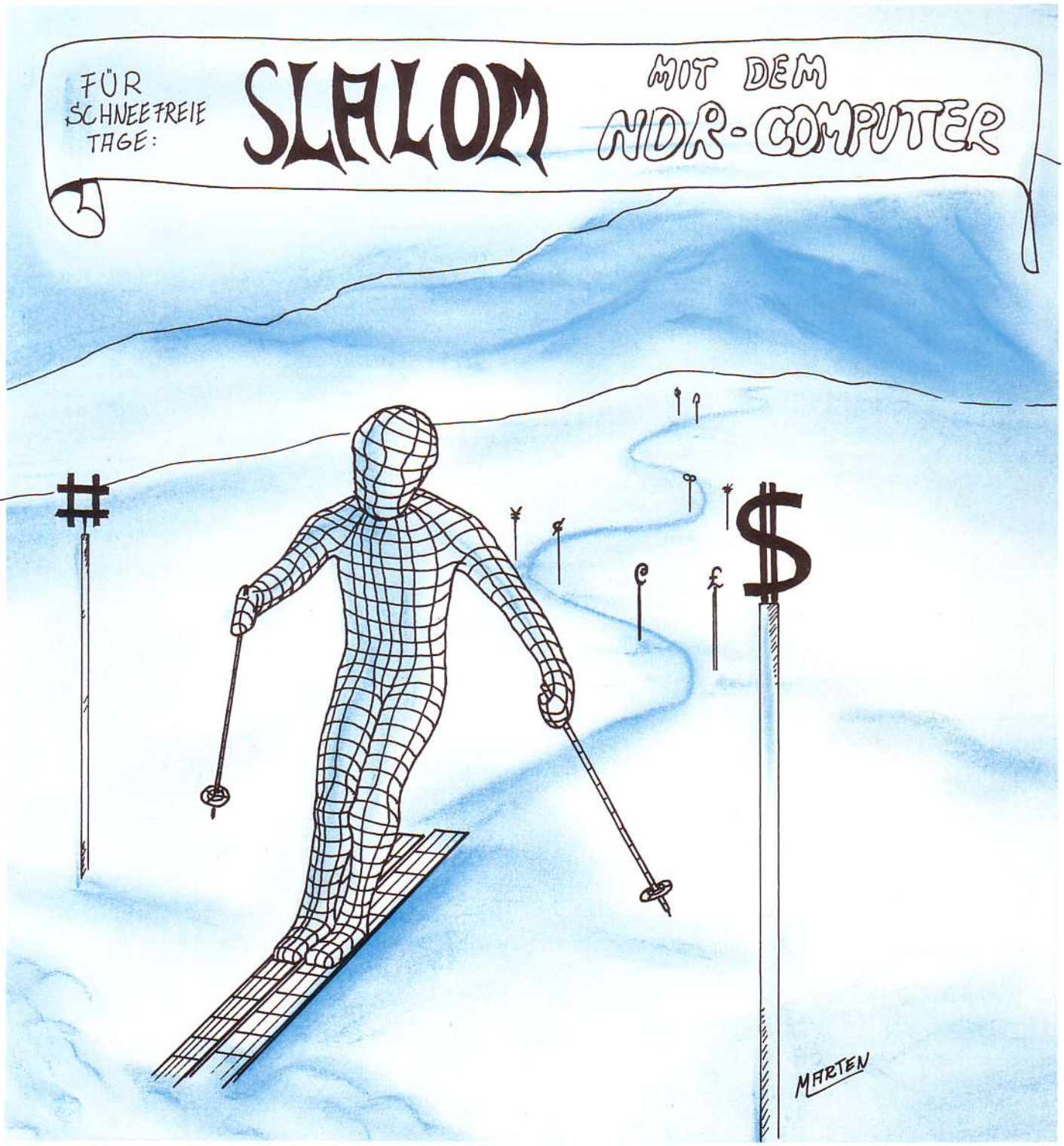
LOOP

26

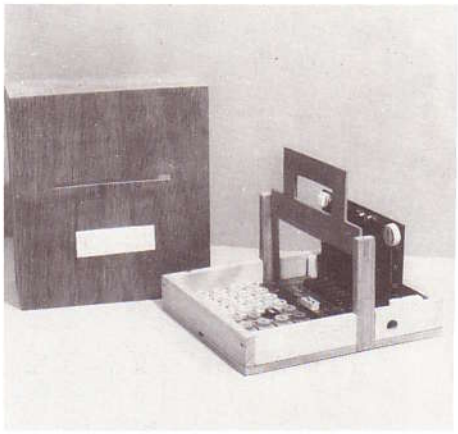
Februar 1991

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter DM 3,50

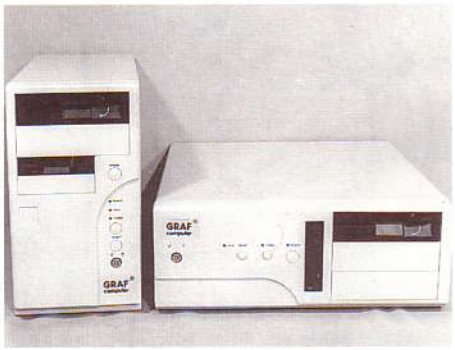


Leitartikel

CPU Z80



Ein maßgeschneidertes Gehäuse für das NDR-Einsteigerpaket



Die neuen Gehäuse des mc modular AT

Neue Gehäuse für den modular AT

CPU 680XX

CPU68ooo

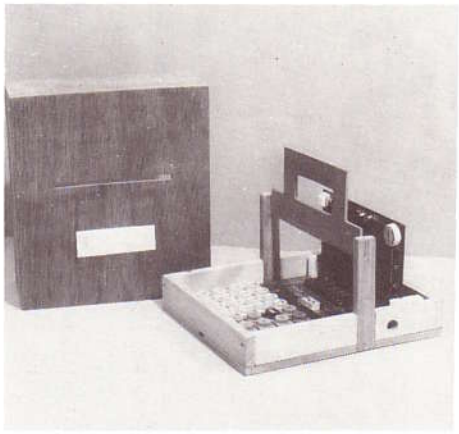
NDR CPU 8088

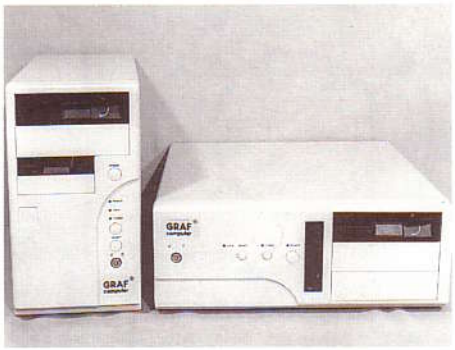
mc-modular AT

Grundlagen

Leitartikel

CPU Z80





Die neuen Gehäuse des mc modular AT

Neue Gehäuse für den modular AT

CPU 680XX

CPU68ooo

NDR CPU 8088

mc-modular AT

Grundlagen

Reinhard Brune

"Slalom" mit dem Einsteiger-Paket

Nicht nur Skifahrer können mit diesem 0,2kB kurzen Spielprogramm ihre Wendigkeit trainieren. Es ist als Anregung für Einsteiger in die Maschinensprache des Z80 gedacht und fordert dazu auf, eigene Ideen in Programme umzusetzen. Schauen Sie sich doch einmal diese oder jene gefundene Teillösung genauer an. Bestimmt fällt auch Ihnen etwas dazu ein...

Die Spielidee

Die achtstellige 7-Segment-Anzeige stellt eine zweiteilige Slalomstrecke dar. Von links kommen zufällig wechselnde Hindernisse (Tore) auf den "Ski" (waagerechter Strich ganz rechts) zu. Mit der Plus- und der Minustaste kann der Ski den Toren ausweichen. Kommen Tor und Ski zur Deckung, gilt das als Fehler. Eine der noch brennenden LEDs (Bonus) wird dann gelöscht. Nach acht Fehlern sind alle LEDs abgeschaltet, das Spiel endet. Die insgesamt durchfahrenen Tore werden gezählt und als erreichte Punktzahl angezeigt.

Natürlich wird die Fahrt immer schneller. Schließlich ist auf Teil 1 der Bahn eine Reaktion nicht mehr möglich. Wer bis dahin noch genügend Bonus und ein bißchen Glück hat, übersteht die rasante Fahrt und gelangt auf den zunächst langsamen Teil 2, auf dem aber die Abstände zwischen den Toren halbiert sind. Das Tempo nimmt auch hier wieder zu. Irgendwann sind acht Fehler gemacht und das Spiel schließt mit der Punkbewertung ab. Über die Änderung der Parameter läßt sich der Schwierigkeitsgrad an die steigende Kondition anpassen.

Das Programm setzt sich aus folgenden Teilen zusammen:

1)	INIT	(Parameter initialisieren)
2)	LEDS	(Bonus anzeigen)
3)	TAST	(Tastatur abfragen)
4)	LAGE	(Stellung der Ski neu bilden)
5)	TEMPO	(Fahrttempo erhöhen)
6)	KOLL	(Fahrfehler registrieren)
7)	ENDE	(Punktzahl anzeigen, wenn Bonus verbraucht)
8)	KOMBI	(Ski- und Torbild kombinieren)
9)	TRANS	(Tore rücken weiter)
10)	ABST	(Abstand einhalten, ändern)
11)	ZUF	(neues Tor bilden, Schleife nach LEDS)

Verwendete ROM-Routinen

Es ist oft vorteilhaft, im ROM bereits vorhandene Routinen von Anwenderprogrammen aus aufzurufen. Die Schnittstellenbedingungen zum Übergang auf solche ROM-Routinen sind dann genau einzuhalten.

Dieses Spielprogramm greift auf drei Routinen zurück:

0033	CLEAR	Anzeige löschen
0009	ANZEIGE	Anzeige und Tastaturabfrage
0021	PRTDEZ	Hex-Dez-Umwandlung

Die Routine CLEAR benötigt keine Übergabedaten, sie löscht das Anzeigefeld.

ANZEIGE bringt den Inhalt des Anzeigespeichers (8000h bis 8007h) einmalig auf das Anzeigefeld und übergibt den gefundenen Tastencode im Akku. War keine Taste gedrückt, wird FFh übergeben. Die 7-Segment-Zeichen sind so lange sichtbar, wie ANZEIGE z.B. in einer Schleife ständig aufgerufen wird.

PRTDEZ schließlich wandelt die im HL-Register befindliche Hex-Zahl in eine mehrstellige Dezimalzahl um und legt sie mit der Einerstelle in der Adresse ab, auf die das IX-Register zeigt.

Parameter und Variable

Parameter	Variable	Bedeutung, (Normalwert)
8107h	81B5h	Anfangs-ABSTAND zwischen den Zeichen, (02h)
	81CAh	ABSTAND-Zähler, Maximalwert, (02h Teil 1 oder 01h in Teil 2)
8108h	DE	PUNKTE-Zähler, Anfangswert 0000h
810Fh	8121h	TEMPO-G Spieltempo grob, (05h)
8114h	8123h (8Fh)	TEMPO-F Spieltempo fein,
81CBh		LED-Bild, (FFh für "alle ein")
	8148h	TAST-Zeitschleife, (FFh)
	81CCh	Zeiger auf gültiges SKI-Bild, Lowbyte
81CDh		SKI-Bild für "Oben", (FEh)
81CEh		" " " "Mitte", (BFh)
81CFh		" " " "Unten", (F7h)
	8006h	aktuelles TOR im Anzeige feld, Platz 7
	8007h	KOMBI, Bild aus Überlappung von SKI und TOR

Die Teilprogramme

1) INIT (Parameter initialisieren)

Nur zu Spielbeginn ist INIT aktiv. Es löscht das Anzeigefeld und definiert ABSTAND, PUNKTE, TEMPO-G, TEMPO-F und das Anfangs-LED-Bild. Die Parameter ABSTAND, TEMPO-G und -F sind hier veränderbar.

8100	CD 33 00	CALL "clear" 7-Segmentanzeige löschen
8103	21 B5 81	LD HL, 81B5h Variable ABSTAND
8106	36 02	LD (HL), 02h ablegen
8108	11 00 00	LD DE, 0000h PUNKTE-Zähler auf Null setzen
810B	21 21 81	LD HL, 8121h Variable TEMPO-G in 8121h
810E	36 05	LD (HL), 05h ablegen
8110	21 23 81	LD HL, 8123h Variable TEMPO-F in 8123h
8113	36 8F	LD (HL), 8Fh ablegen
8115	21 CB 81	LD HL, 81CBh LED-Bild, "alle ein" in
8118	36 FF	LD (HL), FF 81C6h ablegen

Übergabe:
Die 7-Segment-Anzeige ist gelöscht, die Parameter TEMPO und ABSTAND sind definiert, DE enthält den PUNKTE-Stand Null, HL zeigt auf das 1. LED-Bild.

2) LEDS (Bonusstand anzeigen)

Das in 81CBh abgelegte aktuelle LED-Bild gelangt über den Akku nach Komplementierung auf die LEDs (Adresse 02h)

	LOOP LEDS von 81C7h		Einsprung für nächsten Schritt.
811A	3A CB 81	LD A,(81CBh)	Aktuelles LED-Bild holen,
811D	2F	CPL	für LEDs komplementieren,
811E	D3 02	OUT 02	an LEDs ausgeben.

Übergabe:
Die LEDs zeigen den aktuellen Bonus, sonst wie oben.

3) TAST (Tastatur abfragen)

INIT hatte die Schleifenvariablen TEMPO-G und TEMPO-F gesetzt, hier werden sie im Spielverlauf abwärtsgezählt. Die Rücksprungziele von LOOP-G und -F sind 8122h und 8124h.

Die ROM-Routine ANZEIGE bringt den Bufferinhalt (8000h bis 8007h) auf die Anzeige und fragt die Tastatur ab. Im Akku wird der Tastencode übergeben, 5Dh für "+" und 5Eh für "-". Aus 5Dh wird 01h (+1d), aus 5Eh wird FFh (-1d) im Akku gebildet und in LAGE zur Steuerung des SKI-Bild-Zeigers genutzt.

Ist keine Taste gedrückt worden, überspringt das Programm den Teil LAGE.

