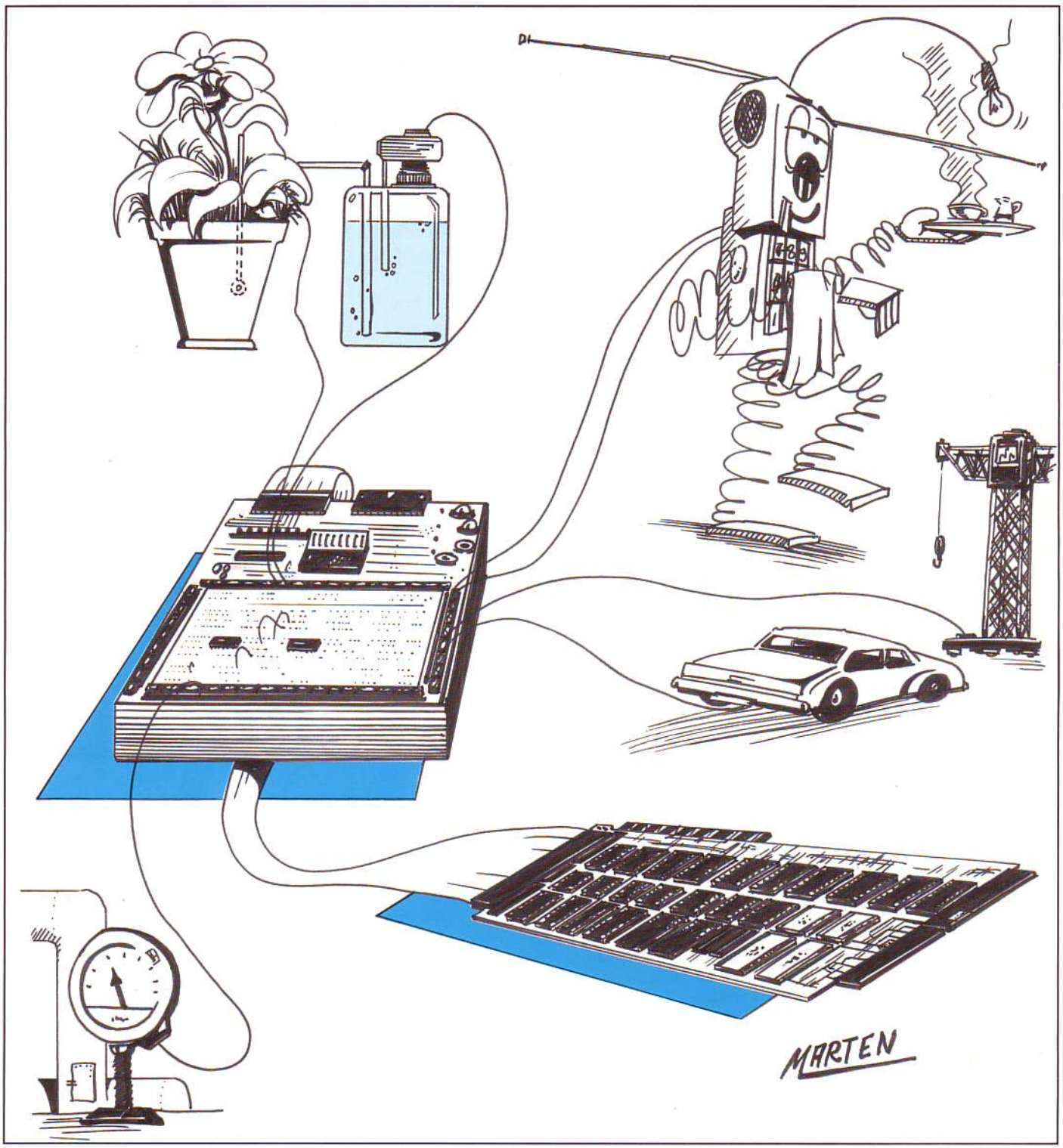
LOOP

22

Juni 1989

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

DM 3,50



Leitartikel

Multi I/O Interface

Universelle Ein- und Ausgabe für NDR- und PC-Bus : Analog/Digital-Wandler, Roboter, Meßgeräte, Maschinen , durch die neue I/O - Karte ansteuerbar ...

CPU Z80

Ökosystem eines Interpreters
Im 3. Teil gehts auf Schnitzeljagd
22/5

Graphik zu Fuß programmiert
Eine GDP - Routine
22/7

Programm "List"
komfortabel und übersichtlich
22/11

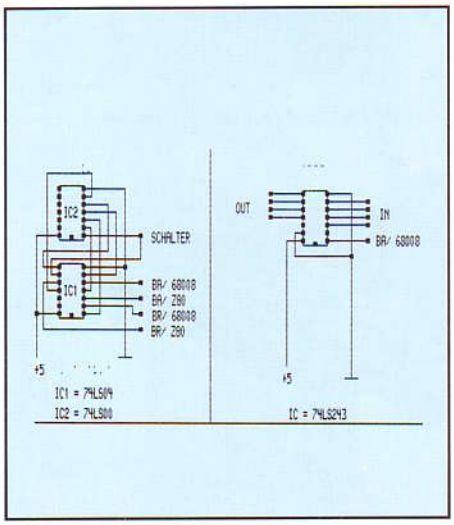
ZEAT patcht ZEAT
22/12

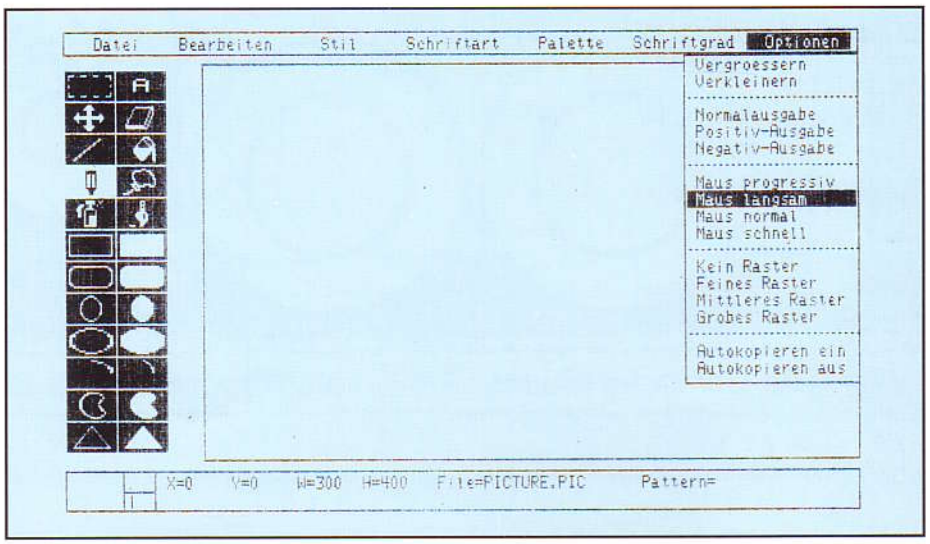
Einstellen der Baud-Rate der SER-Baugruppe unter CP/M 2,2
Menüwahl der Seriellen Schnittstellen für die Installation der Baudrate
22/15

CPU 680XX

Jetzt wird gelesen
22/17

Z80 und 68008 auf einem Bus
Zwei kleine Schaltungen, um beide Prozessoren auf einem Bus laufen zu lassen
22/19





Hauptmenü des Zeichenprogrammes "DRAW"

Es geht auch umständlicher
oder wie man in Assembler schnelle Programme schreibt
22/23

Windows unter Modula 2
Ein kleines Programm zur Window-Verwaltung
22/25

Zeichnen mit dem Computer
Ein umfangreiches Programm - etwa 156 verschiedene Funktionen lassen sich mit der Maus anklicken und ausführen
22/27

Fenster
RL-Basic-Programm, das es ermöglicht, Bildfenster auf der GDP64k auszugeben
22/28

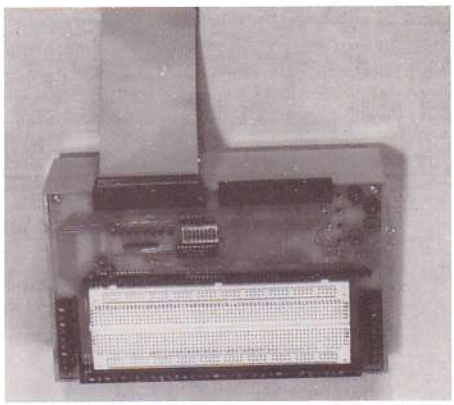
Grundlagen zur Interpolation
Ein Programm in Assembler
22/29

OS-9/68000
4. Teil: Aufbau der Treiber
22/30

mc-modular AT

NEAT-Chipsatz für den mc-modular AT
Zwei preislich günstige Lösungen, einen schnellen, leistungsfähigen Rechner zu erhalten
22/32

ProfiLog
Für Profi`s oder solche, die es werden wollen
22/14



Multi I/O - Karte

Impressum

LOOP Zeitung für Computerbauer

Herausgeber: Gerd Graf

Redaktion: Rolf Dieter Klein, Gerd Graf, Christoph Köhler

Gestaltung: byte & byte software Elisabeth Mayr, MARTEN

Druck: Karl-Heinz Rieder, Kempten

Herstellung und Anzeigenverwaltung: GES GmbH, Magnusstraße 13, 8960 Kempten

Anzeigenpreisliste: 6/89

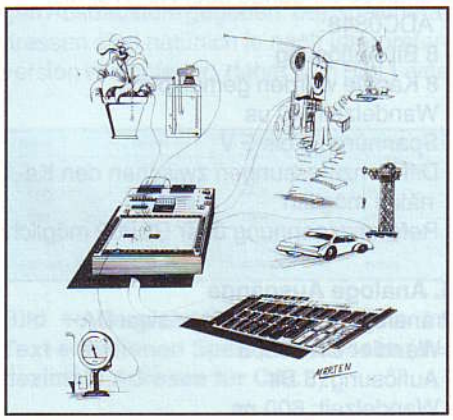
Rubriken

Editorial
Es lebe der NDR-Computer 22/3

Comic - Strip
Parity NONE 22/31

Tips und Tricks
MS/DOS auf der CPU 8088 22/34

Aus der Technik
Key 3 - die neue Tastaturkarte 22/35



LOOP-Titelbild:
Multi I/O - Karte

In eigener Sache:

Software-Partner stellen sich vor
Rolf Lobreyer 22/38

NDR in der Fortbildung
Spengler 22/19

Für Sie gelesen:

DBASE III-Plus-Schulung
Dr. Albrecht 22/4

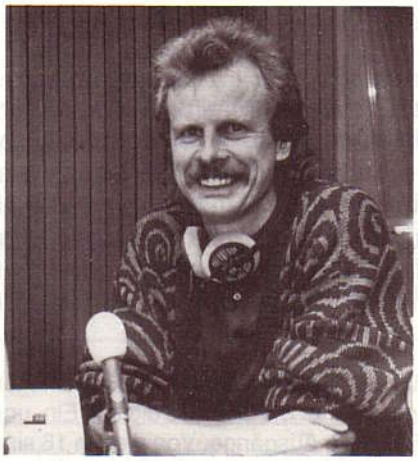
Clipper-Handbuch
Michael Aselmann-Frasch 22/33

Turbo C
Gerhard Renner 22/34

Erfolgreiches Programmieren von 68000er Systemen in Assembler und C 22/37

Kleinanzeigen 22/37

Editorial



Es lebe der NDR-Computer!

Mehr und mehr wird der NDR-Computer zur innerbetrieblichen oder schulischen Ausbildung eingesetzt. Neben den vielen Vorteilen des modularen Systems kritisierten die Anwender lediglich den mechanischen Aufbau des NDR-Systems.

Ein offenes System ist für den rauhen Schulalltag nicht zuverwenden, und der NDR-Computer im Gehäuse eignet sich nicht für Messungen direkt auf der Baugruppe.

Wir haben schon seit Jahren so etwas geahnt und deshalb bei Layouts die "GES-Norm" entworfen. Sie besagt unter anderem, daß, wo immer es sinnvoll und vom Platz möglich ist, auf der Baugruppe neben dem NDR-Bus auch der ECB-Bus untergebracht werden soll.

Heute können wir die Früchte dieser Norm ernten: Wir bauen die wichtigsten Teile des NDR-Computers nun auf Europakarten mit ECB-Schnittstelle und haben damit ein neues Ausbildungssystem auf 19 " - Basis geschaffen.

Damit sind alle Wege geebnet, um das modulare NDR-System in der Ausbildung einzusetzen. Einige Baugruppen müssen noch umgebaut werden, so die Bustest, die es ermöglichen wird, ganz "zu Fuß" Adressen und Daten in den Speicher ein- zugeben. Wir haben dann ein komplettes, breites Ausbildungssystem, von Hexeingabe über CP/M bis zu MS-DOS und OS /9.

Alle neuen Baugruppen, so die neue KEY3 für IBM-Tastaturen mit eigenem Prozessor und die im Layout befindliche EGA-Graphik entsprechen natürlich ebenfalls der GES-Norm - ECB UND NDR-Bus. Dadurch bleibt das NDR-System immer jung - es lebe der NDR-Computer.

Auf der Systems in München (16. bis 20. Oktober) wird das neue Ausbildungs-System, verbunden mit einer der Zielgruppe angepaßten, sehr guten Dokumentation präsentiert.

Kommen Sie auch?

Gerd Graf

Klaus Bischof

Das MULTI I/O Interface

MULTI I/O steht bei uns wirklich für universelle Ein/Ausgabe. Damit ist also nicht eine Karte gemeint, wie sie von vielen Herstellern von kompatiblen Rechnern angeboten wird, mit der sie Zugriff auf einige Standardschnittstellen haben und eventuell noch einen Joystick anschließen können. Wir verstehen unter MULTI I/O Ein- und Ausgaben für den engagierten Elektroniker, Meßtechniker, Steuerungstechniker, Regelungstechniker etc. Der Zugriff auf externe universelle Schaltungen, wie z.B. Sensoren, Schaltern, Ventilen, Schrittmotoren etc. ist mit dieser Karte problemlos möglich. Die Baugruppe ist für den NDR-Bus, als auch für den PC-Bus erhältlich und wird natürlich von LOGSIM und PROFILOG unterstützt.

Nun aber zu den technischen Einzelheiten dieser Karte: Sie bietet 16 digitale Ein- und 16 digitale Ausgänge. Von diesen 16 sind jeweils 8 optoentkoppelt. Mit diesen optoentkoppelten Ausgängen können Sie Spannungen bis 24 V bearbeiten. Zusätzlich stellt die Karte noch 4 umpolbare Leistungsausgänge zur Verfügung, mit denen Sie kleine Motoren (Fischer Technik), auch Schrittmotoren, anschließen können. Die Leistung der Ausgänge: 24V, 1A.

Außer diesen digitalen Ein- und Ausgängen bietet die Karte noch 8 analoge Eingänge und 2 analoge Ausgänge. Die acht analogen Eingänge werden vom A/D-Wandler gemultiplext. Bei diesem Wandler sind nicht nur einfache Ein-Kanal-Messungen gegen Masse möglich, sondern auch Differenzmessungen zwischen zwei Eingängen. Die beiden D/A Wandler arbeiten ebenfalls mit einer Auflösung von 8 Bit und sind über einen Operationverstärker gegen Kurzschluß gesichert.

Diese analogen Schnittstellen sind für die Meß- und Regelungstechniker unter Ihnen sicher ein Leckerbissen, zumal in dem mitgelieferten Handbuch auch für den Einsteiger einfache Beispiele der Meß- und Regelungstechnik beschrieben werden.

Der Aufbau einer solchen Meßschaltung z.B. ist mit der ebenfalls für die MULTI I/O erhältlichen externen Experimentkarte sehr einfach möglich. Sämtliche Ein- und Ausgänge der Karte stehen auf einem 72-poligen Buchsenreihe zur Verfügung. Auf einem Steckfeld können Sie Ihre Schaltung aufbauen und mit den Ein- und Ausgängen der MULTI I/O verbinden. Wer solche Aufbauten meidet und lieber fest verdrahtet, für den gibt es die Möglichkeit, die Ein- und Ausgänge direkt über eine Klemmleiste anzuschließen.

Die technischen Daten

1. Digitale Ausgänge
- 8 Ausgänge mit TTL-Pegel
- 8 optoentkoppelte Ausgänge max. 24 V, (kurzschlußfest)

2. Digitale Eingänge
- 8 Eingänge mit TTL-Pegel
- 8 optoentkoppelte Eingänge, max. 24 V, (kurzschlußfest)

3. Leistungsausgänge
- 4 umpolbare Leistungsausgänge:
- ext. Spannung bis max. 24V Gleichspannung
- max. bis 1 A belastbar
- durch Verwendung von 2 Leistungsausgängen auch Schrittmotoren ansteuerbar
- verschiedene Haltmodi: Auslaufen oder generatorisches Bremsen
- induktive Ausschaltspannungsspitzen über Freilaufdioden abgesichert

4. Analoge Eingänge
8 analoge Eingänge über AD-Wandler ADC0848:
- 8 Bit Auflösung
- 8 Kanäle werden gemultiplext
- Wandelzeit: 40 us
- Spannung: 0 bis 5 V
- Differenzmessungen zwischen den Kanälen möglich
- Referenzspannung über REF02 möglich

5. Analoge Ausgänge
2 analoge Ausgänge über zwei DA-Wandler DAC0808
- Auflösung: 8 Bit
- Wandelzeit: 600 ns
- Spannung: 0 bis 5
- Referenzspannung über REF02 möglich
- Kurzschlußfest

6. Externe Experimentierkarte
- 8 Ausgänge auf LED's geführt
- 8 Eingänge über DIL-Schalter steuerbar
- Alle digitalen und analogen Ein- und Ausgänge auf 72-polige Buchsenreihe gelegt
- die wichtigsten Ein- und Ausgänge auf 36-poliger Klemmleiste abgreifbar
- Steckboard für eigene Musterschaltungen, z.B zum Testen von Schaltungen
- 18 frei Klemmleisten (über Steckfeld beschaltbar)
- wird über zwei 50-polige Kabel mit MULTI I/O verbunden

Für Sie gelesen:

DBASE III-Plus-Schulung

Autor: Dr . Dieter Albrecht
erschienen im Markt & Technik Verlag
mit Beispieldiskette

DBASE III-Plus-Schulung von Dr. Albrecht ist ein Buch, das sich besonders gut für Anfänger eignet.
Dies wird durch eine hervorragende Konzeption erreicht. Das Werk ist aufgeteilt in zwei Bereiche, den Lernbereich und den Antwortbereich. Das Auffinden der Antworten auf die im Lernbereich gestellten Fragen ist somit völlig problemlos. Hervorzuheben ist auch die klar verständliche "Sprache", die, wenn möglich, auf "fachchinesisch" weitgehend verzichtet. Der Schulung liegt eine Diskette mit vielen nützlichen Programmbeispielen bei, was mühsames Eingeben von Daten überflüssig macht.

Dr. Hehl Hans

Ökosystem eines Interpreters

Teil 3: Schnitzeljagd

In Meyer`s Großem Konversations-Lexikon aus dem Jahre 1909 findet man folgende Erklärung der Schnitzeljagd: "Jagdreiten, bei dem die Fährte des Wildes durch Papierschnitzel markiert wird von einem Reiter..."Nun ganz so turbulent soll es im dritten Teil nicht voran gehen. Wir werden die Fährte im Interpreter bedächtig verfolgen und jeden Schnitzel mit der Lupe betrachten.

Auftakt zur Jagd

In der Folge 2 hatten wir uns das Werkzeug hergerichtet und ein wenig damit geübt. Je nach Ausbaustufe des Rechners können und müssen nun verschiedene Programme verwendet werden. Diese Vielseitigkeit ist aber auch unser Problem. Daher werden nach den folgenden, zunächst allgemein gehaltenen Erläuterungen immer Hinweise zur jeweiligen Ausbaustufe gegeben. Die Speicheradressen sind natürlich je nach Interpreterversion verschieden, daher ist in Bild 1 eine

HEBAS 3.1+	HEBAS G4	EHEBAS 1.0	EHEBAS 1.1
47B0	4A39	BA1A	BA1A
19CF	19DD	1BBA	1BBA
2156h	2193h	2321h	23DB
2753d	2767d	4020d	4020d

Bild 1: Vergleichstabellen für die im Text enhaltenen Speicheradressen (d = dezimale Adresse für CALL)

Vergleichstabelle für die im Text angegebenen Adressen enthalten. Ansonsten gelten die Adressen für die HEBAS-Version 3.1+, da es den Quelltext für diese Version gibt.

Wir verwenden nun HEBAS mit dem Diskettenbetriebssystem CP/M. Ohne dieses benötigen Sie den neuen Monitor-Tester FLOMONCG und EHEBAS (Epromversion) auf einer Speicherkarte ROA64 (siehe Folge 1). Also EHEBAS in Sockel 1 und 2, zwei 8K-RAM-Bausteine in Sockel 5 und 6 sowie wegen der Speicherkartenumschaltung von FLOMONCG ein 8K-RAM in Sockel 8 (von links auf die Platine gesehen). EHEBAS erwartet RAM von 8000h bis BFFFh.

Der erste Schnitzel

Wir wollen nun ein Mini-BASIC-Programm schreiben und diesem Programm nachforschen. Dazu eignet sich z.B. folgende BASIC- Zeile:

10 PRINT"AAAA"

Also Rechner und Basic-Interpreter starten und die Zeile eingeben. Dann wird die RESET-Taste gedrückt. Da der Kernspeicher dabei nicht gelöscht wird, ist unser Programm noch da. Wie finden wir nun die Stelle, wo unser Programm steht? Man könnte den gesamten Speicher abschnittsweise anschauen, was sehr mühsam wäre. Einfacher geht es mit dem Suchkommando eines Monitor/Testers wie beim FLOMONCG oder beim mc-Computer mit HEBAS.

Wir geben also nach dem Suchbefehl Y und dem Suchbereich für die vier Bytes A den sedezimalen Wert 41 ein: und beim mc-Computer: Y41 41 41 41. Der Monitor meldet uns einige Adressen, darunter 47B6.

Jetzt können wir den Speicher in diesen Bereichen mit dem Monitor/Tester anschauen. Mit DDT.COM kann man Bytes leider nicht suchen, da wird gleich ab den angegebenen Adressen mit dem D-Befehl gearbeitet. Beim Grundprogramm wählt man den Menüpunkt "Speicher ansehen". Mit FLOMONCG und EHEBAS auf einer zusätzlichen Speicherkarte, z.B. Bank E, müssen Sie Y E:0 9000 41 41 41 41 eingeben.

Bild 2 zeigt für HEBAS31+.COM den Bereich ab 47B0, einmal mit Programmzeile und einmal ohne. Die ersten beiden Bytes BC und 47h stellen die sogenannte Link-Adresse 47BC dar.
Dort steht, wenn eine weitere Programmzeile vorhanden ist, die nächste Link-Adresse, so aber nur zwei Nullen. Genaue-

Teil 1: Aller Anfang ist schwer
LOOP 20

Teil 2: Werkzeug zum Knacken
LOOP 21

Teil 3: Schnitzeljagd
LOOP 22

Teil 4: Geheimsprache
LOOP 23

Teil 5: Ein Boarisch-BASIC
LOOP 24

Teil 6: BDOS und BIOS
LOOP 25

Teil 7: Innereien
LOOP 26

Teil 8: Auf die Plätze, fertig, los
LOOP 27

Teil 9: Patchwork mit Variablen
LOOP 28

Teil 10: Hexeneinmaleins
LOOP 29


```
d47b0,480f

47B0  BC 47 0A 00 95 22 41 41 41 41 22 00 00 00 67 72 <G..."AAAA"...gr
47C0  65 6E 7A E5 20 42 79 74 65 73 20 66 72 65 69 0A enze Bytes frei.
47D0  0A 56 69 65 6C 20 53 70 61 7E 20 20 0A 0A 48 45 .Viel Spaß  ..HE
47E0  42 41 53 20 4E 44 52 2D 4B 6C 65 69 6E 20 43 6F BAS NDR-Klein Co
47F0  6D 70 75 74 65 72 20 56 33 2E 31 2B 28 43 29 20 mputer V3.1+(C)
4800  44 72 2E 20 48 65 68 6C 20 48 61 6E 73 0A 8A 00 Dr. Hehl Hans...

d47b0,480f

47B0  1A 1B 7A 31 0D 0A 53 70 65 69 63 68 65 72 67 72 ..z1..Speichergr
47C0  65 6E 7A E5 20 42 79 74 65 73 20 66 72 65 69 0A enze Bytes frei.
47D0  0A 56 69 65 6C 20 53 70 61 7E 20 20 0A 0A 48 45 .Viel Spaß  ..HE
47E0  42 41 53 20 4E 44 52 2D 4B 6C 65 69 6E 20 43 6F BAS NDR-Klein Co
47F0  6D 70 75 74 65 72 20 56 33 2E 31 2B 28 43 29 20 mputer V3.1+(C)
4800  44 72 2E 20 48 65 68 6C 20 48 61 6E 73 0A 8A 00 Dr. Hehl Hans...
```

Bild 2: Speicher mit und ohne Basic-Programmzeile für HEBAS31+.COM

> Jede Problemlösung verursacht neue Probleme .
> (C) Murphy

re Erläuterungen dazu werden in der 4. Folge "Geheimsprache" gegeben. Dann folgt ebenso umgekehrt und sedezimal angegeben die Zeilennummer 10 der staben und wieder ein Anführungszeichen. Hier

einen Ausschnitt als sog. Hexdump und den dazugehörigen Abschnitt aus dem Quellcode von HEBAS31+.COM. Beachten Sie das erste Byte an Adresse 19CF. Mittels Ihrer DEZHEXBI-Tabelle sehen Sie, daß C1 das große A mit gesetztem Bit 7 ist. Bei den folgenden BASIC-Befehlen, die mit A beginnen, also ABS, ATN, ASC und AND, fehlt jeweils dieser Buchstabe. Nach jedem Befehlswort folgen ein oder zwei Bytes mit gesetztem Bit 7. Dies sind die sog. Token (Schlüsselwörter).

```
d8a1a,8a39

8A1A  26 8A 0A 00 95 22 41 41 41 41 22 00 00 00 04 04 &....."AAAA".....
8A2A  04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 ................
```

Bild 3: Speicher mit Basic-Programmzeile für EHEBAS

bewährt sich aus Folge 1 das Programm DEZHEXBI.BAS, das eine ASCII-Tabelle ausdruckt. Dann folgt ein Null-Byte als Abschluß der Programm-Zeile und zwei weitere als Programm-Endezeichen.
Bei EHEBAS finden Sie Ihre BASIC-Zeile ab Adresse 8A1A, da ab dieser Adresse Programme abgelegt werden. In Bild 3 ist der Speicherinhalt aufgeführt. Nur die ersten beiden Bytes (die Linkadresse 8A26) sind anders.

