END
```

RUBACODE-Programm –

zum Ver- und Entschlüsseln von Dateien, von Rüdiger Bäcker.

Überall dort, wo Daten elektronisch gespeichert werden, gewinnt der Datenschutz zunehmend an Bedeutung. Eine Art, die Daten vor unerlaubtem Zugriff zu schützen, ist es, sie zu verschlüsseln. Dazu gibt es diverse Methoden, eine soll hier beschrieben werden.

Das Programm RUBACODE kann eine Datei unter JADOS ver- und entschlüsseln. Dabei ist es egal, ob es sich bei der Datei um ein ASCII-File, ein Programm oder um Daten handelt.

Eine einfache Art der Codierung ist das Aufaddieren bzw. Subtrahieren eines Wertes auf die einzelnen Bytes. Dabei ist ein Problem dadurch gegeben, daß, wenn immer der gleiche Wert benutzt wird, u.U. eine Regelmäßigkeit erkannt wird und so der Code „geknackt" werden kann.

Dazu ein Beispiel, wird beim nachstehenden Wort immer der gleiche Wert (im Beispiel 1) addiert, so sieht das folgendermaßen aus:

```
Wort vor der Verschluesselung :   W E R K S E I S E N B A H N
                                  + + + + + + + + + + + + + +
                                  1 1 1 1 1 1 1 1 1 1 1 1 1 1
                                  = = = = = = = = = = = = = =
Wort nach der Verschluesselung :  X F S L T F J T F O C B I D
```

Daran ist zu erkennen, daß gleiche Buchstaben auch nach der Verschlüsselung stets gleich sind. Anders sieht das aus, wenn ein variabler Wert hinzuaddiert wird. Dies kann zum Beispiel ein zweites Wort sein. Auch dazu ein Beispiel:

```
Wort vor der Verschluesselung :   W E R K S E I S E N B A H N
                                  + + + + + + + + + + + + + +
Schluesselwort (ADELIG) :         A D E L I G A D E L I G A D
                                  = = = = = = = = = = = = = =
Wort nach der Verschluesselung :  X I W W B L J W J Y K H I R
```

Deutlich zu erkennen ist, daß nun gleiche Buchstaben nicht mehr das gleiche Ergebnis im verschlüsselten Wort ergibt. Dabei ist noch nicht berücksichtigt, daß das Codewort kleine und große Buchstaben enthalten kann. Man kann eine verschlüsselte Datei auch mehrfach hintereinander mit verschiedenen Worten verschlüsseln. So ist eine sehr hohe Sicherheit erreichbar. Das Programm RUBACODE erlaubt die Verwendung eines Schlüsselwortes mit bis zu 32 Buchstaben.

Nach dem Aufruf des Programmes erscheint ein Startmenue. Es muß nun der Name der zu verschlüsselnden Datei eingegeben werden. Dabei ist auch die Dateiextension mit anzugeben (z.B. TEST.TXT). Als zweites wird das Schlüsselwort eingegeben. Dabei ist vor Bestätigen mit CR in jedem Fall zu prüfen, daß man das Wort richtig geschrieben hat. Ist dies nicht der Fall, kann die Datei nicht mehr entschlüsselt werden!
Sind diese Eingaben erfolgt, so muß man zwischen ver- und entschlüsseln wählen oder das Programm mit 'm' beenden.

Nach Ausführung der Funktion gelangt man zur JADOS Kommandoebene zurück.

```
14.09.87-17.08 Listing des Files : 1:RUBACODE.S
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

020000                  *       R U B A C O D E
020000                  *
020000                  *       PROGRAMM ZUM VER- & ENTSCHLUESSELN VON DATEIEN UNTER JADOS
020000                  *
020000                  *       COPYRIGHT (C) 1987 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
020000                  *
020000                  *       V 1.0 - 14.09.87 - FREIGABE FUER LDOP
020000
020000                  ******
020000                  *** VEREINBARUNGEN
020000                  ******
020000
020000                  *** JADOS-FUNKTIONEN
020000
= 0000002C              J_FILELOAD      EQU     44      * DATEI LESEN
= 0000002D              J_FILESAVE      EQU     45      * DATEI SPEICHERN
= 00000007              J_SCHREIBE      EQU      7      * TEXT AUF CO2
= 00000012              J_FILLFCB       EQU     18      * FCB ERZEUGEN
= 00000001              J_GETNAME       EQU      1      * FILENAME VON CONSOLE HOLEN
= 00000003              J_LESE          EQU      3      * TEXT VON CONSOLE LESEN
= 00000029              J_CI            EQU     41      * TASTATUREINGABE HOLEN
020000
020000                  *** GRUNDPROGRAMM-FUNKTIONEN
020000
= 00000014              G_CLRSCREEN     EQU     20      * BILDSCHIRM LOESCHEN
= 00000021              G_CO2           EQU     33      * ZEICHEN AUF BILDSCHIRM
020000
020000                  ******
020000                  *** HAUPTPROGRAMM
020000                  ******
020000
020000                  START:
020000  7E14            MOVEQ #G_CLRSCREEN,D7           * BILDSCHIRM LOESCHEN ZU ERST
020002  4E41            TRAP #1
020004  41FA 0146       LEA HEADER(PC),A0
020008  7E07            MOVEQ #J_SCHREIBE,D7
02000A  4E46            TRAP #6
02000C  41FA 026F       LEA GETTXT(PC),A0               * AUFFORDERUNGSTEXT NACH A0
020010  43FA 028A       LEA FCB(PC),A1                  * ADRESSE DES FCB NACH A1
020014  7E01            MOVEQ #J_GETNAME,D7             * NUN NAMEN VON CONSOLE HOLEN UND IN FCB
020016  4E46            TRAP #6                         * EINTRAGEN
020018  0C00 0000       CMPI.B #0,D0                    * TESTEN, OB FEHLER
02001C  6600 00FB       BNE ERROR                       * JA, DANN FEHLERBEHANDLUNG
020020  41FA 02CC       LEA LADEADR(PC),A0              * ADRESSE, AN DIE DATEI GELADEN WIRD IN A0
020024  7E2C            MOVEQ #J_FILELOAD,D7            * ADRESSE DES FCB STEHT NOCH IN A1
020026  4E46            TRAP #6                         * DANN LADEN
020028  0C00 0000       CMPI.B #0,D0                    * WIEDER TESTEN, OB FEHLER
02002C  6600 00EB       BNE ERROR                       * AUCH DANN WIEDER FEHLERBEHANDLUNG EINLEITEN
020030  41FA 021B       LEA SCHLTXT(PC),A0              * AUFFORDERUNGSTEXT ZUR SCHLUESSELWORT-EINGABE
020034  7E07            MOVEQ #J_SCHREIBE,D7            * AUSGEBEN
020036  4E46            TRAP #6
020038  43FA 0292       LEA INBUF(PC),A1                * ADRESSE DES EINGABEPUFFERS IN A1
02003C  343C 0020       MOVE #32,D2                     * MAXIMALE ZEICHENZAHL IN D2
020040  7E03            MOVEQ #J_LESE,D7                * UND DANN SCHLUESSELWORT EINLESEN
020042  4E46            TRAP #6
020044  0C00 0000       CMPI.B #0,D0                    * AUCH HIER WIEDER TESTEN, OB FEHLER
020048  6600 00CC       BNE ERROR                       * JA, DANN FEHLERBEHANDLUNG
02004C  6100 006A       BSR WORTLENG                    * LAENGE DES SCHLUESSELWORTES ERMITTELN
020050  3B00            MOVE D0,D4                      * D4 IST LAENGE DES SCHLUESSELWORTES
020052  6100 0024       BSR AUSWAHL                     * CODEIEREN ODER DECODIEREN
020056  0C40 FFFF       CMPI #$FFFF,D0
02005A  671A            BEQ.S ST1
02005C  6100 004C       BSR FILELENG
```

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

020060  5341            SUBQ #1,D1                      * UND -1 FUER SPEICHERN
020062  41FA 028A       LEA LADEADR(PC),A0              * VON DORT SPEICHERN
020066  43FA 0234       LEA FCB(PC),A1                  * ADRESSE DES FCB
02006A  7E2D            MOVEQ #J_FILESAVE,D7            * UND DANN VERSCHLUESSELTE DATEI ZURUECK-
02006C  4E46            TRAP #6                         * SPEICHERN
02006E  0C00 0000       CMPI.B #0,D0
020072  6600 00A2       BNE ERROR
020076
020076                  ST1:                            * ABBRUCH MIT 'm' BEI AUSWAHL
020076  4E75            RTS
02007A
02007A                  ******
02007A                  *** UNTERPROGRAMME
02007A                  ******
02007A
02007A                  *** FUNKTIONSAUSWAHL
02007A
02007A                  AUSWAHL:                        * WAEHLEN, OB CODIEREN ODER DECODIEREN
02007A  41FA 016A       LEA WAHLTXT(PC),A0              * TEXT LADEN
02007C  7E07            MOVEQ #J_SCHREIBE,D7
02007E  4E46            TRAP #6
020080  7E29            MOVEQ #J_CI,D7
020082  4E46            TRAP #6
020084  7E21            MOVEQ #G_CO2,D7
020086  4E41            TRAP #1
020088  0C00 0031       CMPI.B #'1',D0
02008C  6700 003B       BEQ CODE
020090  0C00 0032       CMPI.B #'2',D0
020094  6700 005B       BEQ DECODE
020098  0C00 006D       CMPI.B #'m',D0
02009C  6700 0006       BEQ ENDE
0200A0  6000 FFD6       BRA AUSWAHL
0200A4
0200A4                  *** PROGRAMMABBRUCH
0200A4
0200A4                  ENDE:                           * ABBRUCH
0200A4  303C FFFF       MOVE #$FFFF,D0                  * FALSE
0200A8  4E75            RTS                             * RUECKSPRUNG IN HAUPTPROGRAMM
0200AA
0200AA                  *** LAENGE DES FILES ERMITTELN
0200AA
0200AA                  FILELENG:                       * LAENGE DES FILES ERMITTELN
0200AA  43FA 01F0       LEA FCB(PC),A1                  * ADRESSE DES FCB IN A1
0200AE  D3FC 0000001A   ADDA.L #26,A1                   * DORT STEHT LAENGE DER DATEI
0200B4  3211            MOVE (A1),D1                    * DANN DATEILAENGE IN D1
0200B6  4E75            RTS
0200B8
0200B8                  *** LAENGE DES SCHLUESSELWORTES ERMITTELN
0200B8
0200B8                  WORTLENG:                       * LAENGE DES SCHLUESSELWORTES ERMITTELN
0200B8  9040            SUB D0,D0                       * D0 LOESCHEN
0200BA
0200BA                  GLLP:                           * SCHLEIFE
0200BA  5240            ADDQ #1,D0                      * LAENGE = LAENGE +1
0200BC  0C19 0000       CMPI.B #0,(A1)+                 * 0 = ENDEKENNUNG,
0200C0  6600 FFFB       BNE GLLP                        * WENN NICHT, DANN WEITER
0200C4  4E75            RTS                             * NUN STEHT LAENGE IN D0
0200C6
0200C6                  *** FILE VERSCHLUESSELN
0200C6
0200C6                  CODE:                           * DATEI VERSCHLUESSELN
0200C6  6100 FFE2       BSR FILELENG                    * NUN LAENGE DES FILES ERMITTELN, STEHT DANN
0200CA  C2FC 0400       MULU #1024,D1                   * IN D1, IN BYTE
0200CE  5344            SUBQ #1,D4                      * FUER DBRA
```

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

0200D0
0200D0  3604          MOVE D4,D3
0200D2  41FA 021A     LEA LADEADR(PC),A0
0200D6
0200D6                CL1:                      * SCHLEIFE BIS DATEILAENGE
0200D6  3803          MOVE D3,D4                * ZAEHLER LADEN
0200DA  43FA 01F2     LEA INBUF(PC),A1          * DORT STEHT SCHLUESSELWORT
0200DC
0200DC                CL2:                      * SCHLEIFE BIS WORTENDE
0200DC  1019          MOVE.B (A1)+,D0           * EIN BYTE DER DATEI HOLEN
0200DE  D118          ADD.B D0,(A0)+            * BUCHSTABE DES SCHLUESSEL ADDIEREN
0200E0  5341          SUBQ #1,D1                * TESTEN, OB DATEI ZU ENDE
0200E2  6700 000B     BEQ CL3                   * JA, DANN RAUSSPRINGEN
0200E6  51CC FFF4     DBRA D4,CL2               * UND WIEDERHOLEN BIS ENDE SCHLUESSELWORT
0200EA  60EA          BRA.S CL1                 * WIEDERHOLEN BIS DATEIENDE
0200EC
0200EC                CL3:
0200EC  4E75          RTS
0200EE
0200EE                *** FILE ENTSCHLUESSELN
0200EE
0200EE                DECODE:                   * DATEI VERSCHLUESSELN
0200EE  6100 FFBA     BSR FILELENG              * NUN LAENGE DES FILES ERMITTELN, STEHT DANN
0200F2  C2FC 0400     MULU #1024,D1             * IN D1, IN BYTE
0200F6  5344          SUBQ #1,D4                * FUER DBRA
0200F8
0200F8  3604          MOVE D4,D3
0200FA  41FA 01F2     LEA LADEADR(PC),A0
0200FE
0200FE                DL1:                      * SCHLEIFE BIS DATEILAENGE
0200FE  3803          MOVE D3,D4                * ZAEHLER LADEN
020100  43FA 01CA     LEA INBUF(PC),A1          * DORT STEHT SCHLUESSELWORT
020104
020104                DL2:                      * SCHLEIFE BIS WORTENDE
020104  1019          MOVE.B (A1)+,D0           * EIN BYTE DER DATEI HOLEN
020106  9118          SUB.B D0,(A0)+            * BUCHSTABE DES SCHLUESSEL ADDIEREN
020108  5341          SUBQ #1,D1                * TESTEN, OB DATEI ZU ENDE
02010A  6700 000B     BEQ DL3                   * JA, DANN RAUSSPRINGEN
02010E  51CC FFF4     DBRA D4,DL2               * UND WIEDERHOLEN BIS ENDE SCHLUESSELWORT
020112  60EA          BRA.S DL1                 * WIEDERHOLEN BIS DATEIENDE
020114
020114                DL3:
020114  4E75          RTS
020116
020116                *** FEHLERBEHANDLUNG
020116
020116                ERROR:                    * FEHLERBEHANDLUNG (EINFACH REV. 1.0)
020116  41FA 000B     LEA ERRORTXT(PC),A0       * NUR FEHLERMELDUNG
02011A  7E07          MOVEQ #J_SCHREIBE,D7      * AUSGEBEN UND
02011C  4E46          TRAP #6
02011E  4E75          RTS                       * DANN ABBRUCH
020120
020120                +++++++
020120                *** PUFFERSPEICHER UND TEXTABLAGEGEBIET
020120                +++++++
020120
020120                *** TEXTABLAGEGEBIET
020120
020120                DS 0
020120                ERRORTXT:
020120  46656B6C657220  DC.B 'Fehler aufgetreten ---> A B B R U C H  ! !',0
020127  617566676574 72
02012E  6574656E202D2D
020135  2D3E2041204220