8120	06 05	Ld B, 05	Variable TEMPO-G im Register B
		LOOP-G von 8155h	
8122	0E FF	LD C, FF	Variable TEMPO-F in Register C
		LOOP-F von 8152h	
8124	CD 09 00	CALL ANZEIGE	holt Tastencode in Akku
8127	FE 5D	CP 5D	Plus-Taste?
8129	20 04	JR NZ, +4d	nein, Sprung nach 812Fh
812B	3E FF	LD A, FFh	ja, "-1" in Akku
812D	18 06	JR, +6d	Sprung nach LAGE
812F	FE 5E	CP 5E	Minus-Taste?
8131	20 1C	JR NZ, +28d	nein, Sprung nach RÜCK
8133	3E 01	LD A, 01h	ja, "+1" in Akku

Übergabe:
BC enthält TEMPO-G u. -F, DE die PUNKTE, HL zeigt auf das aktuelle LED-Bild, A hält die SKI-Zeiger-Differenz für LAGE bereit.

4) LAGE (Lage des Ski neu bilden)

Der bewegliche Ski wird durch einen waagerechten Strich auf dem rechten Anzeigefeld dargestellt. Er kann drei Positionen einnehmen:

Oben: Code FEh	Mitte: Code BFh	Unten: Code F7h
===	...	...
...	===	...
...	...	===

Die zugehörigen 7-Segment-Codes sind in der Tabelle 81CDh, 81CEh und 81CFh abgelegt.
Der Zeiger SKI-Lage (81CCh) enthält das Lowbyte der Tabellenadresse, in der das gerade gültige SKI-Bild steht.

Mit der von TAST im Akku übernommenen Differenz (+1d, -1d) ändert sich der Inhalt des Zeigers durch Addition. Damit wechselt das aktuelle SKI-Bild. Ist die obere oder untere Position erreicht, wird ein Überschreiten der Tabelle verhindert.
Daß die Tastatur nicht zu früh erneut abgefragt wird, stellt die TAST-Zeitschleife sicher. Sie sorgt auch für die ständige Anzeige des Spielfeldes.

8135	21 CC 81	LD HL, 81CCh	SKI-Zeigeradresse in HL, neue
8138	86	ADD A, (HL)	Lowbyte SKI-Adresse bilden.
8139	FE CC	CP CCh	Ist neue Adresse (81)CC?
813B	20 02	JR NZ, +2d	Nein, zum nächsten Test
813D	C6 01	ADD A, 01h	Ja, Korrektur auf (81)CDh
813F	FE D0	CP D0h	Ist neue Adresse (81)D0h?
8141	20 02	JR NZ, +2d	Nein, überspringe Korrektur
8143	D6 01	SUB A, 01h	Ja, Korrektur auf (81)CFh
8145	77	LD (HL),A	Neue Ski-Lage in 81CCh ablegen
8146	C5	PUSH BC	BC auf Stack retten
8147	06 FF	LD B, FF	TAST-Zeitschleifenlänge
8149	CD 09 00	CALL ANZEIGE	
814C	10 FB	DJ NZ, -5d	Zurück, wenn nicht Null
814E	C1	POP BC	BC zurückholen

Übergabe:
Der SKI-Zeiger enthält die Adresse des neuen SKI-Bildes, A ist frei, BC enthält TEMPO-G u.-F, DE die PUNKTE, HL den SKI-Zeiger.

5) TEMPO (Fahrtempo erhöhen)

Die zeitbestimmenden Tastatur-Abfrageschleifen TEMPO-Grob und TEMPO-Fein enden hier. Zunächst wird TEMPO-F um 1 vermindert und, wenn Null, wird TEMPO-G um 1 zurückgesetzt. Ist TEMPO-G noch nicht Null, erhält TEMPO-F erneut den unter 8123h abgelegten Anfangswert.
Ist jedoch TEMPO-G ebenfalls Null, wird der Anfangswert von TEMPO-F um 1 verringert und dann bei 8164h nach 8123h zurückgeladen. Dadurch beschleunigt sich das Spiel bis zur Raserei. Bei geschicktem Ausweichen ist die TEMPO-F-Vorgabe in 8123h bereits Null, obwohl noch Bonus-LEDs leuchten. Dann öffnet sich bei 815Eh der Streckenteil 2, d.h., ABSTAND wird auf 1 (Zeichenabstand Null) gesetzt.

814F	0D	DEC C	TEMPO-F -1d
8150	20 D2	JR NZ, 8124h	Nicht Null, LOOP F (8124h)
8152	05	DEC B	TEMPO-G -1d
8153	20 CD	JR NZ, 8122h	Nicht Null, LOOP G (8122h)
8155	21 23 81	LD HL, 8123h	Adresse TEMPO-F in HL,
8158	7E	LD A, (HL)	TEMPO-F in Akku,
8159	D6 01	SUB 01h	um 1 veringern. Ist Wert Null?
815B	20 07	JR NZ, +7d	Änderung ABSTAND überspringen
815D	E5	PUSH HL	(HL) = 8123 retten
815E	21 B5 81	LD HL, 81B5h	Adr. ABSTAND in HL f. Strecke 2
8161	36 01	LD (HL), 01h	ABSTAND auf 1d setzen
8163	E1	POP HL	8123h zurück nach HL holen,
8164	77	LD (HL), A	neues TEMPO-F nach 8123h laden.

Übergabe:
Die TEMPO-Schleifen sind durchlaufen, Schleife TEMPO-F ist verkürzt, u.U. ist ABSTAND verringert.
A, BC und HL sind frei, DE enthält die PUNKTE.

6) KOLL (Kollisionsprüfung)

Damit der SKI den Toren ohne Kollision ausweichen kann, kommen als TOR nur Zeichen vor, bei denen ein waagerechter Strich fehlt.
Der Kollisionsfall tritt ein, wenn sich Bildteile von SKI und TOR überdecken. Ein Punktabzug bei den Bonus-LEDs ist die Folge.

Beispiel:

SKI	über	TOR	gefahren
...		===	
===		===	<— ergibt hier Kollision.
...		...	

Bitweise gesehen:

BEh in TOR,	1011 1110,	invertiert,	0100 0001,	im Akku und
BFh in SKI,	1011 1111,	invertiert,	0100 0000,	im B-Register
ergeben nach AND B			0100 0000	im Akku.
mit Kollision in Bit 6			7654 3210	

(HL) zeigt auf die SKI-Lage-Adresse. Dort liegt das aktuelle SKI-Bild. 8006h (Anzeigefeld, Platz 7) enthält das letzte TOR-Bild

vor dem Ski.
Bei Kollision ist das Zero-Flag nicht gesetzt. Mit SLA (HL) wird das im HL-Register adressierte aktuelle LED-Bild linksrotiert und zurückgeladen. Das Zero-Flag überprüft auch hier, ob die rotierte Bitkombination (der Bonus) Null ist. Wenn ja, endet das Spiel im anschließenden Programmteil ENDE, andernfalls - auch wenn keine Kollision stattfand - springt das Programm nach KOMBI.

```
8165  21 CC 81   LD HL, (81CCh)   SKI-Bild-Zeiger in HL,
8168  6E         LD L, (HL)       aktuelle Bild-Adresse in HL,
8169  7E         LD A, (HL)       SKI-Bild in Akku,
816A  2F         CPL              SKI-Invers bilden,
816B  47         LD B, A          in B bereitstellen.
816C  3A 06 80   LD A, (8006h)    Aktuelles TOR in Akku holen,
816F  2F         CPL              TOR-Komplement bilden,
8170  4F         LD C, A          in C bereithalten für KOMBI.
8171  A0         AND B            Kollision?
8172  28 21      JR Z, +33d       Nein, Sprung nach KOMBI.
8174  21 CB 81   LD HL, 81CBh     Ja, LED-Adresse in HL
8177  CB 26      SLA (HL)         Linksrotieren, Rückladen.
8179  20 1A      JR NZ, +26d      Nicht Null, Sprung nach KOMBI
```

Übergabe:
Keine Kollision:
A ist frei.
B enthält das aktuelle invertierte SKI-Bild und
in C liegt das aktuelle invertierte TOR-Bild.
DE enthält PUNKTE, HL die aktuelle SKI-Adresse.
Kollision:
HL enhält die LED-Adresse, sonst wie oben.

7) ENDE (erreichte Punktzahl anzeigen, LEDs löschen)

Der Programmteil ENDE wird aktiviert, sobald KOLL die Bonuspunkte auf Null gesetzt hat. Mit 8180h wird das letzte LED gelöscht.
CALL CLEAR löscht auch die Anzeige. Mit PRTDEZ wird PUNKTE in Dezimalform im Buffer abgelegt und mit der ANZEIGE-Schleife dauernd sichtbar gemacht. Ein neues Spiel ist mit einem beliebigen Tastendruck zu starten.

```
817B  CD 33 00   CALL CLEAR   Anzeige löschen,
817E  3E FF      LD A, FFh    LEDs ausschalten,
8180  D3 02      OUT 02h
8182  62         LD H, D      PUNKTE von DE nach HL
8183  6B         LD L, E      übertragen,
8184  DD 21 07 80 LD IX, 8007h Einerstelle nach IX laden,
8188  CD 21 00   CALL PRTDEZ  Hex-Dez-Umwandlung und
                 LOOP EENDSCHLEIFE
818B  CD 09 00   CALL ANZEIGE Anzeige der PUNKTE.
818E  FE FF      CP FF        Ist eine Taste
                              gedrückt?
8190  C2 00 81   J NZ, 8100   Ja, neues Spiel ab INIT
8193  18 F6      JR -10d      Nein, LOOP END-SCHLEIFE
Programm-Ende
```

8) KOMBI (Ski- und Tor-Bild kombinieren)

Aus dem Ski- und dem letztem Tor-Bild ergibt sich das KOMBI-Bild für die Einerstelle des Anzeigefeldes wie folgt:
Aus C wird das TOR (invers) in den Akku übernommen und mit SKI (ebenfalls invers, in B) geODERt. Im Akku befindet sich dann das inverse KOMBI, das komplementiert und im Anzeige-Buffer unter 8007h abgelegt wird. Der nächste Aufruf von ANZEIGE macht es sichtbar.

```
8195  79         LD A, C          Inverses TOR in A holen,
8196  B0         OR B             inverses KOMBI in A bilden,
8197  2F         CPL              komplementieren und
8198  32 07 80   LD 8007h, A      in Anzeigebuffer ablegen.
```

Übergabe:
A, BC und HL sind frei, DE enthält PUNKTE.

9) TRANS (Tore rücken auf)

Das TOR-Bild aus 8006h (Platz 7 des Anzeigefeldes) ist in das KOMBI-Bild in 8007h (Platz 8) übergegangen. Damit ist Platz 7 frei und die Plätze 1 bis 6 können aufrücken nach 2 bis 7.
Das geschieht mit dem Blocktransfer-Befehl LDDR.
Danach ist der Platz 1 (8000h) frei für ein neues Zeichen.

```
819B  D5         PUSH DE          PUNKTE auf Stack retten
819C  21 05 80   LD HL, 8005h     Erste Quelle,
819F  11 06 80   LD DE, 8006h     erstes Ziel und
81A2  01 06 00   LD BC, 6d        Anzahl der Bytes zum
81A5  ED B8      LDDR             Blocktransfer.
81A7  D1         POP DE           PUNKTE nach DE zurückholen
```

Übergabe:
Wie bei KOMBI

10) ABST (Abstand einhalten, ändern)

Im ersten Teil der Abfahrtstrecke:

Die Tore sind mit einem Abstand von mindestens einem Zeichen gesteckt. Auf den freien Platz 1 der Anzeige muß also entweder ein neues Zeichen oder ein Space gesetzt werden. Hierzu wird die Variable ABSTAND in 81CAh benutzt. Im Spielverlauf enthält sie 02h oder 01h. Im B-Register wird sie um 1 decrementiert.

Ist das Ergebnis 01h, wird es nach 81CAh zurückgeladen. Dann gelangt ein Space (FF) über den Akku nach Patz 1 (8000h).
Ist das Ergebnis 00h, lädt 81AFh für den nächsten Durchgang eine 02h nach 81CAh zurück. Bei ZUF wird ein neues Zeichen gebildet.

Im zweiten Teil der Abfahrtstrecke:

Wenn der TEMPO-Anfangswert 00h erreicht hat, beginnt der zweite Teil der Strecke. Hierzu wird ABSTAND (81CAh) von 02h auf 01h gesetzt. Das DEC B in 81ACh erbringt dann immer 00h, 81AFh lädt stets die 01h nach 81CAh zurück, ZUF bildet das neue Zeichen.

```
81A8  21 CA 81   LD HL, 81CAh     ABSTAND in B
81AB  46         LD B, (HL)       holen (02h oder 01h),
81AC  05         DEC B            um 1 erniedrigen.
81AD  28 05      JR Z, +5d        Wenn Null, Sprung nach 81A
81AF  70         LD (HL), B       Nicht Null, rückladen
81B0  3E FF      LD A, FFh        Leerzeichen in Akku nehmen
81B2  18 0F      JR, +15d         ZUF überspringen
81B4  36 02      LD (HL), 02h     alten Wert ABSTAND erneuern
```

Übergabe:
DE enthält PUNKTE, alle anderen Register sind frei.

11) ZUF (Zufallszahl bilden, Lücke auffüllen)

Der Umfang von Auswertung und Zeichenerzeugung bleibt bei Zeichen mit nur waagerechten Strichen vertretbar.

Es ist damit ein Satz von 8 Zeichen möglich:

Bildsegmente	TOR-Bilder		und	Codes			SPACE	Sonderfall
(a)	===	===	...	...	...	===	...	===
(g)	...	===	===	===	...	...	...	===
(d)	...	...	...	===	===	===	...	===
	FEh	BEh	BFh	B7h	F7h	F6h	FFh	B6h

Dazu die Bitkombinationen:
FEh 1111 1110
BEh 1011 1110
BFh 1011 1111
B7h 1011 0111
F7h 1111 0111
F6h 1111 0110
FFh 1111 1111
B6h 1011 0110

(49h) 0100 1001 (erforderliche Filter-Bits)

Als Quelle für die gewünschte Zufallszahl dient das Refresh-Register R. Es zählt ohne Unterbrechung von 00h bis FFh.

Der "Zufall" unterliegt zwei Einflüssen: dem Zeitpunkt des Auslesens von R in den Akku, (die Regelmäßigkeit wird durch das Tastendrücken gestört) und der sich mit jedem Zeichendurchlauf ändernden Variablen TEMPO-F, die zur Refresh-Zahl addiert wird.