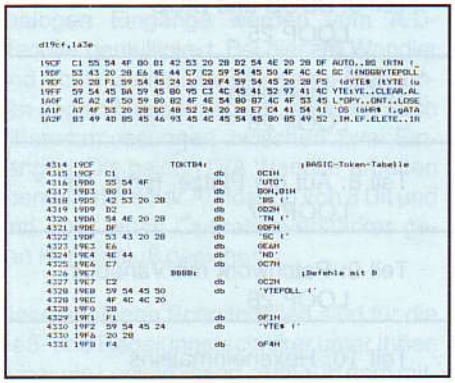


Bild 4: Fehlermeldungen im Interpreter HEBAS31+.COM (Hexdump und Quellcode)

Übungsaufgabe:

Ergänzen Sie Ihren kleinen Basic-Einzeiler durch eine beliebige weitere Programmzeile, z.B. 20 A$ = "BBBB" und schauen Sie sich den Speicher wieder an. Was hat sich verändert?

Noch ein Schnitzel?

Interessant sind weitere Interpreter-Bereiche. Ab 19CF tauchen plötzlich bekannte Begriffe am Bildschirm in der rechten Gruppe auf. Es sind die BASIC-Befehle, allerdings eigenartig abgekürzt. Bild 4 zeigt

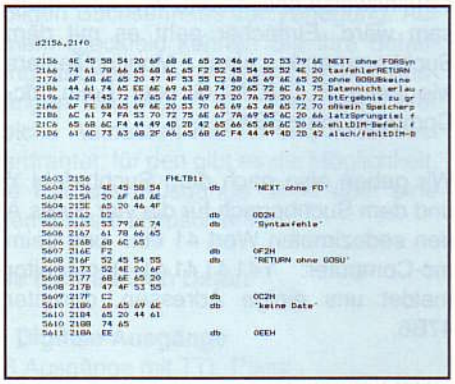


Bild 5: Fehlermeldungen im Interpreter (Hexdump und Quellcode)

Im rechten Teil des Hexdumps wird vom Monitor bei den Zeichen das Bit 7 (man zählt von rechts nach links und beginnt mit 0) einfach ignoriert, so ergibt das Byte D2 ein 'R'. Im Quellcode wird das klarer dargestellt.

FALSCH = FEHLER

Und im Bereich 2156h bis 24FF stehen die Fehlermeldungen, die man meistens am schnellsten beim Programmieren kennenlernt.

Bild 5 zeigt Speicherinhalt und Quellcode-Ausschnitt. Beim letzten Buchstaben eines jeden Fehlertextes ist wieder das Bit 7 gesetzt. Dies wird als Textende-Kennung in der Ausgabe routine für Zeichenketten benützt.

Bei EHEBAS verwenden Sie die in Bild 1 enthaltenen Adressen. Schreiben Sie sich Anfang und Ende der Speicherbereiche auf, die Text-Bytes enthalten. Diese Bereiche dürfen wir nicht mit unserem BASIC-Disassembler bearbeiten, da dieser die Text- Bytes als scheinbare Prozessor-Befehle interpretieren würde.
Wem dies alles zu umständlich ist, kann sich den Quellcode von EHEBAS als Listing kaufen oder als Handbuch vom Disk-HEBAS mit Diskette.

Nur Mut

Für ganz Mutige noch ein kleines Experiment. Ersetzen Sie die beiden letzten Null-Bytes unserer kleinen Basic-Zeile durch irgendeine andere Adresse, z.B. durch 2A2A. Dies geht mit dem Befehl S des Monitors oder des Debuggers DDT.COM. Dann starten Sie den Interpreter mit dem Befehl G 100. Es findet ein Warmstart des Interpreters statt und alles ist noch nor mal. Nun geben Sie den BASIC-Befehl LIST ein. Sie werden sich wundern. Meistens hilft dann nur noch die RESET-Taste. Keine Angst, der Computer geht nicht kaputt.

Ein Virus ???

Ebensoviel Spaß macht der BASIC-Befehl CALL 2753 (HEBAS31+). Damit kann man Laien sehr verblüffen. Meistens läßt sich mit der Leertaste die Ausgabe anhalten, mit <CTRL-E> beenden. Bild 6 zeigt eine Bildschirmausgabe. Manchmal hilft nur noch ausschalten.

Was war geschehen? Im ersten Fall wurde kein Ende der Programmzeile erkannt und weitere Bytes als Programmzeilen interpretiert, die natürlich sinnlos waren. Mit CALL 2753 wird im Interpreter in den Befehlsabschnitt des LVAR-Befehls gesprungen und zufällige Speicherwerte als Variablen ausgegeben.

Nachtrag

Auf der Diskette mit dem Disassembler in Basic aus Folge 2 befindet sich eine ausführliche Erklärung der Funktionsweise des Programmes für den BASIC-Anfänger. Auch die jeweiligen BASIC-Befehle werden in ihrer Funktion erklärt.

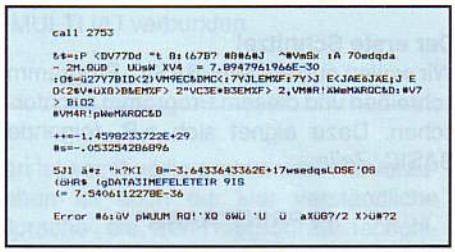


Bild 6: Ein Virus? Zufällige Bildschirmausgabe bei CALL 2753

M. Gujber

Graphik zu Fuß programmiert

Dieser Artikel ist all denen gewidmet,die als Einsteiger versucht haben die Graphikkarte des NDR-Klein Systems direkt anzusprechen und deren Nerven Opfer des Handbuches wurden.

Es ist klar, daß dieses Handbuch einiges voraussetzen muß und nicht alle (Direkt-)Anwender ausreichend ansprechen kann - deswegen auch dieser Artikel. Sicher wollen einige einen Einblick in wichtige Routinen des Grundprogramms bekommen,um diese in eigenen Programmen zu nutzen. Das könnte ja auch der Anfang eines eigenen Grundprogramms werden.
Dieser Beitrag ist auf Programmierneulinge zugeschnitten, es werden die Funktionen des Graphikprozessors erklärt und an Hand von einfachen und anschaulichen Programmbeispielen vertieft. Der Grundaufbau eines Programmes soll deutlich werden und die Programmfunktionen in Umfang und Schwierigkeit langsam anwachsen, wobei aber jedes Programmbeispiel für sich lauffähig und veränderbar ist. Durch einfaches Eintippen, Anschauen, Verändern und Probieren kann der Leser spielerisch einen Einblick in die Anwendungsmöglichkeiten bekommen.

Doch je weniger vorausgesetzt wird, desto ausfürlicher wird der Text und umfangreicher der Artikel. Das ist auch der Grund warum der Inhalt auf zwei Loopausgaben geteilt wird.
In diesem Teil sollen alle Grundeinstellungen behandelt werden und schließlich - als Schmankerl - Buchstaben auf den Schirm gebracht werden. In Übungsprogrammen soll dann das Aussehen der Buchstaben vielfältig geändert werden,das kann man auch als Anfäng eines Textverarbeitungsprogramm ansehen. So, jetzt aber zum "graphic display"-Prozessor.

Es stehen vier verschiedene Seiten zur Verfügung, die entweder beschrieben (Schreibseite) oder angezeigt (Leseseite) werden. Hierfür wird der Port 60h (ein 1 Byte langer Ausgang auf Adresse 60h) angesprochen .

Mit zwei Bits können wir vier verschiedene Zustände darstellen, also vier Bildschirmseiten. Diese beiden Bits, jeweils für die Leseseite und für die Schreibseite werden dem Seiten-Port übergeben.

Bit 7	Bit 6	
0	0	Schreibseite 0
0	1	Schreibseite 1
1	0	Schreibseite 2
1	1	Schreibseite 3
Bit5	**Bit4**	
0	0	Leseseite 0
0	1	Leseseite 1
1	0	Leseseite 2
1	1	Leseseite 3

Abb.: Belegung des Seitenports Wahrheitstabelle Schreib- und Leseseiten

Beispiel 1:
Wir wollen mit einem sehr einfachen Beispiel beginnen: Über den Seitenport soll Schreibseite 0 und Leseseite 0 eingestellt werden.In der oberen Tabelle sehen wir, daß für die Schreibseite 0 die Bits 7 und 6 Null sind und für die Leseseite 0 die Bits 5 und 4 Null sind. Die Bits 3 bis 0 sind unbelegt, und können beliebig belegt werden, d.h. wir geben auf Port 60h den dualen Wert 0000 0000 aus, was dem hexadelzimalen Wert 00h entspricht. Unten sehen Sie das einfache Programm für diese Einstellung.

Adresse	Hexcode	Mnemonic	Kommentar
8800	3E 00	ld a,00h	Lade 00h in Register A
8802	D3 60	out(60h	Gib Inhalt von A auf den Seitenport 60h aus

Um das Ganze etwas zu üben, wollen wir nun verschiedene Kombinationen ausprobieren. Wir wollen nun auf Seite 0 eine Graphik aufbauen, und die Seite 1 auf dem Bildschirm darstellen.
Mit dieser Maßnahme erreicht man, daß ein Bild im Hintergrund (nicht auf dem Bildschirm sichtbar) aufgebaut wird, und in der Zwischenzeit wird die Seite 1 auf dem Bildschirm angezeigt. Wir gehen nun wieder nach folgender Tabelle vor:

Schreibseite 0:	Bit 7 = 0	und Bit 6 = 0
Leseseite 1:	Bit 5 = 0	und Bit 4 = 0

Die Bits 3 bis 0 sind nicht von Bedeutung und werden mit 0 angegeben.

Für diese Kombination ergibt sich der duale Wert 0001 0000b und ein hexadezimaler Wert von 10h. Das obige Programm zur Ausgabe an den Seitenport sieht nun folgendermaßen aus:

8800	3E 10	d a,10h	Lade 10h in Register A
8802	D3 60	out(60h),a	Gib Inhalt von Register A den Seitenport 60h aus

Sie sehen schon, das Ganze ist doch sehr einfach. Damit Sie hier etwas Übung bekommen schlagen wir vor, daß Sie nun noch die folgenden drei Übungsbeispiele durchspielen.

Diese Übung werden wir später noch einmal wiederholen, wenn wir auf verschiedene Seiten geschrieben haben.

1. Schreibseite 2 und Leseseite 3 einstellen
2. Schreibseite 0 und Leseseite 2 einstellen
3. Schreibseite 1 und Leseseite 0 einstellen

Stellen Sie nun zu unserer nächsten Übung wieder Schreib- und Leseseite 0 ein.

Nun begeben wir uns in die Höhle des Löwen und sehen uns den Graphikprozessor etwas näher an. Sie haben sicher, allein schon durch den Namen Graphikprozessor, etwas Respekt vor

diesem Chip und dessen Programmierung.

Dieser Respekt ist aber nicht nötig, denn der hier angesprochene Graphikprozessor EF9366 ist sehr einfach zu programmieren. Ähnlich wie beim Seitenport müssen nur einige Register gesetzt werden, und schon erscheint auf dem Bildschirm ein Zeichen, oder ein Vektor, oder ...

Wie bei allem Neuen müssen wir auch bei unserem Graphikprozessor etwas Zeit investieren, um die Register und Befehle kennenzulernen. Wir beginnen mit den Registern des Graphikprozessors. Dabei haben wir bewußt die bei der GDP64HS nicht benutzten Register weggelassen, um Sie nicht zu sehr zu verwirren.

Adresse	Name / Funktionen	Bits
70h	Status od. Kommando	8
71h	Schreibgerät (CTRL 1)	7
72h	Schriftform (CTRL 2)	4
73h	Schriftgröße (CSize)	8
75h	Vektorlänge X - Koord	8
77h	Vektorlänge Y - Koord.	8
78h	X - Position MSBs	4
79h	X - Position LSBs	8
7Ah	Y - Position MSBs	4
7Bh	Y - Position LSBs	8

Abb.: Übersicht Register

Sie sehen hier nun die wichtigsten zehn Register des Graphikprozessors. Sie werden schon ahnen, was mit dem ein oder anderen Register wohl gemacht werden kann. Das wichtigste Register ist das Kommando- bzw. das Statusregister. Aus dem Statusregister können wir den Status (den momentanen Zustand) des Graphikprozessors abfragen. Bei diesem Statusregister interessiert uns hier nur Bit 2, die restlichen sieben Bit lassen wir vorerst links liegen. Dieses Bit 2 sagt aus, ob der GDP beschäftigt ist, oder ob er bereit ist, ein neues Kommando entgegen zu nehmen.

Damit sind wir auch schon beim Kommandoregister angekommen. Dieses Register wird immer dann beschrieben, wenn ein Befehl ausgeführt werden soll. Doch bevor diese Kommandos gegeben werden können, müssen oft die anderen noch erwähnten Register gesetzt werden. Hier wollen wir nun Schritt für Schritt vorgehen.

Im Grundprogramm wird ständig zwischen Seite eins und zwei umgeschaltet. Nach dem Return in das Grundprogramm muß man sich daher darum kümmern, daß die andere Seite leer ist (oder gelöscht wird) oder, daß man den Rücksprung in das Grundprogramm, beispielsweise durch eine Endlosschleife verhindert (siehe 'Häuschenprogramm). Hier bei diesen Programmen wurde sonst überall der Halt-Befehl verwendet. Dadurch müssen Sie nach jedem Programmablauf einen RESET ausführen. Man hätte hier natürlich ein Schleife einbauen können, daß bei einem bestimmten Zeichen von der Tastatur das Programm unterbrochen wird, dies würde aber die Programme, die bewußt einfach gehalten wurden, doch komplizierter machen.

Löschen des Bildschirmes

Wenn Sie eine Graphik erstellen wollen, ist meist der erste Schritt, den Bildschirm komplett zu löschen. Dazu müssen wir nun folgendermaßen vorgehen:

- die zu löschende Seite als Schreibseite einstellen
- Statusregister abfragen
- Ist der GDP bereit für einen neuen Befehl (Bit 2 = 1)?
  Bit 2 = 0 GDP ist nicht bereit
- Ausgabe des Löschbefehles

Abb.: Ablaufdiagramm

Der Ablauf ist Ihnen sicher klar. Was Ihnen wahrscheinlich noch nicht bekannt ist: Wo finde ich den Löschbefehl?

Zu diesem Zweck sehen wir uns die ersten 16 Befehle des Graphikprozessors etwas näher an (Befehle 00h bis 0Fh). Dazu sollten Sie sich die Befehlstabelle des Kommandoregisters ansehen. In der linken oberen Ecke sehen Sie die Bezeichnungen b3 bis b0 vertikal und b4 bis b7 horizontal aufgelistet. Diese Bezeichnungen entsprechen den Bits 0 bis 7 des Kommandoregisters. Neben bzw. unterhalb der dualen vierstelligen Zahlen sehen Sie, in einer Spalte bzw. Zeile, den hexadezimalen Wert von 0 bis F. In diesem Feld sind 256 mögliche Befehle eingeschlossen. Wir sehen uns nun die erste Befehlsspalte von 00h bis 0Fh näher an. Hier sind grundsätzliche Befehle, wie Rücksetzen von Registern oder bestimmten Bits, oder auch das Löschen des Bildschirmes. Wandern wir die erste Spalte herunter bis zur hexadezimalen Ziffer 4, so sehen wir den Befehl: "Clear screen". Auf Grund Ihrer sicherlich guten Englisch Kenntnisse gehen wir davon aus, daß Sie diesen Befehl als Bildschirm-Lösch-Befehl erkennen. Damit aber nicht genug. In Zeile 6 dieser Spalte sehen wir denselben Befehl gekoppelt mit dem Rücksetzbefehl für die Koordinatenregister X und Y. In Zeile 7 finden wird einen weiteren Bildschirmlöschbefehl, bei dem zusätzlich alle Register auf 0 und das CSIZE-Register auf "minsize" gesetzt wird.

Für unseren ersten Löschversuch verwenden wir einfach mal den Befehl 06h. Stürmische unter Ihnen werden jetzt sicher sofort diesen Befehl an das Kommandoregister (Port 70h) ausgeben wollen. Damit fallen Sie aber, wie man so schön sagt, auf die "Schnauze". Wenn Sie oben das Ablaufdiagramm noch einmal ansehen, wissen Sie sofort, was wir noch tun müssen, damit das alles funktioniert

Wie die zu löschende Schreibseite einzustellen ist, ist uns bereits bekannt. Nun müssen wir nur noch das Statusregister abfragen, ob der Graphikprozessor bereit ist ein Kommando entgegen zu nehmen. Dieses Programm benötigen wir immer wieder; deshalb schreiben wir dieses als Unterprogramm und nennen es "Warte". Das Programm heißt deshalb "Warte", weil das Hauptprogramm hier warten muß, bis der Graphikprozessor bereit ist. Auch dieses Programm ist nicht sehr schwierig. Wir müssen nur das Statusregister des Graphikprozessors (Port 70h) abfragen und "nachschauen", ob Bit 2 0 oder 1 ist. Ist Bit 2 = 0, so ist der Graphikprozessor beschäftigt und wir dürfen nicht ins Kommandoregister schreiben.

Ist Bit 2 = 1 so ist der Graphikprozessor bereit für ein Kommando. Dieses Unterprogramm wird jeweils an das Ende des Hauptpro-grammes gesetzt und dann mit dem "CALL"-Befehl aufgerufen.

Dieses Unterprogramm mit den Namen "warte" wird mit 'ret' abgeschlossen. Jetzt stellt es eine Schleife dar, die man immer wieder durch den Befehl 'call' "Name" (hier also warte) aufrufen kann. Nach dem Befehl 'ret' (von return, das zurückkehren bedeutet) wird in die Zeile nach diesem Aufruf ('call' "warte") zurückgesprungen.

Aus der logischen "UND"-Verknüpfung zwischen der vom Statusregister eingelesenen Bitkombination und der 4h (= 0000 0100b) ergibt sich solange eine Null als Ergebnis, solange Bit 2 = 0 ist. Will man ein Bit eines Bytes isolieren, so wendet man diesen AND-Befehl an; man spricht dann auch von "maskieren" des entsprechenden Bits. Der in Zeile 8810 folgende

Sprungbefehl ist ein bedingter, absoluter Sprung. Der Sprung wird ausgeführt, wenn das "Zero"-Flag gesetzt ist. Dies wiederum ist gesetzt, wenn das zuletzt behandelte Register, in diesem Fall Register A, bei der Maskierung von Bit 2 des Statusregisters, Null ist. Ist das Ergebnis der Maskierung aber 1, so wird der Sprungbefehl nicht ausgeführt und das Unterprogramm beendet.

Adresse	Hex Code	Mnemonik	Kommentar
8800	3E 00	ld a,00h	Lade Register A mit 00h
8802	D3 60	out (60h),a	Schreib- und Leseseite au Null gesetzt
8804	CD 0C	88 call warte	Verzweige zum UP "WARTE"
8807	3E 06	ld a,06h	Lade Register A mit 06h
8809	D3 70	out (70h),a	Inhalt von Reg. A an Kommandoregister des GDP = Bildschirm löschen
880B	76	halt	Halt
warte:			
880C	DB 70	in a,(70h)	Lese Statusregister des GDP
880E	E6 04	and 4	Maskiere Bit 2
8810	CA 0C 88	jp z,warte	Springe nach "Warte", wenn Bit 2 = 0
8813	C9	ret	Rücksprung ins Hauptprogramm

Abb.: Unterprogramm "Warte"

Nun können Sie das Programm im RDK-Grundprogramm eintippen und starten. Der Bildschirm müßte jetzt absolut leer sein.

Daß das Löschen des Bildschirms noch nicht der Weisheit letzter Schluß ist, ist uns und Ihnen wohlbekannt. Deshalb wollen wir nun endlich etwas vernünftiges auf dem Bildschirm darstellen. Um hier wieder einfach einzusteigen, wollen wir einen Buchstaben, und zwar das große "A" auf dem Bildschirm unten links darstellen.

Der Graphikprozessor ist hier sehr wirklichkeitsnah aufgebaut, denn um auf dem Bildschirm etwas zeichnen zu können, muß dem Graphikprozessor gesagt werden ,daß er sich ein Schreibgerät besorgen soll.

Diese Funktion finden wir im CTRL1-Register (Steuerregister 1). Dieses Register wollen wir uns hier etwas näher ansehen. Dabei spielen für uns im Moment nur zwei Bits eine Rolle. Über Bit 1 können wir auswählen, ob ein Schreibstift (Pen: Bit 1 = 1) oder der Radiergummi (Eraser: Bit 1 = 0) verwendet werden soll. Damit ist das entsprechende Gerät aber noch nicht aktiviert. Über Bit 0 dieses Registers kann jetzt der Schreibstift oder der Radiergummi aktiviert werden (Bit 0 = 0: aktiv, Bit 0 = 1: nicht aktiv). Also wählen wir hier, weil wir auf dem Bildschirm etwas schreiben wollen, den Schreibstift aus und aktivieren diesen, d.h. Bit 0 und Bit 1 werden auf 1 gesetzt. Die übrigen Bits dieses Registers werden auf 0 gesetzt.

Ausgabe an CTRL1-Register Port 71h: 03h

Um die oben gestellte Aufgabe erfüllen zu können, müssen wir noch zwei Register vorbelegen: Das CTRL 2 Register und das CSIZE Register. Beim CTRL2 Register sind nur 4 Bits benutzt. Bit 0 und 1 sind für die Vektorgraphik von Bedeutung, werden also für unsere Aufgabe nicht benötigt. Mit Bit 2 können wir festlegen, ob der Buchstabe kursiv oder normal senkrecht geschrieben wird. Mit Bit 3 wird festgelegt, ob der Buchstabe waagrecht oder senkrecht am Monitor erscheinen soll. Hier wählen wir die einfachste Form: waagrecht und nicht kursiv, d.h. Bit 2 = 0 und Bit 3 = 0.

Ausgabe an CTRL2-Register Port 72h: 00h

Das letzte Register, das wir bei unserer Aufgabe beachten müssen, ist das CSIZE-Register (CSIZE = Character Size, was in deutsch soviel heißt wie Zeichengröße). Mit diesem Register kann die vertikale und horizontale Größe des Zeichens, ausgehend vom Grundraster von 8*5 Bildpunkten pro Zeichenfestgelegt werden. Bit 4 bis 7 legen den Faktor auf der der X-Achse und Bit 0 bis 3 legen den Faktor auf der Y-Achse fest. Steht nun in diesem Register der Wert 11h, so wird das Zeichen in einem Grundraster von 5 mal 8 Bildpunkten dargestellt. Steht in diesem Register nun der Wert 22h, so ist das Zeichen sowohl in horizontaler als auch in vertikaler Richtung doppelt so groß. Wir haben mit diesem Register die Möglichkeit, die Zeichen bis 16 mal größer darzustellen als die Grundform. Dabei ergibt sich das kleinste Zeichen, bei 00h und das größte Zeichen, wenn 11h im CSIZE-Register steht. Dazwischen sind alle Möglichkeiten erlaubt. Sie können die Zeichen, z.B. auch sehr lang und schmal (z.B. 15h) oder sehr breit und kurz (z.B. 51h) machen. Ihrer Phantasie sind hier fast keine Grenzen gesetzt.

Eine bequeme Art die Zeichengröße ,genau die Grundgröße festzulegen, ergibt sich über das Kommandoregister. Dieses verfügt über einen leistungsfähigen Befehl ,der gleichzeitig den Bild schirm löscht , die Grundgröße einstellt und alle restlichen Register auf Null setzt . (Befehl 07h im Kommandoregister)

Jetzt wollen wir aber die Größe selbst einstellen ; Für unser Beispiel wählen wir die doppelte Zeichengröße des Grundzeichens.

Ausgabe CSIZE Register auf Port 73h: 22h

Zum Schluß muß natürlich das Kommando für das Zeichnen des Buchstabens noch ausgegeben werden. Dies geschieht wiederum mit dem Kommandoregister. Wenn wir uns wieder die Befehlsübersicht des Kommandoregisters ansehen, können wir feststellen, daß mit den Befehlen 20h bis 7Fh ASCII-Zeichen dargestellt werden können. Das "A" ,das wir darstellen wollen, finden wir mit dem Code 41h.

Ausgabe Kommandoregister auf Port 70h: 41h

Nun können wir unser Programm zusammenstellen. Dabei können wir das vorherige Programm gleich mitverwenden.

Wenn Sie dieses Programm eingegeben und gestartet haben, müßte sich unten links ein kleines "A" zeigen. Mit diesem Programm haben wir nun aber schon die Möglichkeit zu variieren. Wir können z.B die Werte im CSIZE Register ändern, und erhalten dadurch dieses Zeichen größer oder kleiner. Wir könnten auch im CTRL2-Register das Bit 2 ändern und damit ein kursives Zeichen darstellen und wir könnten einen anderen Buchstaben oder ein anderes Zeichen ausgeben. Aber was reden wir, was wir alles tun können; Fangen wir an!

Aufgabe

1. Stellen Sie jetzt das Zeichen kursiv dar

2. Vergrößern Sie das Zeichen, ausgehend vom Grundraster 5*8 in der Vertikalen um das 8-fache und in der horizontalen um das 5 fache

3. Zeichnen Sie statt dem "A" ein "Z", ein "E" und eine "1"

Bei diesen Aufgaben müssen Sie immer nur ein Byte dieses Programmes ändern. Aber was sag' ich Ihnen das, sie werden das längst selbst erkannt haben.

Adresse	Hex Code	Mnemonik	Kommentar
8800	3E 00	ld a,0	Seite einstellen
8802	D3 60	out (60),a	
8804	CD 00 89	call warte	Aufruf Unterprogramm
8807	3E 06	ld a,06h	Bildschirm löschen u.
8809	D3 70	out (70h),a	X- u. Y-Register auf 0
880B	CD 00 89	call warte	Aufruf UP
880E	3E 03	ld a,03h	Geschlossene Bildfläche
8810	D3 71	out (71h),a	
8812	CD 00 89	call warte	UP
8815	3E 00	ld a,00h	Schriftart : normal
8817	D3 72	out (72h),a	
8819	CD 00 89	call warte	UP
881C	3E 22	ld a,22h	Zweifache Grundgröße
881E	D3 73	out (73h),a	
8820	CD 00 89	call warte	UP
8823	3E 41	ld a,41h	ASCII-Code für ein
8825	D3 70	out(70h), A	großes "A"
8827	76	halt	CPU anhalten
warte:			Unterprogramm wartet bis
8900	DB 70	in a,(70h)	GDP bereit ist
8902	E6 04	and 04h	
8904	CA 00 89	jp z, warte	wenn nicht bereit, nochmal abfragen
8907	C9	ret	Rücksprung ins Hauptprogramm

Positionierung eines Zeichens

Sie werden nun meinen, daß auch dies mit der Zeit langweilig wird, denn bisher haben wir immer nur links unten irgendein Zeichen auf den Bildschirm gesetzt. Wir können natürlich auch irgendwo anders auf dem Bildschirm ein Zeichen plazieren. Dazu hat der Graphikprozessor ein X- und ein Y-Register. Da eine Bildschirmseite 512 Bildpunkte in der Horizontalen und 256 in der Vertikalen hat, muß der Ort, wo ein Zeichen stehen soll, in den X- und Y-Registern festgelegt werden. Für die horizontale Auflösung reicht ein 8 Bit Register nicht aus, da damit nur 256 Punkte adressiert werden könnten. Aus diesem Grund wurden für die beiden Koordinatenregister jeweils zwei Register vorgesehen. In unserer Registerliste stehen diese mit X-Register LSB, X-Register MSB, Y-Register LSB und Y-Register MSB. LSB und MSB sind die Abkürzungen für Most Significant Byte und Least Significant Byte, was soviel heißt wie höherwertiges Byte und niederwertiges Byte.

