

Uwe Kötter
Frankenstraße 25
5880 LUDENSCHIED
Tel. (0 20 51) 2 61 92



1. JAHRGANG

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

DM 3,-

SYSTEMS 85 in München – wir sind dabei!



Halle 22 – Stand B7 – Commodore ist neben uns!

Vom 28. 10. bis 1. 11. 85 trifft sich in München alles, was in der Computerbranche Rang und Namen hat – natürlich auch GES mit dem NDR-Computer und dem mc-CP/M-Computer.

Obwohl wir uns vorgenommen haben, Produkte erst dann anzukündigen, sobald sie auch tatsächlich lieferbar sind, hier eine Ankündigung, was wir auf der SYSTEMS alles zeigen. Wir sind natürlich selbst daran interessiert, unsere Produkte so schnell wie möglich auf den Markt zu bringen, damit die Entwicklungskosten wieder erlöst werden und unsere Anwender auf dem neuesten Stand der Technik sind. Wir wollen aber nur ausgereifte und funktionierende Geräte verkaufen – deshalb die vorsichtigen Ankündigungen.

32 Bit für den NDR-Computer Prototyp der CPU68020 auf der SYSTEMS

Der NDR-Computer ist das erste Bauplatz-System, das mit einer echten 32-Bit-CPU, dem 68020 ausgerüstet werden kann. Mit auf der Baugruppe ist ein (optionaler) Gleitkomma-Arithmetik-Prozessor, der ca. 0,5 Millionen Gleitkomma-Operationen pro Sekunde ausführen kann. Damit ist der NDR-Klein-Computer nur noch um den Faktor 1000 langsamer als die CRAY, der schnellste Rechner der Welt.

Die 32 bit CPU paßt natürlich in das modulare Konzept des NDR-Computers – es werden zwei BUS-Baugruppen nebeneinander gestellt; in vier Quadranten werden die Speicherbaugruppen

angeordnet, die Peripherie bleibt.

In München zeigen wir erst den Laboraufbau – lieferbar wird die CPU68020 aber schon im ersten Quartal 1986.

Farbgraphik mit der COL256

Neben dem bisherigen Lieferprogramm des NDR-Computers und des mc-CP/M-Computers wird Farbgraphik der Schwerpunkt unseres Messestandes sein.

Wir zeigen COL256, die wir auch schon in Hannover präsentiert haben. Sie paßt natürlich an alle CPUs des NDR-Computers und kann 256 mal 256 Bildpunkte mit 256 Farben, pro Bildpunkt getrennt wählbar, darstellen.

COL256 ist eine „Memory-Mapped“-Baugruppe, d.h. jedem Bildpunkt entspricht ein Byte im Speicher; der Inhalt dieses Bytes stellt die Farbe des Punktes dar. Damit ist das Ansprechen der Baugruppe sehr leicht möglich.

Da bei 256 x 256 Bildpunkten 64KByte Speicher benötigt werden und die Baugruppe auch mit der CPUZ80 funktionieren soll, werden jeweils 16 KByte eingebündelt.

COL256 wird noch dieses Jahr verfügbar sein; die Software-Unterstützung reicht von einfachen Beispielen bis zu Bibliotheken in C. Die Baugruppe kann auch mit einem Schwarz-Weiß-Monitor betrieben werden und erzeugt dann 64 Graustufen.

LOOP 6 wird als Schwerpunkt diese Karte haben.

ACRT –

Graphik für hohe Anforderungen

Für Anwendungen, die extreme Geschwindigkeit, Window-Technik, Hardware-Scroll und Zoom benötigen, dient die Baugruppe ACRT. Sie ist mit dem neuen Hitachi ACRTC-Controller ausgerüstet. Die Auflösung beträgt 640 x 240 Punkte ohne und 640 x 480 Punkte mit Zeilensprung, der Speicher auf der Baugruppe 1 MByte für ca. 13 Seiten in Farbe.

Drei unabhängige Bildschirmbereiche können definiert werden; über diese kann ein Window (Fenster) überlappend oder durchscheinend gelegt werden. Hardware-Scroll für jeden Bereich getrennt. Zoom von 1 – 16 fach.

Das System belegt zwei Karten, die in 6-fach Multilayer-Technik ausgeführt sind. Es wird deshalb (ab 1986) nur als fertiges System geliefert.

Hardcopy-Maus-Fadenkreuz

Lang erwartet wurde die Hardcopy-Baugruppe, die es ermöglicht, den Bildschirminhalt, also auch Graphik, auf einem Nadeldrucker (z.B. EPSON) auszugeben. Wir zeigen die Hardcopy auf der SYSTEMS – die Baugruppe wird ebenfalls noch dieses Jahr lieferbar sein.

Neben den erwähnten Neuerungen warten noch einige Überraschungen auf Sie! Besuchen Sie uns: Vom 28. 10. bis 1. 11. 1985, Halle 22, Stand B7. Rolf-Dieter Klein und Gerd Graf werden am Stand sein und freuen sich über Ihre Anregungen, Wünsche und Kritik.

Für den Anfänger: Ports und die IOE-Baugruppe

von Gerd Graf

Computer machen erst dann richtig Spaß, falls sie irgendein Gerät steuern oder irgendeine Information, die von außen kommt, verarbeiten können.

Ein Computer besteht immer aus drei grundlegenden Blöcken:

- Die Zentraleinheit, CPU, z.B. der Z80
- Die Speichereinheit, z.B. eine ROA64-Baugruppe oder die Speicher auf der ABC2
- Die Ein-Ausgabereinheit, z.B. IOE, GDP oder KEY.

Die CPU selbst weiß nicht, welche dieser Einheiten wo angeschlossen sind; es ist die Aufgabe des Programmierers, dies der CPU mitzuteilen. Dazu dienen die Befehle

IN adresse und
OUT adresse.

Die Verschlüsselung der Befehle lautet:

IN: DB xxH
OUT: D3 xxH,

wobei xxH für die gewählte Adresse zwischen 00H und FFH steht.

Auch mit minimalen Englisch-Kenntnissen wird klar, daß dies der Eingabe (IN)- und Ausgabe(OUT)-Befehl ist.

Die „adresse“ ist eine Zahl, die zwischen 0 und 255 liegen kann. Die Adresse wird jedoch meist hexadezimal (z.B. 30H, bedeutet $3 \times 16 + 0 = 48$ dezimal) angegeben. Was hat's nun mit dieser Zahl auf sich?

Dazu ein Ausflug in die Hardware: Nachdem die CPU z.B. den Befehl
OUT 30H

aus dem Speicher eingelesen hat, wird dieser Befehl decodiert. Er bedeutet für die CPU: Gib den Inhalt des Akkumulators auf die Adresse 30H als Ausgabebe-

fehl aus. Falls man nun stoppt und die Signale, die die CPU ausgibt, anschaut, erkennt man folgendes:

Datenbus: Hier liegt der Inhalt des Akkus (Register A)

Adressenbus (niederwertigere Hälfte A0 bis A7) Hier liegt die Adresse, die beim Befehl angegeben wurde; bei unserem Beispiel:

```
A7 A6 A5 A4 A3 A2 A1 A0
0 0 1 1 0 0 0 0
---- 3 ---- 0 ----
= Hexadezimal 3 0
```

Steuerleitungen:

IORQ ist 0 Anforderung an eine Ein-Ausgabereinheit (Input/Output Unit) ...

WR ist 0 ... zum Schreiben (two write)

Die Leitung IORQ* geht an alle Ein-Ausgabereinheiten im System. Natürlich dürfen sich nun nicht alle Einheiten gleichzeitig angesprochen fühlen, sondern nur eine, z.B. eine IOE, an die 8 LEDs angeschlossen sind.

Unterschieden wird dies durch die Adresse, die ja am Adressenbus liegt. Man baut nun auf jede Ein-Ausgabekarte einen binär-Vergleicher, der die anliegenden Signale am Adressbus mit der vorher vom Anwender eingestellten Adresse vergleicht. Bekannte Vergleicherbau- steine sind der 74LS85 (kann 4 bit vergleichen) und der 74LS688 (kann 8 bit vergleichen).

Unsere IOE-Karte belegt nun vier „Kanäle“, auch „PORTS“ genannt. Sie könnten mit vier der erwähnten Verglei-

cher auf vier unabhängige Adressen gelegt werden – dies bedeutet einen Bauteileaufwand. Bei der IOE wurde folgender Trick gewählt (vergleiche Schaltbild IOE, Handbuch).

Nur die vier Bits A4 bis A7 werden mit dem Vergleicher J7 verglichen. Preisfrage: Wieviel verschiedene IOEs könnte man in unserem System stecken, falls (zur Vereinfachung) keine weiteren Ein-Ausgabe-Baugruppen verwendet werden?

(Nun nachdenken): Richtig: 16 Karten, denn mit 4 Signalleitungen können 2 hoch 4 = 16 Zustände unterschieden werden. Über die Jumper auf der IOE (JMP1) wird dann die Adresse der Karte eingestellt. Beispiele

Keine Jumper Adresse F0H bis F8H
Jumper auf A4 Also ist A4 = 0, A5 bis A7 = 1
Adresse E0 bis EFH

Schreiben Sie sich die Bitkombinationen mal auf, und zwar so:

```
A7 A6 A5 A4 A3 A2 A1 A0
1 1 1 0 x x x x
```

„x“ bedeutet hier, daß bei *allen* Adressen die Baugruppe selektiert wird.

In der nächsten LOOP werden wir dann erklären, wie das mit den Signalen A3 bis A0 funktioniert.

Bis dahin: Aufgaben:

Studieren Sie zur Vorbereitung mal den Schaltplan der IOE, besonders das Gatter 74LS139.

Versuchen Sie, Mängel der IOE aufzuspüren. TIP: Gehen Adressen für Ein-Ausgabekanäle „verloren“?

Was passiert, falls durch einen Programmfehler eine Ausgabe auf eine Adresse gemacht wird, die gar nicht decodiert wird?

Wohin werden wohl die Daten beim IN-Befehl gelangen?

Wie könnte man eine IOE-Karte einfach prüfen?

Fortsetzung folgt.

Akku-Betrieb mit der SBC2 von Frank Popp 5758 Fröndenberg

Eine feine Sache, wenn nach dem Ausschalten des NDR-Computers Programme und Daten im Speicher erhalten bleiben. Dabei geht es ganz einfach, der Bauteileaufwand ist minimal, nur einige Leiterbahnen müssen aufgetrennt werden.

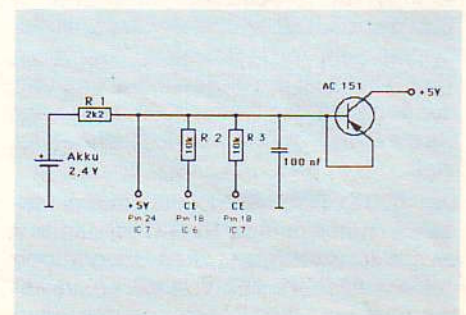
BILD 1 zeigt die Schaltung. Die Stromversorgung der RAMs 6116 (CMOS-Typen verwenden, diese ziehen nur 2 uA!) wird aufgetrennt und die RAMs werden von der gezeigten Akku-Schaltung versorgt.

Im Fall der Akku-Versorgung wird über R2 und R3 der CE-Eingang hochgezogen, damit keine Fehlzugriffe erfolgen können. R1 lädt den Akku, sobald die Versorgungsspannung aus dem Netzteil wieder hergestellt wird.

Hinweis: In der Baugruppe SBC3, die 1986 vorgestellt wird, ist eine ähnliche Schaltung bereits vorhanden.

Noch eine Anregung für Sparsame: Wer seiner SBC2 mehr RAM-Power geben

will, kann einfach weitere RAMs huckepack löten. PIN 18 (CE) wird jeweils abgewinkelt und mit IC5 verbunden, das noch über genügend freie Pins verfügt, um weitere RAMs zu selektieren.



Druckeransteuerung mit dem Z80-Grundprogramm

Wolfgang Reimann

:Das RDK-Grundprogramm von Rolf-Dieter Klein stellt mit seinen Eingabe-, Kontroll- und Testmöglichkeiten eine gute Lösung dar, Anfängern und auch Fortgeschrittenen einen Einstieg in die Assemblerprogrammierung zu bieten.
:Leider hat das Grundprogramm auch einen Nachteil: Es unterstützt keine Drucker-:ausgabe. Diese ist jedoch gerade bei etwas längeren Maschinenprogrammen sehr :nützlich, um den Überblick zu behalten.

:Die Hardware für eine Druckerschnittstelle ist beim NDR-Klein-Computer durchaus :vorgesehen und wird auch vom NDR-Basic und dem CP/M Betriebssystem unterstützt. :Mit einer IDE-Karte und einer kleinen Zusatzplatine (oder ein wenig Bastelei) :wird eine Parallelschnittstelle nach der sogenannten "Centronics Norm" realisiert.

:Auf diese Norm soll hier nur eingegangen werden, soweit es das Verständnis für :die Software erfordert.
:Parallel bedeutet, daß die Daten (D0...D7) gleichzeitig über getrennte Leitungen :übertragen werden, im Gegensatz zu einer seriellen Übertragung wie beim Cassette- :recorder.
:Zusätzlich zu den Datenleitungen benötigt man noch mindestens zwei Steuerleitungen :für das STROBE- und BUSY-Signal.
:Diese beiden Signale steuern die Datenübergabe zwischen Computer und Drucker.
:Ist die BUSY-Leitung log. high, so kann der Drucker keine Daten annehmen. Der :Drucker meldet so dem Computer, daß er beschäftigt ist.
:Geht BUSY auf log. low, so gibt der Computer ein neues Datenwort aus und teilt :dieses durch einen kurzen Low-Impuls auf der STROBE-Leitung dem Drucker mit.

:Ein Programm für die Bedienung der Druckerschnittstelle muß nun die BUSY-Leitung :abfragen, ein Datenwort mit einem ASCII-Zeichen ausgeben und den STROBE-Impuls :erzeugen.

:Die folgende Routine gibt nun ein Zeichen aus dem Akku an den Druckerport.
:Vor dem Aufruf des Programmes über CALL xxxh ist sicherzustellen, daß sich ein :gültiges ASCII-Zeichen oder ein Steuercode im Akku befindet.
:zu beachten ist weiterhin, daß die gesamte Steuerung des Druckers, wie :
: - Wagenrücklauf (Carriage Return) (00h)
: - Zeilenvorschub (Line Feed) (0Ah)
: - Zeilenvorschub bis Blättende (Form Feed) (0Ch)
: von dem aufrufenden Programm vorgenommen werden muß.
:Da eine solche Steuerung für einen gut formatierten Druck leicht aufwendig wird, :gibt es später noch ein nützliches Beispielprogramm.

:Die Beschaltung der IDE-Karte ist in ac-Sonderheft 'Schaltungen & Unterlagen' von :Franz-Verlag auf Seite 74 dokumentiert.
: Centronics Druckerschnittstelle für NDR-Klein-Computer mit RDK-Grundprogramm
: Copyright (C) Wolfgang Reimann Ulmenallee 7 3160 Lehrte
: Stand 29.07.85 Version 1.1

```

ORG 8800H          ; Das Programm beginnt hier auf der Adresse
                  ; 8800h, es ist jedoch nach dem Ändern sämt-
                  ; licher Adressen auch in anderen Speicherbereichen
                  ; lauffähig.

Centronics:
0900 C5          PUSH BC          ; Registerinhalte auf den Stack retten
0901 D5          PUSH DE          ;
0902 E5          PUSH HL          ;
0903 0E 49       LD C,049H        ; Register C wird mit der Portadresse der
                  ; Steuerleitungen geladen

Busy:             ; Anfang der Schleife welche die BUSY-Leitung
                  ; abfragt
0905 ED 40       IN B,(C)         ; einlesen des BUSY-Signals in Register B
0907 CB 40       BIT 0,B          ; ist Bit 0 in Register B = 0 ?
0909 20 FA       JR NZ,Busy       ; wenn nein, zurück an den Schleifenanfang
090B D3 48       OUT (048H),A     ; wenn ja, Datenwort von Akku an Port ausgeben
090D 06 FF       LD B,0FFH       ; jetzt STROBE-Leitung kurz auf low Pegel legen
090F ED 41       OUT (C),B        ;
0911 06 00       LD B,00         ;
0913 ED 41       OUT (C),B        ;
0915 06 FF       LD B,0FFH       ;
0917 ED 41       OUT (C),B        ;

```

```

0B19 E1          POP HL           ; die ursprünglichen Registerinhalte zurückladen
0B1A D1          POP DE           ;
0B1B C1          POP BC           ;
0B1C C9          RET              ; Unterprogramm beendet

```

:Das nun folgende Unterprogramm gibt die beiden Steuerzeichen Carriage Return und Line Feed an den Drucker aus. Damit wird der Druckkopf an den Zeilenanfang zurückgeführt und das Papier eine Zeile weiter transportiert.
:Diese Routine nutzt das vorherige Programm zur Datenübertragung und ist daher nur :zusammen mit diesem lauffähig.

```

CRLF:
0B1D 3E 0D       LD A,0DH        ; Akku mit ASCII-Code für Carriage Return laden
0B1F CD 00 88    CALL Centronics ; Unterprogramm 'Centronics' aufrufen
0B22 3E 0A       LD A,0AH        ; Akku mit ASCII-Code für Line Feed laden
0B24 CD 00 88    CALL Centronics ; wieder Unterprogramm 'Centronics' aufrufen
0B27 C9          RET              ; und zurück springen zu dem aufrufenden Programm

```

:Zum Abschluß nun ein kleines Anwendungsbeispiel für die so geschaffene Drucker- :schnittstelle.

:Es soll der Textbuffer des RDK-Grundprogrammes an den Drucker ausgegeben werden. :Die Adresse des mit Zeilenbuffer beschriebenen Speicherbereiches erfährt man aus dem :aus dem ac-Sonderheft 'Z 80 Grundprogramm'. Der Zeilenbuffer liegt in dem Bereich :von 800H bis 819H und ist 81 Byte lang.
:Hier werden sämtliche Ausgaben des Grundprogrammes auf den Bildschirm zwischen- :gespeichert. Als Kennung für das Ende einer Ausgabezeile wird ein Byte mit 0AH :an das letzte Zeichen angehängt.
:Diese Eigenschaften werden von den folgenden Beispiel zur Druckerausgabe genutzt.

```

Print:
0B28 21 D9 80    LD HL,0800H     ; Registerpaar HL mit der Anfangsadresse des
                  ; Zeilenbuffers laden

Zeichen:
0B2B 7E          LD A,(HL)       ; den Akku mit dem Inhalt der durch HL adressierten
                  ; Speicherzelle laden
0B2C FE 00       CP 0           ; wenn der Inhalt = 0 war, ist das Textende
0B2E 2B 06       JR Z,Ende       ; erreicht, deshalb Sprung zum Ende
0B30 CD 00 88    CALL Centronics ; sonst das Zeichen zum Drucker übertragen
0B33 23          INC HL          ; die Adresse in HL um 1 erhöhen und
0B34 18 F5       JR Zeichen       ; das nächste Zeichen holen

Ende:
0B36 CD 1D 88    CALL CRLF        ; nun Carriage Return und Line Feed ausgeben
0B39 C9          RET              ; und zurück zum Grundprogramm springen

```

:Nach dem Eingeben der drei kleinen Programme mit dem Menue 'Ändern' des Grund- :programmes kann das Programm 'Print' mit Start 8820h gestartet werden.
:Es müßte dann, wenn die Hard- und Software fehlerfrei sind, die Startadresse 8820 :auf den Drucker ausgegeben werden, sonst ist der eingetippte Hexcode oder der :Druckeranschluß noch einmal zu überprüfen.