Aus den so entstehenden 256 möglichen Bitkombinationen läßt sich mit AND 49h der Zeichensatz herausfiltern:

Die Refresh-Zahl	RRRRRRRR	(Die Buchstaben
addiert mit TEMPO-F	TTTTTTTT	können 0
ergibt als "Zufall"	ZZZZZZZZ	oder 1 sein)
Nach Filtern mit AND 49h	01001001	
bleiben als Rest	0E00E00E	
und nach Ivertieren	1X11X11X	

Die Segmente eines 7-Segment-Elementes leuchten, wenn die entsprechenden Bits auf 0 gesetzt sind und wenn Plus an der gemeinsame Anode anliegt.

Das heißt für 1X11X11X:
1 leuchtet nie und X nur, wenn es 0 ist.
Dem 3-Striche-Zeichen B6h kann man nicht ausweichen. Dieser Sonderfall veranlaßt eine erneute Auslosung.

Das neue TOR-Zeichen gelangt auf Platz 1 des Anzeigefeldes und in DE wird die PUNKTE-Zahl verdient heraufgesetzt. Zum Schluß springt das Programm für den nächsten Schritt zurück nach LEDS.

81B6	ED 5F	LD A, R	Refresh-Zahl in Akku
81B8	21 23 81	LD HL, 8123h	HL adressiert TEMPO-F
81BB	86	ADD A, (HL)	Zufallszahl bilden
81BC	E6 49	AND 49h	Bitkombinationen ausfiltern
81BE	FE 49	CP 49h	Ist Zeichen (inv. B6h?)
81C0	28 F4	JR Z, -12d	Ja, neu auslosen.
81C2	2F	CPL	Nein, invertieren und
81C3	32 00 80	LD (8000h), A	in Anzeigebuffer ablegen.
81C6	13	INC DE	PUNKTE um 1 erhöhen
81C7	C3 1A 81	JP 811Ah	LOOP LEDS

Übergabe:
Bis auf DE mit PUNKTE sind alle Register frei.

Allen Einsteigern in die Maschinensprache - insbesondere den Lehrgangsteilnehmern der DBP - viel Freude und Erfolg bei der Slalomfahrt!

Jetzt lieferbar

JADOS V 3.5

Nach der Ankündigung der Leistungsmerkmale des neuen JADOS in der LOOP 24 sind wir nun endlich in der Lage Ihnen dieses Programm anzubieten. JADOS V 3.5 setzt das Vorhandensein des Grundprogramms V 6.2x voraus, da es dessen SCSI-Routinen mitbenutzt.

Mit im Lieferumfang enthalten ist ein völlig neu überarbeitetes zweiteiliges Handuch in gewohnter Ausführlichkeit.Zur Freude für den Systemprogrammierer ist jetzt ein eigenes Handbuch alleine den JADOS-Unterprogrammen (TRAPs) gewidmet. Für Umsteiger bieten wir der bewährten Updateservice an. Schauen Sie doch mal rein, in die Umschlagseite dieser LOOP!

In eigener Sache

Sonderaktion: Software für 680xxer

In der letzten Zeit hat sich in Punkto Versions-Anhebung im Softwarebereich zum NDR-Computer einiges getan: Das 68000er Grundprogramm ist in der Version V6.2 verfügbar, JADOS ist aktuell in der Version V3.5 erhältlich.

Die vorausgehenden Versionen sind ausgereift und bewährt, nur unterstützen Sie nicht die allerneuesten Baugruppen des NDR (PROMER2 und SCSI). Hier gibt es für Einsteiger oder Umsteiger die Chance, preiswert sein System auszurüsten. Natürlich steht für JADOS-Kunden auch nach dem Erwerb einer Sonderangebotes jederzeit der Update-Service zur Verfügung.

In beschränktem Umfang geben wir Software (Disketten und EPROMs) mit beträchtlichen Preisreduzierungen an unsere Kunden weiter. Wir denken vor allem bisherige Z80-Anwender, die in die Welt der Motorola-Prozessoren hineinschuppern wollen. Eine einfache CPU68008-Baugruppe, ggf. eine Speichererweiterung und preiswerte Software - fertig. Schauen Sie auch einmal in unsere Sonderangebotsliste hinein - Sie finden bestimmt etwas.

EPROM-Anwender-Software

Wer den neuen GES-Katalog bereits aufmerksam durchstudiert hat, wird feststellen, daß GES nunmehr keine Speicherbaugruppen mit 8 kByte-Bausteinen (ROA 64) unterstützt. Demzufolge wird auch ein Restposten brandaktueller Software auf 8 k-EPROMs preisreduziert angeboten. Pfiffige Leute schlagen bei diesem Angebot zu und brennen die Software mit dem PROMER2 in 27256er EPROMs um.

Dr. Hans Hehl

Ökosystem eines Interpreters

Teil 6: BDOS und BIOS

In dieser Folge wird nun die Anbindung des Basic-Interpreters EHEBAS bzw. HEBAS an die Hardware mittels Betriebssystem beschrieben.

Damit der BASIC-Interpreter HEBAS Zugang zur Außenwelt erhält, also Zeichen empfangen und ausgeben kann, muß eine Verbindung zwischen Interpreter und Hardware vorhanden sein, die sog. System-Software oder das Betriebssystem. Es sind Maschinensprache-Programme, die aber im Gegensatz zu normalen Programmen dem Benutzer meistens nicht zugänglich sind und automatisch ablaufen.

Begriffe:

Die in der Überschrift genannten Begriffe sollen zunächst erklärt werden. BDOS ist die Abkürzung für Basic Disc Operating System. Es sorgt beim Betrieb eines Diskettenlaufwerkes für eine komfortable Dateiverwaltung.

BIOS heißt Basic I/O System, also Ein-/Ausgabesystem, der Begriff Basic bedeutend "grundlegend" und hat nichts mit der Programmiersprache BASIC zu tun. Das BIOS ist hardwareabhängig und muß an den jeweiligen Rechner angepaßt sein. BDOS und BIOS sind Bestandteile eines Betriebssystems.

Bekannte Betriebssysteme sind z.B. MS-DOS (Microsoft Disc Operating System), UNIX, OS-9 oder das ältere CP/M (Control Program for Microcomputer) von Digital Research. Es gibt aber auch Eigenentwicklungen wie das JADOS von K. Janßen für den NDR-Rechner in der 68000-Ausbaustufe oder die verschiedenen FLOMON-Z80-Versionen von R.-D. Klein, um nur einige zu nennen.

A) Ohne BIOS und BDOS:

Die Besitzer der SBC3 und des Eprom-Interpreters EHEBAS sollen diesmal auch zum Zug kommen. Denn auch hier ist ein Mini-Betriebssystem vorhanden, das allerdings gleich in den Interpreter eingebunden ist. Es beginnt bei Adresse 0 und reicht bis zur Adresse 05B2h.

Bild 1 zeigt Ihnen einen Ausschnitt aus dem Anfang des Interpreters EHEBAS. Diese Sprungtabelle entspricht der BIOS-Sprungtabelle, die beim Interpreter HEBAS für CP/M benötigt wird. Sprungtabellen erleichtern Hardware-Anpassungen und Änderungen, z.B. bei Portadressen, da nur einmal in der Tabelle der Wert geändert werden muß.

```
;*********************************************
;* BASIC-INTERPRETER - QUELLENPROGRAMM       *
;* EPROM-BASIC.180     für  NDR-Rechner (RDK)*
;* Z-80-CPU und SBC3   Vers.: E1.1           *
;* (C) RD-Klein Minimum CI,CO fuer SBC       *
;* (C) Dr. Hehl Hans   03.02.1989            *
;*********************************************

        .Z80

SCRBYT  EQU     1AH     ;NDR-Rechner 1AH
CICOTB  EQU     0       ;Sprungtabelle CICO

SPEND   EQU     0BFFOH  ;letztes Byte im RAM SBC3
                        ;ab BFF1 für CLOAD benötigt
RAM     EQU     8000H   ;Beginn von RAM bei SBC3

        ORG CICOTB      ;für EPROM = 0, sonst 100h
                        ;unter CP/M zum Testen
start1:
        jp start        ;Kaltstart
csts1:  jp csts         ;Consolenstatus
ci1:    jp ci           ;CONIN
co1g:   jp co           ;CONOUT

lo1:    jp lo   ; Ausgabe auf Centronix-Drucker
                ; Adresse IOE auf 48h legen

po1:    jp poo          ; Kassette schreiben
ri1:    jp ri           ; Kassette lesen
```

Bild 1: Ausschnitt aus dem EHEBAS-Quellcode ab Adresse 0

Bild 2 enthält die Routine zur Tastaturabfrage, ob eine Taste gedrückt wurde und ein Zeichen somit verarbeitet werden kann.

```
KEYD equ 68h            ;Daten Tastatur

CSTS:                   ;Zeichen da, = OFFh
        in a,(keyd)     ;Portabfrage
        bit 7,a         ;Test, ob Zeichen da
        jr z,csts11     ;wenn nicht, dann
        xor a           ;Akku löschen und
        ret             ;zurück
CSTS11:
        ld a,0ffh       ;also Zeichen da
        or a
        ret
```

Bild 2: Konsolenstatus: liegt ein Zeichen vor ?

Wesentlich komplizierter sind die Programmteile eines Betriebssystems, das über den Grafik-Prozessor EF9366 der Baugruppe GDP64K die Zeichenausgabe durchführt. So sind vier Bildschirmseiten zu berücksichtigen. Ein Befehl wie DRAWTO (siehe Teil 5, LOOP Nr. 24, Bild 2) erfordert auf Betriebssystemebene eine komplexe Ansteuerung der GDP. Genaueres über die GDP ist auch in der Artikelserie "Graphik zu Fuß programmiert" ab LOOP 22 enthalten bzw. im Sonderheft Nr. 204, Mikrocomputer Schritt für Schritt, des Franzis-Verlages in meinem Artikel über den EF9366.

Bild 3 zeigt den noch einfachen MOVETO-Befehlsteil. Die Cursor-Positionen X,Y werden im HL- und DE-Register übergeben. Zuerst muß mit CALL WAITG gewartet werden, bis der Grafikprozessor bereit ist, dann werden die Register der GDP mit den Werten von HL und DE geladen.

Praxistip:

Betreibt man auf der SBC3 den Basic-Interpreter EHEBAS, so steht kein Grundprogramm zur Verfügung um Speicherbereiche anzuschauen. Hier kann man das in Bild 4 enthaltene kleine Basic-Monitorpro

```
MOVETO:    ;HL = X    DE = Y

call waitg          ;warten bis bereit
ld a,h
out (gdp+8),a       ;X-Register, MSB
ld a,l
out (gdp+9),a       ;X-Register, LSB
ld a,d
out (gdp+0ah),a     ;Y-Register, MSB
ld a,e
out (gdp+0bh),a     ;Y-Register, LSB
ret
```

Bild 3: Der Befehl MOVETO und die Hardware-Anbindung bei EHEBAS

gramm verwenden. Man kann Speicherbereiche anschauen und Bytes ändern. Sicher ist das Programm ausbaufähig.

```
10 REM Hex-Monitor
20 CLEAR 100
30 TS$ = "dez.   hex.  "
40 TZ$ = "0   1   2   3   4   5   6   7   8   9   "
50 TI$ = "A   B   C   D   E   F"
60 CLR
70 PRINT"Speicher anschauen    = 1":PRINT
80 PRINT"Speicher aendern      = 2":PRINT
90    INPUT"Zahl";A: IF A>2 OR A<1 THEN 60
100 CLR: PRINT"sedezimal mit & eingeben"
120 PRINT: INPUT"ab Hex-Adr";B : PRINT
130   IF A=2 THEN 230
140 B=INT(B/16)*16
150   INPUT"bis:";C
160 PRINT TS$;TZ$;TI$
170 FOR I = B TO C
180   IF I/16 = INT(I/16) THEN PRINT:ZR=I:GOSUB 320
190   D=PEEK(I):S=D:GOSUB 290
200 NEXT I:PRINT :PRINT:INPUT"nochmal";X$
210 IF LEFT$(X$,1) ="j" THEN 120 ELSE GOTO 60
220 :
230 REM aendern
240 ZR=B:GOSUB 320
250 D=PEEK(B):S=D:GOSUB 290
260    INPUT"hex mit &";F:IF F>255 THEN 60
270 POKE B,F:B = B + 1:GOTO 240
280 :
290 REM Ausgabe
300 PRINT RIGHT$(HEX$(D),2) + "  ";:RETURN
310 :
320 REM Format
330 ZR$=STR$(ZR):PRINT MID$(ZR$,2,6);SPC(8-LEN(ZR$));
340 PRINT HEX$(ZR); "  "; : RETURN
350 END
```

Bild 4: Ein kleiner Basic-Monitor

B) HEBAS und BIOS:

Im folgenden soll nun die hardwareabhängige Zeichenein- und ausgabe beim CP/M-Interpreter HEBAS beschrieben werden.

Unter CP/M enthält das Betriebssystem im BIOS alle Routinen zur Ein- bzw. Ausgabe eines Zeichens und die Disketten-Grundroutinen. Diese verschiedenen Programmteile werden über eine standardisierte Sprungtabelle erreicht.

HEBAS ist ein BASIC-Interpreter für das Betriebssystem CP/M und verwendet daher diese Sprungtabelle. Der Einsprung in das jeweilige BIOS-Unterprogramm erfolgt durch Addition verschiedener Konstanten zu dem in den Speicherstellen 1 und 2 enthaltenen Wert EA03h (Warmstart-Einsprung) bei einem üblichen 60K-CP/M-System.

Das Unterprogramm hierzu befindet sich im Interpreter im Bereich von BIOSCI bis BIOST2 (Adressen Vers. 3.1: 373Fh - 3769h). In diesem Bereich wird eine dem jeweiligen BIOS-Einsprung entsprechende Konstante zu EA03 addiert und zu der neuen Adresse gesprungen. Es ergeben sich somit zwischen Interpreter, BIOS und dem Monitorprogramm die in Bild 5 enthaltenen Beziehungen.

```
1) vom Interpreter zum BIOS:

BIOSWS:   EA03      -> EA03   Warmstart
BIOSCS:   EA03 + 3  -> EA06   Konsolenstatus
BIOSCI:   EA03 + 6  -> EA09   Zeichen von Konsole
BIOSCO:   EA03 + 9  -> EA0C   Zeichen an Konsole
BIOSLO:   EA03 + C  -> EA0F   Zeichen an Drucker
BIOSPO:   EA03 + F  -> EA12   Zeichen an Stanzer
BIOSRI:   EA03 +12  -> EA15   Zeichen vom Leser

2) vom BIOS zum Monitor:

EA03:          ->   C3 F027:  JP MINI
               oder C3 F024:  JP MAXI
EA06:          ->   C3 F012:  JP CSTS
EA09:          ->   CD F003:  JP CI
EA0C:          ->   C3 F009:  JP CO
EA0F:          ->   C3 F00F:  JP LO
EA12:          ->   C3 F00C:  JP PO
EA15:          ->   C3 F006:  JP RI
```

Bild 5: Beziehung Interpreter - BIOS - Monitor

BIOSCI:

Beispielhaft sollen die Einzelroutinen ab BIOSCI (Bild 6) näher beschrieben werden.Je nach BIOS-Funktion wird der jeweilige Sprungleisten-Offset in den Akku geladen und dann werden wie üblich die Register BC, DE und HL gerettet. Zum Low-Byte der Warmstart-Adresse wird der Offset dann addiert und BIOST2 aufgerufen. Dort steht der Sprungbefehl JP (HL), also wird zur in Register HL angegebenen Adresse gesprungen.
So besitzt der BIOS-Aufruf LO (Druckerausgabe) einen Offset von 0Ch, der zu EA03h addiert wird, dies ergibt dann die BIOS-Adresse EA0Fh, also die Adresse der Drucker-Routine.