Wählen wir einen Punkt in der Mitte des Bildschirmes aus um jetzt ein Zeichen darzustellen. Die Mitte befindet sich in Y-Richtung beim Punkt 128 und in X-Richtung beim Punkt 256. Dadurch ergibt sich für die vier Register folgende Werte

Ausgabe Y-Register LSB auf Port 7Bh: 7Fh
Ausgabe Y-Register MSB auf Port 7Ah: 00h
Ausgabe X-Register LSB auf Port 79h: FFh
Ausgabe X-Register MSB auf Port 78h: 00h

Damit können wir das obige Programm noch einmal erweitern. Das Setzen der Register kann direkt nach dem Löschen des Bildschirmes erfolgen. Stellen Sie das obige Programm nun so um, daß sie ein Zeichen in der Mitte des Bildschirmes darstellen können. Beachten Sie aber bitte, daß Sie bei jedem Zugriff auf den Graphikprozessor das Unterprogramm "Warte" aufrufen.

Haben Sie dies geschafft, so versuchen Sie jetzt einmal diese Koordinaten zu verändern:

1. X/Y-Koordinaten:

450/56
30/230
100/100

Dazu müssen Sie diese Werte natürlich in hexadezimale Werte umrechnen!

2. Verändern Sie im CTRL2-Register Bit 3. Was passiert?

Nun ein komplettes Beispiel, wie so ein Programm aussehen könnte:

Beispiel :

Der Buchstabe 'T' soll in seiner Höhe um den Faktor elf und in der Breite um Faktor zwei gestreckt werden .

Wenn Sie obige Beispiele ausprobiert haben und Schriftform, Buchstabengröße und Position verändert haben, ahnen Sie schon die vielfältigen Möglichkeiten im Umgang mit der Zeichenform.

Will man mehrere Buchstaben hintereinander schreiben,muß man nicht die Position für jede Letter einzeln eingeben, denn innerhalb einer Zeile macht dies der Prozessor selbst.

Das erste Zeichen (kleinste Größe ist der 5 mal 8 Block) einer Zeile muß noch positioniert werden, dann rutscht der Prozessor ganz automatisch mit der X-Koordinate um die Zeichenbreite plus ein Leerpunkt (d.h. ein Vielfaches von 5). Für den Anfang einer jeder Zeile muß man aber beide Koordinaten eingeben,der GDP kann ja nicht ahnen, wo er anfangen soll. So kann man ohne große Neueinstellungen einen Zeilentext auf den Bildschirm bringen. Jetzt kann man schon an sein erstes kleines Textverarbeitungsprogramm denken.

Adresse	Hex Code	Mnemonik	Kommentar
8800	3E 00	ld a,00h	Lade Null in Akku
8802	D3 60	out(60h),a	Inhalt A an Port 60h
8804	CD 00 89	call warte	Aufruf Unterprogramm
8807	3E 07	ld a,07h	Bildschirm säubern und
8809	D3 70	out(70h),a	alle Register auf Null
880B	CD 00 89	call warte	Aufruf Unterprogramm
880E	3E FF	ld a,FFh	FFh als X-LSB
8810	D3 79	out(79h),a	
8812	CD 00 89	call warte	Aufruf Unterprogramm
8815	3E 7F	ld a,7Fh	7Fh als Y-LSB
8817	D3 7B	out(7Bh),a	
8819	CD 00 89	call warte	Aufruf UP
881C	3E 2B	ld a,2Bh	Vergrößerungsbyte' an Port 73H
881E	D3 73	out(73h),a	(CSize-Register)
8820	CD 00 89	call warte	Aufruf UP
8823	3E 03	ld a,03h	Stift einsetzen
8825	D3 71	out(71h),a	
8827	CD 00 89	call warte	Aufruf UP 'warte'
882A	3E 00	ld a,00h	
882C	D3 72	out (72h),a	
882E	CD 00 89	call warte	
8831	3E 54	ld a,54h	Code für 'T' durch A an Port 70h
8833	D3 70	out(70h),a	
8835	76	halt	CPU anhalten
warte:			
8900	DB 70	in a,(70h)	Unterprogramm 'warte'
8902	E6 04	and 4h	
8904	CA 00 89	jp z,warte	
8907	C9		Sprung ins Hauptprogramm

Programmbeispiel

C. Hübner

Programm "List"

Als Ergänzung zu meinem Programm - Schieber (LOOP 11) ist das Programm "LIST" gedacht: Da das Grundprogramm im Ändern - Modus nur drei Programmzeilen anzeigt, sind längere Programme nur schwer überschaubar. Gerade beim Verschieben von Programmteilen kommt es jedoch auf jedes Byte an! Hier schafft "LIST" Abhilfe.

Nach dem Start erfragt es die Adresse, ab der das anzuzeigende Programm, bzw. der interessierende Teil steht. Sobald die

Ein komfortables Programm, um den Speicher mit Hilfe des Z80 Grundprogrammes übersichtlich in einer Tabellenform darzustellen.

Eingabe erfolgt ist, werden 25 Programmzeilen auf dem Bildschirm ausgegeben. Mit der Taste <+> wird der Bildschirm um 12 Zeilen nach oben "gescrollt", und es erscheinen die nächsten Zeilen. Mit der Taste <-> kann man zurückgehen. Hier kann er vorkommen, daß die ersten Befehle falsch angezeigt werden. Dies liegt daran, daß das Programm der Einfachheit halber 50 Bytes ab der aktuellen Adresse zurück geht und so möglicherweise nicht den Anfang eines Befehls trifft.

Die Erfahrung zeigt jedoch, daß es nach wenigen Bytes wieder "einrastet". Mit <R> schließlich kann man eine neue Adresse eingeben; jeder andere Tastendruck beendet das Programm. Die Lage des Programmes auf Adresse 8700h hat zur Folge, daß der Programmspeicher" ab 8800h gar nicht, und der "Symbolspeicher" nur unwesentlich beeinträchtigt wird.

```
                        ;**************************************
                        ;*    Programm zum übersichtlichen    *
                        ;*  Auflisten von Maschinenprogrammen *
                        ;*      unter dem Grundprogramm       *
                        ;*     (C) Christof Hübner            *
                        ;*         Lessingstraße 2            *
                        ;*           6729 Rheinzabern         *
                        ;**************************************

                                TITLE LIST 14.02.1987

                        CLRINVIS    EQU     0033H
                        GETADR      EQU     1B6BH
                        LENGTH      EQU     0D2AH
                        AUSBUF      EQU     807AH
                        SETAUSBUF   EQU     0672H
                        BEFPRINT    EQU     18C4H
                        CI          EQU     0024H

                                ORG 8700H

                        LIST:
8700  CD 33 00                 CALL CLRINVIS   ;BILDSCHIRM LÖSCHEN
8703  CD 6B 1B                 CALL GETADR     ;HL=ANFANGSADRESSE
8706  38 F8                    JR C,LIST       ;FEHLER->NEUEINGABE
8708  D9                       EXX
8709  CD 33 00                 CALL CLRINVIS
870C  D9                       EXX
870D  22 6C 87          SEITE:  LD (START),HL
8710  11 F5 00                 LD DE,245       ;ANFANGSWERT FÜR Y
8713  21 0C 00                 LD HL,12        ;X-POS
8716  3E 21                    LD A,21H        ;SCHRIFTGRÖßE
8718  CD 72 06                 CALL SETAUSBUF  ;AUSGABEPUFFER VOREINSTELLEN
871B  D9                       EXX
871C  06 19                    LD B,25         ;25 ZEILEN PRO SEITE
871E  D9                LOOP:  EXX             ;ZEILENZAHLER IN ZWEITREGISTERSATZ
871F  DD 21 80 80              LD IX,AUSBUF+6  ;FÜR AUSGABE
8723  2A 6C 87                 LD HL,(START)   ;AKTUELLE BEFEHLSADRESSE
8726  E5                       PUSH HL
8727  CD C4 18                 CALL BEFPRINT   ;AKTUELLEN BEFEHL AUSGEBEN
872A  E1                       POP HL
872B  CD 2A 0D                 CALL LENGTH     ;B=LÄNGE DES BEFEHLS IN BYTES
872E  58                       LD E,B
872F  16 00                    LD D,00H
8731  19                       ADD HL,DE
8732  22 6C 87                 LD (START),HL   ;NEUE AKTUELLE BEFEHLSADRESSE
8735  D9                       EXX
8736  78                       LD A,B          ;A=AUSGEGEBENE ZEILENZAHL
8737  D9                       EXX
8738  FE 0C                    CP 12           ;HALBE SEITE VOLL ?
873A  20 03                    JR NZ,W         ;WENN NEIN->WEITER
873C  22 6E 87                 LD (NEU),HL     ;NEUE STARTADRESSE
873F  2A 7C 80          W:     LD HL,(AUSBUF+2);HL=ALTE Y-POS
8742  11 F6 FF                 LD DE,-10       ;ZEILENABSTAND (NEGATIV)
8745  19                       ADD HL,DE
8746  22 7C 80                 LD (AUSBUF+2),HL;NEUE Y-POS
8749  D9                       EXX
874A  10 D2                    DJNZ LOOP
874C  CD 24 00                 CALL CI         ;WARTEN AUF BEFEHLE
874F  FE 2B                    CP '+'
8751  28 0B                    JR Z,WEITER
8753  FE 2D                    CP '-'
8755  28 0C                    JR Z,ZURÜCK

LIST 14.02.1987                                         PAGE  2

8757  CB AF                    RES 5,A         ;KLEIN->GROSSBUCHSTABE
8759  FE 52                    CP 'R'
875B  28 A3                    JR Z,LIST       ;NEUE ADRESSE
875D  C9                       RET
                        WEITER:
875E  2A 6E 87                 LD HL,(NEU)
8761  18 AA                    JR SEITE

                        ZURÜCK:
8763  2A 6E 87                 LD HL,(NEU)
8766  11 CE FF                 LD DE,-50
8769  19                       ADD HL,DE
876A  18 A1                    JR SEITE

                        START:  DS 2
                        NEU:    DS 2

                        ENDE:   END

LIST 14.02.1987                                         PAGE  3

0033 CLRINVIS                   1B6B GETADR
0D2A LENGTH                     807A AUSBUF
0672 SETAUSBUF                  18C4 BEFPRINT
0024 CI                         8700 LIST
870D SEITE                      871E LOOP
873F W                          875E WEITER
8763 ZURÜCK                     876C START
876E NEU                        8770 ENDE


no fatal error(s)


SUM=2D62
CRC=C851


 ; Hexdump des LIST - Programmes
*
*
*D 8700,877F
8700  CD 33 00 CD 6B 1B 38 F8 D9 CD 33 00 D9 22 6C 87 M3.Mk.8xYM3.Y'l.
8710  11 F5 00 21 0C 00 3E 21 CD 72 06 D9 06 19 D9 DD .u.!..>!Mr.Y..Yü
8720  21 80 80 2A 6C 87 E5 CD C4 18 E1 CD 2A 0D 58 16 !..*l.eMD.aM*.X.
8730  00 19 22 6C 87 D9 78 D9 FE 0C 20 03 22 6E 87 2A ..'l.YxYþ. .'n.*
8740  7C 80 11 F6 FF 19 22 7C 80 D9 10 D2 CD 24 00 FE ö..v..'ö.Y.RM$.þ
8750  2B 28 0B FE 2D 28 0C CB AF FE 52 28 A3 C9 2A 6E +(.þ-(.K/þR(£I*n
8760  87 18 AA 2A 6E 87 11 CE FF 19 18 A1 32 B4 76 DA ..**n..N...!24vZ
```

Programm zum übersichtlichen Auflisten von Maschinenprogramme

Jost-Reimer Hoof

ZEAT patcht ZEAT

Nach Auslieferung des neuen Monitors FLOMONCG beginnt sich die NDR-Computerwelt auf Z80-Basis entscheidend zu wandeln. Lassen sich die großen Programmpakete wie TURBO PASCAL oder WordStar ohne Schwierigkeiten betreiben, so gibt es mit ZEAT noch Probleme. Auch diese sind lösbar, wie folgender Beitrag zeigt.

Handwerkzeug zum Patchen

Gepatcht wird immer dann, wenn ein vorhandenes Programm abgeändert werden soll, der Sourcecode, d.h. die Quelle, nicht zur Verfügung steht. Der Patcher versucht den Maschinencode zu analysieren, ihn zu disassemblieren oder sogar mit geeigneten Analyseprogrammen zu bearbeiten. Hat man sich in den Programmablauf eingearbeitet, so lassen sich gewisse Umbauten des Programms vornehmen. Als Handwerkzeug dient ein Disassembler, ein Texteditor und ein Tester. Alles das ist im ZEAT-Programm von Christiani enthalten, und deshalb eignet es sich hervorragend dazu.

Problemanalyse von ZEAT

Das Programmpaket, ein unheimlich starkes Utility mit einem TESTER ähnlich DDT, einem nicht ganz so starken Texteditor EDIT, einem Assembler und einem Modem, das meines Wissens das einzige Programm dieser Art ist, mit dem unser NDR-Rechner ohne Datenverlust über Packed Radio oder Akustik-Koppler/Telefon mit anderen Rechnern Informationen und Daten austauschen kann. Und es wäre jammerschade, wenn ein solches Programmpaket unter FLOMONCG nicht mehr einsetzbar wäre!

ZEAT gibt es in 2 EPROMs je 8 mal 8K Byte, die auf die Steckplätze 1 und 2 der Bankboot-Karte gesetzt werden (auf Steckplatz 0 sitzt das FLOMON 3.2). Außerdem gibt es von ZEAT eine 18k große Diskettenversion, die aber die EPROMs benötigt. Mit der EPROM-Bestückung kollidiert aber FLOMONCG, denn dieser Monitor belegt den Steckplatz 0 und 1 auf der Bankboot-Karte.

Problemlösung

Analysiert man das disassemblierte ZEAT-Programm, so stellt man fest, daß zu zwei Zwecken auf die EPROMs zugegriffen wird:

1. zum Testen der Copywrite-Meldung und
2. um sich die Assembler-Tabelle von weniger als 400 Byte zu holen.

Es gibt 3 Lösungsmöglichkeiten:

1. Man entschließt sich, die aus den beiden EPROMs benötigten Teile auf 1 EPROM zusammenzuschieben und das EPROM auf den Steckplatz 2 (Adresse 4000h - 5FFFh) zu setzen.
2. Dieser Teil wird beim Laden von ZEAT in einen dort vorhandenen RAM-Bereich kopiert.
3. Dieser benötigte Teil wird auf der Bank 0 untergebracht. Die 1. Lösungsmöglichkeit mit dem EPROM auf Steckplatz 2 halte es nutzt den Speicherplatz ab 4000h.

Die 2. Lösungsmöglichkeit bietet sich als elegante Möglichkeit unter der Auflage an, FLOMONCG aus dem Quellcode selbst zu assemblieren und den Datenbereich in dem Programm-Modul GEN.SUB ab 4400h statt ab 4000h zu wählen. Um sich diese Arbeit zu ersparen, schlage ich die Lösung 3 als beste vor, da sie bei mir funktioniert.

Checkliste für das Patchen von ZEAT mit ZEAT

1. Schritt: Hilfsfile ZE-ASM.TAB mit Assemblertabelle aus EPROM 2 herstellen und abspeichern.
2. Schritt: ZEAT laden, TPA mit "F 0100, 4A00, 0" säubern.
3. Schritt: mit EDITOR Programm 1 eingeben und genau prüfen.
4. Schritt: im TESTER ZEAT laden mit RF
5. Schritt: im TESTER das Hilfsprogramm ZE-ASM.TAB nach 4530h laden mit RF 4530 und Filename.
6. Schritt: Programm 1 assemblieren
7. Schritt: im TESTER prüfen, ob Änderungen an richtiger Stelle.
8. Schritt: neues ZEAT abspeichern mit "SF 0100, 4A7F" für gesamten Patch. Wer nur Patch-Teil 1 macht, braucht auch weniger abspeichern.

Bild 1: Checkliste

Das Handwerkliche

Beim Patchen nutze ich 2 Möglichkeiten: Das Vorspiel ist in beiden gleich: Ich lade ZEAT, gehe in den TEST-Modus und säubere die TPA mit "F 0100, 4A00, 0". Dann lade ich mit RF = Read File das zu patchende Programm in die TPA und notiere mir das Ende, d.h. ab welcher Adresse "frei" gemeldet wird.

Habe ich wenige Byte abzuändern, arbeite ich mit dem "S"- Kommando ("S- Adresse" in Hex) .Das Programm gibt mir den Inhalt dieser Adresse aus, den ich nun beliebig in Hex abändern kann. (Man bedenke, bei einem Buchstaben eine "0" vorzusetzen!).

Soll mehr gepatcht werden, oder soll das Gepatchte gleich dokumentiert werden, so benutze ich den Editor von ZEAT. Ich schreibe einfach die Befehle in Assembler. Ganz wichtig ist, daß die entsprechende Adresse mit der "ORG"-Anweisung in HEX-Zahl festgelegt wird. Ist die Befehlsliste fertig, wird sie vorsorglich abgespeichert und dann assembliert. So wird an den Adressen, die in den ORG-Anweisunge festgehalten sind, die Speicherwerte entsprechend geändert.

Das Nachspiel ist für beide Verfahren wieder gleich: Das gepatchte Programm wird im TEST-Modus überprüft, entweder mit der Dump-Kommando "D", besser noch mit List-Kommando "L" und der entsprechenden Adresse. Wenn alles in Ordnung ist, wird das Programm mit dem Save-File-Kommando "SF 0100, Endadresse" und dem neuen Filename abgespeichert.

ZEAT-Patch

Nun kommt die heiße Phase, der eigentliche Patch im ZEAT-Programm mit ZEAT: Wie sich an der Unterteilung des Programmes 1 (Bild 1) mit "*******" erkennen läßt, besteht das Programm aus vier Teilen:

1. Teil: Patch, damit ZEAT unter FLOMON 3.2 ohne EPROMs läuft.
2. Teil: Patch für FLOMONCG, dabei ist

der 1. Teil Voraussetzung.
3. Teil: Patch für ein neues Titelbild.
4. Teil: Fehlerberichtigung für EDITor-Kommando (Vertauschen von 'Y'- UND 'Z'-Kommando ==> WordStar-Kompatiblität).

Im 1. Teil werden 4 Speicherstellen, die FFh enthalten, mit ASC- Zeichen, z.B. "ZEAT", überschrieben. Außerdem wird die Laderoutine für die Assembler-Tabelle abgewandelt und die Relokationstabelle angepaßt. Diese wird zum Umrechnen des geladenen ZEAT-Programms zum tatsächlich ab A000h abgelegten Programm EPROM nach 4530h plaziert (s.u.). Nun kann man ZEAT ohne EPROMs unter FLOMON 3.2 und sicher auch unter den anderen FLOMON-Versionen betreiben.

Im 2. Teil wird ZEAT an FLOMONCG angepaßt. Dazu gehört, zu Beginn das vom Monitor mit 84 Zeichen pro Zeile aufgespannte Fenster auf 80 Zeichen zu verkleinern. Beim Verlassen von ZEAT muß dieses Fenster wieder auf 84 Zeichen vergrößert werden.

Im 3. Teil wird das Titelbild, das unter FLOMONCG merkwürdig flackert, durch ein Ersatzbild ersetzt. (Bekanntlich läßt sich beim FLOMONCG Grafik und Text nicht mischen!) Jeder kann das Textfile auch nach eigenen Wünschen anders gestalten.
Im 4. Teil habe ich eine Korrektur von zwei EDIT-Kommandos vorgeschlagen, nämlich das 'Y'- und 'Z'-Kommando zu vertauschen. Bei mir löscht jetzt 'Y' eine Zeile, während 'Z' den Bildschirm um eine Zeile nach oben scrollt.

Wie bekomme ich die Assemblertabelle an die richtige Stelle?

Im 2. ZEAT-EPROM befindet sie sich ab Adresse 1C56h bis 1F72h (tatsächliche Adresse auf der BankBoot-Karte ist 5C56h bis 5F72h). Programm 2 (Bild 2) kopiert sie von der BankBoot nach 8100h. Um das zu erreichen, habe ich ZEAT (alt) aufgerufen, mit PACK 5000 die TPA auf 8871h vergrößert und im TEST-Modus die gesamte TPA gesäubert (F 0100, 8870, 0). Dann wird Programm 2 eingegeben und aufgerufen. Die sich dann ab 8100h befindliche Assemblertabelle wird nun abgespeichert (SF 8100, 841C *> A-ASM.TAB).
Die Checkliste für das gesamte Vorgehen steht in Bild 2.

Testen

Für das Testen ist wichtig, daß das Programm 1 sehr genau abgetippt und penibel verglichen wird. Vor dem Patch sollte man die Stellen im Ur-ZEAT, die abgeändert werden, ausdrucken und nach dem

```
                        TITLE ZEAT-Patch                        28.01.1989
;=======================================================================
;       Jost-Reiner Hoof, 2305 Heikendorf
;                       Wilhelm-Ivens-Weg 87
;                       Tel 0431 - 24 20 70
;-----------------------------------------------------------------------
;       Assembler-Liste wird direkt nach DB76h geladen. (27.01.89)
;-----------------------
;       Mit diesem Programm läßt sich die Disketten-Version von ZEAT
;       so patchen, daß die EPROMs nicht mehr benötigt werden.
;
;       1. Teil:
;               Patch, damit ZEAT unter FLOMON V.3.2 ohne EPROMs läuft
;       2. Teil:
;               Patch für FLOMONCG, dabei 1. Teil Vorbedingung
;       3. Teil:
;               Neues Titelbild
;       4. Teil:
;               Korrektur von 2 Fehlern im Original-ZEAT
;-----------------------------------------------------------------------
;       Arbeitsschritte:
; 1. Schritt: Hilfsfile ZE-ASM.TAB mit Assemblertabelle aus EPROM 2
;           herstellen und abspeichern.
; 2. Schritt: ZEAT laden, TPA mit "F 0100, 4A00, 0" säubern.
; 3. Schritt: mit EDITOR Programm 1 eingeben und genau prüfen.
; 4. Schritt: im TESTER ZEAT laden mit RF
; 5. Schritt: im  TESTER das Hilfsprogramm ZE-ASM.TAB nach  4530h
;             laden mit RF 4530 und Filename.
;           Beginn ist "LD8AP1LD8AP9RLD819ALD8R9AEX8"
; 6. Schritt: Programm 1 assemblieren
; 7. Schritt: im TESTER prüfen, ob Änderungen an richtiger Stelle
; 8. Schritt: neues  ZEAT  abspeichern mit "SF  0100,  4A7F"  für
;             gesamten Patch.  Wer nur Patch-Teil 1 macht,  braucht
;           auch weniger abspeichern.
;-----------------------------------------------------------------------
Stop:   equ     0
System  equ     0005h

;################  1. Teil für FLOMON 3.2 ######################

                ORG     02BFh                   ; hier stehen 4 x FFh
02BF  5A 45 41 54       DB      'ZEAT'
;-------------------------

                ORG     1FD0h   ; hier stand: F3 AF D3 C8 01 1B 03 2A 20 20 11 00 ...

1FD0  01 30 03          LD      BC, 0330h               ; Größe der Assembler-Liste
1FD3  21 30 45          LD      HL, 4530h               ; Startadresse
1FD6  11 76 3A          LD      DE, 3A76h               ; Zieladresse, hinzu kommt Offset
1FE1  ED B0             LDIR                            ; Kopier-Befehl
1FE3  C9                RET                             ;
;-------------------------

                ORG     415Fh                   ; Relokationstabelle korrigieren
415F  B0 10 B4 10       DB      B0h, 10h, B4h, 10h      ; statt B0h, B0h, 01, 00

;################  2. Teil für FLOMONCG ########################
;                                               Teil 1 ist Voraussetzung !!!
                ORG     100h
0100  C3 00 45          JP      4500h                   ; statt C3 2A 01 JP 012Ah
                                                        ; hierin wird ZEAT umgeleitet
;-----------------------------

                ORG     4500h
4500  06 04               ld      b, 4                    ; Fenster von 84 auf 80 Zeichen
4502  C5                   push    bc                      ; verkleinern
4503  1E 1B                ld      e, 1Bh                  ; ESC
4505  CD 1B 45             call    etextaus
4508  1E 24                ld      e, '$'                  ; '$'
450A  CD 1B 45             call    etextaus
450D  1E 6C                ld      e, 'l'                  ; 'l' = Window rechts verkleinern
450F  CD 1B 45             call    etextaus
4512  C1                   pop     bc
4513  10 ED                djnz    eloop
4515  C3 2A 01             JP      012Ah                   ; Einsprung-Stelle im ZEAT
4518  0E 02          etextaus:
451A  CD 05 00             LD      C, 2
451D  C9                   CALL    System
                           RET
                   ;-----------------------

                           ORG     0077h                   ; in EXIT-Funktion
0077  C3 E4 1F             JP      Exitl

                   ;-----------------------------------
                   ;                               ; Routine zum Vergrößern des Windows
                           ORG     1FE4h
                   Exitl:
1FE4  06 04                ld      b, 4                    ; 4 mal ein Zeichen nach rechts
                   loop:
1FE6  C5                   push    bc
1FE7  1E 1B                ld      e, 1Bh                  ; ESC
1FE9  CD 00 20             call    textaus
1FEC  1E 24                ld      e, '$'                  ; '$'
1FEE  CD 00 20             call    textaus
1FF1  1E 72                ld      e, 'r'                  ; 'r' = Window rechts vergrößern
1FF3  CD 00 20             call    textaus
1FF6  C1                   pop     bc
1FF7  10 ED                djnz    loop
1FF9  2A 51 04             LD      HL, (0451h)
1FFC  22 01 00             LD      (0001h), HL
1FFF  C7                   rst     0
                   ;-----------------------
                   textaus:
2000  0E 02                LD      C, 2
2002  CD 05 00             CALL    System
2005  C9                   RET

                   ;#####################  3. Teil neues Begrüßungsbild  ################

                           ORG     024Ah
024A  C3 5A 02             JP      025Ah                   ; Unterbinden des Ausgabe
                   ;--
                           ORG     0260h
0260  C3 B1 03             JP      03B1h                   ; Unterbinden der Ausgabe
                   ;--
                           ORG     02FCh                   ; Sprung in eigene Ausgaberoutine
02FC  18 01                JR      weiter
02FE  00                   NOP                             ; nötig wegen Relokationsumrechnung
                   weiter:
02FF  C3 B0 48             JP      48B0h                   ; Hier wird neues Titelbild ausge-
                                                           ; geben
                   ;-----------------------

                           ORG     48B0h
48B0  11 B9 48             LD      DE, Text                ; neues Titelbild
48B3  0E 09                LD      C, 9
48B5  CD 05 00             CALL    0005h
48B8  C9                   RET
                   ;-----------------------
                   Text:                                   ; hier läßt sich eigenes Bild
```