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 4

02013C  42205220552043
020143  204B2020212021
02014A  00
02014B
02014B  00            DS 0
02014C                HEADER:
02014C  52205520422041  DC.B 'R U B A C O D E - Programm zum codieren/decodieren von Files unter JADOS'
020153  2043204F204420
02015A  452020205072 6F
020161  6772616D6D207A
020168  756D20636F6469
02016F  657265 6E2F6465
020176  636F6469657265
02017D  6E20766F6E2046
020184  696C657320756E
02018B  746572204A4144
020192  4F53
020194  0D 0A 0A      DC.B 13,10,10
020197  434F5059524947  DC.B 'COPYRIGHT (C) 1987 by Ruediger Baecker - Postfach 4111 - 5820 Gevelsberg'
02019E  485420284320
0201A5  313938372062 79
0201AC  205275656461 67
0201B3  657220426165 63
0201BA  6B657220D2050
0201C1  6F737466616368
0201C8  2034313131202D
0201CF  203538323020 47
0201D6  6576656C736265
0201DD  7267
0201DF  0D 0A 0A 0A 00  DC.B 13,10,10,10,0
0201E4
0201E4                DS 0
0201E4                WAHLTXT:
0201E4  0D 0A 0A 0A   DC.B 13,10,10,10
0201E8  426974746520 77  DC.B 'Bitte waehlen Sie :',13,10,10
0201EF  61656B6C656E20
0201F6  536965203A0D
0201FC  0A 0A
0201FE  31203D3D3E2046  DC.B '1 ==> File codieren',13,10
020205  696C6520636F64
020206  696572656E0D
020212  0A
020213  32203D3D3E2046  DC.B '2 ==> File dekodieren',13,10
02021A  696C652064656B
020221  6F6469657265 6E
02022B  0D 0A
02022A  6D203D3D3E207A  DC.B 'm ==> zurueck zu JADOS',13,10,10
020231  757275656636B20
02023B  7A75204A41444F
02023F  530D 0A 0A
020243  20203D3D3E2000  DC.B '  ==> ',0
020244
020244  426974746520 53  SCHLTXT:      DC.B 'Bitte Schluesselwort eingeben (max. 16 Zeichen) : ',0
020251  636B6C75657373
020258  656C776F727420
02025F  65696E67656265
020266  6E202B6D61782E
02026D  203136205A6569
020274  636B656E29203A
02027B  2000
02027D  4E616D65206465  GETTXT:       DC.B 'Name der zu lesenden Datei : ',0
020284  722074752066 65
02028B  73656E64656E20
020292  446174656920 3A

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 5

020299  2000
02029B
02029B                *** PUFFERSPEICHER
02029B
02029B  00            DS 0
02029C                FCB:        DS.B 48       * PUFFER FUER FCB
0202CC                INBUF:      DS.B 34       * PUFFER FUER SCHLUESSELWORT
0202EE                LADEADR:                  * HIERHIN WIRD FILE GELADEN
0202EE
0202EE                END.

0EBCA0  Ende-Symboltabelle
```

68000 — YOGIDOS UND JADOS, RL-BASIC

Scop-Programm mit AD 10x1 und 680XX

von Bruno Sontheim

Programmbeschreibung:

In der letzten *LOOP* wurde die Jados-Tooldiskette vorgestellt. In der Programmsammlung befindet sich auch ein Scop-Programm für den AD 10x1. Dieses Programm soll nun etwas näher beschrieben werden.

Nach dem Start des Programms werden zwei Fenster auf dem Bildschirm sichtbar. Im oberen Fenster wird das momentane Eingangssignal dargestellt, im unteren Fenster können Signale zwischengespeichert oder 2 verschiedene Kurven miteinander dargestellt werden. Am rechten Rand sind die einzelnen Einstellungen und deren Bedienung dargestellt.

Nachfolgend sind die technischen Daten tabellarisch aufgeführt.

- 5 Empfindlichkeitseinstellungen 2.5 V/T, 1.0 V/T, 0.5 V/T, 0.2 V/T, 0.1 V/T
- Zeitbereich von 5ms – 1s in 8 Abstufungen
- Raster (abschaltbar)
- Dehnung x5
- Triggerung negativ und positiv, Triggerniveau einstellbar
- Einzelmessung, Auslösung durch Tastendruck oder High-Signal an Port 30h Bit 0
- Zwischenspeicherung des Signals im unteren Fenster mit Anzeige der Attribute
- Offsetspannung positiv und negativ einstellbar

- automatische Anpassung der Zeitbasis an unterschiedliche Taktfrequenzen

Wie die unterschiedlichen Funktionen zu handhaben sind, soll nun detailliert dargestellt werden.

– Die Empfindlichkeit:

Durch Drücken der Taste „0" wird die Empfindlichkeitseinstellung aktiviert. Zwischen den beiden Fenstern erscheint der Text „neue Empfindlichkeit mit Leertaste wählen". Das bedeutet, daß mit jedem Tastendruck der nächste Bereich angewählt wird. Der Vorgang ist zyklisch, d.h. nach dem letzten Wert wird wieder die erste Einstellung aktiv. Der gültige Bereich wird unterlegt (invers) dargestellt, ist also eindeutig erkennbar.

Durch Drücken der Taste „ESC" kehrt man wieder in das Normal-Programm zurück. Alle nachfolgenden Messungen werden nun mit der neuen, eingestellten Empfindlichkeit vorgenommen und angezeigt.

– Die Horizontalablenkung (t/Teil):

Die Auswahl der Zeitbasis wird genauso gehandhabt wie die Einstellung der Empfindlichkeit. Nur wird hier die Einstellung durch Drücken der „1"-Taste eingeleitet.

– Verbinden:

Das ist eine Option, die bei analogen Oszillographen nicht zu finden ist. Die Messung von Wechselgrößen mit AC-Converter ist nur ausschnittweise möglich. Angenommen, die Abtastfrequenz des AD 10x1 betrage 10kHz und es soll eine Wechselspannung mit der Frequenz 500 Hz gemessen werden. Die Periodendauer des Messignals beträgt 2ms. Mit Hilfe dieser Angaben kann nun die Anzahl der Messungen pro Periode berechnet werden:

$n = 0.002/(1/10000(1/s)) = 20$

In diesem Fall erhält man 20 Punkte zur Darstellung einer Periode. Ist die Funktion „Verbinden" inaktiv, dann werden nur diese 20 Punkte auf dem Bildschirm abgebildet. Aktiviert man „Verbinden", so wird dem Betrachter eine lückenlose Messung (analoges Oszilloskop) vorgetäuscht.

Die Funktion „Verbinden" ist nach dem Start des Programms immer aktiv. Die Taste „2" ist als Toggle-Switch programmiert.

– Dehnung:

Diese Funktion ist vom normalen Oszilloskop übernommen. Die Zeitbasis wird mit 5 multipliziert. Das Ein- bzw. Abschalten wird mit der Taste „4" durchgeführt.

– Die Triggerung:

Die Triggerung wurde ausschließlich mit der Software realisiert. Dabei ist darauf zu achten, daß bei „falscher" Zeiteinstellung eine Triggerung nicht immer gewährleistet ist. Das Programm wartet maximal 10x (der eingestellten Zeitbasis) auf den Triggerwert. Ist der Triggerwert dann noch nicht gemessen worden, wird das Signal ohne Triggerung dargestellt. Eine Begrenzung der Triggerabfrage ist unerläßlich, da sich das Programm sonst „aufhängen" würde (Gleichspannung, Triggerniveau zu hoch). Es empfiehlt sich also, um eine ruhige Anzeige zu bekommen, die Zeiteinstellung so vorzunehmen, daß mindestens eine Periode dargestellt wird. Natürlich muß darauf geachtet werden, daß der Betrag des Triggerwertes kleiner als der Betrag der Signalamplitude ist. Die Einstellung des Triggers wird durch das Drücken der Taste „5" eingeleitet. Mit den Tasten „1" und „2" wird der Schwellwert eingestellt, mit den Tasten „+" und „–" die Triggerart. „+" bedeutet, es wird mit ansteigender Flanke getriggert, „–" bedeutet, es wird mit der fallenden Flanke getriggert. Die Triggerart wird im Menü angezeigt. Die Anzeige des Trigger-Menüs wird mit Hilfe eines kleinen Pfeils am linken Rand des oberen Fensters durchgeführt. Die Triggereinstellung wird durch das Drücken der Taste „ESC" beendet.

– Die Einzelmessung

Mit dieser Option ist es möglich, einmalige Vorgänge zu erfassen. Früher, als noch keine Digitaloszilloskope zur Verfügung standen, mußten einmalige Vorgänge (z.B. Einschwingverhalten eines RC-Gliedes) fotografiert(!) werden. Hier wurden zwei Möglichkeiten geschaffen, die Aufzeichnung eines Signals auszulösen:

1. durch Drücken der Leertaste,
2. High-Pegel an Bit 0 Port 30h

Alle anderen Einstellungsmöglichkeiten sind direkt aufrufbar. Das heißt, die Zeiteinstellung oder die Empfindlichkeit läßt sich direkt aufrufen.

Der Einzelmessungs-Modus wird durch Drücken der „ESC"-Taste verlassen.

– Das Speichern:

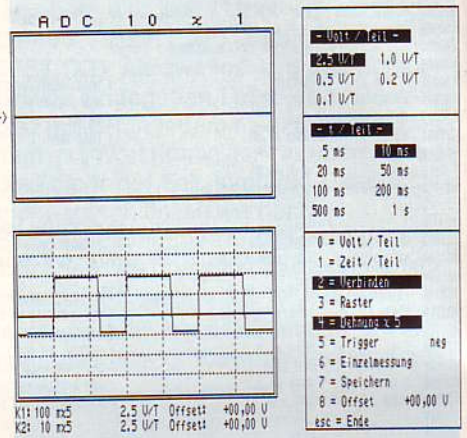
Auch hier stehen drei verschiedene Möglichkeiten zur Auswahl. Mit „1" wird das Signal vom oberen Fenster ins untere Fenster kopiert. Die Attribute werden unter dem 2. Fenster dargestellt. Mit „2" kann ein zweites Signal angezeigt werden. Mit „3" werden beide Signale gelöscht und das Fenster ist bereit, das nächste Signal aufzunehmen. Ende = „ESC".

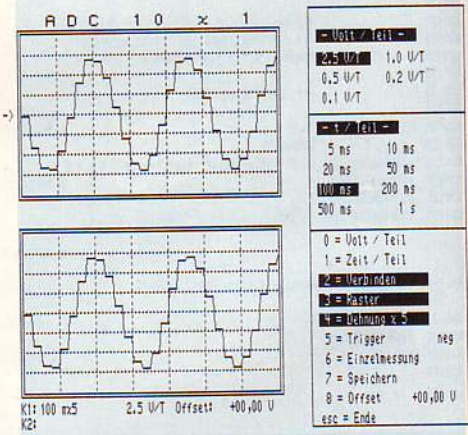
– Der Offset:

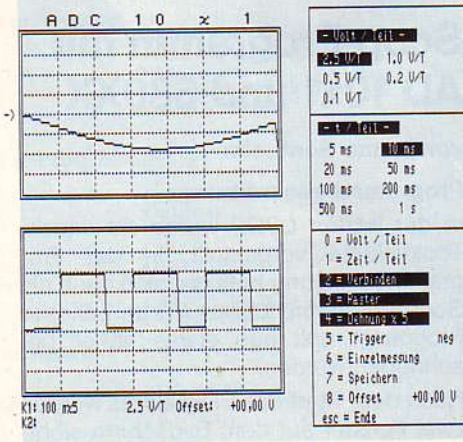
Der Offset dient dazu, Gleichspannungs-Anteile auszugleichen. Die eingestellte Offsetspannung wird angezeigt. Die ungeraden Werte hängen mit dem Meßbereich des AD-Wandlers zusammen. Natürlich beeinflußt der Offset auch das Trigger-Niveau, denn sonst wäre der Sinn des Offsets, das Messen von Wechselspannungen mit Gleichstromanteil, verfehlt. Eine Meßbereichserweiterung ist damit allerdings nicht möglich.

– Programmende:

Mit ^C kommt man ins Betriebssystem zurück.







Quick-Sort

von Dieter Wettrich und Gerhard Roßmanith, Moltke-Straße 2, 7907 Langenau

Wir wollen hiermit ein kleines, ausgefeiltes Sortierprogramm für die *LOOP*-Leserschaft beisteuern.

Bisher arbeiteten wir mit dem NDR-Z80 CP/M-Vollausbau. Die entwickelten Sortierroutinen waren uns zu langsam. Nach dem Umstieg auf die 68008 CPU ergab sich die Möglichkeit, eine schnellere generelle Sortierroutine in Assembler zu entwickeln. Auf Grund guter Erfahrungen mit Quick-Sort bot sich dieses Verfahren auch hier an.

Nachdem die vorliegende Programmroutine bei uns in mehreren Programmen zufriedenstellend arbeitet, möchten wir sie auch anderen NDR-Benutzern zur Verfügung stellen.

```
Programmbeschreibung:
---------------------
                1. Satzlänge variabel (max. 64 KByte)
                2. Anzahl Sätze abhängig nur von Speichergröße
                3. 4 Sortierfelder im Satz möglich
                      (auf- oder absteigend)
                4. Sortiersteuerung durch Parameter

Parameterbeschreibung:
----------------------
                1.   dc.l   Adresse des zu sortierenden Hauptspeicherbereichs
                2.   dc.w   Anzahl der tatsächlich zu sortierenden Sätze
                3.   dc.w   Anzahl der Bytes eines Sortiersatzes
                4.  Definitionen für Sortierfelder
                4.1  dc.w   Distanz des 1. Sortierfeldes vom Anfang des Sortsatz
                4.2  dc.w   Anzahl Byte des 1. Sortierfeldes
                4.3  dc.b   '+'    für aufsteigende Sortierung
                            '-'    für absteigende  Sortierung
                4.4  dc.b   ' '    nicht belegt

                Definitionen für Sortierfelder 2 - 4 dto.
                     dc.w   0,0,0  Ende Sortierfelder
Einschränkung:
--------------
                1.   Vor dem Hauptspeicherbereich muß ein Arbeitsfeld in
                     der Länge eines Sortiersatzes angelegt werden !
```

Weitere Informationen sind dem dokumentierten Programmlisting zu entnehmen.

Die vorliegende Routine läuft unter anderem in einem Schallplatten-Programm, mit zur Zeit ca. 5000 Titeln. Sortierzeit ca. 25 sec.