Monitor:

Damit ist hier nicht ein Bildschirm gemeint, sondern ein Programm, das die Verbindung zwischen BIOS und Hardware knüpft. Außerdem sorgt es nach dem Einschalten des Rechners für die ersten Steuerungsvorgänge, z.B. Initialisierung der Portbausteine.

Unter CP/M muß freier Speicher ab Adresse 100h vorliegen. Monitorprogramme starten ab Adresse 0 zunächst und verschieben sich selbständig an die obere Grenze des Speichers. Beim NDR- und mc-Computer liegt dann das Monitorprogramm ab Adresse F000h und beginnt wiederum mit einer Sprungtabelle, die im Abschnitt 2 von Bild 5 erkennbar ist.

Manche Programmierer glauben ohne diese Sprungtabellen auszukommen und springen direkt in Monitorroutinen. Nach Monitoränderungen ist dann das Chaos bei veröffentlichten Programmen vorgegeben.

Beim NDR-Rechner bzw. mc-Computer enthalten die Monitorprogramme komfortable Routinen zur Überprüfung des Rechners.
Diese stehen sofort nach dem Einschalten zur Verfügung, ohne daß ein Betriebssystem von Diskette geladen muß. So bietet der Monitor FLOMONCG von R. Nahm für den NDR-Rechner eine zusätzliche Fensterverwaltung, Maussteuerung, Statuszeile und viele nützliche Besonderheiten.

```
BIOSCI:                 ;BIOS-Zeicheneingabe
        LD      A,6     ;Offset = 6
        CALL    BIOST1  ;und BIOS-Sprung vorbereiten
        OR      A       ;
        RET
BIOSCO:                 ;BIOS-Zeichenausgabe
        LD      A,9     ;Offset = 9
        JR      BIOST1  ;und BIOS-Sprung vorbereiten
BIOSRI:                 ;BIOS-Reader-Routine
        LD      A,12H   ;Offset = 18
        JR      BIOST1  ;und BIOS-Sprung vorbereiten
BIOSPO:                 ;BIOS-Punch-Ausgabe
        LD      A,0FH   ;Offset = 15
        JR      BIOST1  ;und BIOS-Sprung vorbereiten
BIOSLO:                 ;BIOS-Drucker-Ausgabe
        LD      A,0CH   ;Offset = 12
        JR      BIOST1  ;und BIOS-Sprung vorbereiten
BIOSCS:                 ;
        LD      A,3     ;Offset 3 = Konsolenstatus
BIOST1:                 ;
        PUSH    BC
        PUSH    DE
        PUSH    HL
BIOSWS:                 ;BIOS-Warmstart
        LD      HL,(1)  ;dort Adresse BIOS

;       LD      HL,CICOTB ;bei EHEBAS EQU 0

        LD      E,A     ;Offset nach DE
        LD      D,0
        ADD     HL,DE   ;und dazu addieren
        CALL    BIOST2  ;alles vorbereitet
        POP     HL
        POP     DE
        POP     BC
        RET
BIOST2:                 ;zur BIOS-Sprungleiste
        JP      (HL)
```

Bild 6: Quellcode-Ausschnitt des BIOS-Einsprungs bei HEBAS bzw. EHEBAS

C) HEBAS und BDOS:

Das BDOS hält für Benutzerprogramme die Routinen bereit, die für einen Datentransport von und zur Peripherie (Konsole, Drucker, usw.) sowie zu Massenspeichern benötigt werden.

Allgemein wird mit CALL 5 ins BDOS verzweigt, denn ab Adresse 5 befindet sich der Sprungbefehl zum Anfang des BDOS. Dies ist nur scheinbar ein umständliches Verfahren, aber es ermöglicht verschieden große Speicherbereiche.

Die Nummer der Dienstleistungsroutine (z.B. 15d = Datei eröffnen) wurde vorher in Register C der Z80-CPU abgelegt. In den Registern D und E kann eine Speichera-

```
360C  BDOS13: Rücksetzen des Systems        Nr. 13 (0D)
3613  BDOS14: Laufwerk selektieren          Nr. 14 (0E)
3055  BDOS15: Eröffnen einer Datei          Nr. 15 (0F)
301C  BDOS16: Schließen einer Datei         Nr. 16 (10)
3417  BDOS17: Suche nach erstem Namen       Nr. 17 (11)
35FB  BDOS18: Suche nach nächstem Eintrag   Nr. 18 (12)
34EF  BDOS19: Löschen einer Datei           Nr. 19 (13)
3077  BDOS20: Sequentielles Lesen           Nr. 20 (14)
303A  BDOS21: Sequentielles Schreiben       Nr. 21 (15)
2D5A  BDOS22: Datei erzeugen                Nr. 22 (16)
351C  BDOS23: Datei umbenennen              Nr. 23 (17)
3606  RESET:  Aktuelles Laufwerk melden     Nr. 25 (19)
3072  BDOS26: DMA-Adresse festlegen         Nr. 26 (1A)
```

Bild 7: BDOS-Funktionsaufrufe in HEBAS (Vers. 3.1)

dresse abgelegt werden. Rückmeldungen des BDOS erfolgen über Register A bzw. HL. Je nach BASIC-Befehl erfolgt der BDOS-Aufruf CALL 5 an verschiedenen Stellen im Interpreter HEBAS, z.B. bei den BASIC-Befehlen RESET, LOAD und bei DIR, etc. Die jeweiligen Einsprünge mit ihren BDOS-Funktionen und Labels sind in Bild 7 aufgeführt.

Die Hauptroutinen des Interpreters finden sich bei den Labels BDOS00, BDOS02 und BDOS4. Meistens werden die Register BC, DE und HL vor dem Sprung (CALL BDOS) gerettet und die BDOS-Funktions-Nummer von Register A nach Register C gebracht. Je nach Befehlsart kann z.B. noch ein DMA-Aufruf vorgeschaltet sein, so beim Einsprung BDOS3.

Die BDOS-Funktionen 7 und 8 werden nicht benützt, da beim mc-CP/M-Computer das I/O-Byte eigens abgelegt und ausgewertet wird. Sie entfallen auch beim NDR-Computer.

Nichts Neues:

Unter CP/M 2.2 wurden BDOS-Funktionen bis zur Nr. 40 verwendet, unter CP/M 3.0 kamen weitere Funktionen hinzu, so z.B. die Funktion Nr. 152: Generieren eines FCB's. Den BDOS-Aufruf CALL 5 finden wir auch unter CP/M-86 wieder, dort als CALL 0E0h. Unter MS-DOS heißt es allerdings Interrupt 21h. Die Verwandtschaft mit CP/M zeigt sich in den CP/M-kompatiblen Funktionsaufrufen von MS-DOS.

Vorschau:

Im nächsten Teil dieser Serie werden "Innereien" des Basic-Interpreters HEBAS erläutert. So muß der Interpreter eine Warteschleife absolvieren, bis der Benutzer ein Zeichen über die Tastatur eingibt. Weiterhin sind die verschiedensten Ein- und Ausgabekanäle definierbar, was nicht erst unter MS-DOS möglich war. Dies und auch Optionseinstellungen werden genauer unter die Lupe genommen.

Traurig:

Als Autor einer solchen Serie schreibt man für kargen Lohn und investiert viel Zeit. Da wäre es sehr schön, wenn mal der Leser die Feder ergreifen würde und seine Meinung über diese Artikel und seine Wünsche der Redaktion kundtun würde. Aber vielleicht ist es ein gutes Zeichen, daß es keine Proteste gibt. Seit der Autor einen Lehrauftrag (Praktikum Maschinensprache) an der Fachhochschule München hat, weis er, wie bitter nötig eine Ausbildung in solchen Grundlagen ist (dort bildet man übrigens noch mit der 6809-CPU aus!!). Wie modern ist da der NDR-Rechner, nur eben nicht bekannt.

Literatur:

Feichtinger, Arbeitsbuch Mikrocomputer, Franzis Verlag
Hehl, Mikrocomputer Schritt für Schritt 2, Sonderheft Nr. 204, Franzis Verlag
Plate, Betriebssystem CP/M, Franzis Verlag

Claus Littmann

CP/M80: CCP raus und ZCPR rein

Für einen unbedarften Leser mögen die Abkürzungen in der Überschrift ein spanisches Dorf sein - der eingefleischte CP/M-Anwender ahnt zumindest schon beim Begriff "CCP" was Sache ist: dem altgedienten Befehls-Interpreter des CP/M wird am Stuhlbein gesägt. Unser erfahrener Leser Claus Littman hat einmal etwas "über den Zaun geschaut" und im immer noch existierenden CP/M-Public-Domain Fundus etwas gegraben und dabei ein fast in Vergessenheit geratenes Schnäppchen hervorgekramt: den ZCPR V1.6. Doch lesen Sie selbst, wie CP/M 2.2 im Gebrauchswert mit etwas Initiative aufgemöbelt werden kann:

Der CCP: Standard - aber auch nicht mehr

Wer einige Zeit mit dem Betriebssystem CP/M arbeitet, wird sich bald einige Verbesserungen wünschen.
Es ziemlich lästig, daß beim Auslisten von Dateien immer mit ^S auf interessante Stellen reagieren zu müssen. Das seitenweise Auflisten von Dateien wäre da eine schöne Sache; doch wie bring ich's CP/M bei?

Wenn man mit 2 oder mehr Laufwerken arbeitet, wäre es auch schön, wenn ausführbare Programme nur auf einem Laufwerk sein müßten und die Daten auf dem anderen; wie stellt man das am besten an?

Beim Löschen von Dateien wäre es gut, wenn CP/M die gelöschten Dateien nochmal auflisten würde. Bei Fehlern könnte man dann noch etwas zu retten versuchen; die Realisierung dieses Wunsches wird gleich verraten.

Der ZCPR: Der Elegante

Diese und noch andere bislang unerfüllte Wünsche können leicht erfüllt werden. In der großen Menge von Public-Domain-Programmen für CP/M-80 gibt es eine ganze Gruppe von Programmen, die den CCP, der für alle diese Probleme zuständig ist, ersetzen. Sehr einfach funktioniert das mit dem Programm ZCPR-16.

Individuelle Anpassung

Zunächst müssen im Quelltext die gewünschten Optionen ausgewählt werden. Der Quelltext ist sehr gut kommentiert. Mit etwas Englisch, zur Not mit Hilfe eines Wörterbuches, dürfte dieser Schritt kein Problem darstellen.
Zu beachten ist aber, das manche Optionen von anderen abhängen. Ferner können nicht alle Optionen gleichzeitig genutzt werden, da nicht mehr als 2048 Bytes zur Verfügung stehen. Das Einhalten der Obergrenze sollte im Print-File geprüft werden. Bei Platzproblemen kann z. B. auf die DIR-Funktion verzichtet werden. Hierfür gibt es ja mittlerweile guten Ersatz.

Assemblierung

Wenn alle Optionen wunschgemäß eingestellt sind, kann die ZCPR-Quelle übersetzt werden. Die ursprüngliche Quelle ist für den MAC-Assembler von Digital Resarch ausgelegt worden. Der MAC-Assembler wurde von Digital-Reserach mit dem Betriebssystem CP/M+ ausgeliefert und ist mittlerweile nicht mehr erhältlich. Evtl. kann die Übersetzung auch mit ASM vorgenommen werden. Dann müßen aber

alle Macros im Quelltext expandiert werden.
Die hier sehr viel genutzte bedingte Assemblierung soll ASM ja auch können. Ich habe eine Übersetzung mit ASM aber nicht getestet.
Anmerkung der Redaktion: Wie uns der Autor mitteilte, hat er den Umbau auf den Microsoft M80/L80 mittlerweile ebenfalls durchgeführt.

Einbindung des ZCPR ins BIOS
Wenn alles richtig gelaufen ist, sollte sich die Datei 'ZCPR-16.HEX' auf der Diskette befinden. Jetzt sollte mit SYSGEN80 eine Datei mit dem alten Betriebssystem erzeugt werden. Für alle, die nicht mehr (oder noch nicht) wissen, wie das geht, nochmals die Prozedur in Kurzform:

SYSGEN80 aufrufen, Source auf A, Destination auf <RETURN>, anschliessend mit SAVE 60 CPM60.SYS <RETURN> die Datei CPM60.SYS erzeugen.

Dann DDT aufrufen und CPM60.SYS laden. Jetzt mit IZCPR-16.HEX das Laden der Hexdatei vorbereiten. Die Hex-Datei muß mit einem Offset eingelesen werden. Bei einem normalen NDR-System beträgt dieser Offset 3580h. Man gibt also R3580 ein. Dann wird DDT mit ^C verlassen.

Nun wird wieder SYSGEN80 aufgerufen. Dabei wird die Frage nach der Source lediglich mit <RETURN> beantwortet. Bei Destination wird dann A angegeben. Diese Einbauprozedur entspricht übrigens exakt der in der LOOP Nr. 12 auf Seite 9 beschriebenen Prozedur.

Wer sein BIOS schon mit den Änderungen für einen automatischen Programmstart beim Kaltstart versehen hat, kann den Namen des zu startenden Programms natürlich im Quelltext angeben. Dabei darf natürlich die Längenangabe nicht vergessen werden.

Wer die in der LOOP Nr. 12 angebenen Änderungen noch nicht durchgeführt hat, sollte das bei der Gelegenheit nachholen.