Bild 2: Programm 1

```
                                                          ; einbauen
4889  1A 0A 0A 0A 0A 0A           DB     1Ah, 0Ah, 0Ah, 0Ah, 0Ah, 0Ah

488F  20 20 20 20 20 20 20 20     DB '       ZZZZZZ   EEEEEEE      A      TTTTTTT', 0Dh, 0Ah
48C1  20 20 20 20 20 20 20 20     DB '          Z     E           A A        T', 0Dh, 0Ah
48F0  20 20 20 20 20 20 20 20     DB '         Z      EEE        A   A       T', 0Dh, 0Ah
491F  20 20 20 20 20 20 20 20     DB '        Z       E         AAAAAAA      T', 0Dh, 0Ah
494E  20 20 20 20 20 20 20 20     DB '       ZZZZZZZ  EEEEEEE  A       A     T', 0Dh, 0Ah
497D  0A 0A 0A                    DB     0Ah, 0Ah, 0Ah

4980  20 20 20 20 20 20 20 76     DB     '      von Technischen Lehrinstitut Dr. Ing P. Christiani'

49B9  0D 0A 0A                    DB     0Dh, 0Ah, 0Ah
49BC  20 20 20 20 20 20 20 28     DB     '      (1985) Günther Bauholzer', 0Dh, 0Ah, 0Ah
49DE  20 20 20 20 20 20 20 28     DB     '      (1989) gepatcht v. JoHo - Software'
4A07  0D 0A 0A                    DB     0Dh, 0Ah, 0Ah

4A0A  20 20 20 20 20 20 20 20     DB     '          Editor',  0Dh, 0Ah, 0Ah
4A1C  20 20 20 20 20 20 20 20     DB     '            Assembler',  0Dh, 0Ah, 0Ah
4A33  20 20 20 20 20 20 20 20     DB     '              Tester',  0Dh, 0Ah, 0Ah
4A49  20 20 20 20 20 20 20 20     DB     '                Modem', 0Dh, 0Ah, 0Ah

4A60  24                          DB     '$'

                          ;######### 4. Teil = Fehlerberichtigung ##################
                                                          ; im EDITor wird Y- und Z-Kommando
                                                          ; vertauscht, damit WS-kompartiebel
                                  ORG    258Ah            ;
258A  1A                          DB     1Ah              ; statt 19 = Y

                                  ORG    259Fh            ;
259F  19                          DB     19h              ; statt 1A = Z

                          ;######################### Ende ################ (28.01.1989) ##########

0000 STOP                         0005 SYSTEM
4502 ELOOP                        4518 ETEXTAUS
1FE4 EXIT1                        1FE6 LOOP
2000 TEXTAUS                      02FF WEITER
4889 TEXT


no fatal error(s)


SUM=83CF
CRC=7350
```

Patch prüfen, ob wirklich an dieser Stelle die Änderung vorgenommen wurde. Beliebter Fehler ist, bei der ORG-Anweisung die Adresse ohne "h" für HEX zu schreiben. Die Folgen sind fatal, da ZEAT diese Adressen als Dezimalzahl interpretiert und in Hex umrechnet. Jetzt sollte man die erzeugte und abgespeicherte Version von ZEAT starten. Meldet sich ZEAT mit dem vertrauten Begrüßungsbild bzw. dem durch Teil 3 erzeugten Bild, gehe man in den Tester und starte das List-Kommando "L". Wird ein typisches Assembler-Listing ausgegeben, dann ist schon viel gewonnen.
Zur Kontrolle sollte man noch ein kleines Assemblerprogramm, z.B. dieses Patchprogramm, in den EDITOR einlesen und assemblieren. Gibt es Unregelmäßigkeiten, muß nochmals alles peinlich genau wiederholt werden.
Der endgültige Test erfolgt dann mit herausgenommenen EPROMs. Ich hoffe, daß dann ZEAT zur Zufriedenheit arbeitet.

Viel Spaß beim Patchen wünscht der Autor.

Vorab-Info für die Erweiterung zum LogSim (mehr in LOOP23)

Der ProfiLog

Für Profi's oder solche, die es werden wollen.

Die Möglichkeit, digitale Schaltungen am Bildschirm zu simulieren, wurde bereits in der LOOP 20 mit Hilfe des Programmes LogSim vorgestellt. Der Logiksimulator von Rolf Dieter Klein, wurde nun von Ihm wesentlich erweitert und perfektioniert.

Der einfache Logiksimulator LogSim ermöglichte es ja bereits, die grundlegenden Digitalen Schaltungen am Bildschirm zu simulieren. Jedoch wünschte man sich etwas mehr als die einfachen Grundelemente wie z.B: AND, OR oder XOR. Dies ist jetzt durch die erweiterte Version des Programmes LogSim möglich. Der ProfiLog bietet nun die Möglichkeit, komplexe Schaltungen am Bildschirm zu simulieren. Von der Bedienung hat sich gegenüber LogSim nichts geändert. Die Bildschirmmaske und die komfortable Bedienung mit der Maus wurden, bewußt beibehalten. So kommen auch Umsteiger, die bereits den LogSim kennen, sofort mit dem Profilog zurecht.
Was jedoch erheblich erweitert wurde, ist die Anzahl der zur Verfügung stehenden Gatter und die jetzt einzusetzenden Flip/Flop's. Damit ist es jetzt durchaus möglich, ein Mikroprozessor-Minimal-System zu entwerfen bzw. zu entwickeln. Auch die jetzt zu Verfügung stehenden Speichergatter erweitern das Einsatzgebiet des ProfiLogs's auf ein Vielfaches. Das Handbuch von ProfiLog wurde komplett überarbeitet und bietet so noch mehr Information für die Anwendung des Simulators. Zahlreiche Beispiele auf der Originaldiskette wie z.B. Parallel/Seriell Umsetzer, 8 bit Zähler, Modulo zwei Zähler, 4 bit Addierer/ Subtrahierer bis zur ALU runden den Lieferumfang ab.

Um einen generellen Überblick über die erweiterten Funktionen zu vermitteln. Hier die Daten des ProfiLog's in Kurzform.

-Logische Grundbausteine
.AND, NAND, OR, NOR (- Bis zu acht facher Ausführung)

-Flip-Flop's
.D-Flip
.R/S-Flip
.J/K-Flip + J/K mit 3 Eingängen

Erweiterte Gatter
. Addierer, Subtrahierer (4 bit)
. Multiplexer 4-fach
. Demultiplexer 4-fach und 8-fach
. Latch 4-fach
. Speicher 8*4
. ALU
. A/D Wandler 8-Bit Ausgang
. Zähler 4-fach

- Hilfsmittel
. 4-bit Anzeige
. 8-bit Anzeige (mit Zweierkomple ment)
. 8-bit Anzeige (dezimal)

Udo Peters

Einstellen der Baud-Rate der SER-Baugruppe unter CP/M 2.2

Anschrift:
Udo Peters
Walther-Rhatenau-Str. 3
6080 Groß-Gerau

Nachdem ich schon sehr viel von der Zeitschrift LOOP profitiert habe, möchte ich auch ein paar Informationen beisteuern, die für den einen oder anderen NKC-Besitzer mit CP/M 2.2 interessant sein könnten.

Das Programm N/SER ermöglicht es, die Baudrate für die Installation der seriellen Schnittstelle per Menü zu wählen. Außerdem erhält man eine Rückmeldung über die erfolgreiche Installation. Im Gegensatz zum Z80-TOOL-Programm "ASSIGN.COM" kann hier die Routine beliebig geändert und erweitert werden.
Das abgedruckte Listing kann bei mir auf Diskette (5 1/4") bestellt werden.

```
;**********************************
;* SER Ein/Ausgabeprogramm        *
;* Installiert SER-Unterprogramme *
;* fuer CP/M mit NDR-kLEIN-       *
;* Computer. Rolf-Dieter Klein    *
;* V 1.0  850522                  *
;* Auswahl der Baudrate durch Menue: *
;* Udo Peters, Walther-Rathenau-Str. 35 *
;* 6080 Gross-Gerau    06130      *
;**********************************
;
;
ser     equ     0f0h            ;Basisadresse SER-Baugruppe
;
;
ri      equ     0f006h
poo     equ     0f00ch
;
freemem equ     0f031h
free    equ     0f8b0h
;
coneinf equ     1               ;Tastatur lesen
strausf equ     9               ;Stringausgabe
bdos    equ     5               ;bdos-Aufruf
;
cr      equ     0dh             ;carriage return
lf      equ     0ah             ;line feed
;
menue:
        ld      de,mentxt       ;Menue fuer die
        ld      c,strausf       ;Auswahl der Baud-
        call    bdos            ;rate ausgeben
;
eingabe:
        ld      c,coneinf       ;Zeichen von der
        call    bdos            ;Tastatur holen
;
        cp      '1'
        jp      z,erstes
        cp      '2'
        jp      z,zweites
        cp      '3'
        jp      z,drittes
        cp      '4'
        jp      z,viertes
        cp      '5'
        jp      z,fuenftes
        cp      '6'
        jp      z,sechstes
        jp      0               ;Warmstart
;
erstes:
        ld      b,91h           ;50 Baud
        ld      hl,baud
        ld      de,ebd
        call    transbd
        jp      start
;
ebd:    db      '  50$'
;
zweites:
        ld      b,96h           ;300 Baud
        ld      hl,baud
        ld      de,zbd
        call    transbd
        jp      start
;
zbd     db      '  300$'
;
drittes:
        ld      b,98h           ;1200 Baud
        ld      hl,Baud
        ld      de,dbd
        call    transbd
        jp      start
;
dbd:    db      '1200$'
;
viertes:
        ld      b,9ah           ;2400 Baud
        ld      hl,baud
        ld      de,vbd
        call    transbd
        jp      start
;
vbd:    db      '2400$'
;
fuenftes:
        ld      b,9ch           ;4800 Baud
        ld      hl,baud
        ld      de,fbd
        call    transbd
        jp      start
;
fbd     db      '4800$'
;
sechstes:
        ld      b,9eh           ;9600 Baud
        ld      hl,baud
        ld      de,sbd
        call    transbd
        jp      start
;
sbd:    db      '9600$'
;
transbd:
        ld      a,(de)
        cp      '$'
        ret     z
        ld      (hl),a
        inc     de
        inc     hl
        jp      transbd
;
start:  ld      a,b             ;Uebernahme der Baudrate
        out     (ser+3),a
        ld      a,0bh           ;keine Paritaet, Freigabe
        out     (ser+2),a
;
        ld      ix,free         ;freier Speicherplatz
```

Listing des Programms: N/SER

```
        ld      hl,si           ;Quelle
        push    ix
        pop     de              ;Ziel
        ld      bc,lensi        ;Laenge
        ldir
;
        ld      (ri+1),ix       ;Vektor RI neu definiert
        ld      de,lensi
        add     ix,de           ;naechster Platz
;
        ld      hl,so           ;Quelle
        push    ix
        pop     de              ;Ziel
        ld      bc,lenso        ;Laenge
        ldir
;
        ld      (poo+1),ix      ;Vektor POO neu definiert
;
;Druckerausgabe bleibt unveraendert
;
        ld      de,lenso
        add     ix,de           ;naechster Platz
        ld      (freemem),ix    ;neuer freier Platz
;
        ld      de,endtxt
        ld      c,strausf
        call    bdos
;
        call    0               ;und Warm-Start
        ret
;
;Verschiebbare Routinen:
;
si:
        in      a,(ser+1)       ;Zeichen da ?
        and     8               ;wenn
        jr      z,si            ;nein, dann warten
        in      a,(ser)         ;und sonst holen
        ret
;
lensi equ $-si
;
;
so:
        in      a,(ser+1)       ;DSR frei, DSR auf +12V
        and     40h
        jr      nz,so           ;nein, dann warten
        in      a,(ser+1)       ;Ausgabe frei?
        and     10h
        jr      z,so            ;nein, dann warten
        ld      a,c             ;sonst Wert ausgeben
        out     (ser),a
        ret
;
lenso equ $-so
;
mentxt:
        db      1ah
        db      '*******************************************',lf,cr
        db      '*                                         *',lf,cr
        db      '* Installation der seriellen Schnittstelle *',lf,cr
        db      '*                                         *',lf,cr
        db      '* PUN:=SEROUT:                RDR:=SERIN: *',lf,cr
        db      '*                                         *',lf,cr
        db      '*******************************************'
        db      lf,lf,cr
        db      'Folgende Parameter sind fest eingestellt:',lf,cr
        db      'Datenbits:     8',lf,cr
        db      'Stopbits:      2',lf,cr
        db      'Paritaet:      keine',lf,lf,cr
        db      'Bitte Baudrate waehlen:',lf,lf,cr
        db      '1 =   50 Baud      2 =  300 Baud',lf,cr
        db      '3 = 1200 Baud      4 = 2400 Baud',lf,cr
        db      '5 = 4800 Baud      6 = 9600 Baud',lf,lf,cr
        db      '##'
;
endtxt:
        db      '   -> '
        db      'Serielle Schnittstelle installiert mit '
baud:   ds      4
        db      ' Baud',lf,lf,cr,'$'

        end
```

Günter Renner

Jetzt wird gelesen!

Teil 1: Der verrückte Bootsektor
Loop 17

Teil 2: Eine gefährliche Operation
Loop 19

Teil 3: Log. physikalischer Verwirrspiel
Loop 20

Teil 4: Sage mir, was Du hast
Loop 21

Teil 5:Jetzt wird gelesen
Loop 22

Teil 6: Ärger mit Ä
Loop 23

Das Beschreiben von Datenträgern ist bisweilen eine gefährliche Sache. Vor allem dann, wenn man sich in fremdes Territorium - sprich Systeme - begibt. Es soll deshalb erst einmal mit einer harmloseren Operation begonnen werden, durch die allenfalls Speicherinhalte zerstört werden können: dem Lesen von Dateien von der Diskette.

Diese Aufgabe erledigt die Menuefunktion 'load'. Zunächst wird der Benutzer zur Eingabe des Dateinamens aufgefordert. Dann geht es mit dem bekannten Testen der Disk weiter. Stimmen die Parameter nicht überein, wird mit der eigentlichen Arbeit gar nicht erst begonnen.

Nun wird das Inhaltsverzeichnis durchsucht mittels des Unterprogramms 'lookfor'. Alle 112 Namenseinträge des Hauptinhaltsverzeichnisses werdenverglichen. Wird keine Übereinstimmung mit dem Inhalt des Namenspuffers gefunden, kehrt dieses Unterprogramm mit gesetztem Minusflag zurück, und das Programm 'load' meldet das Nichtvorhandensein des gesuchten Eintrags. Ist der Eintrag jedoch gefunden, zeigt (a4) auf das entsprechende Entry. Das sehr einfache Unterprogramm 'firstclu' liest nun den Startcluster ins Register d2.w, sodaß damit gleich der erste Cluster der Datei bearbeitet werden kann. Hier begegnet einem wieder die verschiedene Anordnung der Bytes eines Wortes, die prozessorabhängig ist und durch Rotation korrigiert werden muß.

Da nun die wenigsten Dateien nur aus einem Cluster bestehen, sollte man wissen, wie es weitergeht. Auskunft darüber gibt das Unterprogramm 'nextclu'. Es verwendet den Inhalt von d2.w als Zeiger auf die aktuelle FAT-Position und liest deren Inhalt nach d3.w. Von den geladenen 16 Bits interessieren hier natürlich nur 12 davon; die vier übrigen gehören zu einem anderen Eintrag.

Die Hardware gestattet nur ein Arbeiten mit 16 Bit breiten Worten. Man muß deshalb den Wert eines Eintrags mit einer UND-Verknüpfung herausmaskieren, ehe man ihn weiterverwenden kann. Vor dieser Maskierung ist bei allen ungeraden FAT-Positioned.h. Clusternummern,noch eine Rechtsverschiebung um vier bits erforderlich, damit man die richtigen 12Bits auch tatsächlich erwischt. Diese müssen stets in d3.w unten und der unbenutzte Rest oben stehen. Viel einfacher hätte man es da natürlich mit einer 16-Bit-FAT, wie man sie auf größeren Festplatten vorfindet; die Väter dieses Filesystems haben sich aber wohl aus Platzgründen zu dieser programmiertechnisch aufwendigeren Lösung entschieden. Eine gute Seite kann man dieser Sache aber immer abgewinnen: wer sich ein Buch über MS-DOS kaufen will, um in den 'Niederungen' zu programmieren, sehe erst einmal darin nach, wie verständlich oder wie verwirrend die Logistik und die Handhabung der FATs darin beschrieben ist...

Zur Erinnerung: der gewonnene Wert kann zum einen den nächsten Cluster bezeichnen, zum anderen das Ende der Datei, aber auch einen reservierten oder fehlerhaften Cluster kenntlich machen. Der Einfachheit halber wird im vorgestellten Monitor der Wert des FAT-Eintrags nur darauf überprüft, ob er außerhalb des Bereichs der möglichen Clustenummern liegt. Ist er innerhalb dieser Grenzen, gibt der gefundene Wert den nächsten Cluster an, so daß man ihn einfach nach d2.w laden und damit erneut die Routine 'cluster' aufrufen kann. Liegt er außerhalb derselben, ist das Dateiende erreicht, oder aber die FAT ist fehlerhaft.

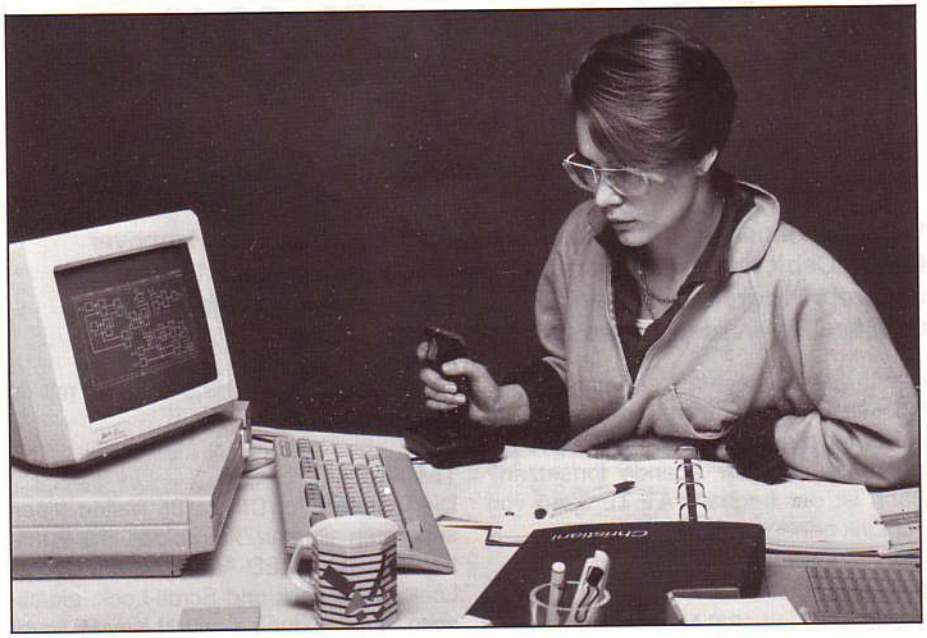
Das Programm 'load' enthält nun eine Schleife, die fortlaufend Cluster nach Cluster einliest. Das geht solange, als 'nextclu' einen Wert liefert, der einer möglichen Clusternummer enspricht. Andernfalls wird die Schleife abgebrochen und zuletzt geprüft, ob ein Floppyfehler aufgetreten ist.

In einer weiteren Folge werden wir es wieder mit solchen Clusterketten zu tun haben. Dann, wenn Programme zum Löschen und Speichern von Dateien vorgestellt werden. Das nächste Mal ist es noch nicht so weit. Denn es gibt noch ein kleines Problem, das bei Textfiles auftreten kann: Umlaute und andere Sonderzeichen!

Günter Renner
Schloßbühlstr. 11
7206 Emmingen-Liptingen

```
68008-UMSCHALTUNG       MACRO-80 3.4    01-Dec-80       PAGE    1
                        page 68


                        TITLE 68008-UMSCHALTUNG
                        .z80
                        ;*****************************************************************************
                        ;**                                                                         **
                        ;**     Umschaltprogramm auf 68008 mit Hardware-Zusatzschaltung             **
                        ;**                                                                         **
                        ;**     Das Programm wird von CP/M80 aufgerufen. Es kopiert sich dann       **
                        ;**     auf Adresse 2000h auf die Bank-Boot. Bei einem Reset kann dann      **
                        ;**     mit FLOMON-Menue-Punkt 3 "2000h Starten" der 68008 wieder           **
                        ;**     aufgerufen werden. Lauffaehig unter FLOMON3.2.                      **
                        ;**                                                                         **
                        ;**     letzte Aenderung 26.02.1988                                         **
                        ;**                                                                         **
                        ;**     (C) K. Neumaier, Ahornstrasse 9, 8311 Altfraunhofen, Tel 08705/868  **
                        ;*****************************************************************************
                        ;
00C8                    bank    equ     0c8h            ;Bank-Boot Umschaltport
0080                    baout   equ     80h             ;Bank-Boot aus
0000                    baon    equ     00h             ;Bank-Boot an
0090                    b68008  equ     90h             ;Bank-Boot aus, 68008 an
8000                    zsp     equ     8000h           ;Zwischenspeicher-Adresse
2000                    bankad  equ     2000h           ;Speicher-Adresse Bank-Boot
                        ;
0000'                   aseg
                        org 100h
                        ;
0100    21 010E         start:  ld hl,copy11            ;kopiert Datenblock von copy11 nach zsp
0103    11 8000                 ld de,zsp
0106    01 0088                 ld bc,copy2e-zsp
0109    ED B0                   ldir
010B    C3 8076                 jp copy2
                        ;
010E                    copy11:
                                .phase bankad
2000    21 200E         copy1:  ld hl,copy1end          ;kopiert Datenblock von copy1ende nach zsp
2003    11 8000                 ld de, zsp
2006    01 0068                 ld bc, bootende-bootstart+lastende-last
2009    ED B0                   ldir
200B    C3 8054                 jp last
200E                    copy1ende:
                                .dephase
                        ; 68008 Boot-Programm aus "Schritt fuer Schritt 2"
                        ; Adresse 29h das Boot-Programmes wurde von 80h auf 90h geaendert
011C                    bootstart:
011C    00 00 8F FE             db      000h,000h,08fh,0feh,000h,000h,000h,008h
0120    00 00 00 08
0124    36 3C 00 31             db      036h,03ch,000h,031h,041h,0f9h,000h,000h
0128    41 F9 00 00
012C    80 00 43 FA             db      080h,000h,043h,0fah,000h,00eh,010h,0d9h
0130    00 0E 10 D9
0134    51 C9 FF FC             db      051h,0c9h,0ffh,0fch,04eh,0f9h,000h,000h
0138    4E F9 00 00
013C    80 00 10 38             db      080h,000h,010h,038h,070h,000h,011h,0fch
0140    70 00 11 FC
0144    00 90 FF C8             db      000h,090h,0ffh,0c8h,041h,0f8h,004h,000h
0148    41 F8 04 00
014C    0C 90 5A A5             db      00ch,090h,05ah,0a5h,080h,001h,067h,008h
```

```
0150    80 01 67 08
0154    D1 FC 00 00             db      0d1h,0fch,000h,000h,010h,000h,060h,0f0h
0158    10 00 60 F0
015C    0C 68 60 00             db      00ch,068h,060h,000h,000h,020h,066h,0e8h
0160    00 20 66 E8


68008-UMSCHALTUNG       MACRO-80 3.4    01-Dec-80       PAGE    1-1


0164    0C 68 60 00             db      00ch,068h,060h,000h,000h,024h,066h,0e0h
0168    00 24 66 E0
016C    4E E9 00 24             db      04eh,0e8h,000h,024h
0170                    bootende:
                                .phase zsp+bootende-bootstart
8054    3E 80           last:   ld a, baout             ;Bank-Boot ausschalten
8056    D3 C8                   out (bank),a
8058    21 8000                 ld hl, zsp              ;Datenblock auf Adresse 0
805B    11 0000                 ld de, 0                ;kopieren
805E    01 0054                 ld bc, bootende-bootstart
8061    ED B0                   ldir
8063    3E 90                   ld a,b68008             ;Bank-Boot ausschalten
8065    D3 C8                   out (bank), a           ;und 68008 einschalten
8067    76                      halt
8068                    lastende:
                                .dephase
                        ;
                                .phase zsp+bootende-bootstart+copy1ende-copy1+lastende-last
8076    21 8000         copy2:  ld hl, zsp              ;kopiert Datenblock von zsp nach bankad
8079    11 2000                 ld de, bankad
807C    01 0076                 ld bc,copy2-zsp
807F    3E 00                   ld a, baon              ;Bank-Boot anschalten
8081    D3 C8                   out (bank), a
8083    ED B0                   ldir
8085    C3 2000                 jp copy1
8088                    copy2e:
                                .dephase
0196                    datend:


68008-UMSCHALTUNG       MACRO-80 3.4    01-Dec-80       PAGE    S


Macros:

Symbols:
B68008  0090    BANK    00C8    BANKAD  2000    BAON    0000
BAOUT   0080    BOOTEN  0170    BOOTST  011C    COPY1   2000
COPY11  010E    COPY1E  200E    COPY2   8076    COPY2E  8088
DATEND  0196    LAST    8054    LASTEN  8068    START   0100
ZSP     8000


No  Fatal error(s)
```



Kompakt-Kurs

Digital-Computer-Labor

NEU

200 Seiten reichbebildertes Lehrmaterial mit Sammelordner, Register und Logik-Simulator für den Commodore C64 / C128 oder IBM- und kompatible Computer mit MS-DOS. Gesamtpreis DM 448,–

Lernen Sie die Bauelemente und Grundschaltungen der Digitaltechnik mit einem ausgezeichneten Logik-Simulator kennen.

Der Lehrgangsaufbau

Der Lehrstoff ist in zwei Fachgebiete aufgeteilt; die Digitaltechnik und die Steuerungstechnik. In der Digitaltechnik machen Sie zunächst Bekanntschaft mit den Grundlagen. Theorie und Praxis gehen Hand in Hand. Sie lernen einfache Schaltungen mit dem Logik-Simulator auf dem Bildschirm darzustellen und auszutesten.

Eine Zeichnung auf dem Bildschirm, die man ausprobieren kann – ein einmaliges Erlebnis sinnvoller Computeranwendung. Nahezu alle Schaltungen, die im Lehrmaterial beschrieben sind, können mit dem Logik-Simulator ausgetestet werden. Das Aufbauen von Schaltungen hat damit ein Ende. Sie werden begeistert sein.