```
*                                   T E S T P R O G R A M M
*
*                                   zu Quick-Sortierroutine


q:      lea     param,a0            * Parameter für Sortierlauf
        bsr     sort                * Aufruf der Sortierroutine
        rts                         * fertig - ( Rücksprung ins Grundprogramm )


        ds.l    0
param:  dc.l    arb                 * zur sortierender Hauptspeicherbereich
        dc.w    20                  * Anzahl der tatsächlich zu sortierenden Sätze
        dc.w    31                  * Anzahl Bytes, je Sort-Satz

                                    * Definitionen für 1. Sortierfeld
                                    * ( max. 4 Sortierfelder möglich )

        dc.w    5                   * Distanz des 1. Sortierfeldes im Sort-Satz
        dc.w    8                   * Anzahl Bytes des 1. Sortierfeldes im Sort-Satz
        dc.b    '+'                 * Kennzeichen '+' für aufsteigende Sortierung
                                    * des 1. Sortierfeldes
        dc.b    ' '                 * nicht belegt

                                    * Definitionen für 2. Sortierfeld
        dc.w    25,3
        dc.b    '-',' '             * Kennzeichen '-' für absteigende Sortierung

                                    * Definitionen für 3. Sortierfeld
        dc.w    16,4
        dc.b    '+',' '

                                    * Definitionen für 4. Sortierfeld
                                    * ( kein 4. Feld im Sort-Satz )
        dc.w    0,0
        dc.b    0,' '

zw:     ds.b    31                  * Arbeitssatz für Sort
                                    * Länge = Sort-Satz muß direkt vor dem zu
                                    * sortierendem Hauptspeicherbereich liegen !

                                    * Testtabelle
arb:    ds.b    3100                * 100 Sätze je 31 Byte
        org     arb
        dc.b    ' 1  55555555  3333    444    '
        dc.b    ' 2  22222222  2222    222    '
        dc.b    ' 3  33333333  4444    555    '
        dc.b    ' 4  33333333  4443    555    '
        dc.b    ' 5  33333333  4434    555    '
        dc.b    ' 6  55555555  3333    445    '
        dc.b    ' 7  22222222  2222    221    '
        dc.b    ' 8  33333333  4444    554    '
        dc.b    ' 9  33333333  4443    553    '
        dc.b    ' 10 33333333  4434    552    '
        dc.b    ' 11 55555555  3333    446    '
        dc.b    ' 12 22222222  2222    220    '
        dc.b    ' 13 33333333  4444    556    '
        dc.b    ' 14 33333333  4443    557    '
        dc.b    ' 15 33333333  4434    558    '
        dc.b    ' 16 55555555  3333    443    '
        dc.b    ' 17 22222222  2221    221    '
        dc.b    ' 18 33333333  4433    555    '
        dc.b    ' 19 33333333  4433    554    '
        dc.b    ' 20 33333333  4433    556    '
        dc.b    ' 21 55555555  3332    444    '
        dc.b    ' 22 22222222  2220    999    '
        dc.b    ' 23 33333333  4444    555    '
        dc.b    ' 24 33333333  4443    553    '
        dc.b    ' 25 33333333  4434    558    '
        org     arb+3100
        ds.l    0

                                    T E S T P R O G R A M M - E N D E
* ---------------------------------------------------------------------------


                                    * Q U I C K - S O R T I E R R O U T I N E
                                    *
                                    * (c) by Gerhard Roßmanith und Dieter Wetterich
                                    *        7907 Langenau, Moltke - Str. 2

sort:   movem.l d0-d7/a0-a2/a4-a6,zwreg
        movea.l (a0)+,a2            * Anfangsadresse Sort
        move    (a0)+,d1            * Anzahl zu sortierende Saetze
        move    (a0)+,d6            * Laenge eines Satzes
        ext.l   d6                  * erweitern auf Wortlaenge
        suba.l  d6,a2               * Adresse auf ersten Satz
        bsr     loe                 * loeschen Satz 1
        move    #3,d5               * Maximale Anzahl Sortierbegriffe
        lea     vergl,a6            * Adresse Vergleichsschleife
s01:    move    (a0)+,d3            * Distanz vom Anfang des Satzes
        move    (a0)+,d4            * Laenge des Sortierbegriff's
        subq    #1,d4               * fuer DBRA Schleife
        addq.l  #2,a0               * A0 auf naechsten Sortierbegriff
        cmpi.b  #'-',-2(a0)         * Absteigend Sortieren
        beq.s   s01a                * Ja
        cmpi.b  #'+',-2(a0)         * Aufsteigend Sortieren
        beq.s   s01b                * Ja
        move    #$4e71,(a6)         * Ende Sortierbegriffe NOP einsetzen
        bra.s   srtbeg              * Start des Sortiervorgangs
s01a:   bsr     loehi               * Nein  erster Satz Sortbegr. auf $FF
        move.b  #$65,24(a6)         * BMI.S setzen
        move.b  #$62,26(a6)         * BHI.S setzen
        bra.s   s01c                * Adresse + Distanz setzen
s01b:   move.b  #$62,24(a6)         * BHI.S setzen
        move.b  #$65,26(a6)         * BMI.S setzen
s01c:   move    #$6002,(a6)         * Sprungbefehl einsetzen
        move    d4,6(a6)            * Anzahl Byte fuer DBRA einsetzen
        move    d3,12(a6)           * Distanz fuer Sortierbegr. im Sortsatz
        adda.l  #32,a6              * auf naechste Vergleichsschleife
        dbra    d5,s01              * setzen
        bra.s   srtbeg              * Start des Sortiervorgangs


                                    * Sortierbegriff fuer absteigende
                                    * Sortierung auf $FF
loehi:  movea.l a2,a5               * Adresse 1 Satz in Arbeitsregister
        adda.l  d3,a5               * Auf Anfangsadresse im Satz setzen
        move.l  d4,d0               * Anzahl Bytes setzen
loehi1: move.b  #$ff,(a5)+          * $FF in Sortierbegriff eintragen
        dbra    d0,loehi1           * fertig
        rts                         * ja zurueck
                                    * Ersten Satz auf $00 fuer
                                    * Aufsteigende Sortierung
loe:    move    d6,d0               * Satzlaenge
        subq.l  #1,d0               * -1 fuer DBRA Schleife
        movea.l a2,a6               * Adresse 1 Satz in Arbeitsregister
loe1:   move.b  #0,(a6)+            * $00 in kompletten Satz
```

```
        dbra    d0,loel         * fertig
        rts                     * ja zurueck

srtbeg: nop                     * S O R T  beginn
        lea     stak,a4         * Anfangsadresse Stapel laden
        move.l  a2,(a4)+        * Adresse Teilbereich-links auf Stapel
        mulu    d6,d1           * Adresse des letzten Satzes ermitteln
        add.l   a2,d1           * Anfangsadresse dazu
        move.l  d1,(a4)+        * und als Teilbereich-rechts auf Stapel
srt01:  move.l  -(a4),d2        * Index-rechts vom Stapel laden
        move.l  (a4),d4         * Index-links  vom Stapel laden
srt02:  move.l  d2,d3           * H-Ind-rechts von Index-rechts laden
        move.l  d4,d5           * H-Ind-links  von Index-links  laden
        move.l  d2,d0           * Hilfselement
        sub.l   d4,d0           * ermitteln
        divu    d6,d0           * (Index-rechts - Index-links)/Satzlaenge
        ext.l   d0              * ohne Rest
        lsr.w   #1,d0           *   / 2
        mulu    d6,d0           *   * Satzlaenge
        add.l   d4,d0           *   + Index-links
        movea.l d0,a5           * ins Adressregister uebertragen
srt03:  cmp.l   d2,d5           * Tauschelement von links suchen
        bge.s   srt04           * H-Index-links >= Index-rechts
        move.l  d5,d0           * Adresse zu Vergleichendes Element
        bsr.s   verg1           * Vergleichen H-Ind-links-Elem mit Hilfselement
        bcc.s   srt04           * H-Ind-links-Elem  >=        wird Tauschelement
        add.l   d6,d5           * H-Index-links + 1
        bra.s   srt03           * weiter von links suchen
srt04:  cmp.l   d3,d4           * Tauschelement von rechts suchen
        bge.s   srt05           * H-Ind-rechts <= Index-links
        move.l  d3,d0           * Adresse zu Vergleichendes Element
        bsr.s   verg1           * Vergleichen H-Ind-rechts-Elem mit Hilfselemet
        bls.s   srt05           * H-Ind-rechts-Elem <=        wird Tauschelement
        sub.l   d6,d3           * H-Index-rechts - 1
        bra.s   srt04           * weitersuchen von rechts
srt05:  cmp.l   d5,d3           * wenn H-Ind-links > H-Ind-rechts
        bmi.s   srt07           * dann kein Tauschen
        movea.l d5,a0           * Adresse linkes  Tauschelement
        movea.l d3,a1           * Adresse rechtes Tauschelement
        cmpa.l  a0,a5           * Wenn Hilfselement getauscht wird
        bne.s   srt05a          * nein                (linkes Tauschelement)
        movea.l a1,a5           * ja dann Adresse mittauschen
        bra.s   srt05b          *    und weiter mit Tauschen
srt05a: cmpa.l  a1,a5           * Wenn Hilfselement getauscht wird
        bne.s   srt05b          * nein                (rechtes Tauschelement)
        movea.l a0,a5           * ja  dann Adresse mittauschen
srt05b: move.l  d6,d0           * Satzlaenge
        subq.l  #1,d0           * - 1 fuer DBRA-Schleife
srt06:  move.b  (a0),d1         * ein Byte von A in Zwischenbereich
        move.b  (a1),(a0)+      * ein Byte von B nach A
        move.b  d1,(a1)+        * Byte aus Zwischenbereich nach B
        dbra    d0,srt06        * noch Byte's zum Tauschen
        add.l   d6,d5           * H-Ind-links  + 1
        sub.l   d6,d3           * H-Ind-rechts - 1
srt07:  cmp.l   d5,d3           * H-Ind-links <= H-Ind-rechts
        bpl.s   srt03           * ja
        move.l  d2,d0           * groesse Teilbereich rechts
        sub.l   d5,d0           * Index-rechts - H-Ind-links
        move.l  d3,d1           * groesse Teilbereich links
        sub.l   d4,d1           * H-Ind-rechts - Index-links
        cmp.l   d1,d0           * groesseren der beiden Teilbereiche
        bpl.l   srt09           * Teilbereich rechts groesser
        cmp.l   d3,d4           * Ind-links >= H-Ind-rechts
        bpl.s   srt08           * ja kein Eintrag in Stapel
        move.l  d4,(a4)+        * Index-links  auf Stapel
        move.l  d3,(a4)+        * H-Ind-rechts auf Stapel
srt08:  move.l  d5,d4           * Index-links aus H-Ind-links setzen
        bra.s   srt11
srt09:  cmp.l   d2,d5           * H-Ind-links >= Index-rechts
        bpl.s   srt10           * ja  kein Eintrag in Stapel
        move.l  d5,(a4)+        * H-Ind-links  auf Stapel
        move.l  d2,(a4)+        * Index-rechts auf Stapel
srt10:  move.l  d3,d2           * Index-rechts aus H-Ind-rechts setzen
srt11:  cmp.l   d2,d4           * Index-rechts > Index-links
        bcs     srt02           * ja
        cmpa.l  #stak,a4        * Stapel abgearbeitet
        bne     srt01           * nein
        bra     sorte

                                * Vergleich für 1. Sortierbegriff

verg1:  nop                     * wird durch Sprung ersetzt
        bra.s   verge           * Fertig mit vergleichen
        move    #0,d1           * Anzahl zu vergleichender Bytes
        move.l  #0,d7           * Distanz vom Satzanfang
        movea.l a5,a0           * Anfangsadresse Hilfsfeld
        adda.l  d7,a0           * Adressregister auf Sortierbegriff
        movea.l d0,a1           * Index auf Sortierelement
        adda.l  d7,a1           * Adressregister auf Sortierbegriff
ve1:    cmpm.b  (a0)+,(a1)+     * Vergleich
        bhi.s   lik
        bcs.s   rek
        dbra    d1,ve1

                                * Vergleich für 2. Sortierbegriff

verg2:  nop
        bra.s   verge
        move    #0,d1
        move.l  #0,d7
        movea.l a5,a0
        adda.l  d7,a0
        movea.l d0,a1
        adda.l  d7,a1
ve2:    cmpm.b  (a0)+,(a1)+
        bhi.s   lik
        bcs.s   rek
        dbra    d1,ve2

                                * Vergleich für 3. Sortierbegriff

verg3:  nop
        bra.s   verge
        move    #0,d1
        move.l  #0,d7
        movea.l a5,a0
        adda.l  d7,a0
        movea.l d0,a1
        adda.l  d7,a1
ve3:    cmpm.b  (a0)+,(a1)+
        bhi.s   lik
        bcs.s   rek
        dbra    d1,ve4
                                * Vergleich für 4. Sortierbegriff
verg4:  nop
        bra.s   verge
        move    #0,d1
        move.l  #0,d7

        movea.l a5,a0
        adda.l  d7,a0
        movea.l d0,a1
        adda.l  d7,a1
ve4:    cmpm.b  (a0)+,(a1)+
        bhi.s   lik
        bcs.s   rek
        dbra    d1,ve4
verge:  move    #$4,ccr
        rts
lik:    move    #0,ccr
        rts
rek:    move    #1,ccr
        rts
sorte:  movem.l zwreg,d0-d7/a0-a2/a4-a6
        rts
zwreg:  ds.l    14
stak:   ds.l    60              * Puffer für Adressen der noch nicht
                                * bearbeiteten Teilbereiche
        end
```

Dynamische Speicherverwaltung für JADOS

von Klaus Janßen

Der vorliegende Beitrag befaßt sich mit dem Thema „dynamische Variablen" und stellt eine Realisierung für eine dynamische Speicherverwaltung für das Betriebssystem JADOS vor.

Zunächst soll die Frage beantwortet werden, was dynamiche Variablen sind; insbesondere der Begriff „dynamisch" bedarf einer Erklärung. Bei der Entwicklung eines Programms benötigt man in der Regel eine Anzahl Platzhalter, in denen man Daten speichert, die sich zur Programmlaufzeit ändern können. Diese Platzhalter nennt man Variablen. Man unterscheidet statische und dynamische Variablen. Der Speicherbedarf für die statischen Variablen wird bereits bei der Programmierung festgelegt und ändert sich während des Programmlaufs nicht mehr. Im Gegensatz dazu werden dynamische Variablen nicht bei der Programmierung, sondern während des Programmlaufs angelegt und gegebenenfalls auch wieder freigegeben.

Es gibt nun verschiedene Möglichkeiten, solche dynamischen Variablen anzulegen. Das wohl bekannteste Verfahren stellen die lokalen Variablen in einer Pascal-Prozedur dar. Sie werden wie folgt angelegt:

```
PROCEDURE beispiel;
VAR
   anton: INTEGER;
   berta: INTEGER;
BEGIN
   (* irgendein Programmcode *)
END;
```

Die Variablen „anton" und „berta" gelten nur für die Zeitdauer, in der die Prozedur „beispiel" die Programmkontrolle besitzt, und sie sind auch nur der Prozedur alleine bekannt. Die Variablen sind daher dynamisch und lokal. Im Assembler sehen die Verhältnisse ähnlich aus:

```
   anton EQU 0
   berta EQU anton+4
beispiel:
   suba.l #8,a7          * schafft 8 Byte Platz auf Stack
   * irgendein Programmciode
   adda.l #8,a7          * entfernt die 8 Byte vom Stack
   rts
```

Das zweite Verfahren soll etwas ausführlicher beschrieben werden. Es handelt sich hierbei um dynamische Variablen, die nicht nur innerhalb einer Prozedur, sondern im gesamten Prgramm gültig sind. Wozu dienen diese dynamischen Variablen?

An einem kleinen Beispiel soll dies einmal näher untersucht werden. Nehmen wir einmal an, ein Entwickler möchte die Windowtechnik mit überlappenden Windows auf dem NKC realisieren. Für jedes Window benötigt er einige Variablen zur Ver-

waltung der Windows (Attribute, Cursor, etc.) sowie ein Abbild des Windowinhaltes. Gäbe es keine dynamischen Variablen, so müßte der Entwickler schon bei der Programmierung die Zahl und Größe der Windows festlegen. Da er nicht wissen kann, wie groß der Anwender die einzelnen Windows festlegt, muß jedes Window für die maximale Größe ausgelegt werden. Dies ist jedoch eine immense „Verschwendung" von Speicherplatz.

Hier greifen nun die Vorteile der dynamichen Variablen. Sie werden erst dann angelegt, wenn sie gebraucht werden. Auch die Größe der Variablen wird genau dem Bedarf angepaßt. Und wenn sie nicht mehr benötigt werden, dann werden sie wieder freigegeben.

Im Folgenden wird ein einfaches, aber effizientes Verfahren vorgestellt, mit welchem dynamisch Speicherplatz angefordert und freigegeben werden kann. In der Initialisierungsroutine „initheap" wird die Speicherverwaltung vorbereitet. Als Parameter werden die Startadresse und Größe in Bytes des gesamten zu verwaltenden Speicherbedarfs übergeben. Mit „allocate" wird Speicherplatz aus diesem Pool angefordert. Als Parameter wird die Größe in Bytes übergeben. Wenn noch genügend Platz im Pool ist, dann wird ein Zeiger auf den angelegten Bereich zurückgegeben; ansonsten 0. Den Zeiger muß man sich irgendwo merken, z.B. in einem Register oder besser – in einer statischen Variablen. Zurückgegeben wird der Speicherbereich mit „dealloc". Als Parameter wird der Zeiger benötigt, den „allocate" geliefert hat.

Die Verwaltung arbeitet nach folgendem Prinzip. Jeder Speicherbereich benötigt 6 Byte Verwaltungsinformation, nämlich einen Status, der angibt, ob der Bereich frei oder besetzt ist und einen Zeiger auf den nachfolgenden, benachbarten Speicherbereich. In der „initheap"-Routine werden drei Speicherbereiche angelegt und verkettet:

```
        ------------------------
first:  | Status: besetzt      |
        | - - - - - - - - - - |
        | Nachfolger:       o-|---
        ------------------------  |
                                  |
                                  |
        ------------------------  |
        | Status: frei         | <-
        | - - - - - - - - - - |
        | Nachfolger:       o-|---
        | - - - - - - - - - - |  |
        |                    |  |
        |   F r e i e r      |  |
        |   B e r e i c h    |  |
        |                    |  |
        ------------------------  |
                                  |
                                  |
        ------------------------  |
last:   | Status: besetzt      | <-
        | - - - - - - - - - - |
        | Nachfolger: keiner   |
        ------------------------
```

Das Element „first" dient als Einstieg in die Kette, das Element „last" als Ausstieg. Der dazwischenliegende Block ist der eigentliche Speicherpool.

In der Routine „allocate" wird beginnend bei „first" die Kette abgesucht, bis der kleinste freie Speicherbereich gefunden wird, der die angeforderte Zahl Bytes aufnehmen kann. Ist der Bereich größer, so wird der Rest abgetrenn und als freier Bereich eingekettet.

In der Prozedur „dealloc" wird der Bereich wieder als frei eingetragen. Darüberhinaus wird untersucht, ob das vorliegende und das nachfolgende Element ebenfalls frei sind. Wenn ja, dann werden

```
        ------------------------
first:  | Status: besetzt      |
        | - - - - - - - - - - |
        | Nachfolger:       o-|---
        ------------------------  |
                                  |
                                  |
        ------------------------  |
        | Status: besetzt      | <-
        | - - - - - - - - - - |
        | Nachfolger:       o-|---
        | - - - - - - - - - - |  |
        |                    |  |
        |   Angeforderter    |  |
        |   B e r e i c h    |  |
        |                    |  |
        ------------------------  |
                                  |
                                  |
        ------------------------  |
        | Status: frei         | <-
        | - - - - - - - - - - |
        | Nachfolger:       o-|---
        | - - - - - - - - - - |  |
        |                    |  |
        |   F r e i e r      |  |
        |   Restbereich      |  |
        |                    |  |
        ------------------------  |
                                  |
                                  v
```

die Bereiche zusammengefaßt. Dadurch wird sichergestellt, daß der Speicher nicht im Laufe der Zeit „zersplittert" wird. Das nachfolgende Listing enthält die besprochenen Routinen.

Die Routinen sind im Zuge der Entwicklung meines neuen Programmeditors gründlich getestet worden. Wer das Listing nicht abtippen will, kann es von mir gegen formatierte Diskette und 10,– DM (V-Scheck oder Briefmarken) im JADOS-Format beziehen.

Wer möchte, kann mir gerne über seine Anwendung der Routinen berichten. Meine Anschrift:

Klaus Janßen,
Hannixweg 74 · 4150 Krefeld 1

```
*************************************************
* HEAP - Management
*-----------------------------------------------
* Routinen:
*   allocate -> Heapelement anfordern
*   dealloc  -> Heapelement freigeben
*   initheap -> Heap einrichten
*
* Datum: 30.07.1987
* Autor: Klaus Janssen
*
*************************************************


* Konstanten
minsize EQU     60      * Mindestgroesse des Heaps in Bytes
elmsize EQU      6      * Verwaltungsgroesse eines Elements

h_free  EQU      0      * Heapelement ist frei
h_occu  EQU      1      * Heapelement ist besetzt


* Datenstrukturen
* Heapverwaltung
status  EQU      0      * Heapelementstatus: 0 = freies Element
                        *                    1 = besetztes Element
succ    EQU      2      * Zeiger auf physikalisch benachbartes Element
                        * GESAMTGROESSE: 6 Bytes


* Lokale statische Variablen
* (nicht fuer EPROM-Betrieb geeignet !)

first:  dc.l    0       * Zeiger auf erstes Heapelement
last:   dc.l    0       * Zeiger auf letztes Heapelement


*-----------------------------------------------
initheap:
* Zweck: Einrichten des Heaps
* Parameter:
*   d0.l: Startadresse des Heaps
*   d1.l: Heapgroesse in Bytes
* Bemerkung: Die Startadresse muss gerade sein
*            Die Heapgroesse wird aufgerundet, so dass
*             sie durch 6 teilbar ist
* Kennung: JAB70727
*-----------------------------------------------
  movem.l d0-d2/a0-a1,-(a7)
  cmp.l   #minsize,d1
  bge.s   ok_size
  move.l  #minsize,d1
ok_size:                  * Heap hat Mindestgroesse oder mehr
  move.l  d1,d2           * Aufrunden der Groesse, so dass sie
  divu    #elmsize,d2     *  durch elmsize teilbar ist
  swap    d2
  and.l   #$0000FFFF,d2   * Rest ist in D2
  beq.s   no_correction
  sub.l   d2,d1           * Aufrundung erfolgt !
  add.l   #elmsize,d1

no_correction:
  lea     first(pc),a0    * first := Startadresse
  move.l  d0,(a0)
```

```
movea.l d0,a0          * first^.status := occupied
move.w  #h_occu,status(a0)

move.l  d0,succ(a0)    * first^.succ := first_free_element
add.l   #elmsize,succ(a0)

lea     last(pc),a1    * last := first + heapsize - elmsize
move.l  d0,(a1)
add.l   d1,(a1)
sub.l   #elmsize,(a1)

movea.l (a1),a1        * last^.status := occupied
move.w  #h_occu,status(a1)

move.l  #0,succ(a1)    * last^.succ := NIL

adda.l  #elmsize,a0    * first_free_element^.status := free
move.w  #h_free,status(a0)

move.l  a1,succ(a0)    * first_free_element^.succ := last

E_initheap:
  movem.l (a7)+,d0-d2/a0-a1
  rts      * initheap


*-----------------------------------------------
allocate:
* Zweck: Anfordern eines Heapelements
* Parameter:
* > a0.l: Zeiger auf angefordertes Element
*         NIL, wenn kein Platz mehr
*   d1.l: Groesse des Elementes
* Bemerkung: Die Zeigeradresse muss gerade sein
*            Die Elementgroesse wird aufgerundet, so dass
*             sie durch 6 teilbar ist
* Kennung: JA870727
*-----------------------------------------------
  movem.l d0-d7/a1,-(a7)
  move.l  d1,d2          * Aufrunden der Groesse, so dass sie
  divu    #elmsize,d2    *  durch elmsize teilbar ist
  swap    d2
  and.l   #$0000FFFF,d2  * Rest ist in D2
  beq.s   no_corr1
  sub.l   d2,d1          * Aufrundung erfolgt !
  add.l   #elmsize,d1

no_corr1:
  add.l   #elmsize,d1    * size := size + elmsize
  movea.l first(pc),a1
  movea.l succ(a1),a1    * help := first^.succ

  move.l  #$7FFFFFFF,d7  * min := maxint
  move.l  #0,d6          * addr := NIL

alloc_lp:                * REPEAT
  cmp.w   #h_free,status(a1)    * IF help^.status = free THEN BEGIN
  bne.s   next_el

  move.l  succ(a1),d5           * i := help^.succ - help + elmsize
  sub.l   a1,d5

  cmp.l   d1,d5                 * IF i >= size THEN BEGIN
  blt.s   next_el
  cmp.l   d7,d5                 * IF i < min THEN BEGIN
  bge.s   next_el

  move.l  d5,d7                 * min := i
  move.l  a1,d6                 * addr := help
                                * END
                                * END
next_el:                        * END
  movea.l succ(a1),a1   * help := help^.succ
  cmp.l   #0,succ(a1)
  bne.s   alloc_lp      * UNTIL help = NIL

  cmp.l   #0,d6         * IF addr <> NIL THEN BEGIN
  beq.s   no_space

  movea.l d6,a0         * addr^.status := occupied
  move.w  #h_occu,status(a0)
  move.l  succ(a0),d4   * help := addr^.succ

  move.l  a0,succ(a0)   * addr^.succ := addr + size
  add.l   d1,succ(a0)

  sub.l   d1,d7         * IF (min-size) > elmsize THEN BEGIN
  cmp.l   #elmsize,d7
  blt.s   no_split

  movea.l succ(a0),a1           * help1 := addr^.succ
  move.w  #h_free,status(a1)    * help1^.status := free
  move.l  d4,succ(a1)           * help1^.succ := help

no_split:                       * END
  adda.l  #elmsize,a0           * addr := addr + elmsize

bra.s     E_allocate    * END

no_space:               * ELSE addr := NIL
  movea.l #0,a0

E_allocate:
  movem.l (a7)+,d0-d7/a1
  rts      * allocate


*-----------------------------------------------
dealloc:
* Zweck: Rueckgabe eines Heapelements
* Parameter:
*   a0.l: Zeiger auf Element
* Bemerkung: Die Zeigeradresse muss gerade sein
* Kennung: JA870730
*-----------------------------------------------
  movem.l d0-d7/a0-a4,-(a7)

  suba.l  #elmsize,a0   * Zeiger auf Verwaltungsteil des Elements richten

  lea     first(pc),a1  * IF addr < first THEN return
  movea.l (a1),a1
  cmpa.l  a1,a0
  blt     E_dealloc

  lea     last(pc),a1   * IF addr >= last THEN return
  movea.l (a1),a1
  cmpa.l  a1,a0
  bge     E_dealloc

  * Physikalischen Vorgaenger suchen
    lea     first(pc),a1  * pred_el := first
    movea.l (a1),a3

  pred_search:
    cmpa.l  succ(a3),a0   * WHILE pred_el^.succ <> addr DO
    beq.s   pred_found    * pred_el := pred_el^.succ
    movea.l succ(a3),a3
    bra.s   pred_search

  pred_found:
    movea.l succ(a0),a4   * succ_el := addr^.succ

    move.w  #h_free,status(a0)    * addr^.status := free

    cmp.w   #h_free,status(a3)    * IF pred_el^.status = free THEN
    bne     looksucc

  * Vorgaengerelement ist frei, dann zusammenfuegen
    move.l  a4,succ(a3)           * pred_el^.succ := succ_el

                                  * END
  looksucc:
    cmp.w   #h_free,status(a4)    * IF succ_el^.status = free THEN BEGIN
    bne     E_dealloc

  * Nachfolgerelement ist frei, dann zusammenfuegen
    move.l  succ(a4),succ(a0)     * addr^.succ := succ_el^.succ
    cmp.w   #h_free,status(a3)    * IF pred_el^.status = free THEN
    bne.s   E_dealloc