Natürlich - die Version 1.6 von ZCPR nicht die letzte. Ich plage mich derzeit mit den Problemen der Installation von ZCPR2. Hierzu sind jedoch nicht ganz unproblematische Änderungen am BIOS erforderlich. Die hiermit verbundenen Problemen habe ich noch nicht gelöst. Aber auch schon die Version 1.6 bietet schon einigen Komfort, den man dem guten alten CP/M gar nicht zutrauen würde.

Wo gibt es den ZCPR?

Nun bleibt eigentlich nur noch ein Problem: Wie kommt man an ZCPR ran? Das ist eigentlich ganz einfach. ZCPR ist Public-Domain. Er kann also zu Selbstkosten weitergegeben werden. Wer mir also eine vorformatierte Diskette (5.25" oder 3.5") im NDR-80-Spur-Format und einen SAFU schickt, wird die erforderlichen Dateien hoffentlich bald in seinem Briefkasten vorfinden.

Natürlich werde ich auch einige kurze Programme zum Ersatz von verschiedenen residenten Komandos mit kopieren.

Bezugsquelle:
Claus Littmann
Zum Spring 15
3155 Edemissen/Plockhorst

Jost-Reimer Hoof

Berichtigung

Im Artikel "Bei mir piept es" in der LOOP 21, Seite 16 hat sich leider ein Fehler eingeschlichen:

Baut man den Bell-Aufruf in das nachladbare FLOMON32.COM ein, das es bis jetzt nur auf der Quell-Diskette für FLOMONCG gibt, so muß die Sprungadresse, die bei 1A75h eingebaut werden muß, nicht 0239h, sondern 423Dh heißen, weil FLOMON32.COM auf die BankBoot ab 4000h kopiert und gestartet wird.

Jost-Reimer Hoof

Teil 3

Einsatz von FLOMONCG

Fenster öffnen, Fenster schließen ist kein Problem mehr, denn heute stelle ich weitere Spielereien mit FLOMONCG vor.

3. Cursor bewegen

Beim Demo-Programm DEMO-FE3.PAS (Bild 3), das in Pascal geschrieben ist, wird vor dem schon bekannten Hintergrund ein Fenster geöffnet, in das das für dieses Programm gültige Menü geschrieben ist. Dann wird mit der Prozedur "Hauptfeld" ein neues Fenster geöffnet und voll mit Zahlen geschrieben. Mit den Tasten 2, 4, 6 und 8 kann man den Cursor nach allen Seiten bewegen, mit "5" wird an der Stelle gelöscht, an der Cursor steht. Diese Stelle wird mit einem Blank überschrieben. Mit der Taste 7 kann man einzelne Buchstaben an die Cursorposition schreiben. Das ist zwar etwas umständlich, aber trotzdem interessant.

Mit der Taste 1 wird die augenblickliche Cursorposition abgefragt und in die oberste Zeile geschrieben. Mit Taste 3 springt man in ein weiteres Fester, um das Cursorattribut neu zu setzen. Endlich werden mit Taste 9 weitere editortypische Funktionen ermöglicht, wie Zeile löschen und hinzufügen, Buchstaben löschen und einfügen, Löschen bis Zeilenende und Fensterende. Am besten probieren Sie alles aus und vertiefen sich dann in das Listing, das wie die anderen modular aufgebaut ist.

4. Fadenkreuz

Mit dem Demo-Programm DEMO-FE4.PAS (Bild 4) kann man auf spielerische Weise das Fadenkreuz umdefinieren. Dabei werden die im FLOMONCG möglichen Kurzvektoren, die der Grafikprozessor verarbeiten kann, als Information auf dem Bildschirm ausgegeben. Die Eingaben erfolgen in einem eigenen Fenster, damit die Eingabe-Information immer sichtbar bleibt. Für das Verständnis ist es hilfreich, sich auch den Artikel "Der Grafikprozessor EF 9366" von Dr. Hans Hehl im mc- Sonderheft 2 anzusehen.

Will man ein bestimmtes Fadenkreuz-Symbol in einem Programmverwenden, z.B. eine Meßmarke in Form eines "C", so schlage ich folgende Vereinbarung in einem Pascal-Programm vor:

```
write (chr(27), '0', '1ÖTZXP');
```

In einem Z80-Programm müßte der String lauten:

```
DB 27, '0'      ; Fadenkreuzvereinbarung
                ; (0 = Null!)
DB '1ÖTZXP'     ; Stopzeichen
```

Ein kleines Fadenkreuz erhält man durch Eingabe von:

```
'1XXXZZ0^^1^^0ZZ1XXX'   (0 = Null)
```

Man sollte spielerisch an dieses Problem herangehen.

5. Notizbuch

Die Möglichkeit, Bildschirminhalte in ein Notizbuch zu schreiben, um sie in einem anderen Programm zu verwenden, ist eine tolle Sache unseres FLOMONCG. Wie das aus dem Monitor heraus gemacht wird, hat Rüdiger Nahm in seiner Beschreibung unter Punkt 4.2 gut und nachvollziehbar beschrieben.

Welche Möglichkeiten aus FLOMONCG selbst angeboten werden, zeigt das Demo-Programm DEMO-FE5.PAS (Bild 5). Hier werden mehrereFenster geöffnet. In das oberste Fenster wird das Wort "DIR *.*" geschrieben. Dann erfolgt im 1. Schritt das Schreiben des Fensterinhaltes in das Notizbuch. Danach wird der Notizbuchinhalt am Bildschirm angezeigt. Zum Schluß wird die Information in die Konsoleingabe eingeschleust. Wenn man das Programm unter Turbo Pascal mit "R" gestartet hat, wird der Notizbuchinhalt manchmal vor das Programm geschrieben, also in das Programm eingefügt. Es muß dann wieder gelöscht werden, da es ja nur Spielmaterial ist.

Hat man dieses Demo-Programm als COM-File übersetzt und ruft es auf, so wird DIR ausgeführt. Bei meinem System wurde aber nur der erste Eintrag angezeigt. Die weiteren Informationen werden als Programmnamen vom System interpretiert und versucht, solche Programme aufzurufen, was natürlich nicht gelingt. Zur Beruhigung kann ich sagen, daß keine Schäden an Programmen, Disketten und dem System nach dieser Prozedur festgestellt werden konnte. Dieses Programm sollte ja nur zeigen, was mit den Kommandos um das Notizbuch alles möglich ist.

6. Lokal-Modus

Das Demo-Programm DEMO-FE6.PAS (Bild 6) beschäftigt sich mit dem Lokal-Modus. Wird er eingeschaltet, so werden alle Zeichen von der Tastatur direkt ans Terminal geschickt. Jetzt lassen sich vornehmlich Cursorbewegungen durchführen. Die Tasten, die bei mir etwas bewirkten, habe ich als Info mit auf den Bildschirm gegeben.

Den Lokal-Modus schaltet man aus, indem "ESC D x" von der Tastatur eingegeben wird. Wer keine ESC-Taste hat, benutze einfach CTRL-Ä, dann "D" und eine beliebige Taste, z.B. "x".

Eingesetzt habe ich das Demo-Programm, um ASCII-Zeichen-Grafiken zu erstellen, die dann nach Beendigung des Programmes über die Monitor-Notizbuchfunktion gesichert und in ein Editor-Programm eingeschleust wurde. Natürlich geht solche Arbeit auch anders.

7. Monitor-Modus

Mit dem Demo-Programm DEMO-FE7.PAS (Bild 7) kann man den Monitor-Modus ein- und ausschalten. Beim eingeschalteten Monitor-Modus werden keine Steuerzeichen mehr ausgeführt, sondern die Steuerzeichen in Klammer (<>) als Hexwert ausgegeben. Lassen Sie sich das mal von Ihrem Rechner vorspielen. Das Programm sollte als COM-File übersetzt werden, bevor es gestartet wird. Dann kann es zum Abschalten des Monitor-Modus erneut aufgerufen werden.
Ruft man den COM-File auf, so werden fast nur CR und LF als <0D> und <0A> gezeigt. Wenn man genau hinsieht, sind die letzten beiden Zeichen vor dem blinkenden Cursor die Prompt-Zeichen vonCPM.
Zum Abschalten starten Sie DEMO-FE7.COM erneut und könnenzwischen den Steuerzeichen in Klammern den bekannten Text wiederfinden. Geben Sie "2" ein, und der Spuk ist vorbei.

(wird fortgesetzt)

```
PROGRAM DemoFenster3;
        {*************************************************
         *  Testprogramm fuer WINDOW-Technik von FLOMONCG  *
         *  hier: Cursor-Bewegung / Attribut / Loeschmoeglichk. *
         *-------------------------------------------------*
         *  J.-R. Hoof, Heikendorf   * LastUpdate 29.12.1989 *
         *      Tel 0431 - 24 20 70                        *
         *************************************************}

 VAR
  cha   : Char;

PROCEDURE Hintergrund;
VAR
 i   : integer;
 Stri : STRING [80];
 StringGross : STRING [80];
Begin
 writeln;
 writeln (' ', chr(27), ')',    { Invers ein }
          ' Demo - Programm fuer Window - Gestaltung ',
          chr(27), '(');   { Invers aus }
 Writeln;
 Stri := 'N D R - Computer * JoHo - Software (C) *  ';
 FOR i := 1 TO 19 DO
  Begin
   StringGross := Copy (Stri, i, 42-i) + Stri + Copy (Stri, 1, i) ;
   writeln ('  ', StringGross);
  End;
End;

{-------------------------------------------------}

 PROCEDURE OpenWindow        { Fenster oeffnen }
   (Nummer, YGroesse, XGroesse, YAbstand, XAbstand, Kennwert : integer);
                  { Kennwert 1 = unsichtbar, sonst sichtbar }
  Begin
   write (#27, '$O', Nummer, Chr (YGroesse+32), Chr (XGroesse+32),
                  Chr (YAbstand+32), Chr (XAbstand+32));
                  { Window unsichtbar eroeffnen }
   IF Kennwert <> 1 THEN
    write ( #27, '$E');    { Window sichtbar machen }
  End;

{-------------------------------------------------}

 PROCEDURE CR;              { erzeugt Carriage Return }
  Begin Write (chr (13)); End;

 PROCEDURE LF;              { erzeugt Line Feed }
  Begin Write (chr (10)); End;


PROCEDURE CursorLinks;      { erzeugt Cursor links }
  Begin Write (chr (19)); End;

 PROCEDURE CursorUnten;     { erzeugt Cursor nach unten }
  Begin Write (chr (22)); End;

 PROCEDURE CursorOben;      { erzeugt Cursor nach oben }
  Begin Write (chr (5)); End;

 PROCEDURE CursorRechts;    { erzeugt Cursor rechts }
  Begin Write (chr (12)); End;

 PROCEDURE FensterEnde;           { Fenster schliessen }
  Begin write (#27, '$C'); End;

{-------------------------------------------------}

PROCEDURE ZeileNeu;        { Moeglichkeiten von Editor-Funktionen }
 VAR  Wert : Char;
 Begin
  OpenWindow (3, 11, 25, 11, 10, 0);  { oeffnet 3. Fenster }
  Writeln (' Waehle: ');
  Writeln (' 1 = Zeile einfuegen ');
  Writeln (' 2 = Zeile loeschen  ');
  Writeln (' 3 = Zeile bis Ende  ');
  Writeln ('       loeschen  ');
  Writeln (' 4 = Zeichen einfuegen');
  Writeln (' 5 = Zeichen loeschen ');
  Writeln (' 6 = Bildschirm bis ');
  Writeln ('     Ende loeschen ');
  Write  ('  ===>  ');
  Read (KBD, Wert);
  FensterEnde;
  CASE Wert OF
   '1' : Write (Chr (27), 'E');     { Zeile einfuegen }
   '2' : Write (Chr (27), 'R');     { Zeile loeschen }
   '3' : Write (Chr (27), 't');     { Zeile bis Ende loeschen }
   '4' : Write (Chr (27), 'Q');     { Zeichen einfuegen }
   '5' : Write (Chr (27), 'W');     { Zeichen loeschen }
   '6' : Write (chr (27), 'y');     { Bildschirm bis Ende }
  End;                        { loeschen }
 End;

{-------------------------------------------------}

PROCEDURE Menu;
 Begin
  OpenWindow (1, 14, 25, 6, 5, 0);  { oeffnet Fenster }
  Writeln (' Waehle aus:  ');
  Writeln (' 1 = Cursorposition ');
  Writeln (' 2 = Cursor nach unten ');

 Writeln (' 3 = Cursorattribut ');
  Writeln (' 4 = Cursor nach links ');
  Writeln (' 5 = Loeschen an Pos. ');
  Writeln (' 6 = Cursor nach rechts ');
  Writeln (' 7 = Schreiben      ');
  Writeln (' 8 = Cursor nach oben  ');
  Writeln (' 9 = Zeile/Zeichen ');
  Writeln ('     manipulieren ');
  Writeln (' 0 = Ende');
 End;


PROCEDURE Loeschen;        { loescht Zeichen durch Ueberschreiben }
 Begin
  Write (' ');
  CursorLinks;
 End;

PROCEDURE Schreiben;       { Schreibt Zeichen }
 Begin
  OpenWindow (3, 2, 20, 15, 1, 0);
  Write ('Gib Zeichen ein: ');
  Read (KBD, Cha);
  FensterEnde;          { aktives Fenster schliessen }
  Write (Cha);
 End;

PROCEDURE CursorAttribut;   { setzt Cursorattribute }
 VAR  Wert : integer;
 Begin
  OpenWindow (3, 8, 25, 12, 1, 0);  { oeffnet 3. Fenster }
  Writeln (' Gib Attribut ein: ');
  Writeln (' 1 = kein Cursor');
  Writeln (' 2 = Block, blinkend');
  Writeln (' 3 = Block ');
  Writeln (' 4 = Unterstrich, bl.');
  Writeln (' 5 = Unterstrich ');
  Write  ('  ===>  ');
  Read (Wert);
  Write (chr(27), '$', 'C');        { aktives Fenster schliessen }
  Write (#27, '.', Wert - 1);
 End;

PROCEDURE CursorPosition;          { Cursorposition abfragen }
 VAR  xx, yy : Char;
 Begin
  Write (chr(27), '?');
  Read (KBD, yy);
  Read (KBD, xx);

Write (chr(27), '=', chr(32 + 0), chr(32 + 4));
  Write ('Cursor: y = ', (Ord(yy) - 31):2, ',  x = ', (Ord(xx) - 31):2);
  Write (chr(27), '=', yy, xx);

 End;

PROCEDURE Hauptfeld;               { Hauptfeld beschreiben }
```

```
VAR  i : integer;
 Begin
  OpenWindow (2, 15, 40, 4, 35, 0); { 2. Fenster oeffnen  }
  Write (chr (30));                 { Cursor HOME }
  LF;
  LF;
  FOR i := 1 TO 12 DO
   Begin
    Write ('12345678909876543210123456789098765432 1');
    CR;
    LF;
   End;                             { Cursor positionieren }
  Write (chr(27), '=', chr(32 + 7), chr(32 + 17));
  Write ('  *  ');
 End;

{======================  Hauptprogramm  ======================}

Begin
 write (#27, '*');        { Bildschirm loeschen }
 write (#27, '$@');       { alle Windows schliessen + neu initialisieren }
 Hintergrund;             { Hintergrund in Window 0 schreien }
 Menu;
 Hauptfeld;
 Write (chr(27), '=', chr(32 + 1), chr(32 + 4)); { Cursor positionieren }
 Write ('====> Taste  ');
 REPEAT
  Read (KBD, Cha);
  CASE Cha OF
   '2' : CursorUnten;
   '4' : CursorLinks;
   '6' : CursorRechts;
   '8' : CursorOben;
   '5' : Loeschen;
   '7' : Schreiben;
   '3' : CursorAttribut;
   '1' : CursorPosition;
   '9' : ZeileNeu;
  End;
 UNTIL (Cha = '0');
 write (#27, '$@');        { alle Windows schliessen + neu initialisieren }
End.
```