:Wie hier gezeigt wurde, läßt auch von Maschinenprogrammen aus die Druckerschnittstelle :recht einfach bedienen.
:Zum Verschieben der hier vorgestellten Beispiele auf andere Speicheradressen in :RAM ist es nur nötig die Eingangsadressen für Centronics,CRLF und Print umzurechnen. :sonst können die drei Programme an beliebiger Stelle in RAM eingegeben und gestartet :werden.

:In einer weiteren Folge soll dann ein Programm vorgestellt werden, daß einen :Hex-Dump (ähnlich wie beim Speicher ansehen) auf den Drucker ausgibt. Des weiteren :ist ein Disassembler für Z 80 Programme geplant. Die Ausgabe soll hier :wahlweise auf dem Bildschirm oder Drucker erfolgen.

Der in LOOP 3 von der Firma GES angebotene BASIC-Interpreter HEBAS soll hier einmal etwas näher beschrieben werden.

Der Interpreter läuft auf einem Standard-CP/M-Computer mit der Z-80 CPU, sowie deutschem Zeichensatz mit Umlauten, unterstützt hervorragend die Arbeit mit Diskettenlaufwerken unter CP/M und ersetzt das 8K-BASIC des Grundprogrammes.

Was jedoch die Arbeit mit diesem Interpreter erheblich erleichtert, sind einerseits die deutschen Fehlermeldungen und andererseits die schon vorhandenen zahlreichen sog. TOOL-KIT-Befehle wie AUTO, FIND, REPLACE usw., die bei anderen bekannten Rechnertypen erst zusätzlich geladen werden müssen oder mittels EPROM zur Verfügung gestellt werden.

HEBAS

Dr. Hans Hehl

Als Besonderheit muß auch die Tatsache erwähnt werden, daß die Ein- und Ausgabekanäle beliebig definierbar sind, d.h. daß z.B. das Inhaltsverzeichnis der Diskette oder auch Programmteile in eine Datei geschrieben werden können, die dann unter BASIC verändert werden kann.

Weiterhin gibt es ein ausführliches (ca. 65 Seiten), deutsches Handbuch dazu. Bis Sommer 1986 wird sogar ein komplettes Quell-Listing der Maschinensprache-Routinen mit Kommentaren und Erklärungen zur Verfügung stehen, so daß es für denjenigen, der die Z80-Ma-

schinensprache beherrscht, leicht möglich ist, seine eigenen BASIC-Befehle einzubauen. Dieses Listing gibt es dann auf Diskette, so daß mit einem handelsüblichen Z80-Assembler gearbeitet werden kann.

Nachfolgend sind einige BASIC-Befehle aufgeführt, die bei anderen Rechnern oft nicht direkt zur Verfügung stehen.

DATE\$	Datumseingabe und Verarbeitung
AUTO	automatische, beliebige Zeilennumerierung
BYTE	Zugriff auf einzelne Bytes von Dateien
COPY	Beliebiges Duplizieren von Programmteilen
DEF	Mehrzeilendefinition (auch Buchstaben)
DIR=	Ausgabe in Dateien, damit bearbeitbar
ERROR	Definition beliebiger Fehlermeldungen
ERL, ERR, RESUME XY	umfangreiche Fehlerbehandlung

EXCHANGE	schneller Variablen-tausch (sortieren)
FIND	Suchen von Zeichen bzw. Zeichenketten
HEXS bzw. &	Dezimal- und Sedezimal-Umwandlung
IF, THEN, ELSE, EXIT	besondere Schleifen-funktion
INP, OUT	A/E-Zugriff (IOE, Floppy-Controller)
INPUT LINE	Eingabe aller Zeichen (auch Komma!)
INSTR	Suchen von Zeichen in Strings
MAT READ, MAT WRITE	Matrix-Befehle
MIDS	zusätzlich: Austausch von Stringteilen
IKILL	Löschung der Speicher-reservierung (DIM)
LOADGO	Laden und Starten eines Programmes
LOG, LOG10	Natürlicher und dek. Logarithmus
LOOKUP	Suchbefehl für Dateien im Directory
LVAR	Ausgabe der Variablen-inhalte
MERGE	Verbinden von Programmen
MOD	Modulo-Funktion (Divisionsrest)
OPTION	Veränderung der E/A-Kanal-Parameter
PI und EE	(Eulersche Zahl e) Werte gespeichert
PRECISION	Arithmetik auf 11 Stellen
PRIVACY	Schutz durch Paßwort (alle Befehle)
REM	im Text auch Kleinbuch-staben erlaubt
RENUMBER	Neunummerierung von Programmzeilen
REPLACE	Ersetzen von Zeichen im Programm
RESTORE XY	Datazeiger auf beliebige DATA-Zeile
RND, RANDOMIZE	echte Zufallszahlen (aus Refresh)
SAVE, RESAVE	Speicherung im ASCII-Format möglich
TRACE	Ausgabe der bearbeiteten Zeilennummern
USING	Umfangreiche Formatie-rung (Zahlen, Texte)
VARPTR	Adressenangabe der Stringzeiger

Für diejenigen, die HEBAS, schon besitzen, folgt nun eine Rubrik TIPS und TRICKS bei HEBAS, die auch in den folgenden LOOP-Serien enthalten sein soll. Es werden Hinweise zum Einsatz gegeben und Leseranfragen beantwortet:

1) Bildschirmlöschung bei den Befehlen CLOSE=0 oder OPEN=0, "O"

Der Monitor FLOMON verwendet für die Bildschirmlöschung das ASCII-Zeichen 1Ah. Damit nun unter HEBAS dies wieder funktioniert, muß an Speicherstelle 3380h der vorhandene Wert 0Ch durch 1Ah ersetzt werden.

2) Der Interpreter belegt im Speicher den Platz von 100h bis 47FFh, wobei Basic-Programme den Interpreter ab 45D7h überschreiben und der Kaltstartteil des Interpreters somit nicht mehr zur Verfügung steht.

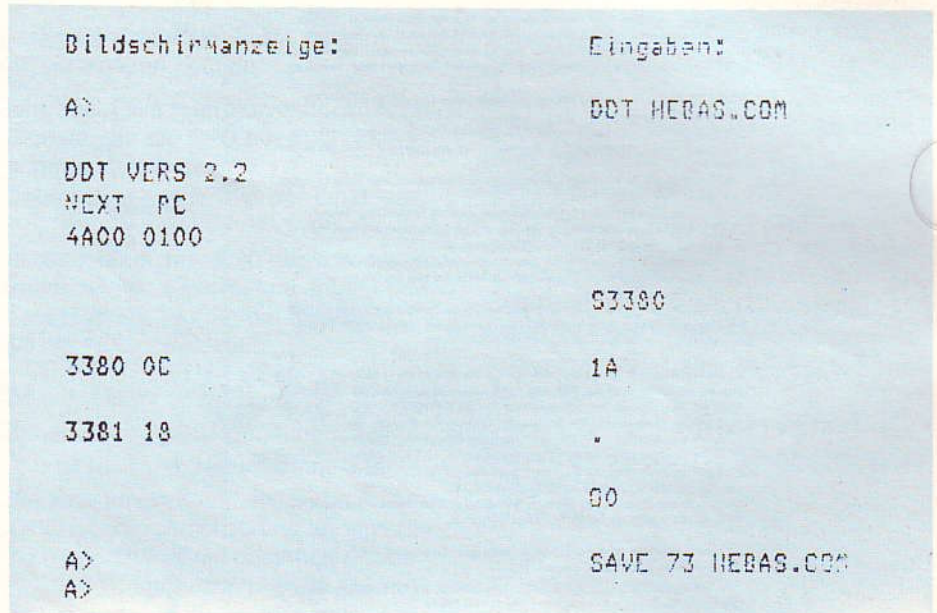
Das läßt sich ändern, wenn ab Adresse 46BAh die Bytes D8h und 45h (= Adresse 45D8h) durch eine neue Adresse ersetzt werden, z.B. 4800h. Es müssen dazu folgende Bytes ab Adresse 46BAh eingegeben werden: 00 und 48. Das Byte vor der neuen Adresse im Speicher muß unbedingt den Wert 00 besitzen, sonst erscheint beim Befehl RUN die Meldung

„Syntaxfehler“. In unserem Beispiel muß also die Speicherstelle 47FFh den Wert 0 enthalten.

Mit dem Befehl CALL &45EA kann man nun den Kaltstart wiederholen. Viel interessanter ist aber nun die Tatsache, daß so BASIC-Programme an beliebigen Stellen im Speicher liegen können. Mit dem CP/M-Dienstprogramm DDT.COM können die Änderungen am

Interpreter auf Diskette geschrieben werden. Die dazu notwendigen Eingaben sind in Bild 1 enthalten, wobei mit dem Befehl SAVE 73 HEBAS.COM ein geringfügig größerer Bereich auf Diskette zurückgeschrieben wird.

Die HEBAS-Diskette kostet nur DM 98,- und ist ab Lager lieferbar. Voraussetzung: NDR-Computer oder mc CP/M-Computer mit CP/M2.2.



TIPS UND TRICKS BEI HEBAS

Dr. Hans Hehl

Möchte man mit dem Grundprogramm (im EPROM auf Adresse 2000h auf der BANK-BOOT-Karte) das normalerweise ab Adresse 45D7h abgelegte BASIC-Programm analysieren, so geht das nicht. Zugeschaltet ist ja die BANK-BOOT-Kar-

te und nicht die dynamische Speicherkarte (64K-Byte). Erst ab Adresse 8000h kann man auf diese Karte zugreifen. Verschiebt man das Grundprogramm von der BANK-BOOT-Karte auf die dynamische Speicherkarte, so kann der Speicher ab 45D7h bearbeitet werden.

```

;Transfer Programm für Grundprogramm mit FLOMON-Monitor
;Grundprogramm von Bank 0, Adresse 2000 nach
;Adresse A000h dynamische Speicherkarte

```

```

0000      2380
          asg
          org 3380h

8800      start:
8800      21 2000 LD HL,2000H ;Adresse erstes Byte
8803      11 A000 LD DE,A000H ;Adresse Ziel
8806      01 2000 LD BC,2000H ;Programmlänge 8K-Byte
8809      ED 00  LDIR          ;verschieben
880B      3E 00  LD A,00H    ;Akku mit 00h (Bit 7 = 1
                        ;Bank 0 aus) laden
880D      D3 08  OUT (000H),A ;Akku Wert nach Port 08h
                        ;nach 2000h verschieben

880F      21 A000 LD HL,A000H ;Adresse erstes Byte
8812      11 2000 LD DE,2000H ;Adresse Ziel
8815      01 2000 LD BC,2000H ;8K-Byte Länge
8818      ED 00  LDIR          ;verschieben
881A      C3 2000 JP 2000H    ;Einsprung in Monitor

          end start

```


Leider ist der Bereich des BASIC-Interpreters von Adresse 2000h bis Adresse 3FFFh durch das Grundprogramm überschrieben.

Abhilfe schafft nur eine Neuassemblierung des Grundprogrammes für eine Adresse am Speicherende. Man kann z.B. den Bereich unter dem sog. BDOS nehmen, z.B. den Adressenbereich BC00h bis DC00h. Das Grundprogramm wird dazu auf Diskette abgespeichert und ein kleines Verschiebeprogramm von Diskette geladen wie eine COM-Datei.

Speichern und Laden von Daten mit dem 8K BASIC

Rolf-Dieter Klein

In dem kleinen BASIC gibt es nur Befehle zum Laden und Speichern von Daten. Das Gleiche gilt auch für GOSI.

Hier soll ein einfacher Weg gezeigt werden, auch Daten zu speichern und zu laden. Das Beispiel wurde für die Vollausbau-CPU-Version gewählt, geht aber auch bei der SBCII-Version, wenn man die Adressen entsprechend wählt. Zunächst einmal benötigt man ein Maschinenprogramm, das man in den RAM-Speicher legen muß. Bild 1 zeigt das Programm. Es speichert einen Block von genau 128 Bytes auf Kassette. Ebenfalls kann es einen solchen Block von Kassette laden. Dabei wird jeder Block mit einer Prüfsumme versehen, um Übertragungsfehler beim Laden erkennen zu können. Der Datenblock steht am Anfang des Programms, hier auf Adresse A000H. Die Startadresse muß so gewählt werden, daß sie nicht mit dem Programmteil von BASIC kollidiert. Danach folgt eine Speicherzelle STATUS, in die das Maschinenprogramm den Erfolg eines Ladevorgangs als 0, und den Mißerfolg als 1 ablegt.

Bei Adresse A082 beginnt das Unterprogramm SCHREIBEN und bei A084 das Unterprogramm LESEN.

Bei Aufruf von SCHREIBEN wird der Datenblock, also 128 Bytes auf Kassette gespeichert. Bei Aufruf von LESEN, wartet das Programm, bis es einen gültigen Datenblock findet und lädt diesen in den Speicher. Stimmt die Prüfsumme nicht, so wird der Wert 1 nach STATUS gespeichert. Die Unterprogramme RI und POO befinden sich im BASIC-Interpreter, und sind über eine Sprungtabelle erreichbar. Achtung, wer den Interpreter ab Adresse 0 laufen läßt, der muß die Adressen entsprechend ändern, hier wurde sie für einen Start ab Adresse 4000h festgelegt.

Interface MC3810 / NDR-KLEIN-Computer

von Rolf-Dieter Klein

Der BOSTON-Datenrecorder MC3810 ist ein für die CAS-Schnittstelle sehr gut geeigneter Kassettenrecorder. Das Schaltbild zeigt das nicht ganz einfache Anschlußkabel an die CAS und eine

mögliche Erweiterung, um den Motor des Recorders zu steuern. Der Recorder mit Kabel und Netzteil kann direkt von GES bezogen werden – Preis derzeit DM 129,-.

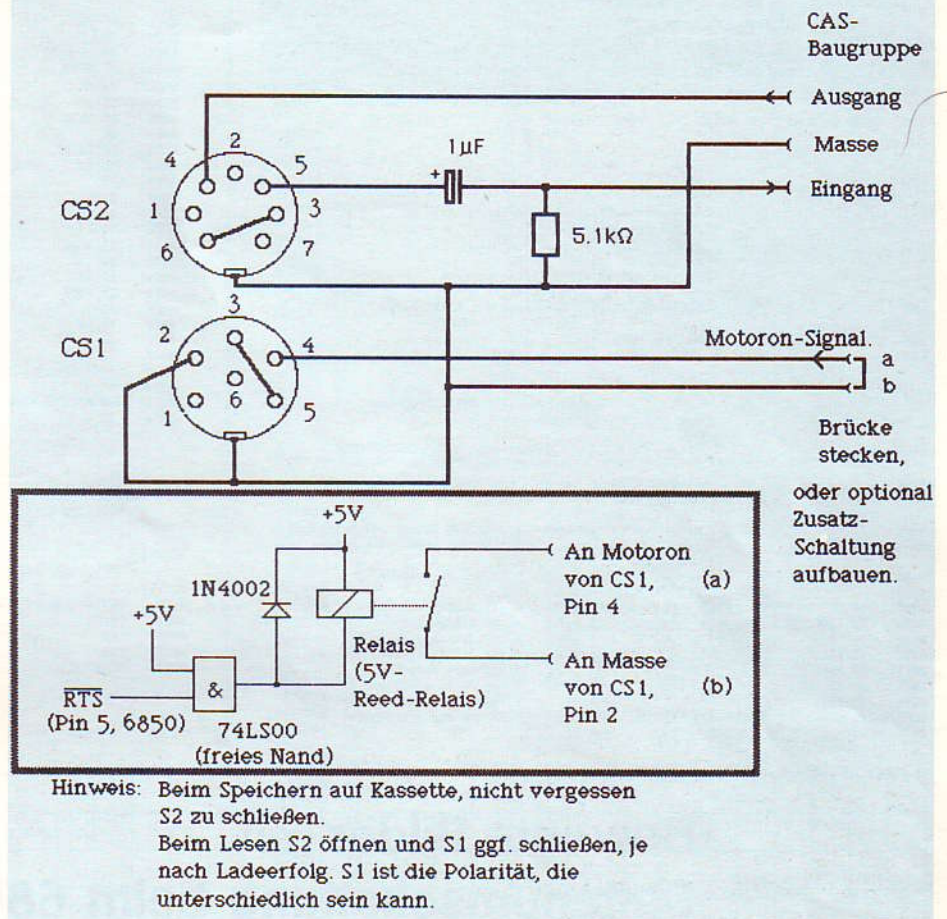


Bild 2 zeigt ein BASIC-Programm, das dieses Unterprogramm aufruft. Dazu muß das Maschinenprogramm entweder mit dem Grundprogramm in den Speicher eingetragen werden, oder als Datenliste im BASIC-Programm ergänzt werden, die dann mit POKE-Befehlen in den Speicher übertragen werden. Das BASIC-Programm trägt mit Hilfe von POKE-Befehlen die Datenwerte in den Buffer ein, und ruft dann mit einem CALL-Befehl das Maschinen-Unterprogramm auf. Danach können die Daten mit Hilfe eines weiteren CALL-Befehls wieder zurückgelesen werden. Dabei muß natürlich zwischendurch der Kassettenrecorder zurückgespult werden. Die Anweisungen, werden aber vom BASIC-Programm selbst ausgegeben.

Wenn man den Kassettenrecorder fernsteuern will, so kann man z.B. auf einer IOE-Baugruppe ein Relais unterbringen,

das dann den Kassettenmotor ein- oder ausschaltet. Dann ist auch ein automatischer Betrieb möglich, wenn man z.B. Meßdaten erfassen will.

Achtung, einer Erweiterungsmöglichkeit besteht auch auf der CAS-Baugruppe, wenn man den Ausgang RTS vom IC 6850 über einen freien Inverter (7400 nehmen), an ein kleines Relais (Ausgang des Inverters an Relais, Relais an +5 V, Schutzdiode nicht vergessen), schaltet und damit den Recorder steuert. Dann kann man mit den Befehlen des 6850 das RTS-Signal und somit den Motor steuern. Siehe auch Datenblatt 6850 von Motorola. Das Programm läßt sich natürlich erweitern, so könnte man die Blocklänge variabel gestalten o. ä. Der Phantasie sind keine Grenzen gesetzt. Wer eine interessante Lösung besitzt, sollte uns schreiben.