  * Wenn auch Vorgaenger frei, dann Element ausketten
    move.l  succ(a4),succ(a3)     * pred_el^.succ := succ_el^.succ

  E_dealloc:                      * END
    movem.l (a7)+,d0-d7/a0-a4
    rts      * dealloc
```

CP/M 68 K, C UND MODULA

CP/M 68K unter JADOS

von Rüdiger Bäcker

Heute etwas für Umsteiger von CP/M 68K auf JADOS!? – Umsteiger von CP/68K auf JADOS? – Ja – richtig, so etwas gibt es und das auch zu Recht, wie jeder bestätigen wird, der mit JADOS arbeitet.

Auch ich gehöre zu diesen Umsteigern – oder, angesichts der Leistungen von JADOS besser Aufsteigern. Mein Problem war, daß ich einige Daten von CP/M 68K zu JADOS transferieren wollte. Die Lösung ist das Programm LOADCPM, mit dem das CP/M 68K als JADOS-Datei ab-

gespeichert wird und von JADOS aus gestartet werden kann.

Die Realisierung war ganz einfach, man muß lediglich das im Speicher stehende CP/M 68k mit dem DSAVE-Befehl von JADOS auf Disk speichern. Als Name sollte dabei CPM68.DAT angegeben werden.

Um das CP/M in den Speicher zu bekommen, muß man es zunächst ganz normal booten. Die Adresse, an der das CP/M dann steht, ist die beim Relozieren angegebene. Das nachstehende SUBMIT-File zeigt diese Adresse, die natürlich von System zu System verschieden sein kann:

```
as68 -f g: b:$1.s
1068 -f g: -r -ucpm -0 b:cpm.rel cpmlib b:$1.o
reloc -b18000 b:cpm.rel b:cpm.sys
```

Mit diesem SUBMIT-File würde das CP/M also an Adresse $18000 stehen. Das CP/M selbst hat eine Länge von ca. 32k, wenn die Winchester- und Fastfloroutinen mit eingebaut werden.

Mit DSAVE werden also dann ab Adresse $18000 insgesamt 32k als Datei CPM68.-DAT abgespeichert.

Das Programm LOADCPM lädt nun das CP/M 68K immer an die angegebene Ladeadresse – die natürlich ggf. angepaßt werden kann – und startet es. Das CP/M 68K meldet sich dann mit dem bekannten 'A)'.

```
10.08.87-08.14 Assprint - (C) 1987 by R. Baecker - Listing des Files : LOADCPM.S
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

020000                  *       L O A D C P M
020000                  *
020000                  *       PROGRAMM ZUM LADEN UND STARTEN VON CP/M 68K UNTER J A D O S
020000                  *
020000                  *       COPYRIGHT (C) 1987 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
020000                  *
020000                  *       V 1.0 - 10.08.87
020000
020000                  *******
020000                  *** VEREINBARUNGEN
020000                  *******
020000
020000                  *** LADEADRESSE FUER CP/M 68K - MUSS AN SYSTEM ANGEPASST WERDEN !
020000
= 00018000              LADEADRESSE     EQU     $18000  * DORTHIN CP/M 68K LADEN
020000
020000                  *** JADOS-FUNKTIONEN
020000
= 0000002C              FILELOAD  EQU 44
= 00000012              FILLFCB   EQU 18
020000
020000                  *******
020000                  *** H A U P T P R O G R A M M
020000                  *******
020000
020000                  *** CP/M 68K LADEN
020000
020000                  LOADFILE:                       * CP/M 68K LADEN - HAUPTPROGRAMM
020000  6100 0012       BSR MAKEFCB                     * FCB FUELLEN
020004  41F9 00018000   LEA LADEADRESSE,A0              * DORTHIN LADEN !! GGF. A N P A S S E N !!
020008  7E2C            MOVEQ #FILELOAD,D7
020006  4E46            TRAP #6
020006  4EF9 00018000   JMP LADEADRESSE                 * UND CP/M 68K STARTEN
020014
020014                  *** FCB ERZEUGEN
020014
020014                  MAKEFCB:                        * FILENAME HOLEN UND FCB ERZEUGEN
020014  41FA 0010       LEA NAMETXT(PC),A0              * AUFFORDERUNGSTEXT
020018  43FA 0016       LEA FCB(PC),A1                  * ADRESSE FCB, BLEIBT NACH FUNKTION ERHALTEN
020018  41FA 0008       LEA NAMETXT(PC),A0
020020  7E12            MOVEQ #FILLFCB,D7
020022  4E46            TRAP #6                         * NAME HOLEN UND IN FCB EINTRAGEN
020024  4E75            RTS                             * AUFRUF ALS UNTERPROGRAMM
020026
020026                  *******
020026                  *** DATENABLAGE / PUFFERSPEICHER
020026                  *******
020026
020026                  DS 0                            * AUF EVEN
020026  43504D363B2E44 NAMETXT: DC.B 'CPM68.DAT',0      * CP/M MUSS ALS DATEI CPM68.DAT AUF DISK STEHEN
020020  415400
020030
020030                  DS 0                            * AUF EVEN
020030                  FCB:
020030                  DS.B 48                         * FCB=PUFFER
020060
020060                  END.
```

Modula 2 –

Tips und Tricks von Rolf-Dieter Klein

Da immer wieder Probleme beim Arbeiten mit Dateien aufgetaucht sind, soll hier einmal anhand zweier Beispiele der Umgang damit beschrieben werden.

Besonders elegant ist das Arbeiten mit dem ReadBlock- und WriteBlock-Befehl. Damit kann man eine beliebige Datenstruktur auf einmal bearbeiten. In Bild 1 wird eine Variante mit dem Namen viel deklariert. Es handelt sich um ein Feld mit 20 Einträgen, das aus einer Struktur besteht.

Im Hauptprogramm werden zunächst ein paar Testwerte in das Feld geschrieben. Danach wird mit Create eine Datei erzeugt und mit einem Befehl WriteBlock das gesamte Feld auf Diskette geschrieben. WriteBlock kennt die Größe des jeweils angegebenen Variablennamens. Daher kann man auf recht flexible Weise Daten verarbeiten. Die Variable State liefert noch das Ergebnis des Schreibvorgangs; hier wird aber der Einfachheit halber nichts weiter abgefragt.

Im zweiten Beispiel wird die Datei mit ReadBlock wieder eingelesen. Anschließend wird der Inhalt zu Testzwecken auf den Bildschirm ausgegeben.

Modula 2 erlaubt eine Vielzahl von Dateioperationen, die in den Bibliotheken verfügbar sind. Es ist nicht immer ganz einfach, damit gleich zurecht zu kommen, da Modula 2 eine sehr strenge Sprache ist. Es empfiehlt sich daher, die Beschreibungen des Handbuches genau zu lesen, dann funktionieren die Operationen auch, wie man es erwartet.

```
MODULE test;

FROM Files IMPORT File,FileState,Create,Close,Open,BinTextMode,
      ReplaceMode,ReadWriteMode;
FROM Binary IMPORT ReadBlock,WriteBlock;
FROM SimpleIO IMPORT WriteString,WriteLn,WriteCard,WriteChar;

VAR
 fd : File;
 state : FileState;
 i : INTEGER;
 viel : ARRAY[1..20] OF
         RECORD
          name : ARRAY[0..15] OF CHAR;
          index: CARDINAL;
         END;

BEGIN
 Open(fd,"testdatei",binMode,readOnly,state);
 ReadBlock(fd,viel,state);
 Close(fd,state);
 FOR i:=1 TO 20 DO
  WriteString(viel[i].name);
  WriteChar(' ');
  WriteCard(viel[i].index,5);
  WriteLn;
 END;

END test.
```

```
MODULE test;

FROM Files IMPORT File,FileState,Create,Close,Open,BinTextMode,
      ReplaceMode,ReadWriteMode;
FROM Binary IMPORT ReadBlock,WriteBlock;

VAR
 fd : File;
 state : FileState;
 i : INTEGER;
 viel : ARRAY[1..20] OF
         RECORD
          name : ARRAY[0..15] OF CHAR;
          index: CARDINAL;
         END;

BEGIN
 FOR i:=1 TO 20 DO
  viel[i].name := "Hallo";
  viel[i].index := i;
 END;
 Create(fd,"testdatei",binMode,replace,state);
 WriteBlock(fd,viel,state);
 Close(fd,state);

END test.
```

Aus der SW-Ecke

Alle von GES ausgelieferten Disketten sind mit einer Beschreibung versehen. Der Name ist „Liesmich" und kann von allen Betriebssystemen gleich aufgerufen werden.

Bei CP/M 2.2, CP/M 68K und JADOS: TYPE LIESMICH.

Mit den Tasten CTRL-S kann die Anzeige am Bildschirm angehalten werden. Weiter gehts bei CP/M xxx mit CTRL-S und bei JADOS mit CTRL-Q.

Der Ausdruck ist natürlich auch möglich:
CP/M xxx → vorher Eingabe CTRL-P
JADOS → Print Liesmich

Die „Liesmich" wird Sie dann mit den wichtigsten Informationen ausstatten.

EPROM- und RAM-FLOPPY

zwei schnelle „Laufwerke" für den mc-CP/M-Computer

Seit einiger Zeit sind für den mc-CP/M-Computer zwei Baugruppen erhältlich, die die Arbeitsgeschwindigkeit dieses Rechners erheblich erhöhen. Eine RAM-Floppy mit einer Kapazität von 128 KB, in Kürze wird auch eine Baugruppe mit 512 KB lieferbar sein, und eine EPROM-Floppy mit einer Kapazität von 320KB. Wir wollen uns in diesem Artikel vor allem mit der Installation und dem Betrieb der beiden Baugruppen beschäftigen. Für die Baugruppen werden von GES zwei BIOS-Versionen geliefert, die auch einen Autostart erlauben. Wenn nun von einer Diskette gestartet wird, so ist es relativ einfach, auf jeder Diskette eine andere Autostartfunktion zu installieren. Da aber auch aus der EPROM-Floppy gebootet werden kann, muß ein Weg gefunden werden, um auch beim Boot aus dem EPROM von Fall zu Fall ein anderes Programm zu starten. Diese Möglichkeit erlaubt uns die Submitfunktion (auch als Stapelverarbeitung bekannt). Dazu muß aber die Datei SUBMIT.COM auf der EPROM-Floppy vorhanden sein. Dann können wir in den CCP folgenden Befehl installieren: SUBMIT C:START.

Wie dies gemacht wird, ist bereits in der *LOOP* Nr. 12, Seite 9 beschrieben. Dieses, so geänderte Betriebssystem, wird als CPM60.SYS auf eine Diskette gespeichert. Nun kann man dazu übergehen, die Programme zusammenzustellen, die man gerne auf der EPROM-Floppy haben möchte. Dabei muß man besonders darauf achten, daß nur 32 Directory-Einträge im Eprom-Bereich liegen. Die Anzahl wird vor allem dann schnell überschritten, wenn sehr viele kleine Hilfsroutinen in die Eproms gebrannt werden sollen. Bevor die Programme dann endgültig auf einer Diskette zusammengestellt werden, muß man die Anzahl der Blöcke für das Laufwerk, auf dem die Programme gesammelt werden sollen, auf weniger als 256 verringern, dann kann man mit dem kleinen Hilfsprogramm aus dem Handbuch das Directory der Diskette direkt für die Eproms verwenden. Dies funktioniert allerdings nur, wenn die Anzahl der Blöcke von vornherein größer als die benötigten Blöcke für die EPROM-Floppy ist. Das Verringern hört sich sehr einfach an, aber es hat sich in der Praxis doch als viel schwieriger herausgestellt, da meist Laufwerk „B" dafür verwendet wird, aber für „A" und „B" immer nur ein gemeinsamer Diskparameterblock vorhanden ist. Nach einer Änderung von „B" hat also auch „A" die gleichen Parameter und somit kann man die Disketten nicht mehr bearbeiten. Deshalb befindet sich auf der Diskette für die EPROM- und RAM-Floppy eine abgewandelte Version des Floppy-BIOS (BIOSEPF), welche für Laufwerk „B" die gleichen Parameter benutzt, wie auch die EPROM-Floppy. Die DIR-Einträge wurden auf 64 begrenzt, die Kapazität beträgt 320 KB. Trotzdem kann eine Diskette im 80-Spur-Format verwendet werden, dessen Kapazität jedoch nicht voll genutzt wird. Wer mit kleineren Diskettenformaten arbeitet (ECMA70), aber doppelseitige Laufwerke besitzt, kann auch mit dem BIOS-Patch-Programm ein entsprechendes Format erzeugen. Das Hilfsprogramm im Handbuch sollte noch mit einer kleinen Lade-Routine versehen werden, dann kann man das Programm eventuell öfter aufrufen und muß es nicht jedesmal mit Hilfe des Debuggers auf 0500H verschieben. Hier die geänderte Version:

```
     MACRO-80 3.43    27-Jul-81          PAGE    1

                                         .Z80
0000'                                    ASEG
                                         ORG     100H
0100    21 010E                          LD      HL,START
0103    11 0500                          LD      DE,500H
0106    01 007A                          LD      BC,ENDE-START
0109    ED B0                            LDIR
010B    C3 0500                          JP      START1
010E                             START:
                                         .PHASE  500H
0500                             START1:
F009                             CONOUT  EQU     0F009H
0500    21 0100                          LD      HL,100H
0503    16 02                            LD      D,2
0505    1E 01                            LD      E,1
0507    E5                       RDSEC:  PUSH    HL
0508    D5                               PUSH    DE
0509    C5                               PUSH    BC
050A    0E D2                            LD      C,11010010B
050C    06 01                            LD      B,1
050E    CD F027                          CALL    0F027H
0511    C1                               POP     BC
0512    D1                               POP     DE
0513    E1                               POP     HL
0514    21 052A                          LD      HL,MSG
0517    CD 051D                          CALL    PRMSG
051A    C3 0000                          JP      0
051D    7E                       PRMSG:  LD      A,(HL)
051E    B7                               OR      A
051F    C8                               RET     Z
0520    E5                               PUSH    HL
0521    4F                               LD      C,A
0522    CD F009                          CALL    CONOUT
0525    E1                               POP     HL
0526    23                               INC     HL
0527    C3 051D                          JP      PRMSG
052A    0D 0A                    MSG:    DB      0DH,0AH
052C    4C 4F 41 44                      DB      'LOADING DIRECTORY OF '
0530    49 4E 47 20
0534    44 49 52 45
0538    43 54 4F 52
053C    59 20 4F 46
0540    20                               DB      'DRIVE B TO 0100H - 04FF'
0541    44 52 49 56
0545    45 20 42 20
0549    54 4F 20 30
054D    31 30 30 48
0551    20 2D 20 30
0555    34 46 46 48
0559    0D 0A                            DB      0DH,0AH
055B    53 41 56 45                      DB      'SAVE IT WITH '
055F    20 49 54 20
0563    57 49 54 48
0567    20                               DB      '"SAVE 4 DIR.DAT" '
056B    22 53 41 56
056C    45 20 34 20