Bild 3 : Pascal-Programm für Cursor- und Editier-Möglichkeiten

```
PROGRAM FadenKreuz;
        {************************************************************
         *   Testprogramm fuer WINDOW-Technik von FLOMONCG          *
         *   hier: Definieren des Fadenkreuzes                      *
   *-------------------------------------------------------*
         *   J.-R. Hoof, Heikendorf    * LastUpdate 30.12.1989  *
         *       Tel 0431 - 24 20 70                            *
        ************************************************************}


VAR  Str : STRING [30];
   Ende : Boolean;

Begin
 Ende := FALSE;
 Write (Chr(27), 'n');              { Maus ein }
 Write (#27, '*');
 Writeln ('   Umdefinieren des Fadenkreuz-Symbols :    e = Ende  ');
 Writeln ('   ================================
           ; ==================JRH');
 Writeln;
 Writeln ('Gib einen String fuer das Fadenkreuz ein: ',
          ' z.B. "1XXXZZ0^^1^^0ZZ1XXX"  ');
 Writeln;
 Writeln ('Vektoren:      [  Z  Y       Laenge 3  ');
 Writeln ('              S  R  Q       Laenge 2    * * * * * * * ');
 Writeln ('               K J I        Laenge 1    * * * * * * * ');
 Writeln ('               C B A        Laenge 0    * * * * * * * ');
 Writeln ('         \  T  L  D $  H  P  X           * * * * * * * ');
 Writeln ('               E F G        Laenge 0    * * * * * * * ');
 Writeln ('               M N O        Laenge 1    * * * * * * * ');
 Writeln ('               U  V  W      Laenge 2  ');
 Writeln ('              ]  ^  _       Laenge 3  ');
 Writeln;
                                { Fenster oeffnen }
 Write (#27, '$O', 1, chr (32+7), chr (32+70), chr (32+15), chr (32+2));
 Write (#27, '$E');             { Fenster sichtbar machen }
 Write (#27, '=', chr (32+0), chr (32+0)); { Cursor an 1. Position setzen }
 REPEAT
   Write ('  Eingabe: ');
   Read (Str);
   IF Str [1] = 'e'
    THEN Ende := TRUE
    ELSE Write (chr(27), '0', Str);
   Writeln (chr (13));
 UNTIL Ende = TRUE;
 Write (#27, '$C');             { Fenster schliessem }
End.
```

Bild 4: Experimente mit dem Fadenkreuz unter FLOMONCG

```
PROGRAM Notizbuch;
        {************************************************************
         *   Testprogramm fuer WINDOW-Technik von FLOMONCG          *
         *   hier: Notizbuch                                        *
   *-------------------------------------------------------*
         *   J.-R. Hoof, Heikendorf    * LastUpdate 29.12.1989  *
         *       Tel 0431 - 24 20 70                            *
        ************************************************************}

 VAR
  cha   : Char;

{-------------------------------------------------}

 PROCEDURE OpenWindow          { Fenster oeffnen }
    (Nummer, YGroesse, XGroesse, YAbstand, XAbstand, Kennwert : integer);
                    { Kennwert 1 = unsichtbar, sonst sichtbar }
  Begin
   write(#27, '$O', Nummer, Chr (YGroesse+32), Chr (XGroesse+32),
                    Chr (YAbstand+32), Chr (XAbstand+32));
                    { Window unsichtbar eroeffnen }
   IF Kennwert <> 1 THEN
    write ( #27, '$E');     { Window sichtbar machen }
  End;

{-------------------------------------------------}

PROCEDURE Weiter;
 Begin
  Write (' ===> RETURN - Taste');
  Read;
  Writeln;
 End;

PROCEDURE FensterSchreiben;   { Text ins Fenster schreiben }
 Begin
  Write ('Info DIR ', Chr(13));
 End;

PROCEDURE NotizbuchSchreiben; { Inhalt des aktiven Fensters wird }
 Begin                        { in Notizbuch kopiert }
  Write (chr (27), '$X');
 End;

PROCEDURE NotizbuchSehen;     { Notizbuch wird sichtbar gemacht }
 Begin                        { keine Fensterkommandos mehr erlaubt ! }
  Write (chr (27), '$V');
 End;

PROCEDURE  NotizbuchWeg;      { Notizbuch wird wieder unsichtbar }
 Begin                        { wieder Fensterkommandos erlaubt }
  Write (chr (27), '$W');
 End;
```

```
PROCEDURE NotizbuchLesen;    { Inhalt des Notizbuches wird in die }
 Begin                       { Konsoleingabe eingeschleusst }
  Write (chr (27), '$Y');    { dh ist wie Tastatureingabe }
 End;

{=======================                                    Hauptprogramm
=======================}

Begin
 write (#27, '*');          { Bildschirm loeschen }
 write (#27, '$@');         { alle Windows schliessen + neu initialisieren }
 OpenWindow (2, 10, 30, 2, 2, 0);    { Fenster oeffnen }
 OpenWindow (3, 10, 30, 10, 20, 0);  { Fenster oeffnen }
 OpenWindow (4, 10, 30, 5, 15, 0);   { Fenster oeffnen }
 OpenWindow (5, 10, 40, 3, 30, 0);   { Fenster oeffnen }
 FensterSchreiben;
 Writeln;
 Writeln (' 1. Info geht ins Notizbuch');
 Weiter;
 NotizbuchSchreiben;                 { Fenster in Notizbuch schreiben }
 Writeln (' 2. Notizbuch wird sichtbar ');
 Weiter;
 NotizbuchSehen;                     { Notizbuch auf Bildschirm anzeigen }
 Write (chr (27), '=', chr (32 + 9), Chr (32 + 1));
 Weiter;
 Writeln (' 3. Notizbuch wird unsichtbar');
 Weiter;
 NotizbuchWeg;                       { Notizbuch unsichtbar machen }
 Writeln (' 4. Notizbuch in Konsoleingabe');
 Writeln ('   Programm kann abstuerzen ');
 Weiter;
 Writeln;
 NotizbuchLesen;                     { Notizbuch in Konsoleingabe einlesen }
 write (#27, '$@');          { alle Windows schliessen + neu initialisieren }

End.
```

Bild 5: Notizbuch-Verwaltung unter FLOMONCG

```
PROGRAM LokalModus;

{*************************************************
          *  Testprogramm fuer WINDOW-Technik von FLOMONCG      *
          *  hier: Lokalmodus setzen                           *
  *-------------------------------------------------*
          *  J.-R. Hoof, Heikendorf   * LastUpdate 29.12.1989  *
          *       Tel 0431 - 24 20 70                          *
         ************************************************************}

 VAR
  cha   : Char;


PROCEDURE Info;
 Begin
  Writeln ('erlaubt sind: ^S, ^D, ^E, ^X, ');
  Writeln ('          ^G = Bell          ^Z = Bildschirm loeschen ');
  Writeln ('          ^^ = Cursor home,    ^H = BackSpace ');
  Writeln ('          ^M = Carriage Return, ^J = Lind Feed ');
  Writeln ('          ^P = Hardcopy ');
  Writeln ('ausserdem:   ^I, ^K, ^L, ^V   sowie alle ASC-Zeichen ');
  Writeln;
 End;

{=======================   Hauptprogramm   ========================}

Begin
 write (#27, '*');          { Bildschirm loeschen }
 Writeln (' Lokalmodus schalten ');
 Writeln ('=======================JRH');
 writeln;
 Writeln ('  1 = Lokal=Modus ein ');
 Writeln ('  2 = Lokal-Modus aus ');
 Write   ('                  ');
 Readln (Cha);
 CASE Cha OF
  '1' : Begin
        Writeln ('+++++ Lokal-Modus eingeschaltet ');
        Writeln ;
        Info;
        Writeln ('Ausschalten mit:  ESC  D  x   (ESC = "CTRL-[") ');
        Write (chr (27), 'DL');
       End;
  '2' : Begin
        Write (chr (27), 'Dx');
        Writeln ('***** Lokal-Modus ausgeschaltet ');
       End;
 End;
End.
```

Bild 6: Arbeiten mit dem Terminal-Mode

```
PROGRAM MonitorModus;
        {*************************************************
          *  Testprogramm fuer WINDOW-Technik von FLOMONCG      *
          *  hier: Monitor-Modus setzen                        *
  *-------------------------------------------------*
          *  J.-R. Hoof, Heikendorf   * LastUpdate 25.12.1989  *
          *       Tel 0431 - 24 20 70                          *
         ************************************************************}

 VAR
  cha   : Char;

{=======================   Hauptprogramm   ========================}

Begin
 write (#27, '*');          { Bildschirm loeschen }
 Writeln (' Monitor-Modus schalten ');
 Writeln ('=======================JRH');
 writeln;
 Writeln ('  1 = Monitor-Modus ein ');
 Writeln ('  2 = Monitor-Modus aus ');
 Write   ('                  ');
 Readln (Cha);
 CASE Cha OF
  '1' : Begin
        Writeln ('+++++ Monitor-Modus eingeschaltet ');
        Writeln ;
        Writeln ('Ausschalten mit:  ESC  X  oder  ESC  u ');
        Write (chr (27), 'U');
       End;
  '2' : Begin
        Write (chr (27), 'X');
        Writeln ('***** Monitor-Modus ausgeschaltet ');
       End;
 End;
End.
```

Bild 7.:Der Monitor-Modus von FLOMONCG von CP/M aus aktiviert.

Hardcopy des Programmes DEMO-AS".COM auf Seite18.

WIR RÄUMEN UNSER EPSON-LAGER *
zu Wahnsinnspreisen

Alles original EPSON-Ware
- solange Vorrat reicht -

Computer:			
70047	EPSON PC mit 2 Diskettenlaufwerken 5 1/4 Zoll 360 KB, PC-Tastatur, Monochrom-Bildschirm, MS-DOS, GW-Basic und Handbuch	DM	698,--
70068	EPSON Handheld-Computer HX 20 - nicht voll funktionsfähig -	DM	198,--
Systemerweiterungen:			
70078	EPSON Expansion Unit für HX 20 extern 16 K	DM	98,--
70075	EPSON 5 1/4 Zoll Diskettenlaufwerk 1,2 MB für PC AX	DM	98,--
Einzelblatteinzüge:			
70018	Einzelblatteinzug für EPSON FX 100	DM	98,--
70087	Einzelblatteinzug für EPSON LX 80	DM	98,--
70017	Einzelblatteinzug für EPSON RX 80	DM	98,--
Aufsatztraktor und Rollenpapierhalterung:			
70067	Aufsatztraktor für EPSON IX 800	DM	38,--
70079	Rollenpapierhalterung für EPSON RX/FX-Serie	DM	19,--
Schnittstellen:			
70050	EPSON IEEE-Interface 8161	DM	38,--
70048	EPSON Centronics-Interface mit 2 K Buffer 8171	DM	48,--
70055	EPSON Serielle Schnittstelle für MX-Serie	DM	38,--
70077	EPSON Serielle Schnittstelle mit 2 K Buffer 7148 für LQ-1500	DM	48,--
70049	EPSON Centronics-Interface für Apple II 8131 (nur Text)	DM	28,--
70051	EPSON Centronics-Interface für Apple II 8132 (Text und Graphik)	DM	38,--
70052	EPSON Interface für Tandy TRS 80 8120	DM	28,--
Kabel zu Schnittstellen:			
70053	Kabel für EPSON IEEE-Interface 8161 (#70050)	DM	10,--
70056	Kabel für EPSON Centronics-Interface für Apple II 8131 (#70049)	DM	10,--
70057	Kabel für EPSON Centronics-Interface für Apple II 8132 (#70051)	DM	10,--
70054	Kabel für EPSON Interface für Tandy TRS 80 8120 (#70052)	DM	10,--
Emulationen und Aufrüstsätze:			
70070	IBM Emulation für EPSON LQ-1500	DM	38,--
70074	EPSON Aufrüstsatz vom FX 100 zum FX 100+	DM	48,--
Nachfüllfarbbänder:			
70073	Nachfüllfarbbänder für EPSON-Drucker	DM	5,--
Software:			
70100	Programmbox für EPSON HX 20 (Demoprogramm) Assembler, Disassembler, Textverarbeitung und DFÜ-Software	DM	10,--
70099	EPSON Textverarbeitung Handy Text auf EPROM für EPSON PX 8	DM	10,--

Alle Preise verstehen sich ab Lager Kempten inklusive der gesetzlichen Mehrwertsteuer
- Bitte Bestellnummern nicht vergessen ! -

* Es handelt sich bei den angebotenen EPSON-Produkten um Inventurüberbestände
Wir hoffen, daß Sie uns auch in den nächsten Jahren Ihr Vertrauen in Sache EPSON schenken

SONDERAKTIONEN AUS DEM NDR-LAGER

NDR-Hardware (Fertiggeräte):

70282	ROB2F Die Leistungsbaugruppe für den NDR: 4 Motorstufen (links/rechts), sowie 2 Analog-Eingänge und Digital-Eingänge machen jedem Funktionsmodell Beine.	DM	169,--
70284	Logikanalysator 10 MHz, 16 Kanäle Der NDR mit Z80 CPU wird zum hochkarätigen Meßgerät, beliebige Triggerbedingungen einstellbar, incl. Steuersoftware (unter CP/M 2.2) auf 5 1/4"-Disk.	DM	169,--
70242	HEXIO2F Die Basis zum NDR-Einsteigerpaket, HEX-Tastatur mit 8stelliger LED-Anzeige. Zwei NDR-Steckplätze voller Länge, erweiterbar.	DM	149,--
70237	ROA64 Speichererweiterung bis 64 kByte, gemischt bestückbar mit statischem RAM (8 kB) und/oder EPROM (8 kB).	DM	69,--
70188	ROA16 Der kleinere Bruder der ROA64, gemischt bestückbar mit statischem RAM (2 kB) oder EPROM (4 kB).	DM	39,--
70111	RAM256 RAM-Erweiterung um 256 kByte, komplett bestückt mit 256kB x 1-Speichern (150 ns).	DM	198,-
70095	RAM64 wie oben, jedoch bestückt mit 64 kB x 1-Speichern (150 ns)	DM	149,--
70285	IOEF Die universelle Experimentierplatine mit großem Lochrasterfeld, 16 TTL-Eingänge und 16 TTL-Ausgänge	DM	59,--
70286	TEAC FD55F Standard NDR-Laufwerk 5 1/4", 720 kB netto	DM	149,--

NDR-Software:

70294	PEDIT38 Der Texteditor der Firma "P1" für CP/M 68k auf 3 1/2" Disk	DM	149,--
70295	PEDIT58 Wie oben, auf 5 1/4" Disk	DM	149,--

Eine Vielzahl weiterer Angbote für NDR und PC/AT wartet auf Sie!
Fordern Sie die "Flohmarkt-Liste" an, kostenlos!