Rolf-Dieter Klein


```

10 REM Schreiben und Lesen von Daten
20 REM auf der CAS
30 DATEN = HEX("A000")
40 STATUS = HEX("A080")
50 SCHREIBEN = HEX("A0B2")
60 LESEN = HEX("A0B4")
70 REM daten in Buffer legen
80 FOR I=0 TO 127
90 POKE DATEN+I,1 : REM einfacher test
100 NEXT I
110 PRINT "Rekorder starten, auf SAVE schalten, ja eingeben";
120 INPUT JA
130 IF JA(">")="ja" THEN 110
140 CALL (SCHREIBEN)
150 PRINT "ok, stoppen; rueckspulen, auf LOAD schalten, ja eingeben";
155 INPUT JA
160 IF JA(">")="ja" THEN 150
170 CALL (LESEN)
180 S1 = PEEK (STATUS)
190 IF S1(">") THEN PRINT "Lese-Fehler": STOP
200 FOR I=0 TO 127
210 PRINT PEEK (DATEN+I)
220 NEXT I
230 PRINT "Ende"]

```

MACRO-80 3.43 27 Jul 81 PAGE 1

```

0000'      ;=00
          ;seg

          ;XXXXXXXXXXXXXXXXXXXXXXXXXXXX
          ; Unterprogramm fuer das
          ; Lesen und Schreiben von Daten
          ; z.B. mit dem BK - BASIC
          ;XXXXXXXXXXXXXXXXXXXXXXXXXXXX

4000      basic equ 4000h : Adresse des BASIC

400C      ri equ 0ch : basic : RI Unterprogramm
400F      poo equ 0fh : basic : PO0 Unterprogramm

          org 0a000h : hier freie Adresse waehlen
          : bei der kein BASIC-Programm
          : liegt.

A000      daten: defs 128 : Datenblock, mit PEEK und POKE
          : erreichbar.
A080      status: defs 1 : gibt an, ob Lesefehler vorhanden.
A081      defs 1 : Reserve

A082 13 02      schreib: jr schreib : Ansprung Datenblock Schreiben
A084 13 2F      lesen: jr lies : Ansprung Datenblock Lesen

A086 06 14      schreib: ld b,20 : 20 mal FF ausgeben
A088          lp1:
A088 0E FF      ld c,0ffh : als Starterkennung
A08A CD 400F    call poo : auf CAS ausgeben
A08D 10 F9      djnz lp1 : und danach
A08F 0E 55      ld c,055h : das Start Zeichen.
A091 CD 400F    call poo : und auch ausgeben
A094 DD 21 A000 ld ix,daten : nun daten ausgeben
A098 21 0000    ld hl,0 : aber mit Pruefsumme (=0)
A09B 06 80      ld b,120 : Anzahl der Datenbytes
A09D C5          lp2: push bc

```

```

A09E DD 4E 00      ld c,(ix+0) : Wert holen
A0A1 06 00      ld b,0
A0A3 09          add hl,bc : Pruefsumme berechnen
A0A4 CD 400F    call poo : und Daten ausgeben
A0A7 C1          pop bc
A0A8 DD 23      inc ix : naechster Datenwert
A0AA 10 F1      djnz lp2 : bis alle 128 ausgegeben
A0AC 4D          ld c,l : dann Pruefsumme ausgeben
A0AD CD 400F    call poo :
A0B0 4C          ld c,h
A0B1 CD 400F    call poo :
A0B4 C9          ret : Ende des Aufrufs.

A0B5 CD 400C    lies: call ri : warten auf Startkennung
A0B8 FE FF      cp 0ffh : sonst weiterlesen
A0BA 20 F9      jr nz,lies
A0BC DD 400C    lpa: call ri : dann solange Lesen, wie FF da.
A0BF FE FF      cp 0ffh :
A0C1 20 F9      jr z,lpa : danach muss Kennung folgen

```

MACRO-80 3.43 27 Jul 81 PAGE 1-1

```

A0C3 FE 55      cp 055h : wenn nicht, dann
A0C5 20 CC      jr nz,lies : neue Suche anfangen.
A0C7 DD 21 A000 ld l,daten : nun daten einlesen
A0C8 21 0000    ld hl,0 : aber Pruefsumme kontrollieren
A0CE 06 00      ld h,120 : Anzahl der Datenbytes
A0D0 C5          lpb: push bc
A0D1 CD 400C    call ri : Wert holen
A0D4 4F          ld c,a : und in Speicher
A0D5 DD 71 00    ld (ix+0),c : ablegen
A0D8 06 00      ld b,0 : Berechnung
A0DA 09          add hl,bc : der Pruefsumme
A0DD C1          pop bc
A0DD DD 23      inc ix : naechster Datenwert
A0DE 10 F0      djnz lpb : bis alle 120 ausgegeben
A0E0 CD 400C    call ri : nun Pruefsumme testen
A0E3 80          cp l : nicht gleich, dann Fehler
A0E4 20 08      jr nz,fehler :
A0E6 CD 400C    call ri : auch msb
A0E9 BC          cp h
A0EA 20 05      jr nz,fehler
A0EC AF          vor a
A0ED 32 A000    ld (status),a : und Status ok ablegen
A0F0 C9          ret : Ende des Aufrufs.
A0F1 3E 01      fehler: ld a,1 : Status=1 bedeutet Fehler
A0F3 32 A080    ld (status),a
A0F6 C9          ret

```

MACRO-80 3.43 27 Jul 81 PAGE 0

```

Macros:
Symbols:
4000 BASIC          A080 DATEN          A0F1 FEHLER
A084 LESEN         A0B5 LIES          A0D0 LPA
A09D LP2          A0C0 LPA          A0D0 LPA
400F PO0          400C RI           A0B6 SCHREIBEN
A0B2 SCHREIBEN   A080 STATUS

```

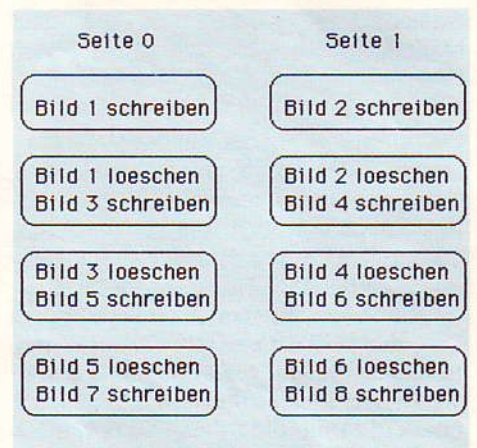
No Fatal error(s)

Bewegte Bilder mit Seitenumschaltung beim 68K-System

Einige haben schon mit dem FIGUR-Befehl aus dem Grundprogramm gearbeitet, um Figuren über den Bildschirm zu bewegen. Der Figur-Befehl arbeitet aber nur mit einer Bildseite und daher ergeben sich Flimmer-Effekte, wenn die Figuren zu groß werden. Man sieht dann an bestimmten Stellen des Bildschirms die Figur sogar überhaupt nicht mehr, da der Löschvorgang seine Zeit braucht. Abhilfe schafft dort nur der Einbau von Verzögerungsschleifen (z.B. SYNC-Aufruf), allerdings wird dadurch das Programm nicht gerade schneller. Der Figur-Befehl benötigt selbst auch einige Zeit, da er einige Umrechnungen der Vektoren durchführen muß. Schneller geht es mit den Kurzvektoren des Graphik-Prozessors, wenn man sich nicht scheut, sie zu kodieren. Ein Unterprogramm mit dem Namen CMDPRINT ist in der Lage, solche

Kurzvektoren direkt und schnell auszugeben. Dabei kann sogar Text mit verwendet werden. Um Flimmereffekte ganz zu vermeiden, ist es möglich, mehrere Bildschirmseiten zu verwenden. Bei der GDP64 sind ja insgesamt vier Bildseiten mit je 512 mal 256 Bildpunkten vorhanden, und schon mit zwei Bildseiten kommt man zu einer flimmerfreien Anzeige. Dabei wird wie folgt verfahren: Man schreibt das Bild in einer unsichtbaren Seite und zeigt dabei eine andere Seite an. Also z.B. es wird Seite 1 angezeigt, und Seite 0 geschrieben. Dann schaltet man um, zeigt also Seite 0 an und man kann nun in aller Ruhe Seite 1 mit neuen Graphiken beschreiben, ohne daß Schreib- und Löschvorgänge störend sichtbar werden.

Bild 1 zeigt ein Schema für das Vorgehen. Es wird zuerst Bild 1 in Seite 0 geschrie-



ben und Seite 1 angezeigt. Beim ersten Mal funktioniert es natürlich noch nicht, da Seite 1 noch nicht beschrieben war. Dann schaltet man um und zeigt Seite 0 an. Dabei wird das Bild sichtbar. Auf Seite

1 kann man nun Bild 2 schreiben. Dann wird umgeschaltet und Seite 1 angezeigt. Bild 2 wird jetzt sichtbar. Nun kann man das Bild 1 der Seite 0 löschen und dann Bild 3 auf Seite 0 schreiben. Dann wird

wieder umgeschaltet usw. Wenn man den Umschaltvorgang selbst noch mit dem Bildwechsel synchronisiert, also nur alle 20 Millisekunden umschaltet, so ist der Umschaltvorgang

selbst auch nicht sichtbar. Mit dem Unterprogramm SYNC ist ein solcher synchroner Wechsel möglich. Bild 2 zeigt ein Programmbeispiel. Eine Figur soll über den Bildschirm bewegt

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

0E9C00          * KLEINES BEISPIELPROGRAMM FUER DIE
0E9C00          * MEHRSEITENBENUTZUNG FUER BEWEGTE
0E9C00          * FIGUREN.
0E9C00          * 00000002          GESCHW EDU 2
0E9C00          START:
0E9C00          323C 0000      MOVE #0,D1      * x=0 IST STARTWERT
0E9C04          4245          CLR D6          * ZAEHLER FUER WECHSEL DER FIGUREN
0E9C06          3001          SCHLEIFE:
0E9C06          MOVE D1,D0      * J.B. x-KOORDINATE VERWENDEN
0E9C08          4E99 000E1034  JSR #SIN       * FUER Y-ACHSE, BEWEGUNG ERZEUGEN
0E9C0E          4BC0          EXT.L D0
0E9C10          81FC 0005      DIVS #5,D0      * AMPLITUDE BEGRENZEN
0E9C14          0640 0064      ADD #100,D0    * UND IN MITTE LEGEN.
0E9C18          3400          MOVE D0,D2     * NEUE Y-KOORDINATE
0E9C1A          41F9 000E9D50  LEA MEN1,A0   * ERSTE FIGUR NEHMEN
0E9C20          0806 0003      BTST #3,D6     * ODER ZWEITE
0E9C24          6700 0008      BEQ WEITER
0E9C28          41F9 000E9D73  LEA MEN2,A0   * DAMIT ERGIBT SICH WECHSEL
0E9C2C          WEITER:
0E9C2C          JSR NEUFIGUR
0E9C34          0641 0002      ADD #GESCHW,D1 * UND IN x-RICHTUNG WEITER
0E9C38          4E99 000E0E3B  JSR #SYNC     * MIN 20MS WARTEN
0E9C3E          6700 FFF8      BEQ WARTER   * SONST KEINE BLEIBENDE BEWEGUNG
0E9C42          0646 0001      ADD #1,D6     * FUER FIGURENWECHSEL
0E9C46          0C41 0200      CMP #512,D1  * UEBER DEN GESAMTEN SCHIRM WANDERN LASSEN
0E9C4A          6F00 FFB8      BLE SCHLEIFE *
0E9C4E          4E75          RTS
0E9C50          NEUFIGUR:
0E9C50          * D1=x, D2=y, A0=FIGURADR
0E9C54          33C1 000E9D9E  MOVEN.L A0/D0-D2,-(A7)
0E9C58          33C2 000E9D9A  MOVE D1,XT1   * X1,Y11 UND ADRT1 SIND DIE
0E9C5C          23C8 000E9DAA  MOVE D2,YT1  * WERTE FUER DAS NEUE BILD
0E9C60          3039 000E9DB6  MOVE D0,XD1  * ZUNAECHST NEUE BILDSEITE ANNAHMEN
0E9C64          3239 000E9DBB  MOVE D0,XD2  *
0E9C68          4E89 000E9E0C  JSR #NEWPAGE
0E9C72          0A79 0001      EDR #1,SCHREIBSEITE * DANN WIRD SCHREIB- UND LESESEITE
0E9C76          000E9DB6  EDR #1,LESESEITE *
0E9C7A          0A79 0001      EDR #1,LESESEITE * FUER DEN NAECHSTEN AUFRUF VERTAUSCHT
0E9C7E          000E9DBB
0E9C80          3239 000E9DA2  MOVE XT3,D1  * NUN WIRD DAS ALTE BILD GELDESCHT
0E9C84          3439 000E9DAB  MOVE YT3,D2  * ES LIEGT ZWEI BILDER ZURUECK,
0E9C88          2079 000E9D82  MOVEN.L ADRT3,A0 * DAHER STEHEN DIE WERTE IN ...T3
0E9C8C          6100 009A      BSR LOESCHE
0E9C90          3239 000E9D9E  MOVE YT1,D1  * DAS NEUE BILD KANN NUN GESCHRIEBEN WERDEN
0E9C94          3439 000E9DAA  MOVE YT1,D2  * ES STEHT IN ...T1
0E9C98          2079 000E9DAA  MOVEN.L ADRT1,A0
0E9CA4          6100 006A      BSR SCHREIBE
0E9CA8          33F9 000E9DA0  MOVE XT2,XT2 * SCHLIESSLICH MUSS ALLES UM EINE
0E9CB4          000E9DA2      MOVE XT1,XT2 * ZEITEINHEIT VERSCHOBEN WERDEN,
0E9CB8          33F9 000E9D9E  MOVE YT2,YT3 * DAMIT SPAETER DAS LOESCHEN KORREKT
0E9CC4          000E9DAB
0E9CC8          33F9 000E9DA4  MOVE YT1,YT2 * DURCHGEFUEHRT WIRD.
0E9CD4          000E9DAB
0E9CD8          23F9 000E9DAE  MOVE.L ADRT2,ADRT3
0E9CE4          000E9DB2
0E9CE8          23F9 000E9DAA  MOVE.L ADRT1,ADRT2
0E9CC0          000E9DAE

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

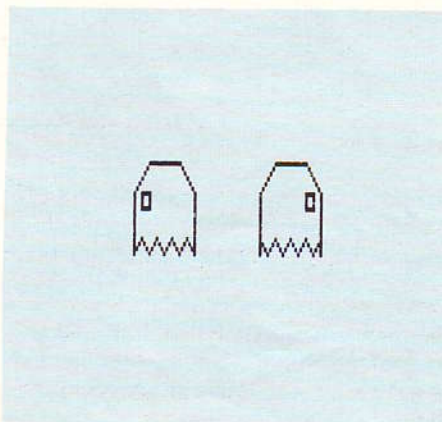
0E9CF0          4CDF 0107      MOVEN.L (A7)+,A0/D0-D2
0E9CF4          4E75          RTS
0E9CF6          TEST:
0E9CF6          41F9 000E9D50  LEA MEN1,A0   * AUSGABE BEIDER FIGUREN AUF DEM SCHIRM
0E9CF8          323C 0064      MOVE #100,D1 * DAMIT TEST MOEGLICH OB DATEN RICHTIG
0E9C00          343C 0064      MOVE #100,D2 * EINGETRAGEN WURDEN.
0E9D04          6100 0016      BSR SCHREIBE
0E9D08          41F9 000E9D73  LEA MEN2,A0
0E9D0C          323C 0096      MOVE #150,D1
0E9D10          343C 0084      MOVE #100,D2
0E9D14          6100 0004      BSR SCHREIBE
0E9D18          4E75          RTS
0E9D1C          SCHREIBE:
0E9D1C          * A0=ADRESSE
0E9D1C          103C 0011      MOVE.B #11,D0 * SCHRIFTGRÖSSE, FALLS TEXTE DA
0E9D20          4E87 60B0      MOVEN.L A0/D1-D2,-(A7)
0E9D24          4E89 000E0D7C  JSR #SETPEN  * D1=x,D2=y
0E9D28          4E89 000E1378  JSR #CDMPRINT * GRUNDPROGRAMM
0E9D30          4CDF 0106      MOVEN.L (A7)+,A0/D1-D2
```

```
0E9D34          4E75          RTS
0E9D36          LOESCHE:
0E9D36          * A0=ADRESSE
0E9D36          103C 0011      MOVE.B #11,D0 * SCHRIFTGRÖSSE, FALLS TEXTE DA
0E9D3A          4E87 60B0      MOVEN.L A0/D1-D2,-(A7)
0E9D3E          4E89 000E0D6E  JSR #ERAPEN  * D1=x,D2=y
0E9D44          4E89 000E1378  JSR #CDMPRINT
0E9D4A          4CDF 0106      MOVEN.L (A7)+,A0/D1-D2
0E9D4E          4E75          RTS
0E9D50          * FIGUREN IN KURZVEKTOREN CODIERT.
0E9D50          MEN1: DC.B %11111001 * START IST LINKS UNTEN.
0E9D54          FD          DC.B %11111010 * DIE LETZTEN DREI BITS BESTIMMEN DIE
0E9D58          F9 FD F9 FD F9 * RICHTUNG, JEDDOCH
0E9D62          DC.B %11111010 * SIND DIE WERTE NICHT WIE BEIM FIGUR-
0E9D66          FA FA FA * BEFEHL CODIERT.
0E9D70          FB          DC.B %11111011
0E9D74          FC          DC.B %FB
0E9D78          FE          DC.B %11111110
0E9D82          FE FE FE DC.B %FE,%FE,%FE
0E9D86          FF          DC.B %11111111
0E9D90          FC          DC.B %FF
0E9D94          FC FC FC DC.B %11111100
0E9D98          03          DC.B %FC,%FC,%FC
0E9DA2          FA FA FA FB DC.B %3 * PEN UP
0E9DA6          02          DC.B %FA,%FA,%FA,%11111000
0E9DAE          00          DC.B %2 * PEN DOWN
0E9DB4          00          DC.B %FB,%FA,%FE,%FC
0E9DB8          00          DC.B %0 * ENDE
0E9DC0          MEN2: DC.B %11111010
0E9DC4          FD F9 FD F9 FD DC.B %FD,%F9,%FD,%F9,%FD,%F9,%FD,%F9
0E9DC8          FC          DC.B %11111000
0E9DCC          FA FA FA DC.B %11111010
0E9DD0          FA FA FA DC.B %FA,%FA,%FA
0E9DD4          FB          DC.B %11111011
0E9DD8          FC          DC.B %FB
0E9DE0          FE          DC.B %11111110

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

0E9D84          FE FE FE          DC.B %FE,%FE,%FE
0E9D88          FF          DC.B %11111111
0E9D92          FC          DC.B %FF
0E9D96          FC FC FC          DC.B %11111100
0E9DA0          03          DC.B %FC,%FC,%FC
0E9DA4          3F          * PEN UP
0E9DA8          FA FA FA FB          DC.B %FA,%FA,%FA,%11111000
0E9DAC          FB FB FB FB FB          DC.B %FB,%FB,%FB,%FB,%FB
0E9DAE          02          DC.B %2 * PEN DOWN
0E9DB0          FB FA FE FC          DC.B %FB,%FA,%FE,%FC
0E9DB4          00          DC.B %0 * ENDE
0E9DB8          * RAM-SPEICHER, ACHTUN, WERTE RICHTIG EINTRAGEN.
0E9DC0          DS 0          * AUF WORTGRENZE LEGEN.
0E9DC4          XT1: DC.W 0    * x-KOORDINATE ZEITPUNKT T1
0E9DC8          XT2: DC.W 0    *
0E9DCC          XT3: DC.W 0    *
0E9DD0          YT1: DC.W 0    * y-KOORDINATE ZEITPUNKT T1
0E9DD4          YT2: DC.W 0    *
0E9DD8          YT3: DC.W 0    *
0E9DE0          ADRT1: DC.L MEN1 * FIGUR-ADRESSE ZEITPUNKT T1
0E9DE4          ADRT2: DC.L MEN1 * ACHTUNG, DEFAULT-WERTE
0E9DE8          ADRT3: DC.L MEN1
0E9DEC          SCHREIBSEITE: DC.W 1
0E9DE0          LESESEITE: DC.W 0
0E9DBA          END
0E9DBA          Ende-Symboltabelle
```

werden. Dabei soll sich die Figur selbst auch noch bewegen. Bild 3 zeigt die beiden Teilbilder der Figur, wie sie auch durch den Aufruf TEST zur Kontrolle ausgegeben werden können. Das Hauptprogramm START berechnet nun die x- und y-Koordinate der Figur und wählt entweder MEN1 oder MEN2 als Figur aus. Das Unterprogramm NEUFIGUR ersetzt nun den früheren Aufruf FIGUR. Damit immer die richtige Figur gelöscht wird, muß sich das Programm jeweils die alten Koordinaten und Adressen der Figuren merken, denn wenn eine Bildseite gelöscht wird, so liegt das bereits zwei Auf-



rufe lang zurück. Daher gibt es einige Speicherzellen, die mit xt1, xt2, xt3 usw. bezeichnet sind. Das Unterprogramm SCHREIBE schreibt eine Figur an eine bestimmte Stelle und LOESCHE löscht eine Figur. Die Figuren selbst sind mit den sogenannten Kurzvektoren codiert (siehe Handbuch Grundprogramm 4.3 oder GDP-Datenblatt). Hier wird eine Länge von drei Punkten als Kantenlänge der Vektoren verwendet, jedoch kann man sie beim GDP auch gemischt verwenden.

Rolf-Dieter Klein

Drucker-Umschaltung unter 68000-Grundprogramm

M. Miltsch, Filiale Hamburg