     MACRO-80 3.43    27-Jul-81          PAGE    1-1

0570    44 49 52 2E
0574    44 41 54 22
0578    20                               DB      0
0579    00                               .DEPHASE
                                 ENDE:
0188                                     END

     MACRO-80 3.43    27-Jul-81          PAGE    S

Macros:

Symbols:
F009    CONOUT          0188    ENDE            052A    MSG
051D    PRMSG           0507    RDSEC           010E    START
0500    START1

No Fatal error(s)
```

Wenn nun die Datei DIR.DAT auf der Diskette abgespeichert ist, kann man daran gehen, mit dem DDT das SYSTEM-Eprom zusammenzustellen. Auf der BIOS-Diskette befindet sich hierfür eine Submitdatei, die das erledigt, deshalb wollen wir darauf nicht näher eingehen. Wer nun oft ohne Autostart arbeiten, aber dessen Funktion nicht ganz verzichten möchte, kann statt des 2764- ein 27128-Eprom verwenden und den Pin 26 des System-Eproms über einen kleinen Schalter zwischen High und Low umschalten. Dann kann ein System mit Autostart und eins ohne generiert werden. Das zweite System steht dann im Eprom ab Adresse 2000H. Ist das System dann ins Eprom gebrannt, so kann man bereits aus der EPROM-Floppy starten. Dazu wird ein geänderter Monitor benötigt. Die Änderung in der MC ist für die FLO1, die im Handbuch der EPROM-Floppy ist für den Betrieb mit der FLOSASI (FLO2) ausgelegt. Die neue Monitor-Version 8.0 ist sowohl für FLO1 als für die FLOSASIS geeignet. Nun kann man die Dateien auf der Spezialdiskette zu Eproms zusammenstellen. Gearbeitet wird dabei mit dem DDT. Ins Eprom 1 kommen die Dateien, welche als erstes im Inhaltsverzeichnis stehen. Dabei ist darauf zu achten, das die nächste Datei auch erst beim nächsten freien Block beginnen kann.

Beispiel:
Eine Datei XXX.COM wird eingelesen, nach dem Einlesen zeigt der DDT NEXT 1F80 an. Eine nun nachfolgende Datei YYY.COM kann also erst bei 2100H eingelesen werden.

Befehlsfolge:
DDT XXX.COM Anzeige: PC 100; NEXT 1F80
IYYY.COM
R2000 Anzeige: PC 100; NEXT
„Neue Endadresse"

Nach dem gleichen Prinzip werden alle Dateien eingelesen bis das Eprom voll ist. Dateien, welche über die Epromgrenzen hinausgehen, sollten vorher in zwei Dateien aufgeteilt werden.

Beispiel:
Eine Datei AAA.COM ist 16 KB lang, im Eprom sind noch 4 KB Platz.

Befehlsfolge:
DDT AAA.COM Anzeige: PC 100; NEXT 4100
GO
SAVE 16 AAA/1.COM
DDT AAA.COM Anzeige: wie oben
M1100,4100,100
GO
SAVE 48 AAA/2.COM

Damit ist die Datei zerlegt und kann auf zwei Eproms aufgeteilt werden. Die Datei AAA/1.COM wird ins fast volle Eprom kopiert, die Datei AAA/2.COM kommt ins nächste Eprom. Die Grenze in der TPA ist für 27128-Eproms 4100H, für 27256-Eproms 8100H. Entsprechend müssen die Dateien nach dem Zusammenstellen der Eproms abgespeichert werden. Bei 27128: SAVE 64 EPROM1. Bei 27256: SAVE 128 EPROM1.

Nachdem alle Eproms zusammengestellt sind, werden die Eprom-Dateien in die Eproms gebrannt. Man hätte natürlich auch die einzelnen Dateien direkt in die Eproms brennen können, nur auf der Diskette läßt sich ein Fehler leichter beheben als später, wenn schon ein Teil falsch in Eprom steht. Bei HEX-Adressen kann man sich schnell verrechnen, aber man muß sich deshalb nicht mehr Arbeit machen als unbedingt nötig. Sind nun alle Eproms gebrannt, so kann man die Schnelligkeit der Baugruppe erst einmal testen. Besonders angenehm ist der schnelle Warmstart bei CTRL C oder beim Beenden eines Programms, wenn aus der EPROM-Floppy gestartet worden ist, diese Baugruppe also als Systemlaufwerk definiert ist.

Bei älteren EPROM-Floppy-Bausätzen wurden, statt der 74HC161 Bausteine vom Typ 74LS161 bestückt, bei diesen Baugruppen kann es bei Vollbestückung mit Eproms zu Problemen kommen (ab 6 – 7 Eproms). Es sollten dann die LS-Typen gegen HC-Typen ausgetauscht werden.

Bei der Installation der RAM-FLOPPY gibt es in der Regel keine Probleme, nach einigen Tests mit dem Monitor und Installation des BIOS, kann dieses ohne Probleme angesprochen werden.

Noch einige Worte zu den neuen BIOS-Versionen:

Beide BIOS-Versionen unterstützen den Autostart. Die RAM-Floppy wird nur beim ersten Kaltstart initialisiert und entsprechend ihrer Größe installiert (128 oder 512 KB). Ein Löschprogramm ist nicht mehr nötig. Die Steprate wird nach dem Kaltstart auf Maximum gesetzt, es entfällt dadurch der Aufruf des Programms SPEED.COM. Dies kann jedoch bei Zugriff auf ältere 8-Zoll-Laufwerke zu Problemen führen, es muß dann vorher die Steprate wieder heruntergestezt werden.

Zum Schluß noch ein Vergleichstest. Übersetzen und Linken des BIOS für die EPROM-Floppy mit M80, L34 und DDT mit Hilfe der Submitfunktion einmal auf Diskette und zum zweiten Mal mit EPROM- und RAM-Floppy. Gemessen wurde mit einem 4 MHz System.

Diskette: 02:35 Minuten
EPROM- und RAM-Floppy: 00:30 Minuten.

Wir hoffen mit diesem Artikel einiges zur Erleichterung bei der Installation der EPROM-Floppy beigetragen zu haben und wünschen nun viel Spaß mit dem TURBO-MC-COMPUTER.

100% formatgetreue Hardcopy

von Uwe Koch, Frankenstraße 25, 5880 Lüdenscheid, Telefon: (02351) 26192

Ergänzung zum Hardcopy-Artikel in LOOP 13a, Seite 20:

In meinem oben genannten Artikel habe ich die optimale Anpassung eines einfachen (nicht Epson-kompatiblen) Matrix-Druckers an die HARDCOPY-MAUS-Karte beschrieben. Dabei ergab sich, auch unter Ausnutzung einiger Tricks, nur ein brauchbares, nicht aber formatgetreues Bild. Da das gleiche Problem auch bei den Epson-kompatiblen auftaucht, möchte ich hier Abhilfe schaffen.

Das im Handbuch der HARDCOPY-Karte abgedruckte Programm ist in der dargestellten Form zwar mit allen epson-kompatiblen Druckern lauffähig, es erzeugt aber ein unsymmetrisches Bild. Dabei hat ein Quadrat, das auf dem Bildschirm z.B. 512 x 256 Punkte hat, auf dem Drukker etwa ein Seitenverhältnis von 2 : 1. Wenn man nun einfach den Zeichen-Ausgabe-Befehl des Programmes verdoppeln würde, so ergebe das, bei gleichzeitiger Anpassung des Grafik-Steuercodes, ein Verhältnis von ungefähr 1 : 1. Aber dann müßten für eine Druckerzeile statt 256 nun 512 Grafik-Daten dargestellt werden. In dem mit 'ESC K n m' eingestellten Grafikmodus sind aber nur maximal 480 Grafikzeichen pro Zeile möglich. Daher würde diese Änderung zwar ein formatgetreues, aber nicht vollständiges Bild (480 von 512 Spalten) ergeben. Auch die Verwendung der Modi mit doppelter bzw. vierfacher Dichte ('ESC L n m' bzw. ESC Z n m') können dieses Problem nicht lösen, da sie die Punkte nur doppelt bzw. viermal so dicht setzen, ohne ein anderes Format zu erzeugen. Daher läßt sich auf einem Drucker, der nur diese Epson-kompatiblen Grafik-Modi kennt, tatsächlich kein formatgetreues Bild erzeugen.

Abhilfe schafft hier nur ein Epson/IBM-kompatibler Drucker, der auch den sogenannten „Grafik-Master-Modus" kennt. Denn hier gibt es neben den oben genannten Grafik-Betriebsarten „Standard", „doppelte Dichte" und „vierfache Dichte" auch noch die Arten „Bildschirm-Grafik Typ I", „Bildschirm-Grafik II" und „Plotter-Grafik". Die beiden Bildschirm-Grafik-Typen dienen einer formatgetreuen Hardcopy bei IBM-kompatiblen Druckern, die bekanntlich ein anderes Bildschirm-Format benutzen (640 x 200 oder 640 x 400 oder . . .).

Die Betriebsart „Plotter-Grafik" arbeitet dagegen mit einem Format von 72 Punkte/Inch. Dadurch ist der horizontale

Punktabstand genauso groß wie der vertikale. Damit lassen sich symmetrische Grafiken mit maximal 576 x 576 Punkten ausdrucken. Unser Bildschirmformat von 512 x 256 Punkten stellt ebenfalls ein symmetrisches Bild dar, weil die Auflösung in Y-Richtung (256) nur halb so groß ist und der Punktabstand daher doppelt so groß ist wie in X-Richtung. Verdoppelt man die Ausgabe von Y-Werten, so erhält man ein Format von 512 x 512, das sich in dem 576 x 576-Feld der Plottergrafik formatgetreu darstellen läßt. Neben dem symmetrichen Bild ergibt sich auch noch eine Bildgröße, die genau der Darstellungsgröße auf einem 12-Zoll-Monitor entspricht.

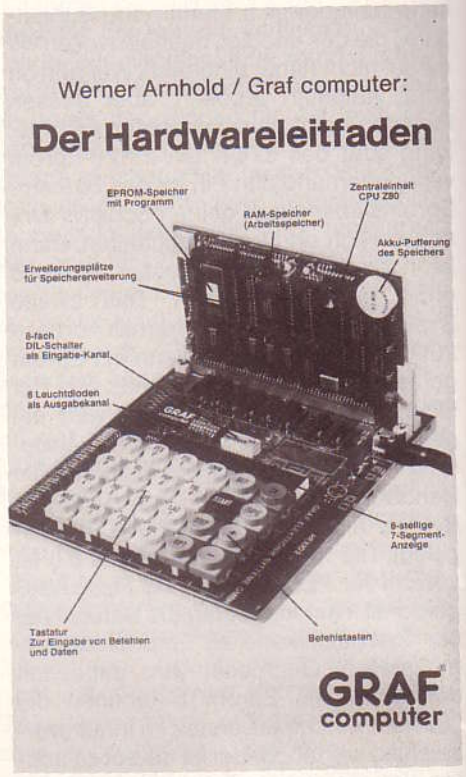
Um nun dieses Format zu verwenden, muß der Initialisierungs-Befehl für eine Grafik-Zeile von ESC 'K' 0 1 auf ESC '*' 5 02. Dabei schaltet ESC '*' 5 auf die Plottergrafik und die Werte 0 2 bedeuten 0+2 x 256, also 512 Punkte/Zeile. Da unsere HARDCOPY-Karte aber „nur" 256 Punkte pro Bildschirmspalte (d. h. Druckerzeile) liefert, muß die Ausgabe verdoppelt werden. Das geschieht durch den zweifachen Aufruf der Druckerausgabe in der Routine PRTLINE, so wie für den Seikosha-Drucker wie in der *LOOP* 13a beschrieben. Damit werden zwar horizontale Linien dicker als vertikale, aber es entsteht eine 1 :1 Hardcopy vom Monitorbild.

Um das so geänderte Programm sinnvoll nutzen zu können, muß es beim Z80 hinter der Flomon-Einsprungtabelle, oberhalb des CP/M abgelegt werden, wie im Handbuch beschrieben. Bei der 680xx-Version ist speziell unter JADOS 2.1 ein anderes Verfahren sinnvoll. Es empfiehlt sich, das Programm nicht mit dem RAM hinter dem Grundprogramm unterzubringen, da dadurch die Symboltabelle beschränk wird, bzw. andere Grundprogrammerweiterungen gestört werden könnten. JADOS 2.1 bietet aber die Möglichkeit, Programme resident im Arbeitsspeicher zu halten. Installiert man COM-Programme im Speicher, so werden sie sogar an der installierten Stelle gestartet. Dieses Verfahren bietet sich besonders für Programme wie HARDCOPY oder aber auch spezielle Druckertreiber an. Daher habe ich das Programm so umgeschrieben, daß es voll relokatibel ist (Voraussetzung für COM-Files) und sich selbst initialisiert. Es braucht nach dem Übersetzen mit ASS HARDCOPY. ASM nur mit REN HARDCOPY.68K HARDCOPY.COM umbenannt und dann mit INST HARDCOPY.COM installiert zu werden. Ruft man es dann mit HARDCOPY auf, so wird es im Speicher an der installierten Adresse gestartet und bindet sich dann in die CI-Routine des EGRUND, in die Interrupt-Vektortabelle und in die RL-Basic-Sprungadresse ein. Danach ist das Programm dann durch Eingabe von (CTRL-§) aufrufbar, immer wenn eine Eingabe erwartet wird. Zusätzlich kann man das Programm jederzeit durch einen Taster zwischen MASSE- und der INT-Leitung starten. Die dritte Möglichkeit ist ein Aufruf über den RL-BASIC-Befehl HARDCOPY.

Beim Z80 funktioniert allerdings nur der Aufruf über (CTRL-§), da ich das HEBAS nicht kenne und ein Interrupt-Aufruf wegen des Interrupts von der Floppy-Karte nicht möglich ist. Ein Aufruf aus HEBAS müßte aber mit CALL $0F80B möglich sein. Bei MBASIC ist es, wegen der anderen Darstellung der Hexzahlen, der Befehl CALL –2037.

Wer sich das Eintippen der Listings sparen möchte, kann eine lauffähige Version für den Z80 unter CP/M 2.2 oder den 68008 unter JADOS 2.1 zusammen mit dem Quelltext für 5,– DM bei Einsendung einer Leerdiskette mit einem frankierten Rückumschlag bei mir erhalten. Das Programm ist für den inzwischen wohl weit verbreiteten Drucker STARNL-10 mit IBM-Interface geschrieben und mit einem Z80 bzw. 68008 getestet. Es ist auf allen Epson/IBM-kompatiblen Druckern lauffähig, die den Plottergrafik-Modus mit 72 Punkten/Inch (576 Punkte/Zeile) beherrschen. Durch Anpassen der Variablen CPU im Quelltext läßt es sich auch für den 68000 oder 68020 übersetzen.