Frühjahrszeit - Messezeit: Wir stellen aus:

besuchen Sie uns auf der...

diadacta ´91 (Düsseldorf)

vom 25.2. - 1.3. Halle 5, Stand G09

CeBit ´91 (Hannover)

vom 13.3 - 20.3. Halle 7, Stand A03

Hardcopy zum Artikel "Einsatz von FLOMONCG"

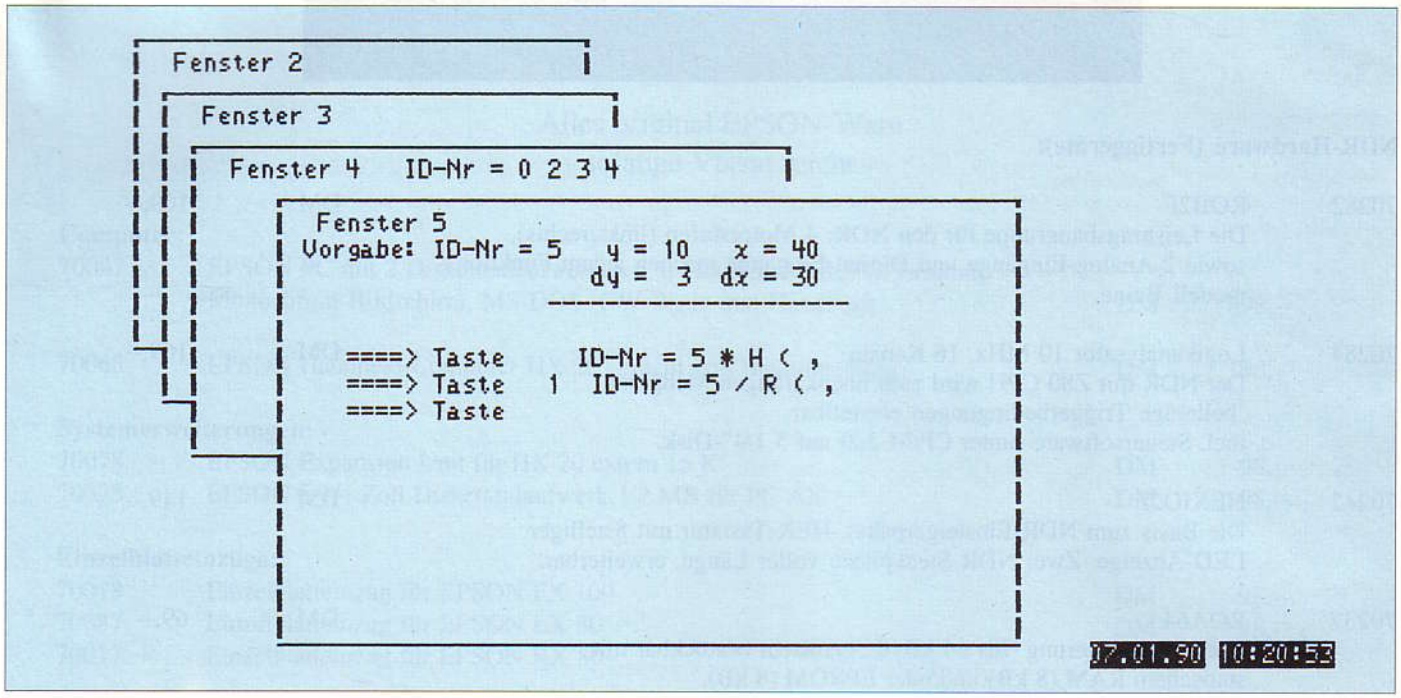


Bild 2.3: Hardcopy des Programmes DEMO-AS2.COM

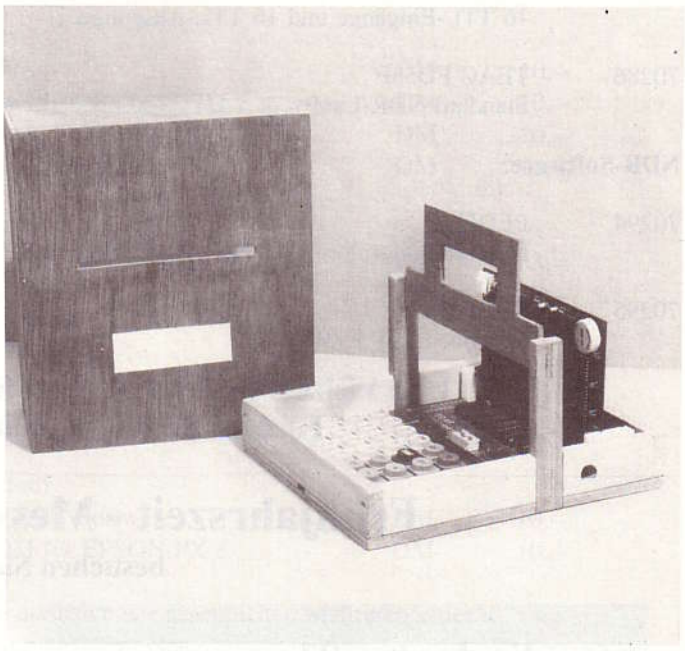
Das Öko-Gehäuse für NDR-Einsteigerpaket

Vor kurzem erst erreichte die LOOP-Redaktion ein sehr originelles Gehäuse für das NDR-Einsteigerpaket. Es stammt vom M.I.K.-Teilnehmer Woltje Kluin aus Norden (hinter MIK verbirgt sich ein Aus- und Weiterbildungskonzept für Mitarbeiter der Deutschen Bundespost).

Wie er uns mitteilte, wollte er einen besonderen mechanischen Schutz um den wertvollen Einsteigercomputer herum aufbauen, den er ja nur als Leihgerät von der Deutschen Bundespost zur Verfügung gestellt bekam.

So hat er dem Einsteigerpaket kurzerhand eine maßgeschneiderte Behausung aus Kistensperrholz verpaßt, die sich sehen lassen kann (siehe Bild). Nach ergonomische Gesichtspunkten optimiert wurde der Transportgriff für das Gehäuse: er dient bei abgenommener Schutzhaube gleichzeitig als Überroll-Bügel.

Dieses Gehäuse könnte durchaus der Prototyp eines "Einsteigerpaket-Laptops" sein... Herr Kluin hat dieses Gehäusemuster freundlicherweise der Redaktion als "Dauerleihgabe " überlassen - nochmals herzlichen Dank dafür. Wir werden es hoch in Ehren halten. Nicht, daß Herr Kluin es nicht mehr gebraucht hätte - im Gegenteil, er hat sich bereits eine zweite Version aufgebaut, mit Fächern für Trafo und Lehrbriefe, um sein mittlerweise eigenens Einsteigerpaket optimal nutzen zu können.



Ein maßgeschneidertes Gehäuse für das NDR-Einsteigerpaket

Dietmar Simons

Komfortables Löschen unter CP/M-80

Dem Aufruf des Herrn Hoof aus der LOOP 24/12 bin ist NDR-Anwender Dietmar Simons gefolgt und hat sich einmal hingesetzt und ein interessantes Programm für FlomonCG entwickelt, das er hiermit den Lesern der LOOP einmal vorstellen möchte.

Es handelt es sich um ein komfortables Löschprogramm mit Fenstertechnik, Statuszeile und invertierter Schrift. Es heißt DELETE.COM und ist vom Aufruf her kompatibel zum CP/M - Befehl ERA.
Das Programm DELETE.COM (V2.0) ist mit dem CP/M - Editor ED geschrieben worden. Anschliessend wurde es mit dem CP/M - Assembler ASM assembliert. Danach wurde der Hex - Code des Programmes mit dem CP/M - Loader LOAD in ein COM - File umgewandelt.
Der Autor mußte das Programm auf diese Weise erstellen, weil er (noch) keinen Z-80 Assembler, geschweige den einen richtigen Texteditor besitzt.
Unter administrativer Zusammenarbeit mit Herrn Hoof (der Autor von FlomonCG), wurde die nun gültige Version 2.0 erarbeitet.

Sie weist folgende Features auf:
1. DELETE ist vollständig mit einer deutschen Bediener-Oberfläche ausgestattet.

2. Die Statuszeile wurde unter die Copyrightmeldung gelegt, sodaß sie nun vollständig erscheint.

3. DELETE fängt Fehle,r die durch einen Aufruf ohne Parameter erfolgen, ab und meldet dies in einem Fenster.

4. Nach Verlassen von DELETE wird wieder der amerikanische Zeichensatz eingeschaltet.

Jetzt aber näheres zur Bedienung von DELETE.COM Ver. 2.0:
Der Aufruf von DELETE ist "nahezu" kompatibel zu dem CP/M-Befehl ERA. Nahezu deshalb, weil beim Aufruf von DELETE mit einigen Extensionen, erst eine andere Datei angezeigt wird und dann erst die gewünschte. Ich arbeite aber daran, diesen Fehler zu beheben.

Beispiele für den Programmaufruf von DELETE.COM:

>DELETE FILE.TYP Löschen einer vollstaendig benannten Datei
>DELETE FILE.??N Löschen einer Datei mit variablem Typ
>DELETE FILE.* Löschen aller Typen einer Datei
>DELETE *.REL Löschen aller Dateien eines bestimmten Types
>DELETE *.* Löschen aller Dateien

Ein Beispiel für den Programmablauf:

```
>DELETE *.PRN
DUMP        .PRN  IHRE EINGABE BITTE : Y
BIOS        .PRN  IHRE EINGABE BITTE : Y
FLT24X      .PRN  IHRE EINGABE BITTE : N
^           ^
Dateinamen & Typ   Eingabe
```

H i n w e i s:
Schreibgeschützte Dateien sind invers dargestellt!

Wird DELETE ohne Parameter aufgerufen, erscheint ein Fenster mit folgendem Text:

Es wurde die Parameterangabe FILE vergessen!
Nach dem Programmabbruch bitte neu aufrufen.
Bitte beliebige Taste druecken.

Drückt man 'I' für Information, dann erscheint das Infofenster mit einer kleinen Bedienungsanleitung. Jede andere Taste führt zu einem Programmabbruch.

Wurde beim Aufruf von DELETE die Parameterangabe 'TYP' vergessen, erscheint ebenfalls eine Fehlermeldung. Von dort kann auch das Infofenster geöffnet werden.

Wurde DELETE richtig aufgerufen, erscheint ein Fenster mit der Aufforderung die Diskette in Laufwerk A zu wechseln. Ist die zu bearbeitende Diskette leer, erscheint eine entsprechende Fehlermeldung in einem Fenster.
Befinden sich Dateien auf der Diskette, dann wird ein grosses Fenster geöffnet und die erste Datei wird angezeigt. Hier steht nun die vollständige Statuszeile zur Verfügung:

CTRL-C = Programm abbrechen ohne eine Datei zu löschen
E = Programm beenden, und die markierten Dateien löschen
Y = Datei löschen
N = Datei nicht löschen
I = Information

Jede Taste außer CTRL-C, E, Y und I hat die gleiche Wirkung wie N.

Wird die Taste 'E' gedrückt, erscheint ein Fenster mit einer Sicherheitsabfrage. Bejaht man die Frage dann werden alle markierten Dateien gelöscht. Verneint man die Frage, dann wird das Fenster geschlossen und das Dateienfenster gelöscht und man bekommt eine neue Möglichkeit.
War keine Datei markiert, dann erfolgt auch keine Sicherheitsabfrage und das Programm wird beendet.

So viel zur Bedienung von DELETE.
Anregungen, Tips und Kritik von Ihrer Seite und natürlich von den NDR-Usern nimmt der Autor natürlich gerne entgegen.

Bezugsbedingungen:

Da das Listing sieben DIN-A4 Seiten umfaßt, eignet es sich nicht für einen Abdruck in der LOOP.

Das Programm DELETE (als COM-File 4K und 12K als Assembler-Quelle) kann vom Autor direkt bezogen werden:

Bitte eine 3 1/2" Diskette, natürlich formatiert, und einen Unkostenbeitrag von 10,- DM senden an:

DiSi-Software Dietmar Simons
Gladbacher Str. 261
5013 Elsdorf - Esch

Herr Simons kopiert zusätzlich zu dem Programm DELETE noch folgende Programme als Quelle und COM-File:

FIXCOPY: ein Kopierprogramm für zwei Laufwerke (Kopierzeit ca. 1,5 min)

ONECOPY: Kopiert eine einzelne Datei.

Georg Neumann

Teil 3

Vorbeugen statt heilen

Der 3. und letzte Teil der Artikelserie über Computer-Viren stellt vorbeugende Maßnahmen vor. Ein besonderer Schwerpunkt ist der Schutz mit Hilfe von Software und dort speziell der Einsatz von kryptographischen Algorithmen. Der Algorithmus INFOCHECK wird vorgestellt. Dieser steht auch als Public-Domain Programm für alle MS/DOS- und alle JADOS - Rechner zur Verfügung.