```
org #0
offset #a000
dc.l #55aa0180
dc.b 'Breit Dr'
dc.l start
dc.l ende
dc.b 1,0,0,0
dc.l 0,0,0
dc.l 0,0

start:
  move #!ciinit2,d7      * Ram auslesen
  trap #1
loop:
  move #!1b,d0          * ESC setzen
  move #!1o,d7          * Drucker initialisieren
  trap #1
  move #!0e,d0          * umschalten auf Breitschrift
  move #!1o,d7
  trap #1
  move #!ci2,d7         * Zeichen lesen
  trap #1
  beq ende
  move #!1o,d7          * Zeichen auf Drucker ausgeben
  trap #1
  bra.s loop
  move #!12,d0          * Drucker auf normalschrift schalten
  move #!1o,d7
  trap #1
ende:
  rts
```

* Dies Programm zeigt wie man einen Drucker
* per Software umschalten kann. Startet man
* das Programm, so druckt es sich selber aus.
* Will man andere Texte ausgeben, so muss man
* vorher einen neuen Textstart auswählen.

Submit-Dateien unter CP/M68K

Mit sogenannten „Submit-Dateien“ besteht die Möglichkeit, „Stapelverarbeitung“ unter den Betriebssystemen CP/M2.2 und CP/M68k zu betreiben.

Das Wort „Stapelverarbeitung“ kommt aus der Vorzeit der EDV, in der Aufrufe von Programmen auf Lochkarten geschrieben wurden; mehrere solche Karten wurden dann zu Stapeln übereinandergelegt.

Bei uns ist der Stapel eine ganz normale Textdatei, die mit der Erweiterung „SUB“ enden muß.

Nun muß meist beim Aufruf eines Programmes, z.B. des Assemblers, einer oder mehrere Parameter, z. B. der Name des zu übersetzenden Programmes eingegeben werden. Dies geschieht durch Angabe des „Platzhalters“ in der Submit-Datei, der Zeichenfolge „\$n“, wobei n von 1 bis 9 zählt.

Das kleine Beispiel EAS, das von Michael Miltsch aus Hamburg zusammengestellt wurde, zeigt dies.

Zunächst ist mit dem Texteditor eine Datei EAS.SUB zu erstellen. Rufen Sie dazu den Editor auf mit

```
EDITRDK EAS.SUB
```

In die Datei wird eingegeben.

```
EDITDK $1.s
```

```
ASSRDK $1.s
```

```
STARTE START
```

Diese dann abgespeicherte Datei wird aufgerufen mit SUBMIT EAS name, wobei statt „name“ der Programmname des zu erstellenden Programmes eingegeben wird. Die so erstellte Submit-Datei wird dann abgearbeitet, d. h. zunächst meldet sich der Editor, Sie geben ein Programm ein. Danach wird der Assembler aufgerufen, das Programm übersetzt und gestartet.

Dieses Beispiel ist nur für fehlerfreie Programmierer gedacht, die sicher sind, keine Syntax-Fehler zu machen, da ja gleich gestartet wird. Zum Trost läßt sich ein Submit-Vorgang auch per CONTROL-C abbrechen.

Weiteres Beispiel, hier wird der Assembler des CP/M68k verwendet:

```
EDITRDK $1.s
```

```
AS68 $1.S
```

```
RELOC $1.o.$1.68k
```

Wird eine Ausgabe auf dem Bildschirm gewünscht, so muß nach AS68 „-P“ stehen, also

```
AS68 -P $1.s
```

Grafik-Beispielprogramm für 68008

```
; Hier ein Programm, das kurz genug ist, um mal  
; einfach so eingegeben zu werden. Kommentar fehlt,  
; versuchen Sie doch zur Übung herauszufinden, wie das  
; Programm funktioniert. Wir können nur sagen:  
; Eingeben lohnt sich...
```

```
;
start:
  move #9,d6 de
  ;
  loop:
    jsr $clr
    jsr $firsttime
    jsr $hide
    addq #1,d6 de
    clr d4 bc
    ;
    inner:
      addq #1,d4 bc
      move d4,d0 bc
      jsr $schreite
      move d6,d0 de
      jsr $drehe
      cmpi #350,d4 bc
      bne inner
      ;
      cmpi #350,d6 de
      bne loop
  ;
  bra start
  ;
```


Buchstabenschlüssel zu den Serien 74.../54.../ und CMOS

von Thomas von Jan, Steufzger Straße 73, 8960 Kempten

- Serie 54... : Mil-Spezifikation zu 74 ... - 55 bis + 125 Grad C
- Serie 54H... : Mil-Spezifikation von 74H ... - 55 bis + 125 Grad C
- Serie 54L... : Mil-Spezifikation von 74L ... - 55 bis + 125 Grad C
- Serie 54LS... : Mil-Spezifikation von 74LS ... - 55 bis + 125 Grad C
- Serie 54S... : Mil-Spezifikation von 74S ... - 55 bis + 125 Grad C
- Serie 74... : TTL-Standardversion
- Serie 74ALS... : Advanced Low Schottky
- Serie 74AS... : Advanced Schottky
- Serie 74C... : TTL-Kompatible CMOS-Version von 74...
- Serie 74F... : kompatibel zu 74S... mit 75 - 80 % weniger Stromverbrauch
- Serie 74H... : High Speed
- Serie 74 HC... : High Speed CMOS mit TTL-kompatiblen Ausgängen
- Serie 74HCT... : High Speed CMOS mit TTL-kompatiblen Ein- und Ausgängen
- Serie 74L... : Low Power
- Serie 74 LS... : Low Power Schottky
- Serie 74S... : Schottky

- CMOS 4000 A/14000C-Serie: 3 - 15 Volt
- CMOS 4000 B/14000BC-Serie: 3 - 20 Volt
- CMOS 4500 A/14500C-Serie: 3 - 15 Volt
- CMOS 4500 B/14500BC-Serie: 3 - 18 Volt
- LOC-MOS HEF 4000B-Serie: 3 - 18 Volt
- LOC-MOS HEF 4500B-Serie: 3 - 18 Volt

... AF / AD / BF / AL / BAL / BDM: Mil-Spezifikation - 55 bis + 125 Grad C
HEF... BD: Mil-Spezifikation - 55 bis + 125 Grad C

Mil-Spezifikation = Militärspezifikation (meist keramischer IC-Körper)



LOOP PRIVAT Berghütte zu vermieten!

Unser Mitarbeiter in unserer Schwesterfirma, Alfred Einsiedler, ist Besitzer einer Berghütte und vermietet diese an LOOP-Leser.

Die Hütte ist landschaftlich herrlich in einem Naturschutzgebiet im Allgäu gelegen, zwischen Sonthofen und Hindelang, dem Ausgangspunkt vieler Berg- und Skitouren. Sie verfügt über eine Quelle, jedoch keinen elektr. Strom (= Urlaub ohne Computer). Die Einrichtung ist komplett, mitgebracht werden muß neben Verpflegung nur Bettwäsche und Handtücher.

Ein Doppelbett, zwei Stockbetten und eine Couch bieten fünf Personen Platz. Gekocht wird auf Gaskochern oder auf dem Kachelofen. Über eine Mautstraße ist, je nach Schneelage, eine Zufahrt möglich.

Der Mietpreis pro Woche beträgt DM 560,- und beinhaltet alle Nebenkosten. Mindestmietdauer eine Woche - die Hütte ist noch kurzfristig frei. Nähere Auskunft gibt Alfred Einsiedler, Telefon 0831-67168, oder die LOOP-Redaktion.



AKTUELL

Neu zum NDR-Computer und mc CP/M-Computer:

- Hardcopy/Maus (mc und NDR)
Hardcopy des Bildschirmes auf Drucker
Bausatz..... 198,00
- CPU68000 - 12 MHz (NDR)
Die echte 16 bit CPU zum NDR-Computer
Bausatz..... 498,00
- REL (NDR) Die 8-fach Relais-Ausgabe
zum NDR-Computer, Bausatz..... 159,00
- RAM 8K x 8 bit (6464) Tiefpreis..... 9,90
- TEAC-Laufwerk FD 55F (800 KByte).... 448,00
- LOOP 5 ist da - ABO nur 20,00 DM.

Kleine Drucker-Routine für CP/M68K

von Michael Milzsch, Hamburg

Durch diese Routine erfolgt ein Textausdruck vom Grundprogramm aus. Der Aufruf erfolgt mit Funktion 23, Textstart ist immer die aktuelle Adresse, z.B. \$9000.

```

start:      .text
            move    #23,d0
            trap    #3                * Sonderfunktion
            move.l  a4,gruadr
            move    #31,d7            * Textstart auf Anfang setzen
            jsr    $420(a4)          * CIINIT2
loop:       movea.l  gruadr,a4
            move    #32,d7            * Zeichen wird eingelesen
            jsr    $420(a4)          * CI2
            movea.l  gruadr,a4
            beq.s   ende             * Wenn Textende erreicht ist
            move    #22,d7            * Zeichen wird auf Drucker ausgegeben
            jsr    $420(a4)          * LD
            movea.l  gruadr,a4
            bra.s   loop             * Wenn nicht erreicht wieder zurück
ende:       rts                      * Textende ist erreicht

            .data
            .even
            gruadr: dc.l 0
end
    
```

Seminare in der Filiale Hamburg

In unserer Filiale Hamburg, Ehrenbergstraße 56, 2000 Hamburg 50, Telefon 040-388151, finden jeweils samstags und, nach Vereinbarung, auch abends, Seminare zum Thema „NDR-Computer“ statt. Auch neue Produkte und Erweiterungen werden vorgestellt.

Aktuell wird eine universelle Übertragungssoftware gezeigt, die es ermöglicht, den NDR-Computer mit Z80 als universelles Terminal zu betreiben.

Der Eintritt zu den Seminaren ist natürlich kostenlos - über die aktuellen Termine informiert unser Filialeiter, Herr Michael Miltsch.

IN&OUT

Leser fragen – Fachleute antworten Stellen Sie auch Ihre Fragen an „loop“

Sehr geehrte LOOP-Redaktion, ich benütze den NDR-Klein-Computer mit der großen DIN-Tastatur vom Elektronikladen Detmold. Es tritt hierbei ein Problem auf, das möglicherweise auch andere NDR-Klein-Computerbenutzer mit einer „fremden“ Tastatur haben: In unregelmäßiger Folge ist der erste Buchstabe eines Wortes verloren und der Inhalt der bis zu 20 Zeichen umfassenden Schnellspeichertasten wird vom Editor nur bis zu den zwei ersten Zeichen akzeptiert. Die Tastatur ist sicher fehlerfrei. Das Problem liegt wahrscheinlich am Editor, der zu langsam ist. Übrigens, wenn mit dem Read-Befehl des Grundprogramms ein Eingabefeld erzeugt wird, gehen keine Zeichen verloren.

Gibt es eine Lösung des Problems ohne eine andere Tastatur zu kaufen?

Wilfried Oehrle
Philosophenweg 19, 7400 Tübingen

Antwort LOOP:

Sie haben recht, es liegt tatsächlich am „zu langsamen“ Editor. Das Problem ist nur durch eine Hardware-Änderung zu beheben, und zwar durch den Aufbau eines kleinen Puffers auf der KEY-Karte. Wir arbeiten daran, damit auch bei sehr schnellen Tastaturen keine Zeichen verloren gehen können. Sobald die Baugruppe verfügbar sein wird, können Sie durch Kauf der neuen Leiterplatte und einiger ICs Ihre alte KEY „aufrüsten“ und dann ohne Probleme arbeiten.

Sehr geehrte Damen und Herren, als Benutzer des NDR-Computers und Bezieher von LOOP habe ich folgende Frage:

Anfang 1985 kaufte ich mir den Datenrecorder von BOSTON MC 3810, weil er in der „mc“ so positiv besprochen wurde.

Die Enttäuschung war groß, als ich ihn auspackte. Er besitzt kein eigenes Netzteil und der NF-Anschluß war mir bisher nicht möglich. Die Stromversorgung ist relativ leicht anzuschließen. Er läuft jetzt, aber das Laden und Speichern von Programmen habe ich noch nicht geschafft. Können Sie mir den Anschluß an den NDR-Computer angeben?

Zur Ihrer Zeitschrift LOOP möchte ich sagen, daß ich sehr froh bin, daß es sie

gibt. Denn ohne praktische Tips und Hilfen ist der NDR-Computer nur die Hälfte wert. Vielen Dank.

Rolf Keßling
Dorfstraße 24, 7850 Lörrach

Antwort LOOP:

Wir haben Ihren Brief zum Anlaß eines eigenen Artikels genommen, den Sie in dieser LOOP finden! Viel Erfolg.

Sehr geehrter Herr Graf, seit November 1984 beschäftige ich mich intensiv mit dem NDR-Computer und bin mittlerweile mit dem vollständigen CP/M-Ausbau fertig geworden.

Nach einigen anfänglichen Schwierigkeiten, die richtigen Unterlagen zu den richtigen Platinen zu bekommen, ging der Zusammenbau sehr schnell und sicher – von 8 Platinen arbeiteten 7 ohne Probleme sofort nach dem Einschalten.

Probleme gab es auf der RAM64/256: Die Karte lief nicht, trotz Vorhandenseins aller wichtigen Signale, auch der REFRESH war in Ordnung. Durch systematisches Austauschen einzelner IC's konnte der Fehler jedoch gefunden werden. Dieser Fehler scheint mir doch einigermaßen kurios zu sein, denn es stellte sich heraus, daß die 5 auf der Platine befindlichen 74LS153 nicht richtig arbeiteten. Diese IC's waren zwar elektronisch dennoch in Ordnung, lediglich das Timing des Bauteils (Multiplexer!) war völlig daneben. Um es nochmals klarzustellen: das Bauteil entsprach genau der Bauanleitung von GES, war jedoch von einem „Billighersteller“ und nicht von „Texas Instruments“, wie sonst bei GES-Bausätzen üblich (mir lag kein Bausatz, sondern nur die Platine vor). Mein Tip an LOOP-Leser wäre erstens: lieber, gerade bei Bauteilen, die zeitkritisch sind, Markenbauteile verwenden, oder gleich einen GES-Bausatz verwenden!

Ein weiterer Punkt: Obwohl ich mich seit einigen Jahren mit Computern beschäftige, und auch schon 6502-Computer gebaut habe, gab es immer wieder die Frage, wie die verschiedenen Buchstaben bei den TTL-Serien zu verstehen seien (74LS... / 74HCT... usw.).

Nach Rücksprache mit Herstellern ergab sich folgende Zusammenstellung, die

vielleicht auch für andere Computer-Bauer interessant wäre:

Thomas von Jan
Steufzer Straße 73, 8960 Kempten

Antwort LOOP:

Herzlichen Dank für den Tip, der wirklich etwas für sich hat. Dies ist auch der Grund, warum die GES-Bausätze manchmal etwas teurer sind als beim Einkauf von (billigen) Einzelteilen. Ihre Zusammenstellung finden wir so interessant, daß wir sie in dieser LOOP abgedruckt haben.

Sehr geehrte Herren!

Vor kurzem baute ich den NDR-Computer in Einsteigerversion (POW5V, SBC2, IOE und Hexio). Alles klappte hervorragend. Der Computer war fertig und das Buch HEXMON war durchgearbeitet. Parallel dazu verwendete ich das Buch „Microcomputer selbstgebaut und programmiert“.

Nun meine Kritik:

1. Für einen absoluten Einsteiger ist die Software und die Programmierung sehr schlecht beschrieben. Daraufhin legte ich mir das Buch „Programmierung des Z80“ von Rodney Zaks zu und man bekam schon einen besseren Einblick. Ich finde, daß für die Programmierung des Computers, gerade in der Grundausstattung, mehr getan werden müßte. Oder sehe ich das vielleicht falsch???

2. Die absoluten Einsteiger kommen in LOOP zu kurz. Es wird viel über Assembler, 68008 und 68000 geschrieben, aber kaum etwas über den Einsteiger, der nur die Grundversion mit HEXIO besitzt. Denn der NDR-Computer soll doch auch ein Computer für den absoluten Einsteiger sein, oder? Deshalb denke ich vielleicht mal an Listings oder Anwenderprogramme in Maschinensprache.

3. Vorschlag: Was halten Sie davon, in jeder LOOP ein Kapitel „Programmierung für den Einsteiger in Maschinensprache“ einzurichten? Denn Sie wissen bestimmt genau so gut wie viele andere, daß die Programmierung in Maschinensprache nicht sehr einfach ist.

4. Nun noch eine Frage: Wie werden bei der IOE die Ports In1, In2, Out1 und Out2 angesprochen (welche Adressen)? Was kann mit den 4 Brücken geändert werden? Und, kann man Daten ein und auslesen, während die HEXIO angeschlossen ist (wenn ja, dann wie)?

Dieses alles soll LOOP nicht schlechtmachen, sondern im Gegenteil. Ich finde es gut, daß es eine eigene Zeitschrift für den Computer gibt, und das sich der Leser mit seinen speziellen Wünschen seines Computers an diese Zeitschrift wenden kann. Wie gesagt, nur die absoluten Einsteiger kommen in LOOP etwas zu kurz.

(Übrigens, daß habe ich schon mehrfach gehört.) Trotzdem, *LOOP ist für den Besitzer des NDR-Computers Gold wert.*

Ich hoffe, Sie können meine Fragen alle beantworten.

Uwe Frenzel
Südweg 48, 2900 Oldenburg

Antwort LOOP:

Sie haben vollkommen recht. Wir planen, gerade die Literatur zum Einsteigerpaket viel ausführlicher zu gestalten, um dem Anfänger wirklich zu helfen. Wir glauben, daß besonders die Zusammenarbeit mit CHRISTIANI und den dort erstellten Lehrbriefen zum Einsteigerpaket – der erste ist schon für DM 38,- von uns oder Christiani lieferbar – hier helfen wird.

Alle LOOP-Leser, die auch mal mit dem Einsteigerpaket begonnen haben, sind hier aufgerufen: Senden Sie uns Ihre Erfahrungen und vielleicht kleine Programme, den Anfängern das Leben leichter machen!

Zu Ihren Anregungen: Sie werden in den nächsten LOOPS viel mehr für Anfänger finden!

Betr.: NDR-Klein-Computer, RDK-Grundprogramm 68K.

In der Version 3.1 des 68008-Grundprogramms wird im Unterprogramm C0 (Zeichenausgabe auf Bildschirm mit Steuerzeichenbeachtung) die Funktion „Cursor down“ (\$16) nicht korrekt ausgeführt.

Die Ursache hierfür ist im Heft „Arbeitsmaterial zur Serie Mikroelektronik: 68008 Grundprogramm“ des Franzis-Software-Service zu erkennen:

Auf Seite 39 heißt es
0016CE 5279 00008224 ADDQ
#1,CURY.

Durch den Wortzugriff wird die Speicherzelle mit der Adresse \$008225 beeinflusst; dadurch wird das Steuerzeichen \$16 als ESCAPE aufgefaßt. Richtig muß es heißen

0016CE 5239 00008224 ADDQ.B
#1,CURY.

Im Grundprogramm braucht also nur der Inhalt der Adresse \$001ACF von \$79 in \$39 geändert zu werden, um eine ordnungsgemäße Funktion herzustellen.

Auch wenn in neueren Versionen des Grundprogramms dieser Fehler bereits beseitigt ist, vermeidet ein entsprechender Hinweis in einer der nächsten Ausgaben von *LOOP* sicherlich vielen Anwendern der Version 3.1, Ärger und lange Fehlersuche.

Dirk Hannemann
Doebnerstraße 28, 3200 Hildesheim

Antwort LOOP:

Danke für Ihren Hinweis – der Fehler ist in der Version 4.3 schon behoben. Nochmals der Hinweis: Alle älteren Versionen der Programme, werden gegen eine Gebühr von DM 10,- pro Eprom, falls Sie die Original-Eproms zurücksenden, umgetauscht! Dies lohnt sich besonders beim Grundprogramm für den 68008, da doch viele neue Funktionen dazu gekommen sind. Auch für den Z80-BASIC-Interpreter (jetzt V1.5) ist dies ratsam.

Siehe auch die Tabelle: Aktuelle Software-Versionen.

GRAMME · PROGRAMME · PROGRAMME · PRO

Squash – Ein spannendes Spiel aus dem LOOP-Wettbewerb

Von Marc Rasch

Squash ist in PASCAL-S programmiert und verwendet 68008-Routinen. Es läuft mit dem 68008-Ausbau.

Das Programm erzeugt ein Spielfeld, einen Schläger und einen Ball, der mit dem Schläger getroffen werden muß. Das Spiel habe ich „Squash“ genannt.

Das Spiel kann mit Schwierigkeitsstufen

angewählt werden und der Schläger würde sich ebenfalls über den neuen Joystick steuern lassen, den ich leider noch nicht habe.

Der Ball bewegt sich über verschiedene errechnete Zufallszahlen, die wiederum verschiedene Bewegungspozeduren aufrufen.

Rolf-D.Klein 68000/68008 Assembler 3.1 (C) 1983