Digitaltechnik

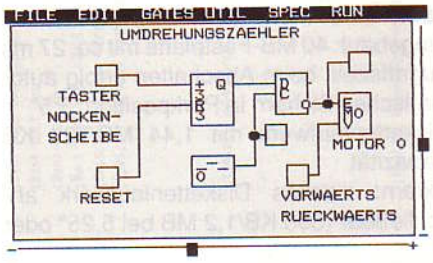
Die Digitaltechnik behandelt alle wichtigen Bauelemente. Wir beginnen mit den logischen Verknüpfungen und gehen über Speicher und Flip-Flops, Schaltalgebra, Zahlensysteme, Codierschaltungen, bis hin zur Mikroprozessortechnik, um nur einige der interessanten Themen zu nennen.

Steuerungstechnik

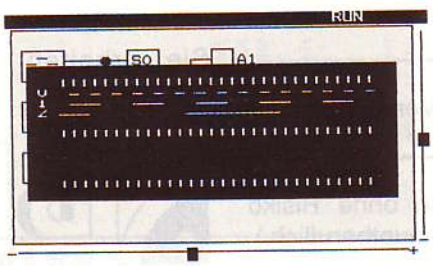
Die Steuerungstechnik hingegen zeigt Ihnen die Anwendung der digitalen Bauelemente. Ablaufsteuerungen, Schaltverstärker, Schrittmotorsteuerungen und vieles mehr wird innerhalb des Kurses ausführlich und leichtverständlich behandelt.

Der Logik-Simulator

Sicher wird Sie der neue leistungsstarke Logik-Simulator begeistern. Über POP-UP-Menüs können die verschiedensten Funktionen angewählt werden. Digitale Verknüpfungen, Schalter, Leuchtdioden, Motor, Lautsprecher, Flip-Flops, Zeitgeber und Zähler sind abrufbare Elemente.



Damit dynamische Prozesse auch leicht verfolgt werden können, kann der Logik-Simulator sogar ein Oszilloskop darstellen. Einfach das Symbol des Oszilloskops anwählen, und schon kann man bis zu 8 Signale verfolgen.



Für wen ist der Kompakt-Kurs interessant?

Für alle, die sich in das Gebiet der Digitaltechnik einarbeiten wollen. Für alle, die den Computer als Werkzeug bei der Schaltungsentwicklung kennenlernen und ausnutzen wollen. Und natürlich für jeden, der sich mit Computeranwendungen und -steuerungen auskennen will.

Was Sie brauchen

Sie brauchen als Vorkenntnisse lediglich Elektronik-Grundlagen und natürlich einen Commodore C64 oder C128 mit Diskettenlaufwerk und Joystick oder einen IBM- oder kompatiblen Computer mit Mouse. Die Software (Logik-Simulator) ist in der Kursgebühr enthalten.

Fangen Sie jetzt an! Die Technik der Computeranwendungen wartet nicht. Bald wird es selbstverständlich sein, Schaltungen aller Art mit dem Computer zu entwickeln. Mit dem Wissen aus diesem Kurs liegen Sie dann ganz vorn.

Christiani
Fortbildung

Postfach 35 000 · 7750 Konstanz
Telefon (0 75 31) 58 01-0

Kei Thomsen

Es geht auch umständlicher!

Oder wie man in Assembler schnelle Programme schreibt.

Leider sieht man es nur zu häufig, wie man den 680XX vergewaltigen kann und damit den Anfängern einen schlechten Stil vormacht. Da mich dieses schon lange stört, möchte ich hiermit zeigen, wie es einfacher geht.

Die Sache mit dem Flags

Die häufigsten falschen Fehler sieht man bei Programmverzweigungen. Da werden Daten aus dem Speicher ausgelesen und dann langwortweise mit Null verglichen. Ich könnte sterben vor Langeweile.

Folgendermaßen sieht das dann meist als Programm aus: :

```
Byte Taktzyklen (68008) -
MOVE.L SPEICHER,D0 6 40 + 10*Wait

CMP.L #0,D0 6 26 + 6*Wait
BEQ UNDSOWEITER 4 18 + 4*Wait
Gesamt: 16 84 + 20*Wait
```

Analysieren wir mal dieses kleine Programm. Zunächst ist da der MOVE.L Befehl.
Dieser ist schon mal gar nicht so schlecht, aber für meinen Geschmack etwas zu lang und dann nicht mal relocatierbar. An diesem Befehl kann man schon erheblich sparen. Hier setzt man dann die programmzählerrelative Adressierung ein. Einfacher gesagt:

MOVE.L SPEICHER (PC) , D0

Dieses (PC) tut schon unheimlich viel.
1. Der besagte Speicher wird relativ zu dem momentanen Programmzähler adressiert und liegt nicht wie vorher auf einer festen Adresse. Das hat schon mal den Vorteil, daß das Programm verschiebbar ist und so in jedem Speicherbereich laufen kann.
2. Der Befehl ist um 2 Byte kleiner als der andere und somit schneller. Das sind beim 68000 schon 2 Taktzyklen und beim 68008 sogar 8 Taktzyklen multipliziert mit der Anzahl der Waitstates.
Als nächstes ist der CMP.L #0,D0 Befehl an der Reihe. Diesen streichen wir zunächst erst mal ganz weg. Warum? Der MOVE Befehl setzt die Flags schon beim Übertragen von Daten entsprechend dem Inhalt. Es wird dabei das Z-Flag auf Eins gesetzt wenn, die übertragenen Daten 0 sind. Sollten die Daten ungleich 0 sein, so steht das Z-Flag auf Null. Gleichzeitig setzt der Prozessor auch das N-Flag, je nachdem, ob die Daten negativ oder positiv waren.
1 = Daten negativ - 0 = Daten positiv. Mit diesen beiden Flags kann man schon viel anfangen. Der Prozessor weiß also, ob die Daten gleich 0, ungleich 0, negativ oder positiv sind. Viel mehr tut der CMP Befehl auch nicht. Also kann dieser wegfallen. Den BEQ-Befehl kann man nicht mehr verbessern, es sei denn das UNDSOWEITER liegt maximal 128 Byte von diesem Befehl entfernt, dann kann man hier
BEQ.S UNDSOWEITER schreiben.
Unser obriges Programm sieht nun so aus:

```
Byte Taktzyklen (68008)
MOVE.L SPEICHER(PC),D0 4 32 + 8*Wait
BEQ UNDSOWEITER 4 18 + 4*Wait
Gesamt: 8 50 + 12*Wait
```

Wie man deutlich sieht, ist die Anzahl der Bytes auf die Hälfte und die der Taktzyklen auf unter 2/3 zurückgegangen. Das ist bei dem 68008 schon eine ganze Menge. Selbst wenn in einem Programm aufgrund einer Null in einem Datenregister verzweigt werden soll, so ist es nicht sehr schön, den CMP.L #0,D0 zu benutzten. Anstatt dieses Befehls sollte man TST.L D0 benutzen, da dieser schneller und kleiner ist. Der TST Befehl braucht nur 8 Taktzyklen gegenüber dem CMP, der 28 braucht.

Tabellenverarbeitung

Dieses Wort ist den meisten Anfängern und auch Fortgeschrittenen kaum bekannt. Dabei macht es viel Spaß, mit Tabellen zu arbeiten, weil diese sehr einfach und übersichtlich zu programmieren sind.
Das Beispiel soll die Abfrage eines Menüs darstellen und die anschliessende Verzweigung nach den einzelnen Menüpunkten.

```
MENUE: MOVEQ #ICI,D7 ; Zeicheneingabe
über TRAP
TRAP #1 ; Wir wollen ja Kompatibel zu
anderen Prozessoren sein.
SUBI.B #'',d0 ; Zeichen von auf 0 bringen
BMI.S MENUE ; Falsches Zeichen, kein
Buchstabe
ANDI.B #$1F,D0 ; Damit Groß und Kleinbuchstaben
LSL.W #2,D0 ; Zeichen * 4 für Tabelle
JSR MENUTAB(PC,D0.W) ; Sprung in die
Tabelle
BRA WEITER ; Hier landet das Programm
nach einem RTS

FEHLER: RTS
MENUTAB
BRA FEHLER ;  / ' nicht vorhanden
BRA TEILA ; A / a
BRA TEILB ; B / b
... ; C..X hier gesamtes Alphabet eintragen
BRA FEHLER ; Y / y nicht vorhanden
BRA TEILZ ; Z / z
BRA FEHLER ; [ / {
BRA FEHLER ; \ /  (ASCII 124)
BRA FEHLER ; ] / }
BRA FEHLER ; / ~
BRA FEHLER ; _ / DEL
```

Bild 1: Menü mit Hilfe einer Tabelle

```
ÄMENUE: JSR CI
CMP.B #'A',D0
BEQ TEILA
CMP.B #'B',D0
BEQ TEILB
... weiter wie gehabt
CMP.B #'Z',D0
BEQ TEIL
... Fehlerausgabe
BRA MENUE
```

Viele schreiben dieses Menü nun folgendermaßen:

Das ist aber ziemlich schreibaufwendig, lang und nicht sehr schnell. Wie man sieht, geht das auch nur mit Großbuchstaben oder noch mal dasselbe für die kleinen Zeichen. Das Ganze nun auf anständige Weise mit einer Tabelle sieht aus wie auf Bild 1.

Wie man hier sieht, ist solch eine Tabelle sehr einfach zu überschauen undauch noch kurz. Es muß dabei nur auf zwei Dinge sehr geachtet werden: Es dürfenkeine BRA.S- Befehle benutzt werden, da in der Tabelle jeder Eintrag an einer bestimmten Stelle stehen muß. Zum anderen muß die Tabelle 32 Einträge haben, da es auch 32 Einsprungmöglichkeiten gibt.

Multiplizieren und Dividieren

Es ist sehr schön, daß der 68000 die DIVS, DIVU, MULS und MULU Befehle hat. Dieses verleitet leider nur zu häufig zu unschönen Programmen. So sieht man sehr oft, daß Daten mit einer 2er Potenz (Potenz im mathematischemSinne), also 2 4 8 16 32 64 128 256 ... Multipliziert oder Dividiert werden.

Divisionen und Multiplikationen mit solchen 2er Potenzen können viel schneller mit Schiebebefehlen ausgeführt werden. Dabei wird ein Links-Schieben als Multiplikation und ein Rechts-Schieben als Division

```
MULU #2,d0 = LSL.W #1,d0
MULS #4,d0 = LSL.W #2,d0
DIVU #8,d0 = LSR.W #3,d0
DIVS #16,d0 = ASR.W #4,d0
```

durchgeführt. Was geschieht nun bei dem Schieben? Beim Links-Schieben wird der Datenwert bei jedem Schieben mit 2 multipliziert. Beim Rechts-Schieben entsprechend durch 2 dividiert. Folgende Tabelle zeigt, wie dieses funktioniert.

Als Anfangswert wird 5 gewählt,der mit 8 multipliziert werden soll. Da 2 hoch 3= 8 ist, wird der Datenwert 3 Bit nach links geschoben. 5 ist Binär 00000101. Diese Daten 3 Bit nach links geschoben ergeben 00101000 = 40. Diesen Datenwert wollen wir nun durch 4 dividieren. Da 2 hoch 2 = 4 ist, müssen wir 2 Bit nach rechts schieben. Das ergibt 00001010 = 10. Also stimmt unsere Rechnung wieder.
Um nun noch das Vorzeichen bei der Division zu berücksichtigen, wird ein anderer Befehl für DIVS als für DIVU benutzt.

Abschließend noch ein Programm, welches den Gebrauch von Tabellen sehr gut wiedergibt.

```
tag   FF.0D.20  FF:00:20
 Motorola 68020/68881 Macro-Assembler                    Seite 1

                ; Testprogramm zur Demonstration von Tabellenverarbeitung.
                ; Es soll ein Menue erstellt werden und danach die Abfrage auf die Zeichen
                ; A bis G und a bis g gemacht werden.

 49FA 0146      start:  lea graftab(pc),a4      ; Adresse der Grafiktabelle laden
 3E1C           graf11: move.w (a4)+,d7         ; Befehl auslesen
 6708                   beq.s graf12            ; war Befehl = 0 dann Tabellenende
 321C                   move.w (a4)+,d1         ; X laden
 341C                   move.w (a4)+,d2         ; Y laden
 4E41                   trap #1                 ; Befehl ausfuehren
 60F4                   bra.s graf11
 49FA 0178      graf12: lea texttab(pc),a4      ; Adresse der Texttabelle laden
 301C           graf13: move.w (a4)+,d0         ; Zeichengroesse auslesen
 6712                   beq.s graf14            ; war Groesse = 0 dann Tabellenende
 321C                   move.w (a4)+,d1         ; Textadresse relativ auslesen
 41FB 1052              lea text(pc,d1.w),a0    ; Textadresse berechnen
 321C                   move.w (a4)+,d1         ; X laden
 341C                   move.w (a4)+,d2         ; Y laden
 3E3C 000A              move #!write,d7
 4E41                   trap #1                 ; Befehl ausfuehren
 60EA                   bra.s graf13
 3E3C 000D      graf14: move #!csts,d7          ; Zeichen testen
 4E41                   trap #1
 4A00                   tst.b d0                ; Test ob 0
 67F6                   beq.s graf14
 3E3C 000C              move #!ci,d7            ; Zeichen lesen
 4E41                   trap #1
 0400 0040              subi.b #'@',d0
 6B0A                   bmi.s fehler1
 0200 0007              andi.b #$07,d0          ; Nur zeichen @ - G / ` - g durchlasssen
 E548                   lsl.w #2,d0             ; X 4 fuer Tabelle
 4EBB 0006              jsr sprung(pc,d0.w)
                fehler1:
 4E75                   rts                     ; Programm Ende
                Fehler:
 4E75                   rts

 6000 FFFC      sprung: bra Fehler
 6000 01BE              bra menueA
 6000 0192              bra menueB
 6000 0196              bra menueC
 6000 019A              bra menueD
 6000 019E              bra menueE
 6000 01A2              bra menueF
 6000 01A6              bra menueG

 52657374617572 text:   dc.b 'Restaurant',0
 616E7400
 4D656E75653A00 text1:  dc.b 'Menue:',0
 41203D20446174 text2:  dc.b 'A = Datensalat mit Knoblauch',0
 656E73616C6174
 206D697420486F
 6F626C61756368
 00
 42203D20436869 text3:  dc.b 'B = Chips a la 68000',0
 70732061206C61
 20363B30303000
 43203D20476573 text4:  dc.b 'C = Geschnetzelte Disketten',0
 63686E65747065
 6C7465204469736
 6B65747465E00
 44203D20537061 text5:  dc.b 'D = Spagetticode',0
 676574746963F

Motorola 68020/68881 Macro-Assembler                    Seite 2

 646500
 45203D20486569 text6:  dc.b 'E = Heisse ICs auf Eis',0
 7373652049437
 206175662045697
 7300
 46203D20477261 text7:  dc.b 'F = Grafik nach Art des Hauses',0
 66696B206E6163
 68204172742064
 657320486175735
 657300

 47203D206F6465 text8:  dc.b 'G = oder fahren sie lieber Bus?',0
 7220666168726
 6E20736965206C
 696568265722042
 75733F00
 42697474652077 text9:  dc.b 'Bitte waehlen Sie:',0
 61656B6C65AE20
 536963A00

                graftab:
 0010 0000 0000         dc.w !clr,0,0           ; Bildschirm loeschen
 0008 00CB 0000         dc.w !moveto,200,0
 0009 00C8 0064         dc.w !drawto,200,100
 0009 012C 00AA         dc.w !drawto,300,170
 0009 0190 0064         dc.w !drawto,400,100
 0009 00CB 0064         dc.w !drawto,200,100
 0009 0190 0000         dc.w !drawto,400,0
 0009 00C8 0000         dc.w !drawto,200,0
 0009 0190 0064         dc.w !drawto,400,100
 0009 0190 0000         dc.w !drawto,400,0
 0000 0000 0000         dc.w 0,0,0              ; Ende

                texttab:
 0021 0000 00F0         dc.w $21,text-text,240,110
 006E
 0022 000B 0014         dc.w $22,text1-text,20,240
 00F0
 0011 0012 000A         dc.w $11,text2-text,10,220
 00DC
 0011 002F 000C         dc.w $11,text3-text,12,210
 00D2
 0011 0044 000E         dc.w $11,text4-text,14,200
 00CB
 0011 0060 0010         dc.w $11,text5-text,16,190
 00BE
 0011 0071 000E         dc.w $11,text6-text,14,180
 00B4
 0011 008B 000C         dc.w $11,text7-text,12,170
 00AA
 0011 00A7 000A         dc.w $11,text8-text,10,160
 00A0
 0011 00C7 0014         dc.w $11,text9-text,20,140
 00BC
 0000 0000 0000         dc.w 0,0,0,0
 0000

 41FA FE9C      menueA: lea text2(pc),a0
 6000 002E              bra menueout
 41FA FEB1      menueB: lea text3(pc),a0
 6000 0026              bra menueout
 41FA FEBE      menueC: lea text4(pc),a0
 6000 001E              bra menueout
 41FA FED2      menueD: lea text5(pc),a0
 6000 0016              bra menueout

Motorola 68020/68881 Macro-Assembler                    Seite 3

 41FA FED8      menueE: lea text6(pc),a0
 6000 000E              bra menueout
 41FA FEEA      menueF: lea text7(pc),a0
 6000 0006              bra menueout
 41FA FF01      menueG: lea text8(pc),a0

                menueout:
 7022                   moveq #$22,d0
 323C 0064              move.w #100,d1
 343C 00F0              move.w #240,d2
 3E3C 000A              move #!write,d7
 4E41                   trap #1
 4E75                   rts             ; Ruecksprung hinter JSR


 Endadresse PC
 Endadresse PC + OFFSET
 Fehler entdeckt
 Ende-Symboltabelle / Anzahl Symbole : 28
```

Programm zur Demonstration von Tabellenverarbeitung. (Menüerstellung und Abfrage der Zeichen A bis G und a bis g

H.Weinert

Windows unter Modula 2

Im folgenden Artikel möchte ich mit Ihnen ein kleines Programm zur Window-Verwaltung unter MODULA-2 entwickeln. Als erstes schlage ich vor, ein Pflichtenheft zu entwerfen. Daran können wir uns später immer orientieren und überprüfen, ob wir auch nichts vergessen haben. Bei der Erstellung des Pflichtenheftes können wir unserer Phantasie erst mal freien Lauf lassen. Abstriche können wir ja immer noch machen. Bevor Sie also das Pflichtenheft weiter unten durchlesen, wäre es doch interessant, ein eigenes zu erstellen.

Pflichtenheft:
Prozeduren zum Öffnen und Schliessen von Windows. Die Anzahl der geöffneten Fenster soll nur durch den Speicherplatz begrenzt sein. Das Fenster soll umrandet und evtl. tituliert werden. Es sollen Prozeduren zum Löschen und zur Neuausgabe eines Windows gestellt werden. (Sehr wichtig für einen Scroll). Darstellung eines Text-Cursors (Underline-Cursor, blinkend). Der Maus-Cursor soll zerstörungsfrei über den Hintergrund bewegt werden. Es müssen also auch Prozeduren zum Abfragen und Setzen der aktuellen Cursor-Positionen gegeben sein).
Für die Eingabe von Zeichen sollen zwei Prozeduren zur Verfügung gestellt werden, die wie 'Terminal.CondRead' und 'Terminal.ReadChar' funktionieren, jedoch die Zeichen nicht auf den Bildschirm ausgeben. Für die Zeichenausgabe soll schließlich eine Prozedur definiert werden, die auch Sonderzeichen für die Fensterumrandung ausgibt. Schließlich soll der GDP direkt angesteuert werden.

Nachdem wir also nun ungefähr wissen, was wir wollen, können wir uns nun an das Programm machen. Da das Programm noch relativ kurz und einfach ist, ersparen wir uns eine genauere Analyse und gehen gleich zum Programm über. Dabei werden wir feststellen, daß wir einige Punkte des Pflichtenheftes verfeinern und andere einschränken müssen.

Programmbeschreibung:
Bei der Programmbeschreibung gehe ich davon aus, daß Sie bereits die Grundkenntnisse in der Programmierung unter MODULA-2 erworben haben. Wenn Sie sich bereits mit der Quelle der GDP-Ansteuerung von 'p1' vertraut gemacht haben, fällt Ihnen sicher sofort die Ansteuerung durch absolut-adressierte Variablen auf. Auf diese Art wird die Zugriffsgeschwindigkeit erhöht (sogar der Syncron-Impuls kann abgefragt werden), und die Programmstruktur einfacher. Leider hat diese Methode auch einen kleinen Nachteil: Die Adressen der Register-Variablen müssen an die verschiedenen Prozessoren der 68000-Reihe angepaßt werden. Für die direkte Ansteuerung des GDP ist es teilweise notwendig, auf einzelne Bits zuzugreifen. Da Modula bekanntlicherweise keinen Datentyp 'BIT' kennt, muß man sich hier selber helfen. Die Lösung liegt in der Bearbeitung von Mengen. Der Compiler reserviert für jedes Element einer Menge ein Bit, das angibt, ob ein Element in der Mengen-Variablen enthalten ist oder nicht. Definiert man also einen Mengentyp mit acht Elementen, so kann man ein beliebiges Byte bitweise manipulieren (—> Byte-Set). Die Variable 'cursorMode' gibt den Zustand des Text-Cursors an (= an, oder ausgeschaltet), 'mouseX', 'mouseY', 'cursorX' und 'cursorY' beinhalten die aktuelle Position des Maus-, bzw. Text- Cursors und 'wDefStart' zeigt auf das zuletzt geöffnete Window. Zu den Unterprogrammen für die GDP-Ansteuerung ist eigentlich nicht viel zu sagen. Wenn Sie die Arbeitsweise des GDP interessiert, können Sie auf umfangreiche Literatur zurückgreifen.

Wenden wir uns nun den Hauptprozeduren zu. 'OpenWindow' öffnet, wie der Name bereits sagt, ein Bildschirmfenster. Bevor das neue Fenster geöffnet wird, muß natürlich erst mal Speicherplatz für den alten Bildschirminhalt und die Zeilenlängen bereitgestellt werden. Anschließend wird getestet, ob die Window-Größe innerhalb des erlaubten Bereiches liegt, und ob genügend Speicherplatz zur Verfügung gestellt wurde.
Sind alle Bedingungen erfüllt, wird der Fensterbereich gerettet und neu initialisiert. Nach dem Rücksetzen des Pointers wird die Window-Definition in die bestehende Liste eingereiht. 'CloseWindow' schließt das zuletzt geöffnete Window. Dafür muß der Window-Bereich erneut gelöscht werden. Anschließend werden die alten Bildschirmdaten ausgegeben. Zur Ausgabe dieser Daten darf keinesfalls 'RePrintWindow' verwendet werden, da in den Daten evtl. Sonderzeichen enthalten sind. Jetzt muß der Pointer zurückgesetzt werden.
Setzt man den Pointer nicht zurück, wird von DEALLOCATE der falsche Speicherbereich an den Heap zurück gegeben - bei (*$P-*) - bzw. das Programm mit einer Fehlermeldung abgebrochen (bei (*$P+*)). Natürlich darf man jetzt nicht vergessen, die Fenster-Definition aus der Liste zu streichen. 'CharOut' gibt ein beliebiges ASCII-Zeichen auf den Bildschirm aus. Die Sonderzeichen 32C - 37C (für die Umrandung des Fensters notwendig - siehe Konstantendefinition) werden über die Kurzvektorbefehle des GDP geplottet.
Mit Hilfe von 'clear' kann das Hintergrundlöschen gesteuert werden. Wurde der Hintergrund nämlich bereits durch 'Block (...) ' gelöscht, so kann 'clear' auf FALSE gesetzt werden. 'CondIn' greift direkt auf die BDOS-Funktion Nr. 6 'DIRECT CONSOLE I/O = dirConIO' zurück. 'dirConIO' erwartet einen Parameter in D1. Ist der Parameter 0FEH (= getState), liefert 'dirConIO' den Konsolstatus zurueck. Dieser wird durch 'succ' abgefragt. Ist ein Zeichen vorhanden (succ = TRUE) wird dieses über die Funktion 'dirConIO' mit dem Parameter 'getChar' eingelesen.
(Zu 'succ' ist noch folgendes zu sagen : Eine boolsche Variable wird vom Compiler dann als TRUE interpretiert, wenn ihr numerischer Wert ungleich Null ist).
'CharIn' wartet mit der Programmfortführung bis ein Zeichen eingegeben wurde. 'CharIn' steuert auch das Blinken des Cursors. Wenn Sie nun das Definitionsmodul 'GDP.DEF' betrachten, sehen Sie zwei Konstanten : 'paintMode = 00H;' und 'eraseMode = 01H'. Wird an den GDP Befehl 01H gegeben, schaltet er um auf Löschmode (und umgekehrt).
Betrachten Sie nun wieder die Prozedur 'CharIn'. In der Zeile 'cursorMode := (cursorMode + 1) MOD 500;' wird für die Inkrementierung von 'cursorMode' gesorgt, gleichzeitig aber sein Wertigkeitsbereich auf 0..499 eingeschränkt. In der darauffolgenden Zeile wird dann 'cursorMode' durch 250 dividiert. Das Ergebnis kann

jetzt entweder 0, für '0 <= cursorMode < 250', bzw. 1, fuer '250 <= cursorMode < 500' sein. Der GDP wird also abwechselnd in den Schreib-, bzw. Löschmode gesetzt 'PrintWindow' dient zur Ausgabe des Windowinhaltes auf den Bildschirm. Ich glaube, wenn Sie mir bis hierher gefolgt sind, können Sie sich diese Prozedur selbst erklären. Wenn Sie alles verstanden haben, können Sie gleich beginnen, das Programm abzuändern, bzw. neue Ideen einbringen. Im folgenden möchte ich Ihnen einige Erweiterungsvorschläge machen.

Invers-/Cursiv-/Underlined-Schrift

Horizontal- und Vertikal-Scroll (natuerlich mit Scrollbar)

Menueleiste - davon ausgehend Pull-Down-Menues

Anpassung an die GDP64HS Darstellung einer Uhr (Window oeffnen, Uhr einzeichnen und bei Tastendruck wieder schließen)

Prozeduren zum Einlesen und Ausgeben von Icons (wahrscheinlich nur mit GDP64HS schnell genug möglich)

Ein-/Ausgabe auf beliebige Files (nicht nur 'CON:')

Grafikwindows (ebenfalls nur mit GDP64HS)

Wenn Sie sich an diese Aufgaben machen, sollten Sie die Struktur von 'Windows' und 'WindowIO' folgendermaßen ändern:

Ihr Modul sollte Prozeduren zum Retten und Zurückschreiben vom Bildschirminhalten beinhalten (natürlich Umrandung ect. inbegriffen; Die Pointerstruktur vom jetzigen 'Windows' übernehmen Sie dann ebenfalls in Ihr Modul). 'Windows' + 'WindowIO' zusammenfassen in 'TextWindows'.

'TextWindows soll dann auf Ihr Modul zurückgreifen.