Uwe Koch,
Frankenstraße 25 · 5880 Lüdenscheid,
Telefon: (02351) 26192)



Neu: Das Buch für den Einsteiger
Best.-Nr. 11112 **DM 9,50**

Der Steckbrief

Lebenslauf:

Vorname: Ralph
Name: Dombrowski
geboren: 02. 04. 1968 in Elmshorn
Wohnort: Gr. Deichstraße 33, 2208 Glückstadt
Ausbildung: 4 Jahre Grundschule danach 9 Jahre Gymnasium mit Abitur als Abschluß im Mai dieses Jahres. Zur Zeit Ableistung des Grundwehrdienstes beim Jägerbataillon 67 in Itzehoe.

Ich programmiere seit etwa drei Jahren und habe mit einem Taschencomputer angefangen. Habe dann in der Schule in einer Computer AG kurze Zeit an einem Apple gearbeitet. Als ich mir einen eigenen Computer kaufen wollte, schlug mir mein Lehrer vor, selber einen Computer zu bauen, da im NDR auch gerade die Sendung zum NDR-Klein-Computer lief. So habe ich zu diesem System gewechselt und bin bis jetzt aktiv dabei geblieben.

Mit der Software-Partner-Tagung begann

meine „Laufbahn" als S.-Partner. Ich programmiere hauptsächlich in 68000 Assembler, weil dies die schnellste Sprache ist und sie von Anfang an zur Verfügung stand.

Folgende Produkte sind von mir für die Firma Graf entwickelt worden:

RD-Diskedit (Umfangreicher Diskettenedit für die 68000-Serie)
RD-Videoverwaltung (Verwaltungsprogramm für bis zu 350 Videokassetten)
RD-Fly Away (Flugzeugspiel)
RD-Demo (3-D-Grafik-Demo für die Cebit 1987)
RD-Breakout (Spiel für die TOOL-Diskette)
RD-DIS Assembler (DIS Assembler mit allen 68008- 68020 und 68881-Befehlen)

Zur Zeit arbeite ich an einem neuen Grundprogramm, das viele neue oder verbesserte Routinen enthalten soll. Auch soll die Bedienungsfreundlichkeit durch eine andere Menüform verbessert werden. Auch Routinen für die Hardcopy-Maus-Baugruppe, die COL256 und die neue GDP sind geplant.

Ich arbeite sehr gerne mit dem NDR-Computer, da ich der Meinung bin, daß man mit diesem System sehr viel lernen kann und man Einblick in wirklich alles hat. Außerdem habe ich hier einen guten Kontakt direkt zur Herstellerfirma, so daß ich Probleme auch einmal direkt besprechen kann, was bei anderen Firmen nicht so möglich ist. Auch ist jetzt mit den neuen Prozessoren 68020 und 68881 und den Farbgrafik-Karten ein sehr leistungsfähiges System vorhanden, das jetzt noch durch umfangreiche Software unterstützt werden muß, womit ich ja dauernd beschäftigt bin.

BRIEFE

Sehr geehrte Herren,

mit Interesse warte ich als Laie, der nicht so „tief drinsteckt", immer auf die neue *LOOP* und die neuen Programme zum Abtippen. Habe mir extra eine Kopflupe gekauft. Lassen Sie nur die Größe der Programme, dann steht mehr drin.

Ich würde es sehr begrüßen, wenn die Programme auf Disketten geliefert werden könnten, eventuell für einen bestimmten Zeitraum und je nach Speichermöglichkeit der angebotenen Diskette; jedoch dann passend für 80 Spuren DS Jados/RL-Basic.

In jedem Heft stolpert man als Laie über neue Fachausdrücke. Warum haben Sie das im ersten Heft angefangene Lexikon nicht weitergeführt?

Begrüßen würde ich die Durchführung Ihrer Planung, ein Betriebssystem zu integrieren, das MS-DOS-Dateien lesen kann. Ich gehe davon aus, daß man dann z.B. Commodore-Disketten lesen und dann entsprechend für NDR-Klein-Computer umbauen kann. Ich nehme an, daß Jados bedeutend besser ist als das fast überall eingesetzte MS-DOS-System.

Weiter begrüße ich die geplante monatliche Erscheinungsweise von *LOOP*. Bei der weiteren Aufnahme von Anzeigen sollte aber darauf geachtet werden, daß die angebotenen Artikel in Ihr System integriert werden können und das „wie" in der *LOOP* besprochen wird.

Dieter Lindinger,
Gutenbergstraße 46 · 8502 Zirndorf

Antwort LOOP:

Danke für Ihren Brief. Die Sache mit den Fachwörtern ist leider nicht so einfach. Wo sollen wir anfangen, was ist ein „Fachwort", und was ist „Umgangssprache"? Immer wieder neu? Alle?

MS-DOS-Dateien können Sie (mit MCOPY) bereits lesen (und schreiben).

Vielen Dank für Ihr Antwortschreiben bezüglich meiner Kritikkarte. Ich hatte überhaupt nicht damit gerechnet, daß die Karte irgendwelche Beachtung finden würde, da meine sonstigen Bemühungen bei GES auf Kritikpunkte aufmerksam zu machen, bei Ihren Mitarbeitern zur Kenntnis genommen wurden, dann aber keine weitere Beachtung mehr fanden. Bezüglich der doppelten Postlaufzeit kann ich sagen, daß Ihr Schreiben nur vier Tage, im Gegensatz zu den sonst üblichen ein bis zwei Wochen, unterwegs war.

Die CPU68000-Platine zeigte keine Funktionsmängel, war jedoch nicht in der sonst üblichen Qualität gefertigt. So fehlte der Bestückungsaufdruck, der Lötstopplack war versetzt und die Baugruppe besaß noch nicht das Normformat. Bei einer Platine zum Preis von DM 60,– dürfte so etwas nicht vorkommen. Diese Tatsache empfand ich als Unverschämtheit, da hierbei der Preis sehr hoch und die Verarbeitungsqualität sehr niedrig angesiedelt war.

Zum Punkt Entwicklungskosten läßt sich sagen, daß die CPU68000 nur eine Abwandlung der CPU68K ist, die ja schon in vierstelligen Stückzahlen verkauft worden sein dürfte. Kosten dürfte hierbei also nur das neue Layout bereitet haben.

Nach Erhalt Ihres Schreibens habe ich mit verschiedenen anderen Besitzern des NDR-Klein-Computers gesprochen. Dabei wurden die folgenden Punkte bemängelt:

Die Preise für die IBM-Baugruppen sind noch zu hoch. Konkurrenzprodukte sind preiswerter und qualitativ gleichwertig. Hierbei ist es egal, ob es sich um ein Motherboard oder um einen passiven Bus mit einer CPU-Steckkarte handelt.

Die Handbücher dürften eigentlich nicht so benannt werden. Packbeilage wäre hierfür eher angebracht. Preise bis zu DM 20,– sind maßlos überzogen.

Die Programme in der *LOOP* sollten auf Diskette erhältlich sein (z. B. vierteljährlich zum Preis von DM 10,– bis DM 15,–).

Die Mailbox ist sehr dürftig. Viele Optionen sind außer Betrieb, so. z. B. Preise. Dort wird darauf verwiesen, daß man eine Preisliste schriftlich anfordern soll.

Entwicklungskosten müssen als Argument für alles mögliche herhalten, dadurch wird dieses Argument jedoch unglaubwürdig.

Der NDR-Computer ist von seinem Aufbau her sehr gut (modular und offen). Daß er nicht von viel mehr Leuten nachgebaut wird, liegt an den hohen Preisen für die Bausätze.

Viele Besitzer wechseln auf andere System, da dort eine bessere Softwareunterstützung vorhanden ist. Siehe auch die Rubrik Kleinanzeigen in *LOOP* 13a.

Ein Messerabatt von 5% ist wohl mehr als dürftig. 10% wären wohl eher ein Kaufanreiz.

Die Adressen in der *LOOP* sollten immer mit der Telefonnummer veröffentlicht werden. Oftmals kann man Probleme am Telefon gleich besprechen.

Die Zeitschrift *LOOP* ist zu kommerziell. Groß und breit wird darüber berichtet, wenn es darum geht Artikel der Fa. Graf zu verkaufen. Direkte Hilfe für den Anwender ist nicht vorhanden.

Auskünfte des Elektronikladens sind kundenfreundlicher. Dort befindet sich ein Ansprechpartner mit technischem Verständnis. Bei GES wird jedoch nur hin- und herverwiesen.

Ein ATARI-Emulator würde viele Softwareprobleme lösen und die 68000-Serie besser auslasten.

Die in der *LOOP* abgedruckten Beiträge müßten besser erläutert werden (Anwenderhinführung). Mittlerweile werden wenigstens die Hardwarevoraussetzungen genannt, welche man zum Ablauf des Programms benötigt.

Die Sound-Baugruppe wird nicht in ausrechendem Maße unterstützt (z. B. bei Fehlermeldungen des CP/M).

Nun noch eine Liste der Produkte, bei denen die Preise gesenkt werden sollten:

Alle „Handbücher".
Das neue Gehäuse (Nr. 10673); DM 298,– ohne die Haltewinkel sind wohl etwas zu viel. Vergleichbare Gehäuse anderer Hersteller (z. B. Fa. Knürr) sind bereits ab DM 120,– zu erwerben. Taiwan-Produkte noch günstiger.
Alle IBM-Baugruppen. Eine AT-CPU-Baugruppe, die auch noch einen Slot auf dem Bus belegt, für DM 1498,– ist immer noch zu teuer (preislich interessant erst ab DM 900,–).
Treiberprogramme (z. B. für die Farbgrafik-Karten).
Festplatten.
Farbgrafik-Baugruppen, bzw. die Platinen.
EPROMs (Grundprogramme, Sprachen usw.).

Ich hoffe, daß dieses Schreiben Ihnen ein wenig den Standpunkt einiger NDR-Computer-Benutzer verdeutlicht und daß es auch in Zukunft noch attraktiv sein wird, einen Computer selber zu bauen. Voraussetzung hierfür ist allerdings eine bessere Softwareunterstützung.

Christian Braden,
Rheinstraße 97 · 6538 Münster-Sarmsheim · Tel.: (06721) 43830

Antwort LOOP:

Es ist sehr wohl so, daß die Kritikkarten bei uns sehr ausführlich gelesen werden. Nicht umsonst steht direkt Herr Graf darauf. Er behält sich das Studium dieser Kritikkarten und deren Auswertung vor.

Wir sind alle nur Menschen und bemüht, aus unseren Fehlern zu lernen und diese zu verbessern. Deswegen möchten wir uns für Ihr langes Schreiben noch einmal bedanken.

Die folgenden Ausführungen sollen keine Rechtfertigung darstellen, sondern Ihnen lediglich den Standpunkt unserer Seite zu erläutern versuchen.

Der Preis einer Leiterplatte besteht grundsätzlich aus zwei oder drei Faktoren. Die Materialkosten der Leiterplatte, die umgeschlagenen Kosten für Entwicklung und Layout, der Gewinn.

Die reinen Leiterplattenkosten sind abhängig von der Auflage und liegen, da wir ausschließlich bei Qualitätsherstellern in Deutschland fertigen lassen, zwischen DM 15,– für eine kleine durchkontaktierte Leiterplatte und DM 200,– für eine Multilayer-Karte. Die CPU-Karte dürfte durch die realtiv geringe Auflage bei ca. DM 38,– Herstellkosten liegen. Layout-Karten, einschließlich Layout-Material und Film sind für eine durchschnittliche Leiterplatte im Bereich von etwa DM 3000,– anzusiedeln. Dies ist wohlgemerkt der reine Layout-Preis. Hier ist noch keine Entwicklung, Prüfung etc. enthalten. Summiert man Entwicklung, Prüfung sowie verschiedene Revisionen auf, werden hier schnell Größenordnungen von DM 10.000,– bis DM 20.000,– erreicht. Bleiben wir im unteren Bereich bei DM 9.000,–. Diese DM 9.000,– müssen auf die voraussichtliche Abnahme der Leiterplatten – rechnen wir zunächst mit ca. 250 Stück – umgelegt werden. Dies ergibt einen Aufschlag von DM 36,– pro Leiterplatte. Auch bei 500 Stück (eine für die CPU68000 normale Stückzahl) reduziert sich der Aufschlag nur auf DM 18,–.

Im dritten Faktor, dem Gewinn, müssen noch die Finanzierung und Verzinsung der Lagerhaltung, der Fertigungskosten berücksichtigt werden.

Zusammenfassend ist zu sagen, daß wir auch bei einem Preis von DM 60,– pro Leiterplatte keinen vernünftigen Deckungsbeitrag erwirtschaften.

Wir glauben jedoch, im Gegensatz zu vielen Kollegen, daß es für unsere Kunden fair ist, wenn wir die Leiterplatten leer anbieten.

Damit reguliert sich auch der Preis für die Bausätze; denn sobald die Einzelteile am Markt in der Summe preiswerter erhältlich sind, als bei uns im Bausatz, gehen unsere Bausatzumsätze zurück und wir müssen reagieren. Dies zur Kalkulation.