```

009C00 *****
009C00 * ASSEMBLER-PROGRAMM
009C00 * SPEICHERPLATZ $9000 *
009C00 * WIRD ALS 1.PROGRAMM GELADEN *
009C00 *****
009C00 *** SQUASH - ASSEMBLER - UNTERPROGRAMM ***
009C00
009C00 BALL:
009C00 41F9 00009C74 LEA FIGUR,A0
009C06 303C 0002 MOVE #2,D0
009C0A 4EB9 00003A0C JSR $FIGUR
009C10 4E75 RTS
009C12 SCHLAEGER:
009C12 343C 0004 MOVE #4,D2
009C16 4EB9 0000091E JSR $MOVETO
009C1C D243 ADD D3,D1
009C1E 4EB9 0000098A JSR $DRAWTO
009C24 343C 0005 MOVE #5,D2
009C28 4EB9 0000091E JSR $MOVETO
009 E 9243 SUB D3,D1
009C30 4EB9 0000098A JSR $DRAWTO
009C36 4E75 RTS
009C38 SCHREIBEN:
009C38 4EB9 00000794 JSR $SETPN
009C3E 4EB9 00009C12 JSR SCHLAEGER
009C44 4E75 RTS
009C46 LOESCHEN:
009C46 4EB9 00000786 JSR $SERAPEN
009C4C 4EB9 00009C12 JSR SCHLAEGER
009C52 4E75 RTS
009C54 NAME:
009C54 41F9 00009C6E LEA TEXT,A0
009C5A 323C 0005 MOVE #5,D1
009C5E 343C 0064 MOVE #100,D2
009C62 303C 0007 MOVE #7,D0
009C66 4EB9 00000D5C JSR $WRITE
009C6C 4E75 RTS
009C6E TEXT:
009 E 5351415348 DC.B 'SQUASH'
009C73 00 DC.B 0
009C74 FIGUR:
009C74 04 05 06 07 00 DC.B 4,5,6,7,0,0,1,2,3,4,10
009C79 00 01 02 03 04

```

```

009C7E 0A
009C7F
0000 Fehler entdeckt
008A8B Ende-Symboltabelle

*****
* PASCAL/S Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein *
* Version nach PASCAL/S von *
* N.Wirth 1976, E.T.H Zuerich *
*****

0 (*****
0 (* PASCAL-PROGRAMM,SPEICHERPLATZ $9C90, *)
0 (* WIRD ALS 2.PROGRAMM GELADEN *)
0 (*****
0
0 PROGRAM SQUASH(INPUT,OUTPUT);
0 VAR ENDE,GESCHWINDIGKEIT,ZAEHLER,A,POSK,POSITIONX,POSITIONY:INTEGER;

0
0 ANTWORT,PAUSE,START,LAENGE,ABSTAND,ABFRAGE:INTEGER;
0 SEED:REAL;
0 PROCEDURE UMRANDUNG;
0 VAR X,Y:INTEGER;
0 BEGIN
0 WRITELN(CHR(1),'E $SETFLIP');
0 FOR X:=5 TO 9 DO
0 BEGIN
0 Y:=3;
0 WRITELN(CHR(1),'E $MOVETO ',0,' ',X,' ',Y);
0 Y:=255;
0 WRITELN(CHR(1),'E $DRAWTO ',0,' ',X,' ',Y);
0 END;
0 FOR Y:=241 TO 245 DO
0 BEGIN
0 X:=5;
0 WRITELN(CHR(1),'E $MOVETO ',0,' ',X,' ',Y);
0 X:=505;
0 WRITELN(CHR(1),'E $DRAWTO ',0,' ',X,' ',Y);
0 END;
0 FOR X:=501 TO 505 DO
0 BEGIN
0 Y:=3;
0 WRITELN(CHR(1),'E $MOVETO ',0,' ',X,' ',Y);

```



```

118 Y:=241;
119 WRITELN(CHR(1),'E $DRAWTO ',O,' ',X,' ',Y);
120 END;
121 FOR Y:=255 TO 255 DO
122 BEGIN
123 X:=5;
124 WRITELN(CHR(1),'E $MOVETO ',O,' ',X,' ',Y);
125 X:=65;
126 WRITELN(CHR(1),'E $DRAWTO ',O,' ',X,' ',Y);
127 END;
128 FOR X:=60 TO 65 DO
129 BEGIN
130 Y:=241;
131 WRITELN(CHR(1),'E $MOVETO ',O,' ',X,' ',Y);
132 Y:=255;
133 WRITELN(CHR(1),'E $DRAWTO ',O,' ',X,' ',Y);
134 END;
135 PROCEDURE SCHLAEGER1;
136 BEGIN
137 IF POSX<=9
138 THEN
139 BEGIN
140 POSX:=10;
141 END;
142 IF POSX=ABSTAND
143 THEN
144 END;
145 END;
146 PROCEDURE TASTATUR;
147 BEGIN
148 WRITELN(CHR(1),'G $FFFFFF68');
149 READ(A);A:=A-53;A:=A*15;
150 POSX:=POSX+A;
151 SCHLAEGER1;
152 END;
153 PROCEDURE PUNKTE;
154 BEGIN
155 IF (POSITIONX)=POSX) AND (POSITIONY<=POSX+LAENGE)
156 THEN
157 BEGIN
158 ENDE:=ENDE+1;
159 WRITELN(CHR(27),'=',CHR(32+0),CHR(32+0),ENDE:4);
160 END;
161 END;
162 PROCEDURE HOCH;
163 BEGIN
164 WHILE (POSITIONX)=20) AND (POSITIONY<=235) DO
165 BEGIN
166 WRITELN(CHR(1),'E BALL ',2,' ',POSITIONX,' ',POSITIONY);
167 TASTATUR;
168 POSITIONX:=POSITIONX+0;
169 POSITIONY:=POSITIONY+GESCHWINDIGKEIT;
170 END;
171 END;
172 PROCEDURE RUNTER;
173 BEGIN
174 WHILE (POSITIONX)=20) AND (POSITIONY)=ABFRAGE) DO
175 BEGIN
176 WRITELN(CHR(1),'E BALL ',2,' ',POSITIONX,' ',POSITIONY);
177 TASTATUR;
178 IF POSITIONY=ABFRAGE
179 THEN
180 PUNKTE;
181 POSITIONX:=POSITIONX+0;
182 POSITIONY:=POSITIONY-GESCHWINDIGKEIT;
183 END;
184 END;
185 PROCEDURE RECHTSHOCH;
186 BEGIN
187 WHILE (POSITIONX)=490) AND (POSITIONY<=235) DO
188 BEGIN
189 WRITELN(CHR(1),'E BALL ',2,' ',POSITIONX,' ',POSITIONY);
190 TASTATUR;
191 POSITIONX:=POSITIONX+GESCHWINDIGKEIT;
192 POSITIONY:=POSITIONY+GESCHWINDIGKEIT;
193 END;
194 END;
195 PROCEDURE LINKSHOCH;
196 BEGIN
197 WHILE (POSITIONX)=20) AND (POSITIONY<=235) DO
198 BEGIN
199 WRITELN(CHR(1),'E BALL ',2,' ',POSITIONX,' ',POSITIONY);
200 TASTATUR;
201 POSITIONX:=POSITIONX-GESCHWINDIGKEIT;
202 POSITIONY:=POSITIONY+GESCHWINDIGKEIT;
203 END;
204 END;
205 PROCEDURE LINKSRUNTER;
206 BEGIN
207 WHILE (POSITIONX)=20) AND (POSITIONY)=ABFRAGE) DO
208 BEGIN
209 WRITELN(CHR(1),'E BALL ',2,' ',POSITIONX,' ',POSITIONY);
210 TASTATUR;
211 IF POSITIONY=ABFRAGE
212 THEN PUNKTE;
213 POSITIONX:=POSITIONX-GESCHWINDIGKEIT;
214 POSITIONY:=POSITIONY-GESCHWINDIGKEIT;
215 END;
216 END;

```

PASCAL/S Pcode-Compiler V3.1 (C) 1984

```

550 PROCEDURE RECHTSRUNTER;
551 BEGIN
552 WHILE (POSITIONX<=490) AND (POSITIONY)=ABFRAGE) DO
553 BEGIN
554 WRITELN(CHR(1),'E BALL ',2,' ',POSITIONX,' ',POSITIONY);
555 TASTATUR;
556 IF POSITIONY=ABFRAGE
557 THEN PUNKTE;
558 POSITIONX:=POSITIONX+GESCHWINDIGKEIT;
559 POSITIONY:=POSITIONY-GESCHWINDIGKEIT;
560 END;
561 END;
562 FUNCTION RANDOM:INTEGER;
563 BEGIN
564 SEED:=SEED*27.182813+31.415917;
565 SEED:=SEED-TRUNC(SEED);
566 RANDOM:=TRUNC(SEED*6);
567 END;
568 BEGIN
569 REPEAT
570 WRITELN(CHR(1),'E $CLRSCREEN');
571 WRITELN(CHR(1),'E NAME');
572 FOR ZAEHLER:=1 TO 15000 DO;
573 WRITELN(CHR(1),'E $CLRSCREEN');
574 WRITELN(CHR(27),'=',CHR(32+4),CHR(32+0),'GESTEUERT WIRD MIT DEN TASTE
N: ');
575 WRITELN(CHR(27),'=',CHR(32+6),CHR(32+0),' L I N K S = 4');
576 WRITELN(CHR(27),'=',CHR(32+8),CHR(32+0),' R E C H T S = 6');
577 WRITELN(CHR(27),'=',CHR(32+10),CHR(32+0),' S T O P = 5');
578 WRITE(CHR(27),'=',CHR(32+12),CHR(32+0),'WAEHLEN SIE DEN SCHWIERIGKEIT
S: ');
579 WRITE('GRAD:');
580 WRITELN;
581 WRITE(CHR(27),'=',CHR(32+14),CHR(32+0),'1 = LEICHT , 2 = SCHWER (<
');
582 READLN(GESCHWINDIGKEIT);
583 IF GESCHWINDIGKEIT=1
584 THEN
585 BEGIN
586 ABFRAGE:=15;
587 GESCHWINDIGKEIT:=5
588 END
589 ELSE
590 BEGIN
591 GESCHWINDIGKEIT:=10;
592 ABFRAGE:=20;
593 END;
594 WRITE(CHR(27),'=',CHR(32+16),CHR(32+0),'WAEHLEN SIE DIE SCHLAEGERGROE
SSE: ');
595 WRITELN;
596 WRITE(CHR(27),'=',CHR(32+18),CHR(32+0),'1 = KLEIN , 2 = GROSS (<
');
597 READLN(LAENGE);
598 THEN
599 BEGIN
600 LAENGE:=15;
601 END;

```

PASCAL/S Pcode-Compiler V3.1 (C) 1984

```

816 ABSTAND:=485;
817 END
818 ELSE
819 BEGIN
820 LAENGE:=25;
821 ABSTAND:=475;
822 END;
823 WRITELN(CHR(27),'=',CHR(32+20),CHR(32+0),'DRUECKEN SIE DIE TASTE > 5
<');
824 WRITELN(CHR(27),'=',CHR(32+22),CHR(32+0),'UND DAS SPIEL BEGINNT. ');
825 REPEAT
826 WRITELN(CHR(1),'G $FFFFFF68');
827 READ(START);
828 UNTIL START=53;
829 POSX:=300;
830 POSITIONX:=95;
831 POSITIONY:=100;
832 ENDE:=0;
833 SEED:=1.2345678;
834 WRITELN(CHR(1),'E $CLRSCREEN');
835 UMRANDUNG;
836 WRITELN(CHR(27),'=',CHR(32+0),CHR(32+0),ENDE:4);
837 REPEAT
838 CASE RANDOM OF
839 0:RUNTER;
840 1:RECHTSHOCH;
841 2:HOCH;
842 3:LINKSRUNTER;
843 4:LINKSHOCH;
844 5:RECHTSRUNTER;
845 END;
846 UNTIL ENDE=15;
847 FOR PAUSE:=1 TO 10000 DO;
848 WRITELN(CHR(1),'E $CLRSCREEN');
849 WRITELN(CHR(27),'=',CHR(32+12),CHR(32+10),'WOLLEN SIE NOCHEINMAL';
850 WRITELN(CHR(27),'=',CHR(32+14),CHR(32+10),'SPIELEN ? (J/N)');
851 REPEAT
852 WRITELN(CHR(1),'G $FFFFFF68');
853 READ(ANTWORT);
854 UNTIL (ANTWORT=74) OR (ANTWORT=78);
855 WRITELN(CHR(1),'E $CLRSCREEN');
856 UNTIL ANTWORT=78;
857 WRITELN(CHR(27),'=',CHR(32+12),CHR(32+7),'ES WAR MIR EIN VERGNUEGEN, ');
858 WRITELN(CHR(27),'=',CHR(32+14),CHR(32+7),'MIT IHNEN ZU SPIELEN !');
859 END.

```

Bereits mit dem Grundprogramm des Single Board Computers (SBC2-Platine) ermöglicht die PROMER-Baugruppe das Lesen und Programmieren von EPROM's. Trotz der Speichermöglichkeit auf Diskette besteht auch unter dem Betriebssystem CP/M immer wieder die Notwendigkeit, ein EPROM zu programmieren. Durch die Leitungsfähigkeit moderner

Produkte

Programmierung von EPROM's unter CP/M

Compiler kann selbst ein in einer Hochsprache geschriebenes Programm die Ansteuerung der PROMER-Baugruppe übernehmen. Im vorliegenden Fall übernimmt ein Turbo Pascal Programm diese Aufgabe. Leider läßt der Umfang des Programms einen Abdruck der Quelle an dieser Stelle nicht zu. Die Programm-

quelle und ablauffähiges Objekt (COM-Datei) kann jedoch über die Firma Graf bezogen werden. Vor dem Aufruf des Programms sollte man sich davon überzeugen, daß das Monoflop der PROMER-Baugruppe einen 50 ms Impuls liefert (siehe Handbuch der Baugruppe).

Nach dem Aufruf meldet sich das Programm mit dem im Bild 1 gezeigten Menue. Wie aus dem Munue ersichtlich, ermöglicht das Programm EPROM's verschiedener Typen (Bild 2) zu prüfen, zu lesen und zu programmieren. Beim Lesen eines EPROM's kann die Ausgabe des Inhalts sowohl direkt auf dem Bildschirm als auch, im sogenannten Intel-Hex-Format, in einer Datei erfolgen. Bild 3 dient als Beispiel für die direkte Ausgabe auf dem Bildschirm. Dabei besteht die Möglichkeit, nur einen bestimmten Bereich des EPROM's und eine beliebige Anfangsadresse für die Ausgabe zu spezifizieren. Bei der Angabe eines \$-Zeichens interpretiert das Programm die nachfolgende Eingabe als sedizimale Zahl.

Zur Programmierung eines EPROM's erwartet das Programm eine Datei, in der sich im Intel-Hex-Format die Daten befinden. Was stellt nun dieses ominöse Intel-Hex-Format dar, und wie erhält man eine solche Datei?

In dem von der Firma Intel erfundenen Format enthält jede Zeile eine bestimmte Menge an Datenbytes, die Anfangsadresse des jeweils ersten Bytes der Zeile und eine Prüfsumme. Es ist nun nicht nötig, selbst die Daten für das EPROM in dieses Format zu bringen. Diese Aufgabe übernehmen die Assembler, die man ja zur Estellung eines Programms (ggfs. für ein EPROM) benutzt. Beinhaltet zum Beispiel die Datei TEST.ASM die Quelle eines Assembler-Programms, so liefert der Befehl ASM TEST die Datei TEST.HEX. In dieser Datei befindet sich nun das übersetzte Programm im gewünschten Intel-Hex-Format. Möchte man das Programm vor der Programmierung des EPROM's noch testen, so kann man dazu das CP/M-Dienstprogramm

```
Anfangsadresse (EPROM) eingeben oder RETURN fuer Voreinstellung:
Endeadresse (EPROM) eingeben oder RETURN fuer Voreinstellung: $df
Ausgabeadresse eingeben oder RETURN fuer Voreinstellung:
```

Adresse	Inhalt des EPROMs in Hexadezimal	Checksumme
0000	00 00 00 31 FF FF 3E C9 32 00 F0 CD 00 F0 3B 38	068B
0010	D1 21 12 00 19 11 00 F0 01 FF 0F ED B0 C3 00 F0	067D
0020	C3 36 F0 C3 39 F0 C3 40 F0 C3 62 F0 C3 76 F0 C3	0AD5
0030	8D F0 C3 A4 F0 C3 B7 F0 C3 BB F0 C3 C0 F0 C3 33	0C15
0040	F0 C3 33 F0 C3 59 F1 C3 20 F1 C3 33 F0 EF F0 91	0E0D
0050	FC 39 F5 C3 52 FB C3 5B F5 3A 01 F1 E6 03 20 09	088B
0060	DB F1 E6 01 28 FA DB F0 C9 C3 EF F0 3A 01 F1 E6	0B1D
0070	0C CA 39 F0 FE 04 C2 F2 F0 DB F3 E6 01 28 FA DB	0A57
0080	F2 C9 3A 01 F1 E6 03 20 0A DB F1 E6 04 28 FA 79	084B
0090	D3 F0 C9 C3 F5 F0 3A 01 F1 E6 30 CA 62 F0 FE 10	0AA0
00A0	C2 F8 F0 DB F3 E6 04 28 FA 79 D3 F2 C9 3A 01 F1	0AB7
00B0	E6 C0 CA 69 F0 FE 80 C2 FB F0 DB F3 E6 04 28 FA	0BCE
00C0	79 D3 F2 C9 3A 01 F1 E6 03 20 09 DB F1 E6 01 C8	08C0
00D0	3E FF B7 C9 C3 FE F0 3A 01 F1 C9 79 32 01 F1 C9	09C9

Zur Fortsetzung des Programms RETURN betaeligen