Anmerkung der Redaktion:
Leider können wir aus Platzgründen nur die grundsätzlichen Prozeduren abdrucken.
Wer Interesse hat kann sich an folgende Adresse wenden:

Helmut Weinert
Schmidtram 47 A
8220 Traunstein

```
(* ------------------------------------------------------------ *)
(* Schliesst das zuletzt geoeffnete Fenster.  Sind keine Fenster mehr offen, *)
(* wird ohne Operation abgebrochen.                             *)
(* ------------------------------------------------------------ *)


PROCEDURE CloseWindow;
VAR
  i, j : CARDINAL;
  blk  : BlockDescriptor;
  ptr  : WindowPtr;
BEGIN
  (* Testen, ob noch Fenster offen sind *)
  IF wDefStart = NIL THEN
    RETURN
  END(*IF*);

  WITH wDefStart^ DO

    (* Fensterbereich loeschen *)
    GetXY (col, row + rowWidth - 1, blk.x, blk.y);
    blk.w := colWidth * signWidth;
    blk.h := rowWidth * signHeight;
    Block (blk, eraseMode, continous);

    (* Alte Zeileninhalte zurueckgeben *)
    lines := lenAdr^;

    (* Alten Bildschirminhalt wieder ausgeben *)
    FOR i := row TO row + rowWidth - 1 DO
      FOR j := col TO col + colWidth - 1 DO
        CharOut (screenAdr^, j, i, FALSE);
        screenAdr := ADDRESS(screenAdr) + 1
      END(*FOR*)
    END(*FOR*);

    (* Pointer auf alten Stand bringen und Speicher an Heap zurueck *)
    screenAdr := ADDRESS(screenAdr) - LONG(colWidth * rowWidth);
    DEALLOCATE (screenAdr, colWidth * rowWidth);
    DEALLOCATE (lenAdr, TSIZE (LineLens))
  END(*WITH*);

  (* Window aus Liste streichen *)
  ptr := wDefStart;
  IF wDefStart # NIL THEN
    wDefStart := wDefStart^.nextWindow;
    DEALLOCATE (ptr, TSIZE (Window))
  END(*IF*);
END CloseWindow;
```

```
PROCEDURE OpenWindow ((*$C-*)window   : Window;
                      (*$C+*)VAR succ : BOOLEAN);

VAR
  ptr      : WindowPtr;
  blk      : BlockDescriptor;
  i, j, len : CARDINAL;
BEGIN
  (* Speicherplatz reservieren *)
  CondAllocate (ptr, TSIZE (Window), succ);
  ptr^ := window;

  WITH window DO

    len := LEN (title);
    CondAllocate (ptr^.lenAdr, TSIZE (LineLens), succ);

    IF succ THEN
      ptr^.lenAdr^ := lines;
      CondAllocate (ptr^.screenAdr, colWidth * rowWidth, succ)
    END(*IF*);

    (* Test, ob Fenster zu gross, oder zu klein ist *)
    IF ~ succ OR
      (colWidth < 3) OR (rowWidth < 3) OR
      (col + colWidth )> numCols) OR (row + rowWidth )> numRows) THEN
      succ := FALSE;
      RETURN
    END(*IF*);

    (* Fensterbereich loeschen *)
    GetXY (col, row + rowWidth - 1, blk.x, blk.y);
    blk.w := signWidth * colWidth;
    blk.h := signHeight * rowWidth;
    Block (blk, eraseMode, continous);

    (* Fenster  initialisieren *)
    FOR i := row TO row + rowWidth - 1 DO

      (* Zeilenlaengen i* Fenster zuruecksetzen *)
      linesAiü := 0;

      FOR j := col TO col + colWidth - 1 DO

        (* Alten Inhalt des Fensters retten *)
        ptr^.screenAdr^ := screenAi,jü;
        ptr^.screenAdr  := ADDRESS(ptr^.screenAdr) + 1;

        (* Fenster umranden : 'screen' mit Sonderzeichen initialisieren *)
        IF i = row THEN
          IF j = col THEN
            (* Ecke links oben *)
            CharOut (luCorner, j, i, FALSE)
          ELSIF j = col + colWidth - 1 THEN
            (* Ecke rechts oben *)
            CharOut (ruCorner, j, i, FALSE)
          ELSIF (titleAj-1-colü # " ") & (j-1-col < len) THEN
            (* Titel ausgeben, jedoch statt 'blancs' waagrechte Linie *)
            CharOut (titleAj - 1 - colü, j, i, FALSE)
          ELSE
            (* Waagrechte Linie ausgeben *)
```

```
            CharOut (horLine, j, i, FALSE)
          END(*IF*)
        ELSIF i = row + rowWidth - 1 THEN
          IF j = col THEN
            (* Ecke links unten *)
            CharOut (ldCorner, j, i, FALSE)
          ELSIF j = col + colWidth - 1 THEN
            (* Ecke rechts unten *)
            CharOut (rdCorner, j, i, FALSE)
          ELSE
            (* Und wieder horizontale Linie *)
            CharOut (horLine, j, i, FALSE)
          END(*IF*)
        ELSIF (j = col) OR  (j = col + colWidth - 1) THEN
          (* Vertikale Linie *)
          CharOut (verLine, j, i, FALSE)
        ELSE
          (* Bildschirminhalt loeschen *)
          screenAi,jü := blanc
        END(*IF*)
      END(*FOR*)
    END(*FOR*);

    (* Pointer zuruecksetzen *)
    ptr^.screenAdr := ADDRESS(ptr^.screenAdr) -  LONG(rowWidth*colWidth)
  END(*WITH*);

  (* Window in Liste einfuegen *)
  ptr^.nextWindow := wDefStart;
  wDefStart := ptr
END OpenWindow;



PROCEDURE CondIn (VAR   ch : CHAR;
                  VAR succ : BOOLEAN);

CONST
  getState = 0FEH;
  getChar  = 0FFH;
BEGIN
  BDOSbyte (dirConIO, getState, succ);
  IF succ THEN
    BDOSbyte (dirConIO, getChar, ch)
  END(*IF*)
END CondIn;
```

Bild 1: Einige Prozeduren zur Windowsteuerung unter Modula 2

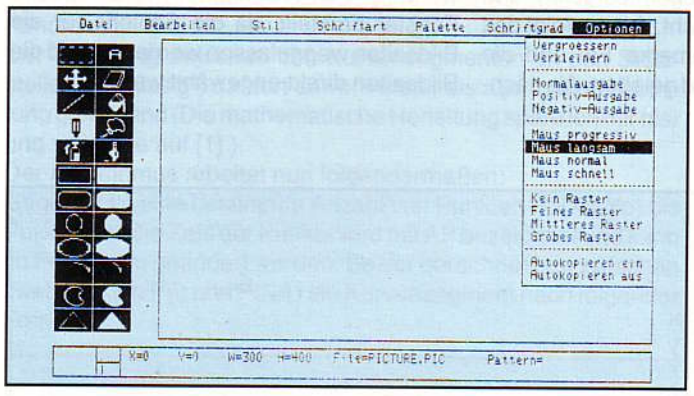
H. Weinert

Zeichnen mit dem Computer

Entwerfen Sie Ihre Zeichnungen in Zukunft ohne Bleistift und Lineal!

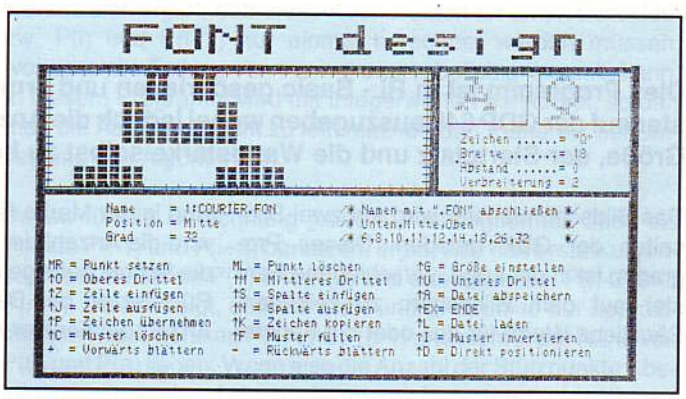
DRAW ist ein universelles Zeichenprogramm, das durch seine übersichtliche Menüsteuerung sehr einfach zu bedienen ist. Der Einstieg ist damit ohne Probleme zu schaffen; schon nach kurzer Eingewöhnungszeit werden Sie keine Schwierigkeiten mehr haben, komplexe Graphiken zu erstellen. Mit seinen zahlreichen Funktionen braucht sich dabei DRAW auch vor PC-Zeichenprogrammen nicht zu verstecken.

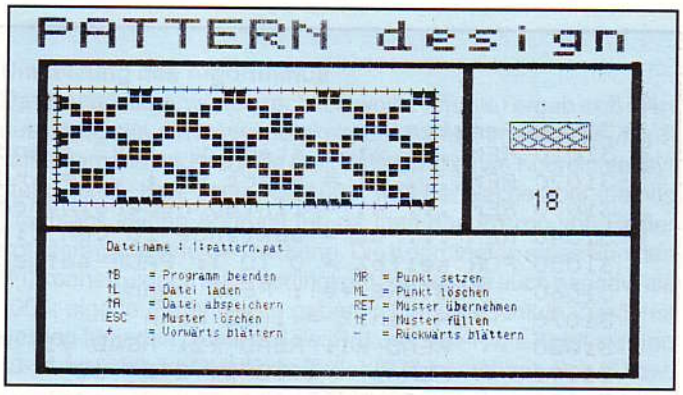
Endlich, ein professionelles Zeichenprogramm für die GDP. Mit DRAW ist es jetzt allen 68000ér Benutzern unter JADOS möglich, mit Hilfe ihres Rechners, Grafiken und Zeichnungen zu erstellen. Das Programm wird über sogenannte Pull-Up Windows mit der Maus gesteuert. Die Funktionen werden durch Ikonen angewählt. Dem Autor Helmut Weinert ist es gelungen, den Read-Modify-Write Modus der neuen GDP64HS zu emulieren,



damit ist dieses Programm auch auf der alten GDP64 lauffähig. Für den Mausanschluß dient die HCOPY oder die neue KEY3. Die HCOPY wird also nur für den Mausanschluß benötigt. An Speicher muß mindestens 128 KByte konfiguriert sein. Die Bankboot-Karte ist nicht notwendig.

Anhand der abgebildeten Hardcopy's kann man schon erahnen wie umfangreich das Programm ist. Etwa 156 verschiedene Funktionen lassen sich mit der Maus anklicken und ausführen.





Um Ihnen in etwa einen Einblick darüber zu verschaffen, hier in Kurzform einige der Funktionen:

Technische Daten:
Abspeichern
Abspeichern als
Laden
Inhaltsverzeichnis auflisten
Arbeitslaufwerk einstellen
Drucken
Drucker installieren
Ende

Bearbeiten der Zeichnungen:
Viewport löschen
Leinwand löschen

Bereiche
Kopieren
Ausschneiden
Invertieren
Löschen
Einsetzen
Einfügen absolut
Einfügen AND
Einfügen OR

Verknüpfen
Absolut, OR, XOR, AND

ZOOM Funktion
Teilbereiche
vergrößern
verkleinern
Positive, Negative Ausgabe
Ganze Seite darstellen

Maus
progressiv, normal, langsam, schnell

Mausraster
keines, feines, mittleres, grobes

Waagerecht spiegeln
Senkrecht spiegeln
Seiten vertauschen

Texte bearbeiten:
Standard
Fett
Kursiv
Schattiert
Unterstrichen
Durchgestrichen
Normal
Punktiert
Linksbündig
Zentriert
Rechtsbündig

Durchsichtig
Undurchsichtig

Zur Musterbearbeitung dient ein eigener PATTERN Editor, in dem es möglich ist eigene Muster zu definieren. Dieser Editor kann ebenfalls mit Hilfe der Maus gesteuert werden. Es können in einer Datei 24 verschiedene Muster abgelegt werden. Natürlich sind beliebig viele solcher Dateien anlegbar.

Das gleiche gilt für den FONT Editor, der es ermöglicht eigene Schriftarten zu definieren.
Sowohl für die Muster als auch für die Schriften stehen bereits Beispiel Dateien bereit. Es kann also gleich mit dem Zeichnen begonnen werden.

Die Druckerausgabe kann bei Bedarf an den jeweiligen Drucker angepaßt werden. Voreingestellt ist ein EPSON Drucker.

Norbert Nissan - Henschke

Fenster

Dies Programm ist in RL- Basic geschrieben und ermöglicht es, Bildfenster auf der GDP 64k auszugeben wobei jedoch die Anzahl der Fenster, die Größe, der Startplatz und die Wandstärke selbst zu bestimmen sind.

Das Bildschirmfenster wird auf zwei Bildseiten der GDP geladen. Dieses Programm ist in der Lage, verschiedene Fenster auf dem Bildschirm zu zeichnen. Sämtliche Werte für das oder die Fenster sind in der Marke Fe! definiert. Als erstes wird die Anzahl der Seiten festgelegt und die Schreibseite gelöscht. Jetzt erfolgt das Rücksetzen der Datamarke Fe! und die Anzahl der Fenster wird geladen. Als nächstes werden die Werte für das jeweilige Fenster geladen und in einer Schleife für die Wandstärke ausgegeben, wobei die Grundposition um die Schrittweite verringert wird. Dies wird solange wiederholt, bis alle GDP-Seiten geladen sind.
Will man mehrere Fenster zu Verfügung haben, so ändert sich die Zeile restore Fe!: read kenn in if kenn1 = 1 then restore Fe1!: read kenn. Jetzt müssen in der Data-Anweisung nur noch die Werte für das Fenster eingegeben werden. Dies Programm ist auch in den Hex-Monitor aufgenommen worden, weil es schnell verschiedene Fenster darstellt, da die Schleife für die Bildseiten weggelassen worden ist und die Bildseiten direkt angewählt werden.

```
31000 fenster!
31010 REM ... x1 = start x  : x2 = ende x
31020 REM ... y1 = start y  : y2 = ende y
31030 REM ... s1 = linien nebeneinander
31040 REM ... s2 = abstande zu den linien
31050 FOR seite% = 0 TO 1: PAGE seite%,1 - seite%: CLPG
31060   RESTORE fe!: READ kenn
31070   FOR zahl% = 1 TO kenn
31080     READ x1: READ x2: READ y1: READ y2: READ s1: READ s2
31090     FOR s = 1 TO s1 STEP s2
31100       CONNECT (x1 + s,y1 + s) - (x2 - s,y1 + s) - (x2 - s,y2 - s) - (x1 +
s,y2 - s) - (x1 + s,y1 + s)
31110     NEXT s
31120   NEXT zahl%
31130 NEXT seite%
31140 RETURN
31150 fe!
31160 DATA 1
31170 DATA 20,490,20,490,5,1
$END
```

Bild 1: Listing des Unterprogramms

A. Ladwig

Grundlagen zur Interpolation

Schon seit einiger Zeit interessierte mich die Möglichkeit der Interpolation. Es gibt nun verschiedene Verfahren, die auch unterschiedliche Zielsetzungen haben.

Bei der Bezierapproximation geht es nun darum, daß ein aufgespanntes Polygonnetz, welches durch eine beliebige Anzahl von Koordinaten definiert ist, von einer parametrischen Kurve angenähert wird. Parametrisch bedeutet dabei, daß die Kurve nicht als Funktion F(x), sondern durch die Punkte X(t) und Y(t) definiert wird. Dieses t durchläuft hierbei Werte zwischen 0 und 1. Dadurch ergibt sich, daß (im Gegensatz zu einer Funktion) zu einem X-Wert mehrere Y-Werte gehören können. Das Ganze läuft im 2D-Raum ab, kann jedoch leicht auf den 3D-Raum übertragen werden, indem man ein Z(t) in gleicher Weise einführt; nur die Darstellung auf dem Bildschirm wird dann schwieriger. Es ist nicht Sinn von Bezier, die vorgegebenen Stützwerte des Polygonnetzes zu durchlaufen (Hierfür benützt man z. B. Kubische Splines oder Blending Functions).
Der Übergang zwischen den Kurvensegmenten an den Stützstellen ist "flüssig", da dort sowohl erste, als auch zweite Ableitung gleich sind (Die mathematische Herleitung spare ich mir hier und verweise auf [1]).
Der Algorithmus arbeitet nun folgendermaßen:
Gegeben ist eine bestimmte Anzahl von Punkten Po(Xo,Yo) bis Pn(Xn,Yn). Die Zahl der Punkte wird mit AP bezeichnet und kann im Programm geändert werden. Bezier berechnet nun zwischen zwei Punkten P(i) und P(i+1) ein Kurvensegment nach folgender Formel:

```
X(t) = ((a3*t+a2)*t+a1)*t+a0      Wobei t zw. 0 und 1
Y(t) = ((b3*t+b2)*t+b1)*t+b0      verläuft
```

ACHTUNG 3,2,1,0 SIND INDEXE !!!!!!!!
man kann diese Formeln auch umschreiben:

```
X(t) = a3*t^3 + a2*t^2 +a1*t + a0
Y(t) =  analog dazu
```

Die Koeffizienten a3-a0 bzw. b3-b0 werden folgendermaßen berechnet:

```
a3 =[ -X(i-1) + 3X(i) - 3X(i+1) + X(i+2) ]/6
a2 =[ X(i-1) - 2X(i) + X(i+1) ]/2
a1 =[ -X(i-1) + X(i+1) ]/2
a0 =[ X(i-1) + 4X(i) + X(i+1) ]/6
```

Für b analog

```
b3 =[ -Y(i-1) + 3Y(i) - 3Y(i+1) + Y(i+2) ]/6
usw.
```

Ferner gilt:geg: 3 Punkte A,B,C

```
IM SEGMENT  A-B
   X(1) = X(0)
   IM SEGMENT B-C
    ==>
```

Wenn man mit X(0) beginnt, braucht man X(1) im selben Segment nicht zu berechnen, sondern bricht einen Schritt vorher ab; woraus sich ergibt, daß diese Koeffizienten für einen Durchlauf zw. P(i) und P(i+1) nur einmal berechnet werden müssen, wogegen die Teilung von t (zw. 0 und 1) beliebig fein sein kann. In diesem Programm wird mit Integerwerten gerechnet, so daß man die Abstufung nicht zu fein machen darf (näheres. bei Umsetzung des Algorithmus).

Da wir für die Berechnung jedes Kurvensegmentes auch die Punkte P(i-1) und P(i+2) benutzen, ergibt sich das erste Kurvensegment zwischen P(1) und P(2) und das letzte zw. P(n-2) und P(n-1). Daraus folgt, daß die Start-und Endpunkte der approximierten Kurve nahe an P(1) und P(n-1) und nicht in der Nähe von P(0) und P(n) liegen. Wenn also die Anzahl der Stützpunkte 7 beträgt, dann werden 4 Kurvensegmente berechnet.

Umsetzung des Algorithmus

Da ich mit Integerwerten rechnen wollte (mußte) ergab sich hieraus einerseits die Notwendigkeit, die Formeln geschickt umzuschreiben und andererseits die Festlegung eines Koordinatensystems mit X- bzw Ymax von 1000, was der Sache aber meines Erachtens keinen Abbruch tut, da man das Prinzip von Bezier trotzdem gut kennenlernen kann. Die Koordinaten eines Punktes P(i) können jedoch auch bedingt größer sein, ja auch negativ, als 1000; eigene Experimente geben schnell Aufschluß. Zunächst werden für jeden Durchlauf zw. P(i) und P(i+1) die Koeffizienten a3-a1 bzw. b3-b1 OHNE die Teiler (6,2) berechnet. Das X bzw. Y wird in einer Zählschleife nach folgendem Prinzip berechnet:

```
0<= j < N   (mit j = Zählvariable und N = Anzahl Punkte
pro Segment)
t ist ja Element von[0;1] und t = j/N

X(t)=[(a3 ohne Teiler*j*j*j)/(6*N*N*N)]
  +[(a2 ohne Teiler*j*j)/(2*N*N)]
+[(a1 ohne Teiler*j)/(2*N)]+[(a0 ohne Teiler)/6]

Y(t) = analog mit b3 ohne Teiler-b0
```

Die Division erfolgt erst nach den gesamten Multiplikationen (pro Klammer), um die unvermeidlichen Rundungsfehler so klein wie möglich zu halten.

Der maximale Wert von N ergibt sich aus der Langwortdeklaration von Nenner3 :

(6*N*N*N) <= 2^32 =4294967296 ==> N <= 894

Bei der Deklaration von Nenner2 (2*N*N) als:
.W ==> N <= 181
.L ==> N <=46340

Wie man also sieht, ergeben sich durch entsprechende Deklarationen ausreichend hohe Werte.
Zur Struktur des Programmes noch folgendes:
Nach den üblichen Definitionen und Initialisierungen werden zunächst die Werte der Stützpunkte eingelesen und festgelegt, ob diese direkt gezeichnet oder vorher aufgelistet werden. Die Nummer des Stützpunktes wird mit 2 multipliziert, da die Koordinaten als Wortwerte abgelegt werden. Die so errechnete Zahl dient dann in D6 als Adreßoffset Danach werden die Nenner berechnet und abgelegt. Bei Änderung von N erspart dies Rechen- und Editierarbeit.

Bei AP Punkten werden ja AP-3 Kurvensegmente berechnet. Da die Hauptschleife mit dem Befehl DBRA abgeschlossen wird, heißt die Anweisung für die Zähl-variable: AP-4 . In dieser Schleife werden dann zunächst einmal die Koeffizienten ohne ihre Teiler berechnet und die Register für den nächsten Stützpunkt adressiert. In der Routine F von T werden dann die Zähler und Nenner von X bzw. Y zusammen gesetzt und für eine definierte Anzahl (=N) werden dann die Koordiaten berechnet und in einem Puffer abgelegt. Diese Routine wird solange aufgerufen, bis alle Kurvensegmente berechnet sind. Danach erfolgt dann entweder direkt die Zeichnung oder erst die Liste der Werte. Nun werden das Koordinatensystem, der Graph und das Polygon gezeichnet.

Änderungen in einigen Details kann jeder nach Belieben vornehmen. Denkbar wäre z.B. eine derart veränderte Eingabe der Koordinaten, daß diese in die Graphik eingeblendet wird, was jedoch den Platz für die Kurven verringert.

Literatur: [1] "Programming Principles in Computer Graphics"
Leendert Ammeraal ISBN: 0 471 90989 0 S. 28 - 33

Listings können angefordert werden bei:

Graf Elektronic Systeme GmbH Telefon: (0831)6211

Volker Wiegand

OS-9/68000

Aufbau der Treiber

In unserer vorläufig letzten Folge wollen wir uns mit dem Aufbau der Treiberprogramme beschäftigen. Diese Treiber sind wichtig für alle, die selbst Hardware an das OS-9 System anpassen wollen und können außerdem sehr viel für das Verständnis des OS-9 beitragen.

Eines der größten Probleme des OS-9 Betriebssystems ist, wie uns allen bekannt ist, die Beschaffung von Informationsmaterial über das System. Leider kann man im Augenblick nur die Aussage treffen, daß es kein Buch oder andere frei zugängliche Literatur über OS-9 gibt oder in absehbarer Zeit geben wird. Man ist also auf die Handbücher, die für OS-9 lieferbar sind, angewiesen. Diese Handbücher sind in Englisch abgefaßt und beinhalten keine Beispiele oder andere Hilfen, um selbst Systemprogramme oder gar Treiber zu schreiben. Allzu viel kann ich an dieser Situation auch nicht ändern, da Microware das Verbot ausgesprochen hat, irgendwelche Informationen der Anpassung an die Hardware, also an den NDR-Computer, zu veröffentlichen. Zu unserem Glück kann Microware dieses Verbot nur für solche Programmteile aussprechen, die auch von Microware selbst stammen. Dazu gehören Beispieltreiber für einige gängige Hardwarebausteine. Die Treiber für den NDR-Computer wurden zum größten Teil von mir selbst geschrieben und daher darf ich sie veröffentlichen. Eine Einschränkung muß jedoch dahingehend gemacht werden, daß diese Treiber von ihrer Struktur her auf den Beispieltreibern aufbauen und Microware diese Strukturinformationen als ihr geistiges Eigentum betrachtet. Daher werde ich im folgenden auf die ohnehin öffentlich bekannten Eigenschaften von Treibern eingehen, um durch eine Veröffentlichung nicht den Unmut Microwares auf mich zu ziehen und stattdessen die von mir komplett geschriebenen Treiber als Beispielprogramme zum Assembler, der in den nächsten Tagen ausgeliefert werden kann, beigeben.

Wenn wir uns die Hierarchie und den modularen Aufbau des OS-9 noch einmal vor Augen führen, erinnern wir uns, daß für Ein-/Ausgabeoperationen Device Drivers und Device Descriptors zuständig sind. Dabei waren die Deskriptoren nichts anderes als Tabellen, die die Hardware näher beschreiben. Die Treiber sind nun die Unterprogramme, die die Informationen aus den Deskriptoren direkt auswerten und die Hardware bedienen, wobei sie sich auf primitive Operationen beschränken. Von diesen Operationen gibt es sieben an der Zahl. Wir wollen sie im folgenden behandeln.

INIT - Gerät initialisieren

Vor der ersten Benutzung muß die Hardware so eingerichtet werden, daß Lese-, Schreib- und Statusinformationen ausgetauscht werden können. Dazu wird die Initialisierungsroutine aufgerufen. Dies geschieht immer, wenn ein neuer Kanal zu einem Gerät geöffnet wird, es kann jedoch auch explizit aufgerufen werden durch den Befehl iniz. In dieser Routine wird zunächst die Adresse der Hardware festgestellt. Wir haben hier eine Besonderheit beim NDR-Computer, die es so in kaum einer anderen OS-9 Implementierung gibt. Wir stellen nämlich fest, um welchen Prozessor es sich handelt (das tun alle anderen auch), und können die Hardware dann auf unterschiedlichen Adressen ansprechen, so daß ein Treiber für alle Busbreiten ausreicht. Als nächstes werden etwaige te von der Interrupttabelle entfernt werden. Ein nach dieser Routine eingehender Interrupt würde von OS-9 nicht mehr zugeordnet werden können und das System in die Knie zwingen.

INT - Interruptroutine

Diese Routine soll hier besprochen werden, obwohl sie nicht in der Sprungleiste des Treiberkopfes vorhanden ist. Sie hat

Puffer für die Ein-/Ausgabe reserviert und die Zeiger darauf zurückgesetzt. Zuletzt muß noch bei den Geräten, die interruptgesteuert sind, der Interrupt eingeschaltet werden (dies sind z.Zt. nur die GDP und SER).