Die Preise für die IBM-Baugruppen sind sehr stark im Schwanken. Als wir die IBM-Baugruppen zur Hannover-Messe CeBit vorstellten, mußten wir amerikanische Leiterplatten einsetzen, die sehr teuer waren. Mittlerweile hat sich der Preis auf DM 1098,– für die CPU-Baugruppe eingependelt. Dies ist im Vergleich für die DM 1000,– für die sonstigen Mainboards ein sicher relevanter Preis, wenn man die Taktrate mit 10/12 MHz und die fortschrittliche Technologie (CHIP Technology) mit ins Kalkül zieht.

Wir leben jedoch auch nur vom Umsatz und sind stets bemüht, günstigere Einkaufspreise zu erhalten.

Nicht ganz so hart sollten Sie unsere Handbücher kritisieren. Natürlich gibt es einige Handbücher, die sehr dünn ausgefallen sind; ganz einfach, weil man über ein solches Produkt (z. B. BUS) nicht sehr viel schreiben kann; sehr viele unserer Handbücher werden jedoch besonders von unseren Kunden gelobt, wie die Handbücher für die FLO oder die SBC3.

Bezüglich der *LOOP*-Programme überlegen wir uns einiges. Wir werden sicherlich hier eine befriedigende Lösung finden. Wir warten noch die Reaktion der *LOOP*-Leser ab.

Völlig recht haben Sie mit Ihrer Kritik zur Mail-Box. Dies hängt effektiv nur damit zusammen, daß wir derzeit keinerlei Kapazität frei haben, diese Mail-Box ordentlich zu bedienen.

Wir sind gerade daran, unser Rechnersystem zu vernetzen und überlegen uns einen teilweisen Zugriff von der Mail-Box direkt auf den Rechner zu ermöglichen. Damit wären z. B. Informationen über den Preis sowie über die Lieferfähigkeit direkt möglich. Hier wollen wir auch unsere Filialen miteinbinden. Dies wird jedoch noch einige Zeit in Anspruch nehmen.

Messerabatt und sonstige Rabatte sind ein betriebswirtschaftlich heißes Eisen. Jeder Rabatt muß vorher in den Preis einkalkuliert werden; jeder Anwender, der „normal" ohne Rabatt kauft, ist damit benachteiligt. Deswegen gewähren wir nur sehr geringe Rabatte und sehen die 5% Messerabatt lediglich als Anreiz an.

Die Adressen in der *LOOP* sind in der *LOOP* 14 erledigt worden. Mit an der *LOOP* sind die Anwender die Gestalter. Natürlich ist *LOOP* immer ein Sprachrohr unseres Hauses, sehr gerne drucken wir aber besonders Anwenderberichte ab, die sich mit Tips und Tricks und Hilfen beschäftigen.

Über die Aussagequalität von Elektronikladen oder uns, ist es müßig zu diskutieren. Wir bemühen uns beide, möglichst ausführliche und umfangreiche Informationen zu geben.

Der Atari-Emulator ist bereits von einem externen Software-Haus in Entwicklung und dürfte uns bald vorgestellt werden.

Die von Ihnen genannten, zu hohen Preise, haben wir sehr ernst zur Kenntnis genommen und werden hierüber in der nächsten Preiskalkulation befinden. Grundsätzlich möchten wir aber die Qualität unserer Produkte hoch halten; Gehäuseimporte lehen wir daher zunächst ab.

Wir setzen auf das Pferd NDR-Computer und werden dieses Produkt auch in Zukunft sehr stark forcieren und unterstützen. Hierbei sind wir auf die Hilfe unserer Kunden und besonders die fundierte Kritik, wie Sie sie in diesem Schreiben gegeben haben, angewiesen. Noch einmal möchten wir uns dafür bedanken.

Wir werden Ihr Schreiben sowie diesen Brief in der nächsten *LOOP* 15 gerne veröffentlichen.

Kleinanzeigen

Verkaufe: GDP 64K, KEY, TAST1 komplett mit Originalgehäuse und -kabel, EBASIC, EGRUND 2000, ESKOP, Handbücher einschl. FLO2, Sonderheft 1+2, LOOP1–4; alle Platinen sind fertig aufgebaut und funktionsfähig, Verhandlungsbasis: DM 350,–, einzeln: ca. ½ Bausatzpreis, Josef Eisele, Tel. (08807) 8254

Lehrgänge zum NDR-Computer

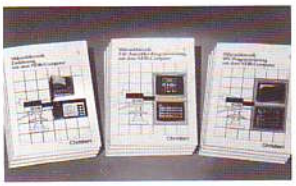
Wenn Sie sich intensiv mit der Hard- und Software des NDR-Computers beschäftigen wollen, helfen Ihnen diese Kurse weiter:

Kompakt-Kurs Elektronik

Ca. 200 Seiten im Format A4 mit Experimentiermaterial, Tonbandkassette und Flip-chart zur Einführung in die Grundlagen der Elektotechnik 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

35519

Briefe - Kontakte - Kleinanzeigen

Die **Volkshochschule Bremerhaven,** Friedrich-Schiller-Haus, Lloydstraße 15, führt Kurse über **Einführung in die speicherprogrammierte Steuerung (SPS)** auf Basis des NDR-Computers durch. Spezielwissen wird nicht vorausgesetzt; geringe Vorkenntnisse aus der Elektronik erleichtern jedoch das Verständnis des Lernstoffs. Teilnehmerzahl: 14.

Dienstag, 18.45 – 21.00 Uhr, ab 15. September 1987. Leitung: Karl-Heinz Donner, Dietrich Nickel. Ort: Schulzentrum Bürgerpark, Gewerbliche Lehranstalten, Raum 1038, Haus 1, Georg-Büchner-Straße 7. Kosten: 18 DM (Material/Miete). Einschreibegebühr: 18 DM (E: 12 DM), sechs Abende.

NDR 68000/8: JADOS-Entwickler verk. ● schnellen Editor JEDI! (ausführliches Info gegen Rückporto) ● mausgesteuertes REVERSI DM 39,– ● menügesteuerten DIASSEMBLER, mit Hexdump und Abspeichern auf Datei DM 50,– ● Diskettenmonitor mit Sektoren-Editor, Suchfunktion und Dateimodus DM 35,– + Porto 3,50 bei V-Scheck, 5,– bei NN! auf Disk 3½" oder 5¼" unter JADOS. K. Janßen, Hannixweg 74, 4150 Krefeld 1.

Verkaufe: NDR-Computer betriebsbereit im Schroffgehäuse, CPU 64180, SBC2, BUS3, IOE, ROA64, POW5, Trafo, CAS, KEY, TAST1 kompl., GDP 64K, 2 ERP.-Grund., Basickurs, EBASIC, EGRUND, Sonderhefte, alle LOOPs, Verh.-Basis 680,– DM. Georg Bögelein, Ganghoferweg 3, 8520 Erlangen, Tel.: (09135) 8444

Verkaufe: NDR-Computer, betriebsbereit, wegen Zeitmangel. Floppy-Disk., Kabel, gr. Tastatur, Monitor, Grundp. CP/M Disk. SCB 2, CAS, POW 5, CPU Z80, 2 x Bus, FLO2, Bankb., CPU 68K, RAM 64/256, ROA 64, GDP 64K, KEY, IOE, CEN, NE2, Gehäuse, versch. EPROMs u. RAMs, gesamte Literatur, diverse Teile. Paul Hammel, Schloßgasse 1a, 8713 Marktbreit, Tel.: (09332) 3926 (abends)

NDR-68000/08-Computer: Verk. 1.) ersten an JADOS angepaßten 2-Paß-Makro-Disassembler. Selbständiges Erzeugen aller Labels (Sprungziele) und Rückwandlung von Texten und Zeichenketten! 2.) Progr. zum grafischen Ausdrukken eines Jahreskalenders auf 12 DIN A4 Seiten mit einem EPSON o. kompatiblen Drucker bis zum Jahr 2013. Benötige JADOS. Info: Olaf Reinhold, Butterberg 1B, 3300 Braunschweig (Rückporto).

Suche Kontakt zu anderen NDR-Computer-Besitzern. Anton Prinzbach, Kirschenweg 14, 7601 Durbach, Tel.: (0781) 42352.

Impressum:

LOOP Zeitung für Computerbauer

Herausgeber: Gerd Graf

Redaktion: Rolf-Dieter Klein, Gerd Graf

Gestaltung und Druck:
Karl-Heinz Rieder, Kempten

Herstellung und Anzeigenverwaltung:
GES GmbH
Magnusstraße 13, 8960 Kempten

Anzeigenpreisliste 1/84

Neue Artikel – Neue Preise!

Texteditor (R. Jork)
für 68000-Systeme, JADOS und CP/M68K

Best.-Nr.	Bezeichnung	Preis DM
11109	3½"	99,00
11110	5¼" 80 Spuren	99,00

Disassembler
für 680xx, auch 68020

Best.-Nr.	Bezeichnung	Preis DM
11101	3½"	49,00
11102	5¼"	49,00

HEBAS auf EPROM
für SBC3 ohne Floppy

Best.-Nr.	Bezeichnung	Preis DM
11103	EPROMs	98,00
11104	Listing dazu	20,00

Schach, neueste Version

Best.-Nr.	Bezeichnung	Preis DM
10873	3½"	79,00
10874	5¼" 80 Spuren	79,00

MODULA-UPDATE auf V. 2.30

Best.-Nr.	Bezeichnung	Preis DM
10878	Nur Modula 5¼"	100,00
10970	Nur Modula 3½"	100,00
10879	Modula Update 5¼" **und** Editor dazu	170,00
	dto., 3½"	170,00

RAM-Floppy 512K

Best.-Nr.	Bezeichnung	Preis DM
11100	Handbuch	20,00
11111	Leiterplatte	50,00
10780	Bausatz	398,00
10781	Fertiggerät	468,00

Window-Verwaltung für NDR und mc

Best.-Nr.	Bezeichnung	Preis DM
11126	Windowverwaltung 3½", 80	49,00
11127	Windowverwaltung 5¼", 80	49,00
11128	Windowverwaltung 8"	49,00

Handbuch (original)
für TEAC-Laufwerke, IBM-kompatibel (engl.)

Best.-Nr.	Bezeichnung	Preis DM
11107	5¼" GFR55	15,00
11108	3½" 35 GFN-21	15,00

ROA 256/1M

Best.-Nr.	Bezeichnung	Preis DM
11028	Handbuch	20,00
11029	Leiterplatte	40,00
11026	Bausatz ohne Batt.	98,00
11027	Fertiggerät ohne Batt.	158,00
11030	Bausatz **mit** Batt.	148,00
11031	Fertiggerät **mit** Batt.	198,00

FLOMON V. 4.2

Best.-Nr.	Bezeichnung	Preis DM
11095	EPROM	75,00
11096	Listing dazu	20,00

Preisänderungen alter Artikel

Z80 Emulator

Best.-Nr.	Bezeichnung	Preis DM
10609	5¼" 80 Spuren	298,00
11065	3½"	298,00
10210	8"	338,00

dBase, Wordstar, Multiplan

Best.-Nr.	Bezeichnung	Preis DM
10331	Multiplan 5¼" 80 Spuren	99,00
10670	Multiplan 5¼" 40 Spuren	139,00
10671	Multiplan 3½" 80 Spuren	139,00
10672	Multiplan 8"	139,00
10442	Wordstar 5¼" 80 Spuren	99,00
10667	Wordstar 5¼" 40 Spuren	139,00
10668	Wordstar 3½" 80 Spuren	139,00
10669	Wordstar 8"	139,00
10193	dBase 5¼" 80 Spuren	99,00
10664	dBase 5¼" 40 Spuren	139,00
10665	dBase 3½" 80 Spuren	139,00
10666	dBase 8"	139,00

Diverse Baugruppen NDR-Computer

Best.-Nr.	Bezeichnung	Preis DM
10401	SPRACHE Bausatz	198,00
10402	SPRACHE Fertiggerät	258,00
10702	Echtzeituhr Smart Watch	98,00
10071	AD8 x 16B	98,00
10072	AD8 x 16F	158,00
10135	CENT2B	98,00
10136	CENT2F	148,00
10378	SASINDRB	89,00
10379	SASINDRF	129,00
10515	RFLOB (128K)	198,00
10516	RFLOF (128K)	248,00
10621	HEXIO2F / Steckernetzteil	248,00
10622	HEXIO2B / Steckernetzteil	198,00
10626	IOE2B	98,00
10625	IOE2F	148,00
10629	ROB2F	248,00
10631	ROB2B	198,00
10634	KEY2B	148,00
10645	FLO3B	248,00
10644	FLO3F	298,00
10774	COL256 Speichererweiterung	98,00
10067	AD10 x 1 B	198,00
10068	AD10 x 1 F	268,00
10341	POW26/22 V2 (mit Wandler) F	98,00
10776	Monitor BGC36 Grundig	1998,00
10286	HEXIOH	20,00

Diverse Programme NDR-Computer

Best.-Nr.	Bezeichnung	Preis DM
10197	EASS0-3 Grundprogramm	95,00
10200	EBASIC	50,00
10201	EBASIC2	50,00
10202	EBIO	35,00
10206	EDATEY	45,00
10207	EDEMO	20,00
10213	EG 68000 Grundprogramm	95,00
10214	EGOSI	50,00
10215	EGOSI2	50,00
10216	EGOSI68 68000	50,00
10217	EGOSICOMP 68008	50,00
10219	EGRUND	30,00
10220	EGRUND2	30,00
10221	EGRUND2000	30,00
10228	EJOGIMON 68008	50,00
10687	EJOGIMON 68000	50,00
10231	EPASCAL 68008	50,00
10232	EPASCAL 68000	50,00
10244	EJOGIDOS 68008	50,00
10686	EJOGIDOS 68000	50,00
10246	EZASS2	50,00
10610	EFLOMON V3.2	49,00
10656	STRUCTA 3½"	98,00
10522	STRUCTA 5¼"	98,00

Diverse Baugruppen mc-CP/M-Computer

Best.-Nr.	Bezeichnung	Preis DM
10499	OUT1B (mc-Computer)	198,00
10500	OUT1F (mc-Computer)	298,00
10511	RAM16B	148,00
10525	SYS1B (mc)	148,00
10526	SYS1F	198,00
10530	TERM1B (9366)	298,00
10531	TERM1F	398,00
10532	TERM1B (9367)	398,00
10533	TERM1F	498,00
10539	TERMC Platine + Eprom	98,00
10544	Termplatine + Eprom	98,00
10475	FLO2MCB (FLO + SASI)	298,00
10728	FLO2MCF (FLO + SASI)	358,00
11042	FLO3MCF	298,00

Disketten für NDR und IBM (mc-modular-AT)

Best.-Nr.	Bezeichnung	Preis DM
10660	Disketten 5¼" QD (NDR) Stück	4,00
11114	Disketten 5¼" HD (AT) Stück	8,00
11113	Disketten 3½" (NDR + AT) Stück	14,00

3½"-Floppy und Speicher zum mc-modular-AT

Best.-Nr.	Bezeichnung	Preis DM
11084	3½" Floppy IBM	378,00
11105	RAM256k x 1, 120 ns (9 Stück für 256KB benötigt!)	12,00

GRAF ELEKTRONIK SYSTEME GMBH

Magnusstraße 13 · Postfach 1610 · 8960 Kempten (Allgäu) · Telefon: (0831) 6211

Teletex: 831804=GRAF · Telex: 17831804=GRAF · Datentelefon: (0831) 69330

GRAF® computer