```
Anfangsadresse (Datei) eingeben oder RETURN fuer Voreinstellung:
Endeadresse (Datei) eingeben oder RETURN fuer Voreinstellung: $5ff
Startadresse (EPROM) eingeben oder RETURN fuer Voreinstellung:
Bitte Dateinamen oder RETURN fuer Abbruch eingeben: test
```

Bitte Programmierspannung anlegen !

Zur Fortsetzung des Programms RETURN betaeligen

EPROM-Adresse: 05FF

Verify

EPROM-Adresse: 05FF

Weiteres EPROM mit diesen Daten programmieren (Y/N) ?

Bitte Programmierspannung abschalten !

Zur Fortsetzung des Programms RETURN betaeligen

derung durch das Programm angelegt werden! Bei angelegter Programmierspannung darf das EPROM auf keinen Fall entnommen oder eingelegt werden!

Dieses Programm wird auf der Diskette CP/M80-Werkzeuge, die ab Januar verfügbar ist, enthalten sein.

Hauptmenue fuer NDR-PROMER

- 1 EPROM auswaelhen
- 2 EPROM Lesen
- 3 EPROM Programmieren
- 4 EPROM Pruefen
- 5 Ende

Bitte Nummer der gewuenschten Funktion eingeben:

EPROM-Select fuer NDR-PROMER

- 1 EPROM des Typs 2716
- 2 EPROM des Typs 2732
- 3 EPROM des Typs 2764

Bitte Nummer des gewuenschten EPROM-Typs angeben:

DDT benutzen. Auch der DDT verarbeitet Dateien in diesem Format. Den Ablauf des Programmiervorgangs für EPROM zeigt dann Bild 4 auf.

Hinweis: Die Spannung zur Programmierung der Speicher darf erst nach Auffor-

Impressum:

LOOP Zeitung für Computerbauer

Herausgeber: Gerd Graf

Redaktion: Rolf-Dieter Klein, Gerd Graf

Gestaltung und Druck:

Karl-Heinz Rieder, Kempten

Herstellung und Anzeigenverwaltung:

GES GmbH

Magnusstraße 13, 8960 Kempten

Anzeigenpreisliste 1/84

Der Makro-Assembler M80

Seit einigen Monaten wird von GES für den NDR-Computer mit dem Betriebssystem CP/M 2.2 und für den mc-CP/M-Computer ein Makro-Assembler M80 (Bestell-Nr. MACRO80, DM 590,-, incl. MWSt.) angeboten.

Beim CP/M2.2 wird nur ein Assembler-Programm für den 8080 mitgeliefert. Für den Z80-Benutzer, der professionell in Maschinensprache programmieren will, ist der M80 das empfehlenswerte Werkzeug, verbunden mit einem leistungsfähigen Text-Editor, wie z.B. Word-Star.

Alle Z80-Programme von Rolf-Dieter Klein, z.B. das Grundprogramm oder EZASS sind mit dem M80 erstellt.

Macro Assembler Überblick

Microsoft Macro Assembler ist ein leistungsfähiges, flexibles Programmierstellungssystem für die Programmierung in Zilog-Z80- oder Intel-8080-Assembler. Das Assemblerprogrammiersystem umfaßt folgende Komponenten:

1. MS-Macro Assembler: ein Übersetzungsprogramm, das aus Assembler-Quelltexten verschiebbliche Objektprogramm-Module zur Ausführung auf Z80- oder 8080-Mikrocomputern erzeugt.
2. MS-Link™: ein bindendes und verschiebendes Ladeprogramm (linking loader), das nach Eingabe eines einzigen Ladebefehls beliebig viele Programme einspeichern kann.
3. MS-CREF™: ein Programm zur Erstellung von Querverweislisten.
4. MS-LIB™: ein Programm zur Verwaltung und Ausgabe von Routinen in Unterprogramm-Bibliotheken.

Merkmale des Microsoft Macro Assembler-Systems

- Programmierstellung in Z80- oder 8080-Assemblersprache.
- Die Programmierproduktivität kann durch Einführung von Macros, die mehrfach benötigte Assembler-Befehlssequenzen definieren, erhöht werden.
- Mehrere unabhängig voneinander assemblierte, speicherverschiebbliche Objektmodule können gemeinsam in anwenderdefinierte Arbeitsspeicherbereiche eingelagert werden.
- Durch bedingte Assemblierung von Programmtext-Abschnitten können aus einem Assembler-Quellprogramm verschiedene Objektprogrammversionen erzeugt werden.
- Objektprogramm-Module, die mit dem Microsoft Macro Assembler erzeugt wurden, können mit Objektprogramm-Modulen, die von einem COBOL-, FORTRAN- oder BASIC-Microsoft-Compiler erzeugt wurden, verbunden werden.

MS-Macro Assembler

Die Einsatzbreite der Macro Assembler Programmiersprache ist praktisch unbegrenzt. Es können ganzheitliche Assemblerprogramme oder dedizierte Assembler-Unterroutinen, die als Funktionsroutinen dienen sollen, erstellt werden.

Assemblerprogrammierung empfiehlt sich besonders für die Realisierung spezieller Ein-/Ausgabesteuer-routinen (i/o driver routines), für die Erstellung von Anwendungssystemen für Rechnernetze und für virtuelle Speichersysteme.



MS-Macro Assembler bietet annähernd gleiche Möglichkeiten wie die Assembler von Großrechnersystemen, ohne daß Umsetzungs-Geschwindigkeit oder Arbeitsspeichervolumen beeinträchtigt werden. Der Assembler belegt 14 KBytes des Arbeitsspeichers und übersetzt zirka 1000 Assembler-Quellcodezeilen in der Minute.

MS-CREF Querverweisliste

MS-CREF erstellt eine alphabetische Liste sämtlicher Variablen eines Programmes. Zu jeder Variablen werden die Nummern derjenigen Zeilen mitgeliefert, in denen die Variable verwendet, bzw. definiert wurde.

MS-LIB Bibliothekenverwaltung

Mit Hilfe von MS-LIB können Unterprogramm-bibliotheken aufgebaut, verwaltet und ausgegeben werden. MS-LIB überführt Unterroutinen in eine Bibliothek, aus der MS-LINK all diejenigen Routinen entnimmt, die während eines Programmlaufes zur Ausführung aufgerufen werden.

KONTAKTE

Ansprechpartner für NDR-Computer Paket E, Z80-Vollausbau, CPU68K:
Christoph Helmers
Ulbarberstraße 36, 2962 Großefehn
Telefon 049/451423

Suche Erfahrungsaustausch mit anderen NDR-Computer-Anwendern.
Ralf Albrecht Behrends
Am Moosschacht 41, 4630 Bochum 1

Verkaufe NDR-Computer Z80 + 68008, div. Zubehör, auch Einzelplatinen.
Karl Helmut Jung
Rüdigerstraße 107, 5000 Köln 91
Telefon 0221/692682

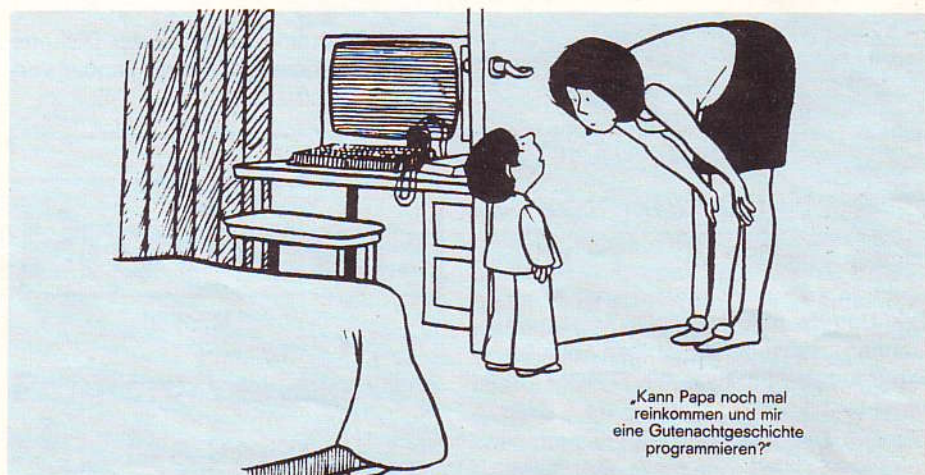
Suche Kontakt zu NDR-Nutzern im Rhein-Main-Gebiet.
K. H. Maisold
Dieburger Straße 77, 6100 Darmstadt
Tel. 06151/45358

Gibt es auch in Österreich, insbesondere Ober-Österreich Interessierte am NDR-Computer? Ich bin aus Linz und möchte das System gerne voll ausbauen.

Markus Kipper
Landstraße 70, A-4020 Linz

Aus: Heft 42 der Zeitschrift „HÖRZU“.

TEXT68K für System 68008 Textverarbeitung und Druckersteuerung für Epson RX80, rel, bootf, 13 Menues, alle Druckparameter, wählb. Block- und Flattersatz, Textoptimierung über Bib-Funkt. in 2 x 2764/250 + Anleitung für 85,- DM bei H.-J. Bessel
Lange Straße 38a, 3304 Wendeburg
Telefon 05302/3585



„Kann Papa noch mal reinkommen und mir eine Gutenachtgeschichte programmieren?“

Theorie und Praxis rund um den NDR-Computer

Mikroelektronik Einführung



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

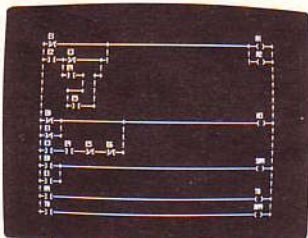
Der Kurs ist auf die HEXIO abgestimmt und ist für alle geeignet, die ihre ersten Schritte in Z 80-Maschinenprogrammierung machen.

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

Der Kurs ist in verschiedene Fachgebiete aufgeteilt und bringt eine Menge Aufgaben, Beispielprogramme und Übungen.

Aus dem Inhalt: Was ist ein Mikroprozessor? * Inbetriebnahme des Computers * Planung von Programmen * Aufbau der CPU * Speicher und Adressen * Datentransfer * Lauflicht * Breakpoints * Hilfsfunktionen * Logo-Elemente * Strukturiertes Programmieren * Label & Call.

SPS-Programmierung



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Dieser Kurs zeigt Ihnen, wie SPS programmiert wird, die Normung, die Anwendungsmöglichkeiten und die verschiedenen Darstellungsarten.

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

Beispielprogramme, Aufgaben und Übungen geben Ihnen die praktischen Erfahrungen und zeigen, wie SPS professionell eingesetzt wird. Nutzen Sie Ihren NDR-Computer für diese moderne Technik voll aus.

Der Kurs ist in folgende Fachgebiete gegliedert: Steuerungstechnik * Digitaltechnik * Methoden zur Beschreibung von Steuerungsaufgaben * Programmierung * Übungen und Tafeln.

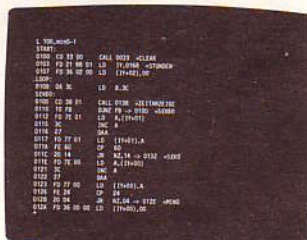
ZEAT-Betriebssystem



Das Betriebssystem beinhaltet in drei EPROMs: Z 80-2-Pass-Assembler, Disassembler, Editor, Debugger, Telefonmodem-Programm, FLOMON 1.5, ausserdem eine ausführliche Dokumentation zum Preis von DM 198,-.

Das Betriebssystem ZEAT benötigt 64-K-RAM (dynamische RAM-Karte). Die EPROMs werden in die BANKBOOT-Karte eingesteckt und sind sofort betriebsbereit. Programmieren Sie Ihren NDR-Computer mit einem Profi-Assembler.

Das Textverarbeitungsprogramm hat volle Bildschirmditierung und kann neben der Programmierung auch zum Textschreiben eingesetzt werden.



Z 80-Assembler-Programmierung

4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Der Kurs ist auf das ZEAT-Betriebssystem abgestimmt und zeigt Ihnen in leicht verständlicher Art, wie der NDR-Computer in Z 80-Assembler programmiert wird, bringt reichhaltig Übungsbeispiele und Anwendungen. Sie werden erstaunt sein, wie leicht diese Art der Programmierung ist. Und Sie lernen, wie man die serielle Schnittstelle bedient und Daten über Telefon übertragen kann.

Die Fachgebiete dieses Lehrgangs sind: Systembeschreibung * Betriebssystem * Programmierung * Testen * Modemprogramm * Listings, Tafeln und Tabellen.

Christiani

Hier abtrennen und im Umschlag einsenden an: Dr.-Ing. P. Christiani GmbH, Techn. Lehrinstitut und Verlag, Postfach 35 691 89, 7750 Konstanz

Bestellcoupon

	Preis je Teil	Gesamtpreis
<input type="checkbox"/> Einführung mit dem NDR-Computer (4 Kursteile)	DM 38,-	DM 152,-
<input type="checkbox"/> Z 80-Assembler-Programmierung (4 Kursteile)	DM 38,-	DM 152,-
<input type="checkbox"/> SPS-Programmierung mit dem NDR-Computer (4 Kursteile)	DM 38,-	DM 152,-
<input type="checkbox"/> Kompakt-Kurs BASIC (angepasst an das RDK-BASIC)	DM 198,-	DM 198,-
<input type="checkbox"/> ZEAT-Betriebssystem (3 EPROMs mit Dokumentation)	DM 198,-	DM 198,-

Name, Vorname

Straße

PLZ, Ort

Datum

Unterschrift

86189

Bild 2 zeigt das komplette Programm. Dabei wurde hier Integer-Arithmetik verwendet, und alle Zahlen sind mit 1000 multipliziert, um Nachkommastellen zu erlauben. Dabei ist c-Real Min mit PMIN bezeichnet und c-Real Max mit PMAX, sowie c-Imaginaer Min mit QMIN und c-Imaginaer Max mit QMAX. Bild 3 zeigt das allgemeine Struktogramm – Sie können so das Programm auch mal in anderen Sprachen ausprobieren. Die Integer-Arithmetik ermöglicht eine sehr schnelle Berechnung, so entsteht das Apfelmännchen innerhalb von ein paar Minuten auf dem Bildschirm. Mit Gleitkommaarithmetik kann es hingegen Stunden dauern.