READ - Daten vom Gerät lesen

Als nächstes wollen wir die Routine besprechen, die für die Übertragung von Daten vom Gerät zum OS-9 verantwortlich ist. Wir müssen hier wie beim Schreiben zwischen den verschiedenen Geräteklassen unterscheiden. Die für uns interessanten Klassen sind RBF-, SCF- sowie PIPE-Geräte. Dabei sind RPF und PIPE blockorientiert, SCF dagegen ist zeichenorientiert. Das bedeutet, daß bei ersteren immer ganze Blöcke bewegt werden, welche z.Zt. in ihrer Größe auf 256 Byte beschränkt sind. Letzteres bewegt immer ein Zeichen auf einmal. Es gibt einen Unterschied zwischen Geräten, die mit Interrupt betrieben werden und solchen ohne. Mit Interrupt, welches die vorzuziehende Betriebsart im OS-9 ist, wird von dieser Routine die Hardware gar nicht angesprochen. Dies geschieht in der Interruptroutine. Die Leseroutine tut nichts anderes, als zu prüfen, ob sich Zeichen im Puffer befinden und zu warten, bis genügend vorhanden sind, um die Anforderung zu erfüllen. Ohne Interrupt werden Zeichen oder Blöcke gelesen und an OS-9 direkt übergeben.

WRITE - Daten zum Gerät senden

Das Gegenstück zur Leseroutine ist die Schreibroutine. Auch hier gibt es einen Unterschied in der Implementierung mit oder ohne Interrupt. Mit Interrupt werden die Daten in einen Puffer geschrieben und ein Flag wird gesetzt, daß Daten zum Schreiben bereitstehen. U.U. wird die Hardware allerdings doch angesprochen, etwa um das erste Zeichen ohne Interrupt zu senden oder einfach Sendeinterrupts freizugeben. Ansonsten muß von dieser Routine die Hardware auch selbst bedient werden. Es ist von großem Vorteil, wenn hier Interrupts eingesetzt werden können. Wir wollen uns dies am Beispiel SER klarmachen. Wenn 10 Zeichen zur Schnittstelle zu schicken sind und kein Interrupt benutzt werden kann, so ist es notwendig, den Treiber während aller 10 Sendezeiten nicht zu verlassen, da man damit rechnen muß, sonst nur etwa alle 20_ms die Möglichkeit zu haben, ein Zeichen loszuwerden (das ist genau die Zeit, nach der die Prozesse neu zugeteilt werden). Das würde aber bedeuren Informationen über die Zahl der Zeichen in den Puffern, Informationen über den Betriebszustand, Formatieren von Disketten, Abfrage von Dateigrößen oder Schreiben von Dateideskriptoren.

TERM - Arbeit mit Gerät beenden

Wie wir wissen, ist es im OS-9 möglich, während des Betriebes Geräte anzumelden (init Routine aufrufen über iniz) und auch wieder abzumelden. Dazu dient die Routine term, welche mit dem Befehl deiniz aktiviert werden kann. Diese Routine hat bei Geräten ohne Interrupt keine Bedeutung, bei solchen mit Interrupt hingegen eine sehr große. Sie muß nämlich dafür sorgen, daß alle noch anstehenden Interrupts noch ausgeführt werden und dann die Interrupts sperren, weil die Geräte von der Interrupttabelle entfernt werden. Ein nach dieser Routine eingehender Interrupt würde von OS-9 nicht mehr zugeordnet werden können und das System in die Knie zwingen.

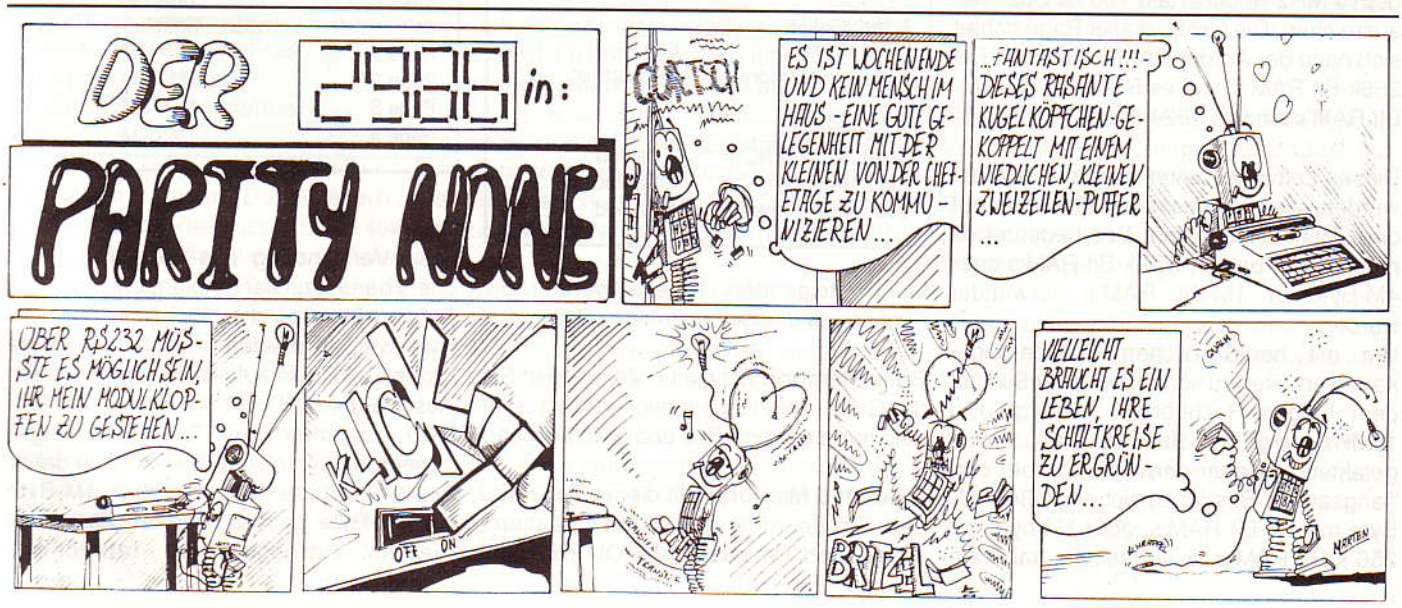
INT - Interruptroutine

Diese Routine soll hier besprochen werden, obwohl sie nicht in der Sprungleiste des Treiberkopfes vorhanden ist. Sie hat aber eine zentrale Bedeutung für die Abwicklung des Datenverkehrs. Dies wird besonders deutlich bei der Betrachtung eines Schreibens auf die SER. Hier hat ja die write Routine nur den Interrupt freigegeben. Dadurch wird sofort ein IRQ ausgelöst und OS-9 verzweigt über seine internen Interrupttabellen, in welche die int Routine eingeschleift wurde, zu dieser Routine.

Hier wird erkannt, daß Zeichen im Puffer vorhanden sind, und daß eine Ausgabe möglich ist. Also wird ein Zeichen aus dem Puffer genommen und ins Ausgaberegister geschrieben. Dann wird OS-9 mitgeteilt, daß der Interrupt erledigt ist. OS-9 kann nun normal weiterarbeiten. Sobald der 6551 das Zeichen abgesendet hat und bereit ist für eine neue Ausgabe, wird der Interrupt wieder ausgelöst. Dies setzt sich fort bis keine Zeichen mehr vorhanden sind und IRQ wird dann gesperrt. Natürlich kann während der Ausgabe der Puffer, der ein Ringpuffer ist, auch beschrieben werden.

Mit diesen Betrachtungen wollen wir unsere kurze Serie über OS-9 zunächst abschließen. Die Einzelheiten der Treiber-Implementierung werden, wie gesagt, beim Assembler mitgeliefert. Zum Lieferumfang des Assembler gehören nämlich die Treiber für SER, CENT, GDP, KEY, sowie FLO2/FLO3. Wir haben bereits damit begonnen, Programme für das OS-9 auf dem NDR Computer zu sammeln und aufzubereiten. In den nächsten Wochen wird ein Rundschreiben an alle OS-9 Benutzer gehen, in dem eine Liste der bisher verfügbaren Public Domain Programme enthalten ist. Diese Programme können dann auf Anfrage geliefert werden.

ch hoffe, daß diese Einblicke in OS-9 Ihr Interesse gefunden haben, und ich bin sicher, daß das Gespann aus NDR Computer und OS-9/68000 noch eine große Zukunft vor sich hat.



H. Klotsche

NEAT-Chipsatz für den mc-modular AT

Der Bedarf an schnellen Rechner steigt immer weiter. Bislang stellte ein Rechner mit einer CPU auf Basis des 80386 Prozessors die beste Lösung dar, wenn da nicht die hohen Anschaffungspreise wären. Mittlerweile gibt es zwei preislich günstigere Lösungen einen schnellen und leistungsgfähigen Rechner zu erhalten.

Die erste Möglichkeit ist ein abgespeckter 80386 Prozessor. Der 80386 SX arbeitet wie sein großer Bruder intern mit 32 Bit, jedoch nach außen nur mehr mit 16 Bit. Die zweite Lösung stellt der NEAT-Chipsatz dar, der auf der Basis eines 80286 Prozessors arbeitet und ihm noch einmal einen ordentlichen Leistungszuwachs verhilft. In diesem Artikel soll gezeigt werden auf welche Art diese Geschwindigkeitserhöhung beim NEAT-Chipsatz erzeugt wird.

Grundlage des NEAT-Chipsatzes ist ein neu überarbeiteter 80286, der Taktfrequenzen bis 16 MHz zuläßt. Bei der bisherigen Speicherverwaltungsart würden nur noch sehr schnelle Speicherbausteine mit dem hohen Rechnertakt zurecht kommen. Da schnelle Speicher teuer sind wurde nach einer neuen Verwaltungsart gesucht, um mit langsameren Speichern auszukommen.

Um auf RAM's zuzugreifen, werden die obere (Reihenadresse) und die untere Adresshälfte (Spaltadresse) nacheinander übertragen und mit Strobesignalen in die RAM's geschrieben. Beim NEAT-Chipsatz wird, wenn die obere Adresshälfte gleich bleibt, nur noch die untere Adresshälfte übertragen. Diese Page-Mode Speicherverwaltung spart natürlich Zeit, so daß nun bei 16 MHz Rechnertakt 100 ns Speicher ausreichen. Die Größe dieser Page richtet sich nach der Art der RAM-Bausteine. Bei 256k-Bit RAM's sind es 512 Byte, bei 1M-Bit RAM's sind es 1024 Byte.

Dieser Zeitvorteil kann aber nur genutzt werden, wenn mindestens 36 RAM's auf der CPU gesteckt sind. Das bedeutet es müssen 1M-Byte in 256 k-Bit RAM's oder 4M-Byte in 1M-Bit RAM's verwendet werden.
Um mit herkömmlichen Erweiterungskarten arbeiten zu können wird der Bus mit dem halben Rechnertakt versorgt. Bei 16MHz wird der Bus also mit 8MHz getaktet. Um aber den Umweg über den "langsamen" Bus für Speicher größer 4M-Byte mit 1M-Bit RAM's, oder 1M-byte mit 256 k-Bit RAM's zu vermeiden, kann ein zusätzliches Board mit 36 RAM's direkt auf der CPU - Karte gesteckt werden.

Da die CPU mit 4M-Byte Speicher (mit Erweiterungsboard 8M-Byte) ausgerüstet werden kann, stellt sich die Frage, wie dieser Speicher genutzt werden kann. Prinzipiell sind Extended- und Expandedspeicher möglich. Zusätzlich können die BIOS- EPROM's in den Bereich zwischen 640K-Byte und unter 1M-Byte kopiert werden. Die jeweilige Speicherdefinition ist im Setup der CPU möglich.

Einige Beispiele dazu:
1. Ausrüstung mit 1M-Byte mit 256 k-Bit Speichern.
Mit diesem Speicherausbau stehen maximal 384 k-Byte Speicher als Shadow- oder Erweiterungspeicher zur Verfügung.

a. 640 k-Byte Hauptspeicher und BIOS kopieren.

Entspricht Standardeinstellung der neuen CPU

Dazu wichtige Punkte im Setup:

1. Setupseite:	
Base Memory	640 K
Extended Memory	not installed
2. Setupseite:	
Shadow BIOS ROM	Enabled
EMS Memory	Disabled
640 - 1024K Relocation	Disabled

In der folgenden Tabelle werden die Funktionen der Setup-Punkte erklärt.

Base Memory: Mit dieser Zeile legen Sie die Größe des Hauptspeichers fest. Hier sind nur die Werte 512 und 640 möglich.

Extended Memory: Mit dieser Zeile wird die Größe des Extended-Speichers angegeben. Um das BIOS ROM in die 384 k-Byte RAM zu kopieren, muß in dieser Zeile unbedingt "not installed" stehen.

Shadow BIOS ROM: In dieser Zeile wird festgelegt, ob das BIOS ROM in den RAM-Bereich iopiert wird (enabled) oder nicht (Disabled).

EMS Memory: Mit diesem Einstellungspunkt wird mit "Enabled" der Erweiterungs-speicher als Expanded-Speicher definiert.

640 - 1024 Relocation: Mit diesem Punkt können Sie bei nicht mehr als 1MByte Speicher die "freien" 384 k-Byte so einblenden, als wäre zusä tzlich zu dem 1MByte Speicher noch 384 k-Byte vorhanden. Dieser Trick ist notwendig, um diese 384 k-Byte überhaupt als Erweiterungsspeicher nutzen zu können. Wenn aber das BIOS ROM in den RAM-Bereich kopiert werden soll, darf diese Funktion nicht ausgeführt werden (Disabled).

b. 640 k-Byte Hauptspeicher 384 k-Byte Expanded-Speicher

1. Setupseite wie unter a.
2. Setupseite

Shadow BIOS ROM	Disabled
EMS Memory	Enabled
640 - 1024k Relocation	Enabled
Page 1	1M - 2M
Page 2	1M - 2M
Page 3	1M - 2M
Page 4	1M - 2M

Die Verwendung des Expanded-Speichers beruht auf der Einblendung von 64k-Byte großen Speicherblöcken. Diese 64k-Byte werden wiederum in 4 x 16k-Byte große Teilblöcke aufgeteilt und dann beim Ausbau der CPU mit 8M-Byte jeden 2M-Byte Speicher ein Teilstück (Page) zugeordnet. Im obigen Fall müssen daher alle 4 Teilstücke in den Bereich 1M-Byte bis 2M-Byte gesetzt werden, da nur in diesem Adressbereich tatsächlich

Speicher vorhanden ist (384k-Byte eingeblendeter Speicher!) Zusätzlich muß dann noch der EMS-Treiber EMM.SYS in die CONFIG.SYS eingebunden werden. Die-ser Treiber wird bei der NEAT 16MHZ-CPU mitgeliefert.

c. 640 k-Byte Hauptspeicher 384 k-Byte Extended-Speicher

1. Setupseite	
Base Memory	640K
Extended Memory	384K
2. Setupseite	
Shadow BIOS ROM	Disabled
EMS Memory	Disabled
640 - 1024k Relocation	Enabled

2. Ausrüstung mit mehr als 1M-Byte

In dieser Konfiguration stehen mit 256k-Bit Speichern 1M-Byte Erweiterungs-speicher und bei 1M-Bit RAM's bis zu 7M-Byte Erweiterungsspeicher zur Verfügung. Die "freien" 384k-Byte Speicher zwischen 640k-Byte und 1M-Byte können dann nur noch als Shadow RAM für das BIOS ROM verwendet werden. Der Grund dafür liegt darin, daß der Adressbereich, in den vorher dieser Speicherbereich gespiegelt wurde, nun tatsächlich mit RAM-Bausteinen belegt ist und es sonst eine Doppelbelegung die Folge wäre. Darum ist auch der Setup-Punkt "640 - 1024k Relocation" unbedingt mit "Disabled" zu versehen.

a. 640 k-Byte Hauptspeicher 1M-Byte Expanded-Speicher

1. Setupseite	
Base Memory	640K
Extended Memory	not installed
2. Setupseite	
Shadow BIOS ROM	Enabled
EMS Memory	Enabled
640 - 1024 Relocation	Disabled
Page 1	1M - 2M
Page 2	1M - 2M
Page 3	1M - 2M
Page 4	1M - 2M

Die Page-Bereiche müssen bei mehr als 1M-Byte Expanded-Speicher natürlich auch in die anderen 2M-Byte großen Teilbereiche günstig verteilt werden, so daß pro 2M-Byte mindestens ein Page-Bereich aktiviert ist.
Zusätzlich muß noch der Treiber EMM.SYS in die CONFIG.SYS eingebunden werden.

b. 640k-Byte Hauptspeicher 1M-Byte Extended-Speicher

1.Setupseite	
Base Memory	640K
Extended Memory	1024K
2.Setupseite	
Shadow BIOS ROM	Enabled
EMS Memory	Disabled
640 - 1024K Relocation	Disabled

Weiter Informationen zu CPU's mit NEAT-Cipsatz sind auch in der CT 3/89 enthalten. Dort wurden zwar Motherboard getestet, aber die Eigenschaften und Leistungen sind auch auf die CPU-Karte für den mc-Modular AT übertragbar.

Abschließend bleibt zu sagen, daß die CPU-Karte mit NEAT-Chipsatz ein günstige und lei-stungsfähige Alternative zur CPU 80386 darstellt. Im Bezug auf die verwendbaren 1M-Bit RAM's läßt mit nur einem Zu-satzboard 8M-Byte Speicher auf der CPU-Karte unterbringen, ohne Bussteckplätze für Speichererwei-terungskarten zu belegen.

Für Sie gelesen:

Clipper-Handbuch

Michael Aselmann-Frasch :
Clipper-Handbuch
iwt-Verlag, Vaterstetten

Schon beim ersten Durchblättern des Buches wird die Zielgruppe für ein solches und ähnliche Bücher klar: Es sind dies die Anwender, die zum Clipper gekommen sind, ohne das dazugehörige Handbuch zu besitzen; denn das Clipper-Handbuch ist der wesentliche Teil des Buches. Einige Sätze sind umgestellt, aber ansonsten ist das Handbuch 1:1 abgeschrieben - leider nicht 1:1, denn die wichtigen Querverweise und die wichtigen Anwendungsbeispiele für die verschiedenen Clipper-Befehle fehlen gänzlich.

Außer über das Handbuch verfügt das Buch nur noch über ein umfangreiches Beispielprogramm, das eigentlich ein Beispiel darstellt, wie man in Clipper nicht programmieren sollte. Allein die Tatsache, daß zum ausgedrucktem Beispiel nachträglich im Foto einige Kommentare dazugepackt wurden, läßt den aufmerksamen Leser verzweifeln. Auch Zeilenumbrüche wurden einfach vorgenommen, obwohl Clipper ohne einen "Strich-Punkt" am Ende einen solchen Umbruch nicht akzeptiert und dies zwangsläufig zum Fehler führt.

Der Autor scheint auch noch nicht festgestellt zu haben, wann man im Clipper Befehle der Übersichtlichkeit wegen groß schreibt und wann klein.

Auch Kommentare, wie z.B. "ab ungefähr hier sollte eine zweite Datei benutzt werden", nachdem 2400 Datensätze geschrieben sind, stimmt einen erfahrenen Clipperprogrammierer eher nachdenklich.

Spätestens innerhalb eines Menü's der Adressverwaltung findet man dann eine Schleife über die gesamte Datei, die eine Nummer einschreibt - ein Beispiel, wie man mit relationalen Daten dann nicht umgehen sollte.

Die sonstigen Hinweise zum Arbeiten mit Clipper sind eher mager, besonders auf das doch komplexere Thema der Overlay-Struktur wird kaum eingegangen; die Fehlermeldungen sind wieder vom Handbuch abgeschrieben.

Zusammengefaßt, ein Buch, dessen Kauf sich sicher nur für die o.e. Zielgruppe vielleicht lohnen wird.

Diese Buchbesprechung gibt die Meinung des Autors, nicht die des Verlags wieder.

Gerd Graf

Volker Stahl

CPU8088 - MS/DOS

Anfang dieses Jahres lötete ich mir die neue CPU8088 zusammen,da ich damit die Möglichkeit sah, endlich auch mit dem NDR-Computer ein Standardbetriebssystem betreiben zu können. Denn für die Betriebssysteme IBM PC/DOS und MS/DOS gibt es die größte Softwarebibliothek der Welt.

Teil 1: NDR PC - IBM PC
Teil 2: PC-Basiswissen
Teil 3: EDLIN.COM
Teil 4: CONFIG.SYS
Teil 5: AUTOEXEC.BAT

Teil 1: NDR PC - IBM PC
Ich selbst habe folgenden Ausbau (den ich auch weiterempfehle):

* CPU8088
* 512 KB RAM
* FLO3
* 2 x 5,25" Diskettenlaufwerke
* GDP64k (nur vorübergehend brauchbar!)
* CENT
* BUS 3
* Netzgerät NE1
* MF-2 Tastatur (IBM XT/AT, 102 Tasten)
* 14" Grün Monitor (nur vorübergehend brauchbar!)
* PC-Gehäuse

Beim ersten Lauf mit meinem DOS bemerkte ich sogleich die negativen Erscheinungen der neuen PC-Konfiguration des NDR-Klein-Computers:

a) Bildschirmausgabe
Die Textausgabe ist nur sehr langsam und das Textscrollen mangelhaft. Tippt man ein Wort schnell ein, so erscheint zwar dieses Wort auf dem Bildschirm, doch mitVerzögerung, man kann zusehen, wie das Wort Buchstabe für Buchstabe aufgebaut wird, naja, sagen wir beinahe. Ein zweiter großer Nachteil der Bildschirmausgabe ist die Grafik, die überhaupt nicht funktioniert. Beide große Nachteile sind der GDP64k zu verdanken. Dank dem hervorragendem BIOS von R.D. Klein emuliert dieses die GDP64k zur Monochromen Textkarte von IBM. Doch auch da ist keine Grafik möglich! Die Verzögerung bei der Textausgabe liegt daran, daß das BIOS jedes Zeichen auf dem Bildschirm 'malen' muß - und dies kostet Zeit! Hoffnungsschimmer: BUS-Adapter der Firma GRAF (ermöglicht Anschluss von PC-Steckkarten) EGA-Karte an NDR-BUS (von Firma GRAF in Vorbereitung!)

b) Diskettenzugriff:
Die Zugriffszeit bei Diskettenoperationen ist wesentlich größer als bei vergleichsweisen PC-Laufwerken.
Auch schäppert der Schreib/Lesekopf viel störender als bei vergleichsweisen PC-Laufwerken. Doch dies sind ja nur kleine Schönheitsfehler und stören nicht weiter!

c) Druckerausgabe:
Mir ist es zuerst nicht gelungen deutsche Umlaute sowie Sonderzeichen auf meinem an der CENT angeschlossenem Epson-Drucker auszugeben. Die Firma GRAF war der Meinung, es liege an der nicht 100% kompatiblen Druckerkarte, doch ich ließ nicht nach und fand doch schließlich noch heraus, wie ich auf meinem Drucker Umlaute ausdrucken kann: Man benütze einen Druckeremulator z.B. LPTGER, der dann automatisch von der AUTOEXEC.BAT, bei jedem Kaltstart installiert wird. Doch ansonsten gibt es keine Schwierigkeiten, abgesehen davon, daß keine Hardcopy von Grafiken möglich sind.

Wie ich schon einmal erwähnte, habe ich mein System in ein PC-Schubgehäuse eingebaut. Erstens sieht es proffesioneller und viel schöner aus, und zweitens ist dann der ganze Kabelsalat aufgeräumt.

Ein solches Gehäuse bietet Platz für bis zu vier Laufwerke, oder zwei Laufwerke und zwei Festplatten, ein Netzgerät, und dem großen RDK-BUS und man bekommt es für weniger als 100,- DM zu kaufen
Angeschlossen habe ich von den drei an der Gehäusefront angebrachten Tastern nur den Reset-Taster. Dazu muß eine Verbindung zwischen diesem Taster und den Lötpunkten auf der CPU8088 Platine (beim Resettaster) hergestellt werden. Auch habe ich den im Gehäuse vorhandenen Minilautsprecher an der CPU8088 angeschlossen. Ein kleiner Fehler: die Sound-Ausgabe ist sehr laut und wirkt auf die Dauer nervtötend. Doch ich habe gleich für Abhilfe gesorgt: einfach den Widerstand R4 auf der CPU8088 gegen einen Trimmer (Trimmwiderstand, einstellbarer Widerstand, ähnlich dem Poti) von 1kOhm auswechseln und schon kann die gewünschte Lautstärke eingestellt werden.

In der nächsten LOOP werde ich im Teil 2 'PC-BASISWISSEN' über die Grundlagen und das Know-how Ihres neuen NDR-PC und seines Betriebssystems berichten.
Bis dahin, viel Spaß mit der CPU8088 -

Volker Stahl.

Für Sie gelesen:

TURBO C

Autor: Gerhard Renner
erschienen im Markt & Technik Verlag
356 Seiten
mit Beispieldiskette
ISBN 3-89090-536-6

TURBO C von Dr. Gerhard Renner, erschienen im Verlag Markt & Technik, ist ein Buch das sich in erster Linie an den schon etwas ***erfahreneren Programmierer*** wendet. Die klare Unterteilung in 25 Kapitel, versehen mit den Kapitelinhalt kennzeichnenden Teilüberschriften, erleichtern den Umgang mit diesem Buch.Es kann somit nach der Durcharbeitung ohne weiteres als Nachschlagewerk herangezogen werden. Das Buch beinhaltet neben einer kurzen Einführung, die wohl eher als kleine Wiederholung für bereits "Vorbelastete" anzusehen ist, ausführliche Informationen über Sprachelemente und Syntax von TURBO C. In die Programmierung unter TURBO C wird anhand vieler praktischer Beispiele eingeführt.

Sehr positiv zu bewerten ist die Tatsache, daß Programmbeispiele nicht mühsam von Hand eingegeben werden müssen. Dem Buch liegt eine Diskette mit einer Vielzahl von Beispielprogrammen bei.

Norbert Grotz

KEY3 - die neue Tastaturkarte

PC-Luxus am NDR-Computer

Bedingt durch den Preisverfall der komfortablen PC/XT-Tastaturen, wurde der Wunsch, solch eine Tastatur am NDR-Computer verwenden zu können, immer größer. Jetzt ist es so weit! Mit der neuen Tastaturbaugruppe KEY3 ist es nun möglich, preiswerte, leicht erhältliche und komfortable PC-Tastaturen an den NDR-Computer anzuschließen. Und auch für eine Maus, inzwischen Standard bei Mikrocomputern, ist eine Schnittstelle vorhanden.

Parallel - seriell - die Zahl der Kabel

Der Unterschied der XT/PC-Tastaturen zu den bisher verwendeten ASCII-Tastaturen liegt erstens in der Datenübertragung und zweitens in der Tastencodierung. Bei einer ASCII-Tastatur werden die Daten parallel übertragen, d.h. jedem Bit des Tastencodes steht eine eigene Leitung zur Verfügung. Da sieben Bits zur Codierung notwendig sind, sind dies sieben Leitungen plus zwei Leitung Versorgungsspannung plus ein Leitung zur Synchronisation (Strobe).
Bei einer XT/PC-Tastatur werden die Daten seriell gesendet, d.h. auf einer Leitung kommt eine Bit nach dem anderen. Auch hier sind noch zwei Versorgungsleitungen und eine Leitung zur Synchronisation notwendig (Takt).