```

radius = 100      ca. Wert
pmin = -2.25     pmin, pmax, qmin, qmax
pmax = 0.75     bestimmen das
qmin = -1.5      Gebiet
qmax = 1.5
anzahl = 100     Iterationszahl
xkooor=0
ykooor=0
p = (pmax-pmin)*xkooor/511+pmin
q = (qmax-qmin)*ykooor/255+qmin
x = 0
y = 0
k = 0
xneu = x*x-y*y + p
yneu = 2*x*y + q
k = k + 1
x = xneu
y = yneu
wiederhole bis k=anzahl max iteration
oder bis x*x+y*y > radius*radius
gebe Punkt bei xkooor,ykooor aus, wenn
(k mod 2) = 1
ykooor = ykooor + 1
wiederhole bis ykooor > 255
xkooor = xkooor + 1
wiederhole bis xkooor > 511

```

Die Formel: $z^2 + c$ läßt sich in Realteil und Imaginärteil zerlegen, man erhält dann:

$f(z) = x^2 - y^2 + p$ im $f(z) = 2xy + q$ wobei x der Realteil von z und y der Imaginärteil von z ist. Ferner ist p der Realteil von c und q der Imaginärteil von c . Bei der Ausgabe des Punktes wird einfach geprüft, ob das Bit 0 des Zählers k gesetzt ist, wenn ja, so wird ein Punkt ausgegeben. Wenn man über eine Farbausgabe verfügt (z.B. später unsere Farbkarte COL256), dann kann man auch einfach k als Farbcode interpretieren und ausgeben.

Das interessante an der Mandelbrotmenge ist, daß wenn man Ausschnitte vergrößert, immer mehr Details sichtbar werden. Dabei gibt es immer neue Bilder. Interessante Regionen finden sich insbesondere in der Nähe des Apfelmännchens. Dabei muß man aber die Anzahl der Iterationen („Anzahl im Programm“) weiter erhöhen, denn je näher man dem Apfelmännchen auf den Pelz rückt, desto länger dauert es, bis ein Punkt zeigt, ob er innerhalb oder außerhalb des Apfelmännchens zu liegen kommt.

Der Wertebereich vom Apfelmännchen aus Bild 1 ist:

c-Real: -2,25 bis 0,75 c-Imaginaer: -1,5 bis 1,5

Wenn man gröbere Ausschnitte wählt tauchen keine neuen Elemente mehr auf, wohl aber bei kleineren Ausschnitten, z.B.:

c-REal: -0,19920 bis -0,12954 c-Imaginaer: 1,01480 bis 1,06707 oder c-Real: -0,95 bis -0,88333 c-Imaginaer: 0,23333 bis 0,3

Bei zunehmender Vergrößerung macht uns allerdings die Integer-Arithmetik zu schaffen, da sie hier nur mit 16 Bit durchgeführt wird. Wer will, kann das Programm auf 32-Bit-Arithmetik erweitern

und dann auch mal stärkere Vergrößerungen versuchen. Beispiel:

c-Real: -0,74591 bis -0,74448 c-Imaginaer: -0,11196 bis 0,11339

Anstelle das Start-z festzuhalten und das c zu verändern, kann man auch für z unterschiedliche Startwerte wählen und nur für ein bestimmtes c untersuchen. Man erhält dann die Darstellung der sogenannten Julia-Mengen.

Wer tiefer einsteigen will, dem sei das Buch „Schönheit im Chaos, Bilder aus der Theorie komplexer Systeme“ empfohlen, erschienen bei: Forschungsgruppe Komplexe Dynamik.

Universität Bremen, 2800 Bremen 83 ISBN-3-920699-65-3

Satellitenbilder im NDR-Klein-Computer

von Willi Sicking

Wettersatellitenempfang ist eine sehr interessante und lehrreiche Tätigkeit. Zum einen erhält man ständig Informationen über das Wettergeschehen, zum anderen erarbeitet man sich mit der Bildverarbeitung Kenntnisse in einem zukunftssicheren Teilgebiet der Computeranwendung. Daher ist es zu begrüßen, daß für den NDR-Klein-Computer eine Pixelgrafikplatine mit einer Auflösung von 8 Bit pro Punkt erscheint, die eine Bildverarbeitung ermöglicht.

Wettersatelliten

Grundsätzlich ist zwischen zwei Arten von Wettersatelliten zu unterscheiden:

1. umlaufende Satelliten

Die Satelliten der amerikanischen NOAA-Serie und der russischen Meteor-Serie überfliegen die Erde in einer polaren Umlaufbahn mit einer Umlaufzeit von ca. zwei Stunden. Durch die Erddrehung erfassen sie so zweimal täglich die gesamte Oberfläche. Empfangen kann man sie nur, wenn sie im „Sichtbarkeitsbereich“ des Beobachters kommen. Die Feldstärken sind recht hoch, so daß schon preiswerte Empfänger und Antennen ausreichen, um einen Bereich von mehreren 1000 km im Umkreis überblicken zu können. Da ja auch viele Lehrer unter den Lesern sind, möchte ich an dieser Stelle anregen, den Satellitenempfang wie auch in anderen Ländern an den Schulen zu fördern. Nicht nur Wetterkunde, sondern auch Himmelsmechanik, Geographie, Elektronik und Antennentechnik können so auf wirkungsvolle Weise dem Schüler nähergebracht werden.

2. geostationäre Satelliten

Diese Satelliten befinden sich auf einer äquatorialen Umlaufbahn, die jedoch mit 36000 km so gewählt ist, daß der Satellit eine Umlaufzeit von 24 Stunden hat und

somit für den Beobachter still steht. So viel mir bekannt ist, gibt es zur Zeit 4 geostationäre Wettersatelliten, die jeweils 90 Grad versetzt sind. Direkt erreichbar für uns ist nur der Meteosat 2. Zu festgesetzten Zeiten werden aber auch Bilder des amerikanischen GOES East übertragen. Zur Zeit kann man so den Verlauf der heftigen Wirbelstürme in den Staaten verfolgen.

Der Aufbau der Anlage

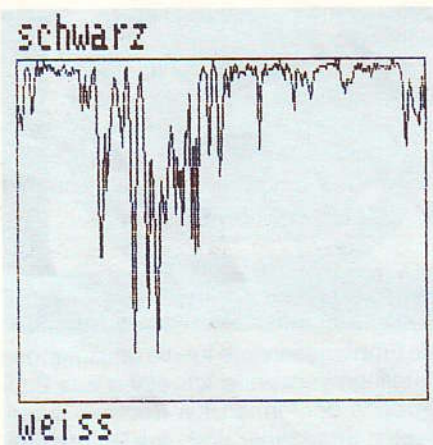
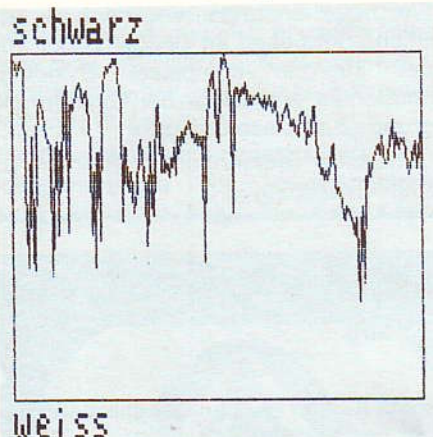
Als Empfangsanlage für die umlaufenden Satelliten verwende ich einen aus Baugruppen der Firma UKW-Technik aufgebauten Empfänger und eine Rundstrahlantenne. Zum Empfang des Meteosat 2 wird vor dem Empfänger ein Konverter geschaltet, der über einen rauscharmen Vorverstärker mit einem 1,2 Meter Parabolspiegel verbunden ist. Beide Antennen sind unter dem Dach montiert. Am Empfängerausgang liegt ein NF-Signal an, bestehend aus einem 2400 Hz Unterträger und dem aufmodulierten Bildinhalt. Zur Zwischenspeicherung wird ein normaler Cassettenrecorder eingesetzt. Besonders bei flachen Durchgängen der umlaufenden Satelliten ist die direkte Aufnahme mit dem Rechner nicht möglich, da das Störspektrum den Empfang stark beeinträchtigt. Zur Wiedergabe dient ein umgebauter Rank-Xerox Telecopierer und ein digitaler Bildspeicher mit einer Auflösung von 256 x 256 x 64 Graustufen. Dieses Gerät besteht im groben aus einem Demodulator mit AD-Wandler, der Taktaufbereitung aus dem 2400 Hz Unterträger und einem Bildwiederholtspeicher.

Aufnahme mit dem NDR-Klein-Computer

Als ich nun vor ca. einem Jahr von Herrn Klein erfuhr, daß er eine Pixelgrafikkarte

mit eben dieser Auflösung und 256 Farben für den NDR-Klein-Computer entwickelt, habe ich mich entschlossen, den Rechner für die Aufzeichnung der Satellitenbilder einzusetzen. Die Vorteile bei einem Rechnereinsatz sind klar:

Schnelles Laden und Abspeichern von Bildern, automatisches Aufzeichnen, Darstellen eines Satellitenfilms, digitale Filterung,



Gammaentzerrung (Grauwertänderung zur Kontrastverbesserung) und alle Arten von Hardcopies.

Da die COL256 zur Zeit nicht lieferbar ist, verwende ich einen 128kByte großen Speicherbereich als virtuellen Bildspeicher. Die Organisation der Daten ist jedoch die gleiche wie in der COL256, also 256 Punkte pro Zeile. Da nun während der Aufnahme mit dem NDR-Klein-Computer keine direkte Kontrolle des Bildaufbaues möglich ist, werden die Daten am Ausgang des AD-Wandlers im digitalen Bildspeicher abgenommen. Auch werden die hier erzeugten Timingsignale zur Übertragung eingesetzt. Zunächst wartet der Rechner auf die abfallende Flanke des Zeilenstartsignals (Zeilehi, Zeilelo). Dann löst jede abfallende Flanke des Pixeltaktes (Pixello) die Übernahme eines Bildpunktes in den virtuellen Bildspeicher aus. So wird das Bild exakt so im NDR-Klein-Computer abgelegt, wie es auf dem Monitor des digitalen Bildspeichers erscheint. Zur Aussteuerung des AD-Wandlers läßt sich jedoch hervorragend die GDP 64-Grafikkarte einsetzen. Das Programm SCOPE bildet jeweils 256 Datenpunkte nach der Übertragung auf dem Bildschirm ab, so daß man eine Kontrolle des Signalverlaufs hat. Es benutzt zwei Bildschirmseiten, damit ein flimmerfreies Oszillogramm entsteht. Mit SC kann man sich die Daten auch nach der Aufnahme noch einmal vorführen lassen, z. B. um die Auswirkung einer Filterung oder Gammaentzerrung am Signalverlauf überprüfen zu können. Die Routinen B1 und B2 bilden je ein Einbitbild, also schwarz und weiß auf dem GDP 64 Bildschirm ab, so daß damit auch ohne

COL256 eine grobe Kontrolle des Bildes und eine Identifizierung der Meteosatbilder anhand der Datenzeile möglich ist. Das kleine Programm UM testet einen Umkopiervorgang von 64kByte, um die maximale Geschwindigkeit bei einem Satellitenfilm simulieren zu können. Dazu kommen jedoch noch 4 Bankumschaltungen, da der Rechner vom Speicher der COL256 nur 16 kByte „sieht“. Falls die COL256 die Erwartungen erfüllt, ist geplant, mit den für den NDR-Klein-Computer existierenden AD-Wandlern einen Demodulator zu entwickeln, so daß dann beliebige Ausschnitte oder Formate digitalisiert werden können. Ebenfalls in Planung ist ein Farbhardcopyprogramm mit dem OKIMATE 20, falls der 68020 nebst Zutaten noch etwas vom Hobbyetat übrig läßt. In der nächsten LOOP folgt ein Hardcopyprogramm mit neun Graustufen und Gammaentzerrung für den RX80, mit dem man auch beliebige andere Grafiken, z.B. die Fractale von Rolf-Dieter Klein, ausdrucken kann.

Die Baugruppen stammen von folgenden Firmen:

UKW-Technik,
Jahnstraße 14, 8523 Baiersdorf
V. Wraase,
Kronsberg 10, 2300 Altenholz/Kiel
SSB-Electronic,
Panzermacherstr. 5, 5860 Iserlohn

Informationen über Meteosat erhält man von:

ESOC,
Robert-Bosch-Straße 5,
6100 Darmstadt

;Programm zur Aufnahme von Satellitenbildern

```

;
; von Willi Stieling
;
; a:=Aufnahme
; sc:=scope
; b1:=Bild 10000-1ffff
; b2:=Bild 20000-2ffff
; um:=Bild von 10000
; nach 20000

org #b000
schw: dc.b 'schwarz'
      dc.b 0
wei:  dc.b 'weiss'
      dc.b 0
      ds 0
grau: dc.w #60 ;Grauschwelle
      ds 0
data  equ $ffffff30
strobe equ $ffffff31

;-----
a:
lea.l #10000,a6 ;Aufnahme vom Bildspeicher
;Startadresse
zeilehi:
move.b strobe,d0
btst.b #0,d0
beq zeilehi ;warten auf high
zeilelo:
move.b strobe,d0
btst.b #0,d0
bne zeilelo ;warten auf low
clr.l d6 ;clear Zeilenzaehler
;neue Zeile
next:
pixelhi:
move.b strobe,d0
btst.b #1,d0
beq pixelhi
pixello:
move.b strobe,d0
btst.b #1,d0
bne pixello ;Pixeltakt

```

```

move.b data,d0
move.b #fff,d7
sub.b d0,d7 ;invertieren
;Daten zum Speicher
move.b d7,(a6)+
addi.w #1,d6
cmpi.w #fff,d6 ;Zeile =256?
ble next
jsr scope ;Zeile darstellen
cmpa.l #30000,a6 ;Datenspeicher Ende?
blt zeilehi
rts

```

```

;-----
scope:
movea.l a6,a1 ;Bildzeile darstellen
suba.l #100,a1 ;Zeile zurueck
sct:
jsr newpage
jsr dfeld ;Umrandung
move.w #40,d1
jsr calc
jsr smoveto ;1. Punkt
scl:
jsr calc
jsr sdrawto ;Datenpunkte verbinden
cmpi.w #140,d1 ;Zeile fertig ?
blt scl
rts

```

```

;-----Einbitbilder
b1:
lea.l #10000,a0 ;Start 1
jsr einbitbild
rts
b2:
lea.l #20000,a0 ;Start 2
jsr einbitbild
rts
einbitbild:
move.l #1ff,d1 ;x:=512
clr.l d2 ;y:=0
jsr smoveto

```


ausgegeben, in diesem Fall: "Druecken Sie eine Taste". Nun wird die Funktion ci() aufgerufen und der Wert an die Variable "zeichen" zugewiesen. Mit print() schließlich wird ein Text sowie das eingegebene Zeichen, ausgegeben.

Nun zum Assembler-Teil. Die Namen "-init" und "-ci" sind als globale Größen definiert, damit kann der Linker sie später identifizieren und die Adressen in das C-Programm eintragen. Das Unterprogramm "-init" ermittelt mit Hilfe des TRAP-Aufrufs im BIOS die Adresse des Grundprogramms. Diese Adresse wird dann in der Variablen "merker" gespeichert.

Das Unterprogramm "-ci", das von C aus als Funktion aufgerufen wird, ruft nach Retten der Register das Unterprogramm CI im Grundprogramm auf. CI liefert als Ergebnis den ASCII-Code des gedrückten Zeichens im Register D0.L. Dieses Register wird aber auch von C als Standard-Register zur Parameter-Rückübergabe bei Funktionen verwendet, so daß das Zeichen schon im richtigen Register steht. Nach dem Rückspeichern der restlichen Register, kann das Unterprogramm daher mit "rts" beendet werden. Wenn man vergißt das Unterprogramm init() im C-Teil aufzurufen, so wird die Routine "fehler" angesprochen.

Falls man mehr als einen Parameter aus dem Assembler-Unterprogramm in das C-Programm übertragen möchte, so muß ein anderer Weg gewählt werden.

her ein Feld, hier z.B. mit 100 Elementen von Typ char.

Mit gets (buffer) wird eine Eingabezeile eingelesen. Die Prozedur gets(), die in der C-Bibliothek eingebaut ist, verlangt die Adresse des Zeichenpuffers als Parameter, denn auch sie muß Parameter, also Werte zurückliefern. Die Adresse wird aber automatisch erzeugt, wenn man nur "buffer" ohne eckige Klammern angibt. Man könnte auch schreiben: "&buffer(0)". Das "&"-Zeichen bedeutet in C "Adresse von". Danach wird unser Unterprogramm wert() aufgerufen. Es erhält drei Parameter. Die Adresse der Variablen "ergebnis" und die Adresse der Variablen "status". Achtung, nicht vergessen das Zeichen "&" vor die Variablen "ergebnis" und "status" zu schreiben. Obwohl der Inhalt von "buffer" nicht verändert wird, wird sie hier dennoch übergeben, da das Unterprogramm WERTE im Grundprogramm die Adresse benötigt.

Danach wird abgefragt, ob der Inhalt von Status Null war, wenn ja, so lag ein Eingabefehler vor, sonst wird der Wert von "eingabe" ausgegeben. Achtung bei der Formatangabe das "L" nicht vergessen, sonst wird ein falscher Wert ausgegeben ("%Ld und %Lx).

Die Eingabe wird solange wiederholt, bis man den Text "End" eingegeben hat. Nun zum Assembler-Teil, der in Bild 4 und Bild 5 dargestellt ist.

Die Unterprogramme "-init" und "-wert" sind als globale Namen definiert. "-init" kennen wir schon von vorher. In "-wert" wird zunächst der Befehl "link a6,=0" ausgeführt, um in A6 eine Adresse zu erhalten, mit der man an die Parameter herankommt. Dann werden die Register gerettet.

Mit movea.l 8(a6),a0 wird der erste Übergabeparameter vom Stack in das Register A0 geladen. Dort steht dann die Adresse von "buffer". Jetzt wird das Unterprogramm WERT aufgerufen. Es liefert im Register D0.L das Ergebnis der Berechnung und im Register D1.W einen Status-Code. Mit movea.l 12(a6),a0 wird die Adresse des zweiten Parameters, hier "ergebnis" nach A0 geladen. Mit movea.l 16(a6),a1 wird die Adresse des dritten Parameters, also von "status" nach A1 geladen. Da alle Parameter Adressen sind, belegen sie auf dem Stack Langwörter, und daher beträgt die Distanz immer 4 (also 8(a6), 12(a6), 14(a6)). Dabei beginnt der erste Parameter immer bei einer Distanz von 8 vom Inhalt von a6 entfernt.

Mit move.l d0,(a0) wird das Ergebnis in die Variable "ergebnis" gespeichert und mit move.w d1,(a1) wird der Status in die Variable "status" gespeichert. Damit ist das Unterprogramm beendet, man darf nur nicht vergessen, die geretteten Register zurückzuspeichern und mit dem Befehl unlk a6, auch den Stack wieder herzustellen.

```

# Unterprogrammaufruf im Grundprogramm mit Adresseübergabe # |
#include <stdio.h>
EXTERN init;
EXTERN wert;

main:
    LONG ergebnis
    int status
    char buffer[100];
    init();
    do {
        printf("Ausdruck eingeben ");
        gets(buffer);
        wert(buffer, &ergebnis, &status);
        if (status != 0)
            printf("Ergebnis: dezinmal %Ld , sedezimal %Lx, n", ergebnis, &ergebnis);
        else
            printf("Eingabe-Fehler: n");
    } while (strcmp(buffer, "End") != 0);

```

Bild 3 zeigt ein weiteres C-Programm. Diesmal soll eine arithmetische Formel von der Tastatur eingelesen werden und das Ergebnis auf dem Bildschirm in dezimaler und sedezimaler Form ausgegeben werden. Dabei soll das Unterprogramm zur Berechnung des arithmetischen Ausdrucks verwendet werden. Das Ergebnis soll eine Langwort-Variable sein, daher wird sie mit LONG im C-Programm deklariert. Achtung, es kommt bei C auf Groß- und Kleinschreibung an.

Eine Variable Status zeigt an, ob das Ergebnis gültig ist oder nicht. Ferner wird noch ein Zeichenpuffer benötigt, in dem der eingelesene Text, also die Formel, gespeichert werden soll. "buffer" ist da-

```

# Routine zum Berechnen eines arith. Ausdrucks - Assemblerteil |
.global init, wert
init:
    movem.l d3-d7/a3-a6, -(a7)
    move #23, a0
    trap #3 # Adresse Grundprogramm berechnen
    move.l a4, merker
    movem.l (a7)+, d3-d7/a3-a6
    rts

wert: # wert(string, ergebnis, status)
    link a6, #0 # fuer Parameter
    movem.l d3-d7/a3-a6, -(a7)
    movea.l 8(a6), a0 # Adresse des Textes
    move #29, d7 # NERT
    movea.l merker, a1
    movem.l a5, -(a7) # retten ist sicherer
    jsr #920(a1) # Aufruf Grundprogramm
    movem.l (a7)+, a6
    movea.l 12(a6), a0 # Ergebnis-Adresse
    movea.l 16(a6), a1 # Status-Adresse
    move.l d0, (a0) # als Langwort ablegen

```

```

move.w d1, (a1) # Status ist Wort
movem.l (a7)+, d3-d7/a3-a6
unlk a6
rts
fehler: rts # Falls Init vergessen wird
merker: dc.l fehler-920
end

```