- 256 Zeichen Tatstaturpuffer
- 40 frei belegbare Funktionstasten
- Abspeichern von selber belegten Funktionstasten möglich
- NDR-Bus
- ECB-Bus
- Mausanschluß
- Unterstützung der Smartwatch (ab PGM 2.0)
- Interrupterzeugung (ab PGM 2.0)
- XT/PC-Tastaturanschlu
- ASCII-Tastaturanschluß
- Befehlswiederholung für 7 Befehle

Bild 1: Technische Daten

ASCII-Code - Positionscode - Logik gegen Luxus

Bei einer ASCII-Tastatur wird, wenn eine Taste gedrückt wird, der zur Taste gehörende ASCII-Code gesendet. Der ASCII-Code ist bei Microcomputern Standard zur Codierung von Ziffern, Buchstaben und Zeichen. So kann der von der Tastatur gesendete Code gleich vom Computer weiterverwendet werden, was beim NDR-Computer auch geschieht.
Eine XT/PC-Tastatur aber sendet nicht den zu einem Zeichen gehörenden ASCII-Code, sondern einen zur Taste gehörenden Positionscode. Positionscode heißt, daß eine Taste nicht nach ihren Bedeutung, sondern nach ihrer Position codiert wird. Dabei wird einfach die Taste links oben (ESC) mit 1 bezeichnet und dann die Tasten fortlaufend durchnumeriert. Wird eine Taste losgelassen wird ihre Position plus 128 gesendet, also z.B. für (ESC) 129. Auch die Sondertasten wie SHIFT oder CONTROL werden genauso behandelt. Dies hat zur Folge das z.B. die Unterscheidung zwischen Groß- und Kleinbuchstaben vom Computer selber vorgenommen werden müssen. Obwohl sich dies vielleicht umständlich anhört, ergeben sich dadurch viel mehr Möglichkeiten die Tastatur, die auch noch vom Rechner selber gesteuert werden können. Die Möglichkeiten sind z.B. Funktionstastenbelegung, Alt-Ebene, Tastennumbelegungen usw.

KEY3 - Computer im Computer

Da beim NDR-Computer aber keine solche Tastaurverwaltung vorgesehen ist muß dies von der Tastaturkarte vorgenommen werden. Wird dies mit herkömmlichen Bauteilen gemacht, kann ein großer Teil der Möglichkeiten nicht genutzt werden. Daher wurde, um allen Komfort nutzen zu können, auf der KEY3 ein eigener Mikroprozessor mit eigenem Speicher und eigenem Programm installiert. Die KEY3 ist also ein kleiner Computer.
Dies klingt für eine 'einfache' Tastaturkarte etwas übertrieben, aber so kann eine XT/PC-Tastatur voll unterstüzt werden. So ist es möglich, auch eine ASCII-Tastatur weiter zu verwenden (natürlich dann ohne zusätzlichen Komfort).
Was ein richtiger Computer ist, hat auch sein eigenes Programm. So auch die KEY3. Mit diesem Programm werden die Umwandlungen von Positionscode in ASCII-Code vorgenommen, SHIFT oder andere Sondertasten erkannt, der Tastaturpuffer verwaltet, die Funktionstasten programmiert ... . Man sieht also sofort, daß das Programm ein sehr wichtiger Teil der KEY3 ist. Durch das Programm kann auch jeder (fast) Benutzer sich sämtliche Sonderwünsche, falls nicht eh schon vorhanden, erfüllen. Auch ist zu erwarten, daß durch Updates des Programmes die Möglichkeiten der KEY3 noch weiter vergrößert werden.

Hardwareteil - der harte Kern

Wie gesagt ist die KEY3 eigentlich ein kleiner Computer. Sie besitzt einen eigenen Prozessor, hat einen eigenen Speicherbereich, hat ihre eigenen Ein- und Ausgabeports und benötigt natürlich auch ihr eigenes Programm. Der Prozessor steuert mit Hilfe des Programms alle Vorgänge auf der Baugruppe. Und wie beim Hauptcomputer auch, sind, um Daten auszutauschen und Adressen anzugeben, hier natürlich auch ein Daten und Adressbus vorhanden (baugruppeninterne Busse). Daneben gibt es für die Ports noch Pufferbausteine, die über Steuerleitungen angesprochen werden. (Siehe Blockschaltbild)

Schieben und speichern - die Tastaturanschlüsse

Links oben im Blockschaltbild sind die Schnittstellen zu den Tastaturen als Pfeile dargestellt. Der obere Eingang dient zum Anschluß einer seriellen PC/XT-Tastatur. Die ankommenden seriellen Daten werden hier mit Hilfe eines Schieberegisters in parallele Daten umgewandelt und zwischengespeichert. Beim Eingang für die parallele ASCII-Tastatur entfällt natürlich die seriell-parallel Wandlung, aber die Daten werden genauso zwischengespeichert.
Die in den Zwischenspeichern liegenden Daten können nun vom Prozessor (CPU Z80) abgeholt und weiterverarbeitet werden. Um die Daten zu holen, wird vom Prozessor über die Dekodierung der entsprechende Zwischenspeicher angesprochen. Der angesprochene Zwischenspeicher legt nun seine Daten auf den Daten

bus und werden dort vom Z80 abgeholt.

Lang- und Kurzzeitgedächtnis - Eprom und RAM

Damit der Prozessor überhaupt weiß, daß er Daten holen soll, benötigt er ein Programm. Dieses Programm steht im Eprom. Das Programm ist auch zuständig für die Weiterverarbeitung der eingelesenen Daten. Die Daten vom seriellen Eingang werden zuerst vom Positionscode in den ASCII-Code umgewandelt und dabei Sondertasten erkannt und gesondert behandelt (z.B. SHIFT,ALT oder auch die Funktionstasten usw.). Die nun erhaltenen Daten werden im RAM zwischen gespeichert. Bei den Daten vom seriellen Port ist eine Umwandlung oder gesonderte Interpretation nicht notwendig. Sie werden unverändert zwischengespeichert.

Ab an den Bus - der letzte Akt der Daten

Die im RAM zwischengespeicherten Daten werden vom Z80 nacheinander immer dann an die Pufferung zum Bus weitergegeben, wenn der Hauptprozessor auf den Dilschalter zugreift. Mit diesem Zugriff bestätigt der Hauptprozessor, daß er die letzten Daten übernommen hat und bereit ist für neue Daten. Diese Art der "Mitteilung" ist auch bei den alten Tastaturbaugruppen KEY1 und KEY2 so vorgesehen und wurde, um kompatibel zu sein, übernommen. Die Pufferung vom Bus ist notwendig, daß der Hauptprozessor mit dem KEYProzessor kommunizieren kann. Die zweite Dekodierung ist für den Hauptprozessor, damit er über den BUS auf die Pufferungsbausteine zugreifen kann und so Daten holen oder abliefern kann.

Klein aber fein - die Maus

Wie bei der Hcopy/Mausbaugruppe kann auch an die KEY3 eine Atari-Maus angeschlossen werden. Diese preiswerte Maus hat sich inzwischen zur NDR Standardmaus gemausert und wird von fast allen Programmen unterstützt. Eine einfache Maus oder ein einfacher Trackball besitzt 4 TTL-Ausgänge, entsprechend den vier möglichen Bewegungsrichtungen (rechts, links, auf und ab). Bei einer Aufwärtsbewegung der Maus erscheinen Rechtecksignale an dem Auf-Ausgange. Ebenso ist es beiden anderen Richtungen. Die Zahl der ausgesandten Impulse wächst proportional mit dem zurückgelegten Weg. Die Zahl der Impulse wird durch vier Zähler auf der Baugruppe gezählt und vom Prozessor des NDR-Computer abgefragt werden.

Bewegt sich nun die Maus nicht rein waage- oder senkrecht, so kann aus der Zahl der empfangenen Impulse in X- und Y-Richtung die Bewegungsrichtung ermittelt werden und somit eine Positionsbestimmung erfolgen.

CPU8088 - KEY3 - zwei Eulen in Athen

Da an der CPU8088 sowieso ein Anschluß für eine PC/XT-Tastatur vorgesehen ist, ist der Einbau der KEY3-Baugruppe in ein 8088-System überflüssig. Nichtsdestotrotz - funktionieren tuts trotzdem, aber halt etwas umständlich.

KEY3 - Tastatur - wer paßt zu wem

Grundsätzlich kann an die KEY3 jede PC- und jede XT-kompatible Tastatur angeschlossen werden. Bei AT- oder DIN 102-Tastaturen ist darauf zu achten, daß die Tastatur auf XT-Betrieb umgeschaltet werden kann. Dies ist zum Beispiel bei der MF-Tastatur der Fall. Auch der Anschluß einer normale ASCII-Tastatur ist möglich.

Null Bock auf no Future - was bringt die Zukunft

Wie schon gesagt ist das Herz der Baugruppe das Programm. Hier wird es in Zukunft, wie bei jedem Programm, weitere Verbesserungen geben. So ist z.B. die Unterstützung der Smart-Watch mit Sicherheit in der nächsten Programmversion enthalten. Die neuen Programmversionen fallen unter die Update-Bedingungen.

Ein Eprom raus, ein anderes Eprom rein, schon ist eine neue Stufe an Tastaturkomfort erreicht, und zwar ganz ohne löten.

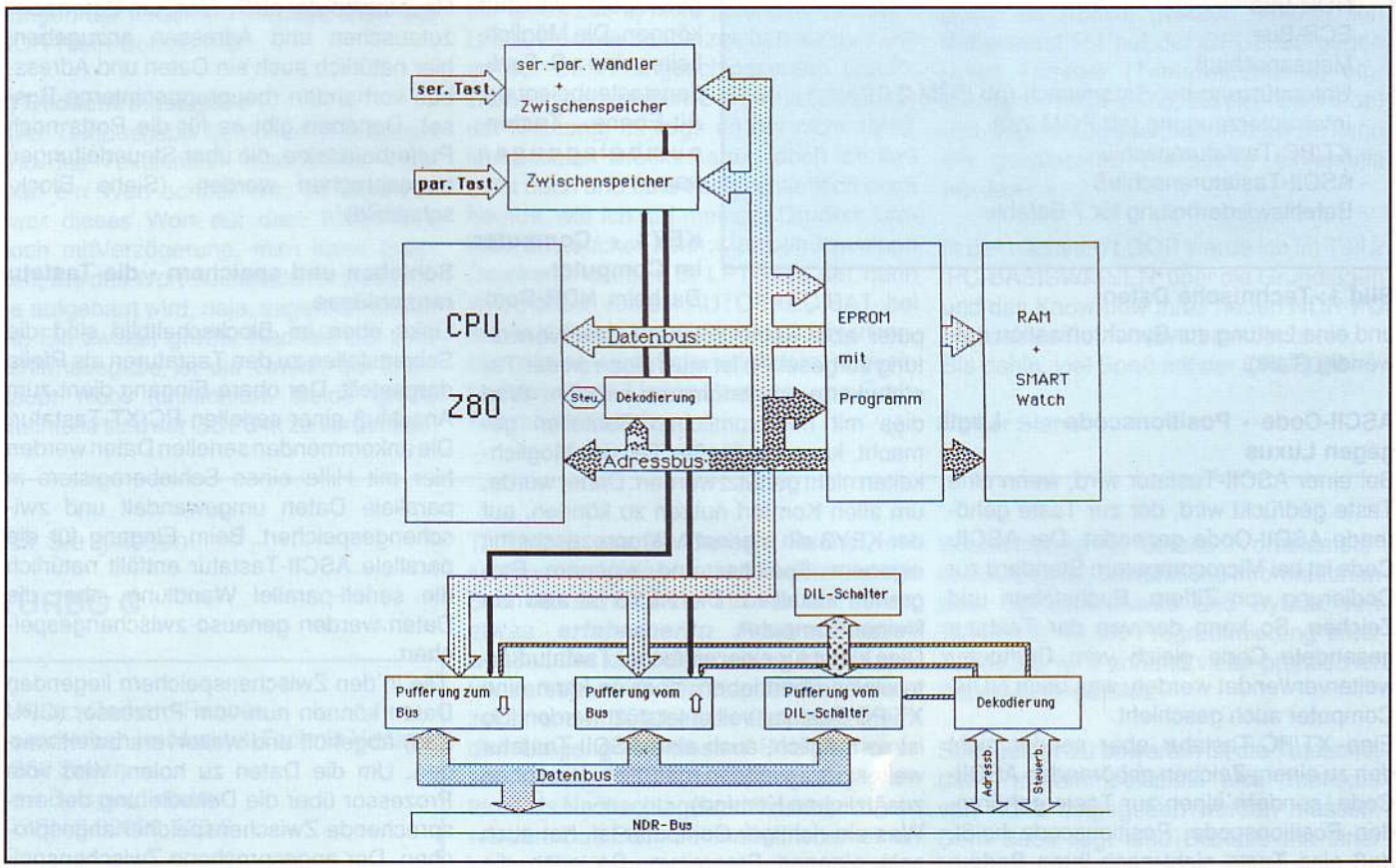


Bild 2: Blockschaltung ohne Mausteil

Für Sie gelesen

Erfolgreiches Programmieren von 68000er-Systemen in Assembler und C.

Autor: Günther Haarmann
stabiler Ringbuchordner,
Format DIN A4,
Grundwerk ca.550 Seiten
Bestellnr: 3400
Preis 92,- DM
Verlag und Vertrieb
Interest-Verlag
Industriestr. 21
8901 Kissing

Dieses Buch eignet sich hervorragend für 68000ér Benutzer, die autodidaktisch den Einstieg in die Programmierung dieser Prozessoren erreichen möchten. Der Themenkreis besitzt ein großes Spektrum. Aufbau und Handhabung, Software-Engineering, Tool's und Utilities sind nur wenige Stichpunkte, die dieses Lehrbuch behandelt.

Die Beispielprogramme sind jeweils in 'C' und in Assembler ausgeführt. Auch das Betriebsystem OS/9 wird behandelt. Hierbei wird der Aufbau und die Handhabung erklärt, bzw. spezielle Dateioperationen angesprochen.
Zu dem Grundwerk erscheinen regelmäßig alle 2-3 Monate ca.120 Seiten als Erweiterung.

Kleinanzeigen

(Fast) zu verschenken: Für den NDR-Computer habe ich noch einige Baugruppen/Zubehör (fast) zu verschenken.
T. Krähmer Tel. 0711/883587

Verkaufe wg. Sys.wechsel: TRAFO2, POW5V, BUS3, CPU Z80, ROA 64 m. 16KB, EHEX2, HEXIOH, IOE, HEXIO, CAS+Rec., KEY2, GDP64, EGRUND2+Lst., 7 x 3Disk, LOOP2-21. Auch einzeln. Preis: VB.
V. Milbrandt, Wurmlinger Str. 38.

UPGADES von bewährten Programmen für 68K-CPU's: BIO_RHYTMUS (mit Datums- und Terminfunktion) sowie HARD-COPY-STEUERUNG (für die HCOPY- und GDP64HS-Karten mit bis zu vierfacher Dehnung) jetzt direkt vom Programmierer lieferbar.
Info anfordern bei EMZET-Soft, Manfred Zehner, Schwenckestraße 2, 2000 Hamburg 20.

Verkaufe: 1 Gehäuse MC-Comp. 100,--; 1 NE1 130,--;1 POW5 20,--; 1 GDP64k 140,-; 1 GDP64 GES-Norm 260,-; 1 IOE 25,-, 1 ROA 64 m. Speicher-6264-15 80,--; 7 RAM8464-12 a 10,--; 1 Eprom SPS f. SBC3 50,--; 1 Eprom EGRUND2 f. SBC3 20,--; 1 Eprom EFLOMON 3.2 mit Listing 30,--; 1 -Eprom ZEAT mit Diskette 130,--; Christiani Kurs Mikroelektronik 90,--; Christiani Kurs f. ZEAT 150,--;
Würth Edgar, Geranienstr. 11, 7022 Leinfelden, Tel.: 0711/752839

Verkaufe NDR-Computer VHB 1000,- DM, lauffähig im SB-Gehäuse mit Monitor 2 x BUS1 + BUSE, POW5V, KEY, TAST1, CAS, IOE, GDP64K, 2 x ROA64, SBC2, CPUZ80, CPU68K, 3A-Trafo, HF-Modulator, BAS-Monitor, EGrund2, Grundpr. V.4,3, PASCAL, BASIC, GOSI, ESKOP, 8 x R8 (64KByte) und div. Literatur.
Kai-Uwe Sülflohn, Theodor-Heuss-Str. 7b, 2400 Lübeck 1,
Tel.: (0451) 51570 (ab 18 Uhr).

Ein Tip für 68000 - 12 Anwenderzur Geschwindigkeitssteigerung ihrer DRAM-Karten: 1. IC9 (74LS74) aus der Fassung nehmen 2. Karte umdrehen und von Pin 5 des ICs eine Verbindung nach PHI auf dem BUS schaffen 3. Pin 5 von IC9 hochbiegen und das IC wieder einsetzen (Pin 5 darf keine Verbindung mehr mit der Fassung haben!). Nach dieser Änderung laufen bei mir 4 DRAM-Karten mit je 8 41256-15 ICs bestückte Karten mit nur noch 1 Wait problemlos.
Olaf Petretti, Hackerstr. 27., 1000 Berlin 41

Software-Partner stellen sich vor

Persönliche Daten:

Rolf Lobreyer
Sonnenhalde 27
7740 Triberg

Hallo,
NDR-Klein-Computer Fans !

Heute möchte ich mich den Loop-Lesern in der Reihe "Loop - Steckbriefaktion" vorstellen.
Ich heiße Rolf Lobreyer, bin am 16. Mai 1963 in Triberg geboren und wohne zur Zeit in Karlsruhe, wo ich meiner Beschäftigung - dem Studieren - nachgehe (davon nachher noch mehr). Nun so fangen ja alle Lebensläufe an, dieser sollte aber ein sogenannter computertechnischer Lebenslauf sein, sodaß ich hier auf meine Tätigkeit in dieser Richtung eingehen werde.

Als ich 1963 das Licht der Welt erblickte, war noch nicht einmal der Mikroprozessor erfunden. So wuchs ich also unbekümmert auf und erst im Alter von 16 Jahren kam ich zum ersten mal mit einem Taschenrechner in Berührung, der gerade die Grundfunktionen +,-, /,* und eine %-Taste besaß. Dieses Wunderwerk der Technik hatten meine Eltern für's heimische Geschäft erworben.
Die Weichen zu meinem Informatik-Studium wurden dann durch den Erwerb eines Apple II Computer mit 48K-Byte Arbeitsspeicher gestellt.
Im 2. Semester wurde uns der Umgang mit einem 68000 Simulator und die Programmierung in 68000er Assemblersprache gelehrt. Der Simulator war natürlich viel zu langsam (basierend auf Apple II unter UCSD-Pascal), worauf ich auf Abhilfe sann. Da ich in den Semesterferien etwas verdient hatte und gerade der NDR-Klein-Computer als erstes 68000er System, welches wirklich erschwinglich war, in der Zeitschrift MC vorgestellt wurde, hatte ich mir einige Bausätze bestellt. Der Aufbau machte, auch ohne die Fernsehsendung gesehen zu haben, keine Schwierigkeiten.
Das System bestand damals aus einer 68008 CPU, 1ROA mit 8K-Byte, 1Kassetteninterface, 1 GDP64k und der Key-Karte, also eigentlich viel weniger als ich schon auf dem Apple II zu verfügung hatte. Aber es machte alles ungeheuer viel Spass! Es begann mit kleinen Assemblerprogrammen, die alle noch in dem 8K-Byte RAM oberhalb des Grundprogramms Platz haben mußten. Es zeigte sich jedoch bald, daß 8K-Byte RAM zu wenig waren. Da die Speicherbausteine noch sehr teuer waren (159.- pro 8K Baustein), griff ich auf die 128K-Karte vom Elektronikladen zurück, die mit den wesentlich billigeren dynamischen Speicherbausteinen 64K*1 zu bestücken waren (256K*1 waren damals noch nicht verfügbar !!)
Nun konnte im Speicher gewütet werden. So begann ich einen Disassembler für den 68008-Prozessor zu schreiben, mit dem ich schließlich beim Loop-Wettbewerb den 3. Platz errang. Das war vor nunmehr drei Jahren.

Um mir ein paar Mark zu verdienen, begann ich einen BASIC-Interpreter zu entwerfen, da ein solcher auf dem NDR-Klein-Computer fehlte. Es dauerte ca. 1 Jahr bis der Interpreter lauffähig war. Mittlerweile habe ich die nächste Generation fertiggestellt, was wieder ca. 1 Jahr Entwicklungs- und Testzeit erforderte. Die Quellen umfassen nun mehr als 20000 (in Worten zwanzigtausend) Programmzeilen Assembler.
Zur Zeit arbeite ich an der Entwicklung eines dazu passenden Übersetzers, der nun nicht mehr in Assembler, sondern in einer höheren Programmiersprach ("C") implementiert wird. Bis zur Fertigstellung dürfte aber noch ein 1/2 Jahr vergehen, da ich noch studiere und die Programmierung nur nebenher betreiben kann, insofern mir die Prüfungen dafür Zeit lassen.
Abschliessend kann ich sagen, daß durch den NDR-Klein-Computer das Studium sehr viel interessanter und praxisorientierter gestaltet werden konnte, da ich die Erfahrungen von Vorlesungen in praktische Arbeiten einfließen lassen konnte.

Hier noch ein paar Tips an die Leser

Verwenden Sie eine höhere Programmiersprache da :
- das Programmieren nicht so fehler - trächtig ist
- die Programme auf andere Rechner leichter übertragbar sind
- die Programme leicht erweiterbar und wartbar sind
- die Programme lesbarer sind
- die Programme schneller erstellt werden können, obwohl:

die Programme meist etwas größer sind langsamer ablaufen.
Der letztgenannte Grund dürfte aber immer mehr in den Hintergrund treten, da immer schnellere Prozessoren auf den Markt kommen und der Hauptspeicher ebenfalls immer größer und schneller wird.
Nur bei besonders zeitkritischen Anwendungen ist Assembler angebracht !!!

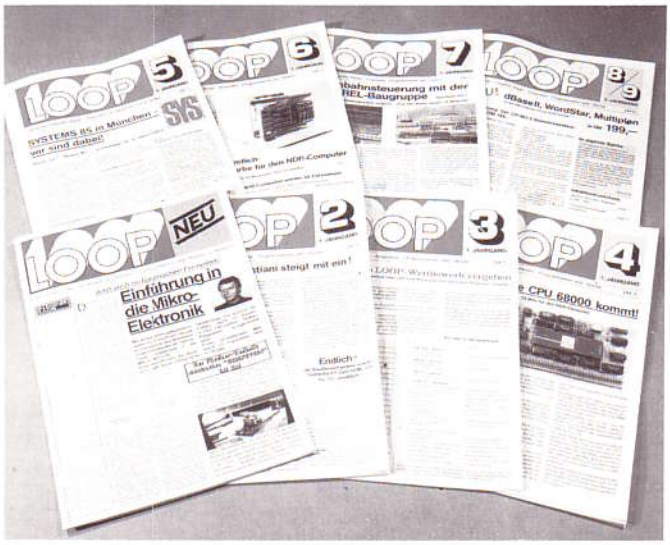
Leider sind höhere Programmiersprachen unter JADOS noch nicht in großer Vielfalt zu erhalten, was sich aber schnell ändern kann. (Solange verweise ich natürlich auf mein BASIC ..)

Fazit : Die gesamte technische Entwicklung der letzten Jahre habe ich mit meinem System gut überstanden. Mittlerweile kann der NDR-Klein-Computer bis zu einem Hochleistungs-68020- System ausgebaut werden, das in punkto Rechengeschwindigkeit wirklich nur durch SUN oder APOLLO - Workstations der 30.000 DM - Klasse übertroffen wird.
Leider kann die Leistungsfähigkeit der Prozessoren nicht in vollem Umfang ausgenutzt werden, da noch keine unterbrechungs-orientierte Ein /Ausgabe und keine Möglichkeit des direkten Speicherzugriffs (DMA) besteht.
Aber selbst diese schönen Dinge können nicht genutzt werden, wenn die Software nicht mitspielt. Es ist deshalb an der Zeit, die Software in verstärktem Maße zu fördern, damit dieses schöne modulare System weiterhin seinen Platz hat.
Anzustreben wäre ein benutzerfreundliches (mit graphischer Benutzeroberfläche + Maus) Mehrprozessorbetriebssystem, das möglichst standarisiert ist. Denn: Ein System ist nur so gut wie die Software, die auf ihm läuft.

Auch nach dem Kauf:

Die Computerzeitschrift *LOOP* ist die Brücke zum Kunden – Programme, Infos, Tips + Tricks!

Jahres-Abo DM 25,–, Probeexemplar kostenlos!



Umfassend informiert Sie unser Katalog.

Schutzgebühr: DM 10,– incl. MWSt.

Bitte bestellen Sie mit anhängender Postkarte!

BESTELLKARTE

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

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10 834	GES-Katalog	10,-
	10061	LOOP-Abo	25,-

Adresse (umseitig) nicht vergessen!

Datum

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

BESTELLKARTE

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

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10 834	GES-Katalog	10,-
	10061	LOOP-Abo	25,-

Adresse (umseitig) nicht vergessen!

Datum

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

Neue Produkte - Neue Preise

Best.-Nr.	Bezeichnung	Preis DM
11359	DRAW58	119,--
11360	DRAW38	119,--
11375	MULTIOB	348,--
11344	MULTIOEXF	248,--
11345	MULTIIOF	398,--
11347	MULTIOH+Disk	40,--
11346	MULTIOP	98,--

Best.-Nr.	Bezeichnung	Preis DM
11312	KEY3B	228,--
11314	KEY3P	49,--
11315	KEY3H	20,--
11313	KEY3F	298,--
40968	EGALAPTOP	8900,--
11371	PROFILOGH	30,--
11352	PROFILOG IBM38	369,--
11353	PROFILOG IBM58	369,--
11362	Erweiterung vom LogSim zum PROFILOG /38	150,--
11361	Erweiterung vom LogSim zum PROFILOG /58	150,--

Bitte Porto nicht vergessen

ANTWORT

GRAF computer

Graf Elektronik Systeme GmbH

Postfach 1610

8960 Kempten

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vorkasse ☐ Bankeinzug

Bankeinzug: Hiermit ermächtige ich die Firma GES GmbH, den Rechnungsbetrag für die auf dieser Karte angegebenen Bestellungen von meinem Konto:

BLZ _____ Konto-Nr. _____

Bank: _____

abzubuchen. Falls mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Kreditinstitutes keine Verpflichtung zur Einlösung.

Datum _____ Unterschrift _____

Bitte Porto nicht vergessen

ANTWORT

GRAF computer

Graf Elektronik Systeme GmbH

Postfach 1610

8960 Kempten

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vorkasse ☐ Bankeinzug

Bankeinzug: Hiermit ermächtige ich die Firma GES GmbH, den Rechnungsbetrag für die auf dieser Karte angegebenen Bestellungen von meinem Konto:

BLZ _____ Konto-Nr. _____

Bank: _____

abzubuchen. Falls mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Kreditinstitutes keine Verpflichtung zur Einlösung.

Datum _____ Unterschrift _____