```

?
B:wert
Ausdruck eingeben: 24+5
Ergebnis: dezinmal 17 , sedezimal 11
Ausdruck eingeben: -10*(67+1023)
Ergebnis: dezinmal -10900 , sedezimal FFFF026C
Ausdruck eingeben: #100
Ergebnis: dezinmal 256 , sedezimal 100
Ausdruck eingeben: 2101110
Ergebnis: dezinmal 46 , sedezimal 2E
Ausdruck eingeben: @schreite
Ergebnis: dezinmal 922248 , sedezimal E1288
Ausdruck eingeben: 1000000/3
Ergebnis: dezinmal 333333 , sedezimal 51615
Ausdruck eingeben: +fff.
Eingabe-Fehler:
Ausdruck eingeben: End
Ergebnis: dezinmal 0 , sedezimal 0
D)

```

Bild 6 zeigt schließlich noch ein Eingabebeispiel.

Rolf-Dieter Klein
(Fortsetzung folgt)

Vorschau LOOP 6

LOOP 6 enthält unter anderem:
 COL256 - die Farbbaugruppe für den NDR-Computer, Beschreibung, Aufbau, Programmierung.
 HARDCOPY mit dem NDR-Computer
 Die neue, interrupt-gesteuerte Hardcopy-Baugruppe.
 Für Einsteiger: Fortsetzung der Ein-Ausgabe, kleines Programmbeispiel.
 Erscheinung: Dezember 1985

AUS DER TECHNIK

Änderungsanweisung FLO2 r 3

Lösung A: READY-Signal für Head Load herausnehmen.

Lösung C: -CS Signal für Datenbustreiber auf GND legen, -CS mit -RD oder auf DIR legen.

Ausführung A: Verbindung Stecker 2 (8") Pin 22 nach Stecker 4 (5 1/4") Pin 34 einlöten.

Ausführung B: Leiterbahn zwischen IC 1 Pin 2 und Motor On Jumper auf Bestückungsseite auf-

Fehler im Handbuch RAM64/256

Auf Seite 18, letzter Absatz heißt es:

Sie läuft mit LS-Bausteinen, wenn IC18 Pin 5-6 aus der Fassung entfernt (abgebogen) ...

Richtig muß es heißen:

Sie läuft mit LS-Bausteinen, wenn IC8 Pin 5-6 aus der ...

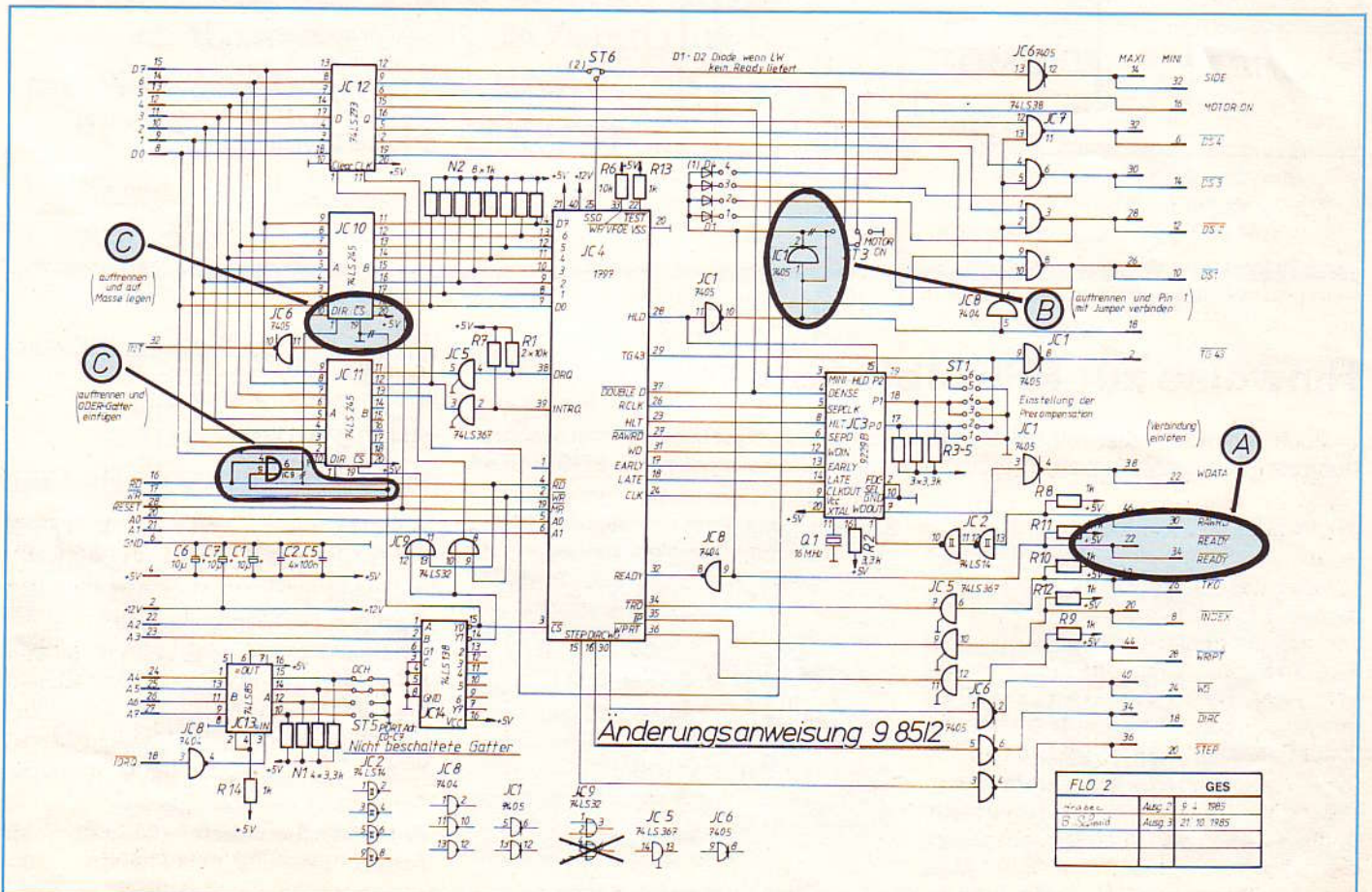
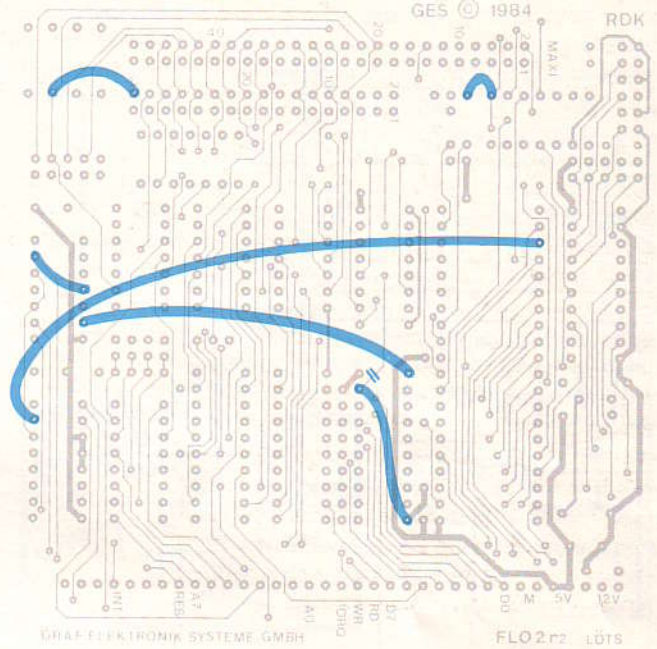
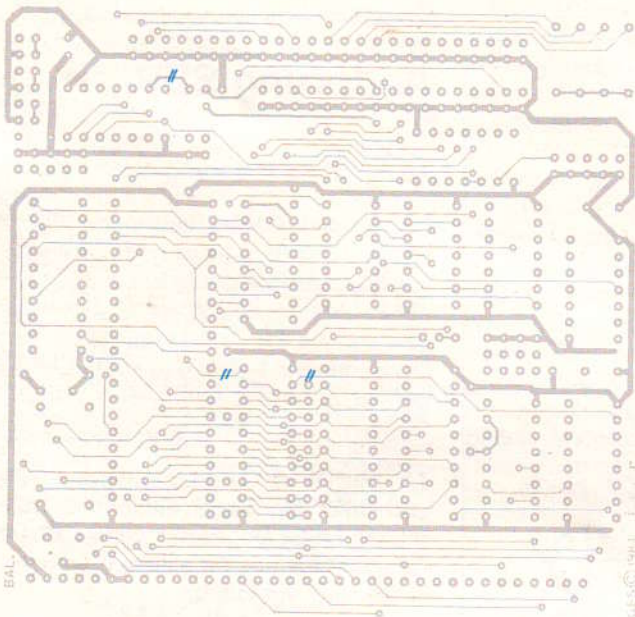
Änderung FLO2 von GS 2 nach GS 3, 9-85-2

Änderungen A, B, C

Grund A: READY-Signal für 5 1/4" Laufwerke ohne Funktion.

Grund B: Motor On über Head Load funktioniert nicht.

Grund C: Fehler beim Schreiben.



trennen. Verbindung IC 1 Pin 1 nach Motor On Jumper (jetzt freier Stift) einlöten.

Ausführung C:

Leiterbahn zu IC 10 Pin 19 direkt am Pin auf Löt- und Bestückungsseite durchtrennen.

Verbindung IC 10 Pin 19 nach IC 10 Pin 10 einlöten.

Leiterbahn direkt an IC 10 Pin 1 (unter Widerstand R 13) auftrennen.

Verbindung IC 4 Pin 4 nach IC 9 Pin 4 einlöten.

Verbindung IC 14 Pin 15 nach IC 9 Pin 5 einlöten.

Verbindung IC 9 Pin 6 nach IC 10 Pin 1 einlöten.

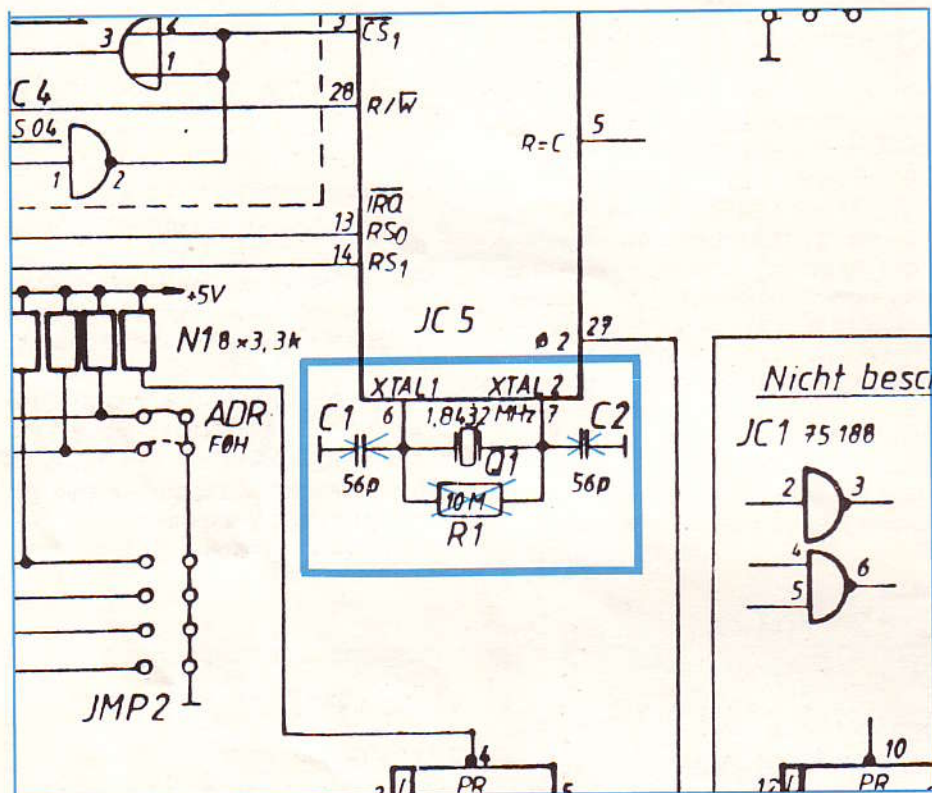
Verbindung IC 14 Pin 15 nach IC 4 Pin 3 einlöten.

Nur bei Fertiggeräten:

Nach Abschluß der Arbeiten Gerätestand von GS 2 nach GS 3 geräten: erhöhen.

Anlage:

Schaltplan Ausgabe 2 mit Änderungen GS 2 nach GS 3 - Bestückungsplan.



Änderung SER

1. Grund der Änderung

Bisher wurde bei der Baugruppe SER der Baustein SY 6551 eingesetzt. Da wir diesen Baustein nicht mehr bekommen, wird der Baustein R 6551 P von Rockwell eingesetzt. Dieser wiederum läuft nicht immer mit dem bisherigen Aufbau. Mit folgender Änderung läuft dieser Baustein sicher:

1.1. Der Quarz Q1 muß entfernt werden und dafür ein Serien-Resonanz Quarz (1,8432 MEC) eingesetzt werden. Dieser Quarz ist bei uns erhältlich, oder schon im Bausatz enthalten.

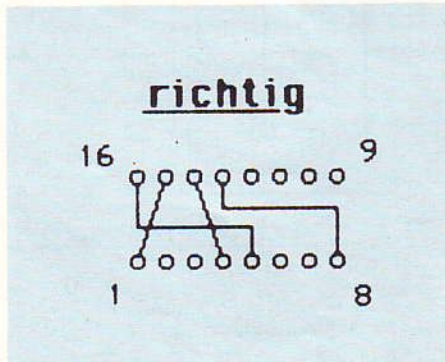
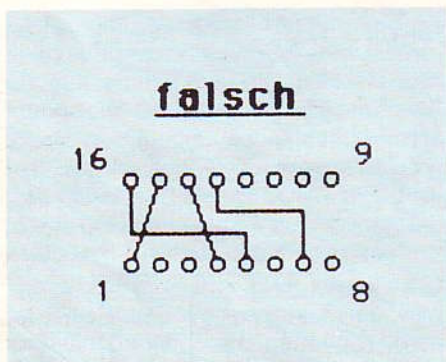
1.2. Die Kondensatoren C1 und C2 und der Widerstand R1 müssen entfernt werden (auslöten oder abwickeln mit dem Seitenschneider).

2. Schaltplanänderung (siehe Seite 2).

3. Keine Layoutänderung!

Hinweise zur PROMER-Baugruppe

Im Buch „Mikrocomputer selbstgebaut und programmiert“, ist in der Promerbeschreibung ein Fehler in der Schaltungsbeschreibung des Kodiersteckers für 2716 EPROM.



Die Verbindung von PIN 13 darf nicht zu PIN 7, sondern muß zu PIN 8 gehen.

CHRISTIANI und GES auf der Hobbytronik Stuttgart

In Stuttgart findet vom 7. bis 10. November 1985 die Hobby-Elektronik 85 statt.

Christiani und GES sind dort vertreten. Sie haben die Möglichkeit, die Christiani-Dokumentationen und Lehrbriefe anzusehen und auch zu bestellen. GES zeigt die neuen Produkte, wie die neue, hochauflösende Farbgraphik COL256 und ACRT.

Besuchen Sie uns Halle 14, Stand 1436

Computerausbildung in Amerika

Bericht aus Phoenix
von Rolf-Dieter Klein

Hierzulande findet der Computer langsam Einzug in die Schulklassen. Dabei sind vorwiegend Gymnasien vertreten, aber auch andere Schultypen.

Wie sieht das in Amerika aus?

Ich hatte das Glück, mehrere Schulen in der Stadt Phoenix, Arizona, besuchen zu können.

Die Computerausbildung beginnt in der ersten Volksschulklasse (Elementary School, first grader). Dabei bekommt die Klasse ca. 1/2 Jahr nach Schulbeginn den ersten Kontakt zum Computer.

Was tut man in der ersten Klasse mit dem Computer? – Man macht Textverarbeitung und Mathematik.

Die erste Stunde: Die Schulklasse wird in den Computerraum geführt, der reichlich mit ca. 30 Computern vom Typ Apple, sowie dazugehörigen Diskettenstationen und einer 56 Mbyte Winchester mit Netzwerk ausgestattet ist. Der Lehrer erklärt

Nach einigen Übungen bekamen die Schüler eine neue Aufgabe. Sie sollten einen kleinen Brief schreiben. Dazu wurde ein Texteditor verwendet. Der Lehrer erklärte noch einige Tasten auf einem großen Modell, und dann gings los. Man hatte den Eindruck, daß die Schüler mehr Schwierigkeiten bei der Rechtschreibung hatten, als mit dem Umgang des Rechners.

So lernen die Schüler rechtzeitig den Umgang mit dem Computer und in den ersten Klassen hauptsächlich Dinge wie Tastatur bedienen und generelle Handhabung. In den ersten Klassen wird sehr viel mit CAI (computer assisted instructions) gearbeitet, noch nicht so sehr mit Programmierung. Dabei kommt der Rechner für den Grammatik- wie auch den Mathematikunterricht zum Einsatz. Diese Form wurde hierzulande oft diskutiert und vielfach verworfen. Einen Vorteil

Die Computerräume sind verglichen mit deutschen Verhältnissen üppig ausgestattet. Manchmal nur eine Rechnerorte, häufig aber verschiedene Rechner, die dann zu Netzen zusammengefaßt sind, um Programme schnell von einer Platte laden zu können. Für einen Distrikt mit mehreren Elementarschulen gab es dann auch noch eine Mikrocomputer-Distrikt-Spezialistin, die die auszubildenden Lehrer betreut und natürlich auch selbst Unterricht gibt. In der Elementary-School und Middle-School verwendet man vorwiegend Home-Computer, während in der High-School Personal-Computer zum Einsatz kommen.

Natürlich brauchen die besuchten Schulen in Phoenix nicht repräsentativ für Amerika zu stehen, doch spricht schon einiges dafür.

Was aber bisher zu kurz kommt, ist, wie auch bei uns, die hardwareorientierte Ausbildung, also die Mikroelektronik.

Denn dafür gibt es auch in Amerika zu wenig geeignete Systeme.

Nicht mehr lange?



Coupon

(ausschneiden und absenden!)

Ja, ich abonniere "LOOP", die Zeitung für
Computerbauer - 5 Ausgaben für DM 20.--
inc. Porto. Scheck oder Schein liegt bei.

Name

Adresse

.....

Graf Elektronik Systeme GmbH
Postfach 1610 8960 Kempten

Bestellung auch per Postkarte oder
bei jeder Bestellung einfach mitbestellen!

GOMOKU

Hinweis der Redaktion:

Der schon lange versprochene Artikel über GOMOKU, dem Hauptpreis des LOOP-Wettbewerbs, wird nun in LOOP 6 erscheinen.

Grund dafür ist der Umfang des Programmes, das uns auch nur handgeschrieben vorlag.

LOOP 6 erscheint noch rechtzeitig vor Weihnachten.

Aufruf für Einsteiger-Software

Dies richtet sich an alle Profis, besonders an die, die mit unserem Einsteiger-Paket (HEXIO) begonnen haben. Schildern Sie uns Ihre Erfahrungen, Ihre Programme und Beispiele – Sie helfen damit uns und vielen neuen LOOP-Lesern!

Zuschriften einfach formlos an die LOOP-Redaktion – eine kleine Überraschung wartet auf Sie.

dann zunächst ein paar grundlegende Verhaltensmaßregeln im Computerraum und die erste Aufgabe. Die Schüler sollen zunächst mit der Zahlentastatur vertraut werden. Dazu hat er ein kleines Programm vorbereitet. Das Programm zeigt drei Würfel mit 0 bis 9 Augen auf dem Bildschirm. Die Anzahl der Augen soll abgezählt und eingegeben werden. Ferner die Summe der Augenzahlen.

Die Schüler setzen sich an den Computer und beginnen mit der Übung. Der Computer stellt immer neue Aufgaben, dann auch einfache Subtraktionsübungen.

hat sie jedoch, wenn man später dann das Programmieren lernt, kann man den Computer schon gut bedienen und braucht nichts mehr über Editor, Drucker, und Speicherung zu erzählen. Ab Klasse 4 bis 6 beginnt dann die Programmierung. Dabei wird meist BASIC als Programmiersprache verwendet. Auf die Frage, ob denn nur BASIC gelehrt wird, sagte mir die Lehrerin: „Nein, natürlich nicht, aber LOGO verwenden wir im Geometrie-Unterricht“. Sprachen wie PASCAL und C folgen dann später, z.B. in der High-School.

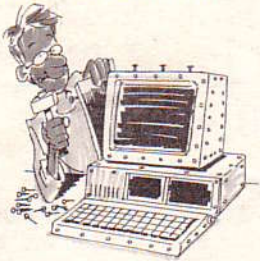
Auf Profis programmiert:



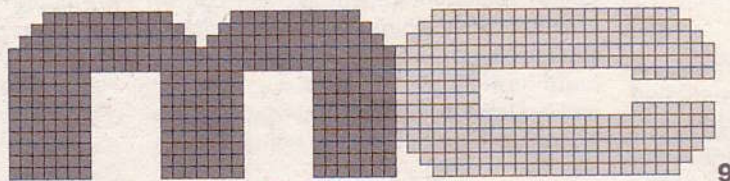
Mit mc kommen Sie jeden Monat auf neue Ideen, wenn Sie selbst programmieren oder Hardware bauen und verändern.



Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.



Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.



Die Mikrocomputer-Zeitschrift

6.50 DM - 55 6S - 7 str. - September 1985

9

68008-Platine für Apple-II

Geknackter Macintosh

Kommunikation mit dem mc-68000-Computer

UCSD-Pascal unter MS-DOS

Erweitertes C-64-Grafikpaket



In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.



Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.

Für DM 6,50 bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.

Ein kostenloses Probeheft schicken wir Ihnen gerne.

Das ist Ihr Gutschein:

mc
Die Mikrocomputer-Zeitschrift

GUTSCHEIN für ein kostenloses Probeheft der mc

Bitte schicken Sie mir die neueste Ausgabe der mc kostenlos an meine folgende Anschrift:

Name

Beruf

Straße

PLZ/Ort

Coupon auf Postkarte kleben, mit 60 Pfennig freimachen und ab damit an

Franzis'

Franzis-Verlag, Abt. Service (L05)
Postfach 37 01 20
8000 München 37