Die "Schwachstelle" Mensch

Der Hauptausbreitungsweg von Viren ist der Diskettentausch zwischen Computer-Benutzern. Besonders Spiele verbreiten sich in geradezu atemberaubendem Tempo und damit auch etwaige in den Spielen verborgene Viren (z.B. "Leisure-Larry" für Atari). Auf das Tauschen von Disketten ist allerdings die Verbreitung von Viren nicht beschränkt und somit sind die Probleme der Benutzer auch nicht unbedingt selbstverschuldet, wie die Software-Distributoren bisher immer gerne behaupteten. Daß man fast nur von Viren auf Rechnern von Benutzern erfährt, liegt auch daran, daß die Firmen meist aus Sorge um ihren guten Ruf entsprechende Viren-Angriffe im eigenen Haus schlichtweg verschweigen. Es kam auch schon vor, daß bei einer großen Softwarefirma die Mutterdiskette, von der die Kopien für die Kunden gemacht wurden, mehrere Monate unbemerkt als besonderen "Service" zusätzlich noch einen Virus enthielt.

Wer nicht glauben kann (oder will), daß so etwas passiert, der soll sich nur in Firmen und Universitäten umsehen, wo überall momentan das Spiel TETRIS zu finden ist (u.a. auch für JADOS-Rechner verfügbar). Prof. Dierstein (DLR, Oberpfaffenhofen) meinte in einem Gespräch: "...das Gefährlichste ist ein spielender Operator...", weil so Viren auch noch höchste Zugriffsrechte erhalten können. Es empfiehlt sich also, die Spielfreude etwas zu zügeln. Da Diskettentausch nicht unbedingt gleich Raubkopieren ist, sollte man solche meist als "Public-Domain" gekennzeichnete Software nur von vertrauenswürdigen Quellen wie z.B. speziellen Magazinen und Firmen beziehen. Diese Programme sind (fast) immer virenfrei und man hat so auch eine rechtliche Grundlage im Falle einer Versuchung (falls in Geschäftsbedingungen nicht anders festgelegt).
Da man vor Virusbefall nie sicher sein kann, sollte man immer von allen wichtigen Programmen mindestens drei Generationen von Backups haben, auf die im Notfall (siehe Teil 2) zurückgegriffen werden kann. Je nach Häufigkeit von Softwaretausch und -kauf sollten diese Sicherungskopien alle 1 - 3 Monate angefertigt werden. Soweit möglich, sind diese Kopien auch auf Infektionen hin zu prüfen. Diese Maßnahme erfordert allerdings ein großes Maß an Sorgfalt und Selbstdisziplin.

Hardware schützt Software

Wer bereit und fähig ist, einige Modifikationen an der Hardware auszuführen, erhält somit einen fast perfekten Schutz.
Ein sehr guter und zugleich sehr billiger Hardware-Schutz ist z.B. ein kleiner Ein/Aus - Schalter. Wenn man sich statt einer Festplatte zwei kleinere mit je etwa der Hälfte an Speicherkapazität zulegt, so wird die eine Festplatte zur Trägerin der wichtigen Daten erklärt. In ihre Schreib/Lese - Leitung wird der oben erwähnte Schalter eingebaut. Dann werden die Daten kopiert und sind nach Betätigung des Schalters - klack - perfekt geschützt. An der anderen Disk wird nichts verändert. Dorthin kommen dann temporäre oder relativ unwichtige Daten.
Pech hat man nur, wenn die wichtigen Daten bereits einen Virus enthalten (z.B. Virus auf Originaldiskette).

Teurer und z.T. überlistbar sind sog. Prüfsummengeneratoren. Diese erzeugen anhand spezieller kryptographischer Algorithmen vor der Ausführung eines Programms dessen spezifische Prüfsumme und starten es erst, wenn der Vergleich mit der an einer bestimmten Stelle gespeicherten Prüfsumme des originalen Progamms keinen Unterschied ergibt. Das Problem ist der benötigte korrekte Vergleichswert des unveränderten Programms. Sind diese Werte aber in EPROMs untergebracht, so sind sie auch von Viren nicht modifizierbar. Somit können Generatoren als der beste Schutz überhaupt angesehen werden, nur leider mit sehr geringer Flexibilität.
Welche Kriterien an die Algorithmen gestellt werden müssen, wird im Kapitel "Software schützt Software" genauer behandelt.

Die kostenlose Schutzmaßnahme darf natürlich nicht vergessen werden: Der Schreibschutz von Disketten. Jedem ist diese Möglichkeit bekannt, nur wenige nutzen sie aus. Bei der 5 1/4" - Diskette ist der Klebestreifen auf der seitlichen Einkerbung zu befestigen, bei der 3 1/2" - Diskette das kleine Fensterchen zu öffnen und die Diskette ist vor schreibendem Zugriff geschützt. Es gab zwar Diskettencontroller, die sich um den Schreibschutz nicht kümmerten, heute populäre Controller (z.B. WD 1772 beim Atari und WD 1792 beim NDR) achten aber sorgsamst darauf und verweigern solchermaßen nicht erlaubtes Beschreiben. Leider gibt es aber viele Programme, die unbedingt temporäre Dateien einrichten wollen. Diese Daten sind oft umlenkbar und gehören daher entweder in die RAM-Disk oder auf spezielle Disketten bzw. Festplatten, die dann regelmäßig gelöscht oder neu formatiert werden sollten.

Software schützt Software

Das reichhaltigste Angebot an Schutzmaßnahmen bietet natürlich die Software selbst. Folgende Methoden sind laut [3] recht weit verbreitet:

- Überwachung von Systemzuständen (Monitoring)
- Vernichten des Virus (z.B. Anti-Viren)
- Impfung des Virus (Vaccine)
- Verschlüsselungsmethoden

Diese Liste entspricht auch einem Ausschnitt aus dem Virus Katalog (siehe auch [3] und Teil 2).
Die Überwachung von Systemzuständen ist nur für multitaskingfähige Rechner relevant. "...Einige Betriebssysteme stellen Systemroutinen zur Überprüfung von System-Vektoren oder zur Veränderung von Programm-Files zur Verfügung..."[3]. Veränderungen werden gemeldet und, wenn nicht erwünscht, zurückgewiesen.

Auf Virendetektoren und die damit kombinierten Anti-Viren wurde bereits in Teil2 eingegangen. Es sei nur noch bemerkt, daß interessanterweise die Zahl der Anti-

Viren heute bereits um einiges höher ist als die Zahl der eigentlichen Viren.
Leider haben viele Anti-Viren ungewollte Seiteneffekte, die die positive Wirkung wieder aufheben. Besonders deswegen wende man sich auch hier am besten an vertrauenswürdige Quellen (eine solche Adresse siehe wiederum Teil 2).

Originalprogramm Teil 1		Originalprogramm Teil 1
Virusprogramm	====>	Virusprogramm
		Korrekturdaten
Originalprogramm Teil 2		Originalprogramm Teil 2

Eine Impfung mit einem entschärften Virus, auch Vaccine genannt, soll die Verseuchung der so behandelten Datei vortäuschen. Ein Virus, der eine Datei daraufhin untersucht, ob diese bereits mit einer Kopie von diesem Virus verseucht ist, findet die gesuchten, aber nur vorgetäuschten Merkmale und verzichtet auf eine (eigentlich erstmalige) Infektion durch sich selbst. Die Schwachstellen dieser Schutzmöglichkeit sind, daß zu einem Virus wie bei Schlüssel und Schloß ein spezielles Impf-Virus benötigt wird und daß nicht alle Viren die Dateien vor einer Infektion auf bereits bestehende Verseuchung hin untersuchen. Auch wird bei erfolgter Impfung eine Infektion durch einen anderen Virus nicht verhindert.

Die Methoden der Verschlüsselung sind seit Jahrhunderten beliebte Mittel zur geheimen Nachrichtenübertragung. Somit kann man hier bereits auf einen reichen Schatz an Erfahrungen, die Sicherheit und Wirksamkeit betreffend, zurückgreifen. Es gibt sogar die Codierungtheorie (z.B. [4]), ein Spezialgebiet der Mathematik, und wiederum davon einen Ableger, die Kryptographie (z.B. [2]).

Letztere liefert uns die benötigten theoretischen Grundlagen für bewiesenermaßen nicht mehr zurückrechenbare Algorithmen zur Erstellung eindeutiger Prüfsummen (bekanntestes Beispiel hierfür ist die Passwort-Funktion bei UNIX-Rechnern). Der große Aufwand ist deswegen nötig, da sonst sog. halbintelligente Computer-Viren, die auch als HICV bezeichnet werden, die Möglichkeit hätten, in der verseuchten Datei Korrekturdaten einzuschleusen, die die originalen Prüfsummen wiederherstellen würden (siehe Bild 1). Zur sicheren Verschlüsselung bieten sich zwei Varianten an: die lokale und die globale Methode.

Die Wirkung der lokalen Methode beruht auf dem Einsatz von geheimen Schlüsseln. Entweder wird eine Prüfsumme im direkten Zusammenhang mit dem Schlüssel erzeugt, die anschließend noch mit dem Originalwert verglichen werden muß, oder man verknüpft die gesamten Daten direkt mit diesem Schlüssel (siehe z.B. [1]). Spezieller Nachteil der letzteren Version: die Daten müssen vor jedem Gebrauch entschlüsselt werden, außerdem wird ein Virusbefall eines Programms nur durch dessen evtl. Nicht-Funktion bemerkt.

Genereller Nachteil der lokalen Methode ist, daß der Schlüssel bei einer Speicherung im Rechner entdeckt werden oder bei einer Eingabe von außen "abgehört" werden kann.

Die globale Methode braucht keine geheimen Schlüssel. Meist werden sogar die Daten selbst als Schlüssel benutzt. Die verwendeten Verfahren entsprechen einer "kollisionsfreien one-way Hashfunktion". Dabei werden die Daten in einer durch einen Algorithmus fest vorgegebenen Art und Weise rotiert, transponiert, substituiert und schließlich auf eine bestimmte Byteanzahl (4-8 Bytes) komprimiert.

Dieses Ergebnis wird auch Signatur genannt. Die resultierende Wahrscheinlichkeit, daß ein Originalprogramm und dessen verseuchte Version die gleiche Prüfsumme haben, liegt bei ca. 10^{-19} (also sehr, sehr unwahrscheinlich). Ein Vergleich der errechneten und der originalen Signatur muß zwar immer noch stattfinden, ist aber durch geeignete Hardware-Unterstützung, wie oben beschrieben, gut realisierbar.

Der Signatur-Algorithmus INFOCHECK

Ein Algorithmus, der die soeben genannten Eigenschaften hat, ist der von der Firma INFOSYS entwickelte INFOCHECK, der als Public-Domain-Programm für MS/DOS- und OS2-Rechner verteilt wird [5]. Dieses Programm wurde im übrigen vom Autor auf den NDR-Klein-Computer unter JADOS portiert und wird von der Firma GRAF dankenswerter Weise ebenfalls als Public-Domain beziehbar sein. Für einen Abdruck ist das Listing leider zu lang, was durch Betrachtung von Bild 2, das den prinzipiellen "Schaltplan" des Algorithmus skizziert,sicherlich leicht einzusehen ist.

Die errechnete Signatur ist zwar so eindeutig wie D-8960 die Postleitzahl von Kempten ist, doch hat das Programm nur Zweck, wenn es regelmäßig benützt wird. Dabei geht man folgendermaßen vor:

Von allen wichtigen Dateien errechnet man die Signaturen. Diese werden (am bequemsten in einer Datei frei wählbaren Namens) gemerkt. Bei der nächsten Prüfungsserie werden die aktuellen Signaturen errechnet und mit den alten bzw. originalen verglichen. Wenn die Signaturen in einer Datei gemerkt wurden, dann sollten die neuen Signaturen in einer anderen Datei gespeichert werden, weil so der Vergleich auf Gleichheit mit FILECOMPARE sehr bequem machbar ist. Achtung: auf gleiche Prüffolge der Dateien achten !

Eine wichtige Verwendung des INFOCHECK wäre, daß Firmen die Signatur der von ihnen vertriebenen Programme als zusätzliche Information im Handbuch und/oder auf dem Diskettenaufkleber veröffentlichen. So kann jeder Benutzer zu jeder Zeit seine Programme auf Virenbefall prüfen. Eine weite Verbreitung dieses Prüfprogramms und die Bekanntmachung der Signaturen aller veröffentlichter Programme wäre ein möglicherweise großer Schritt in Richtung einer virenfreien EDV-Welt.
Es wäre schön, wenn wir LOOP-Leser und die Firma GRAF hierfür die Vorreiterrolle übernehmen würden.

Georg Neumann
Ackermannstr.12
8000 München 40

P.S.: Für alle, die meinen, Viren würden sich nie bei ihnen finden lassen: Viel Spaß an den kommenden Freitagen, den 13ten !!

Anmerkung der Redaktion: Das Programm INFOCHECK liegt der LOOP-Redaktion vor. Eine Demonstration anhand beigefügter Beispielsdateien ist sehr beeindukkend. Bei entsprechender Nachfrage der NDR-Anwender nach den Signatur-Programm INFOCHECK unter JADOS, sind wir gerne bereit dieses Programm (incl. Source) im Pubilc-Domain Preisrahmen zu vertreiben. Als Aufwandsentschädigung werden demzufolge die reinen Disketten-Kopierkosten in Höhe von DM 30.- (incl. Mwst.) veranschlagt.
Alternativ wäre auch die Aufnahme von INFOCHECK in eine weitere JADOS-TOOL Disk denkbar.

Wie denken die NDR-Anwender über den Vorschlag von H. Neumann, die Prüfsumme von neu veröffentlichten Programmen mit zu veröffentlichen? - Schreiben Sie uns!

Literatur:

[1] Bäcker, Rüdiger
"RUBACODE Programm"
LOOP Nr.15 S.18-19 1987

[2] Beutelspacher,A. Universität Gießen
"Kryptologie"
Vieweg-Verlag Braunschweig 1987

[3] Brunnstein, Klaus Universität Hamburg
"Zur Klassifikation von Computer-Viren: Der 'Computer Virus Katalog' "
Proceedings der 19. GI-Jahrestagung 1989

[4] Heise,W., Quattrocchi,P. TU München, Universität Padua
"Informations- und Codierungstheorie"
Springer-Verlag Berlin Heidelberg 1989

[5] Krause, Georg INFOSYS GmbH Bodenheim
"INFOCHECK"
Begleitheft zum Public-Domain-Programm INFOCHECK 1989

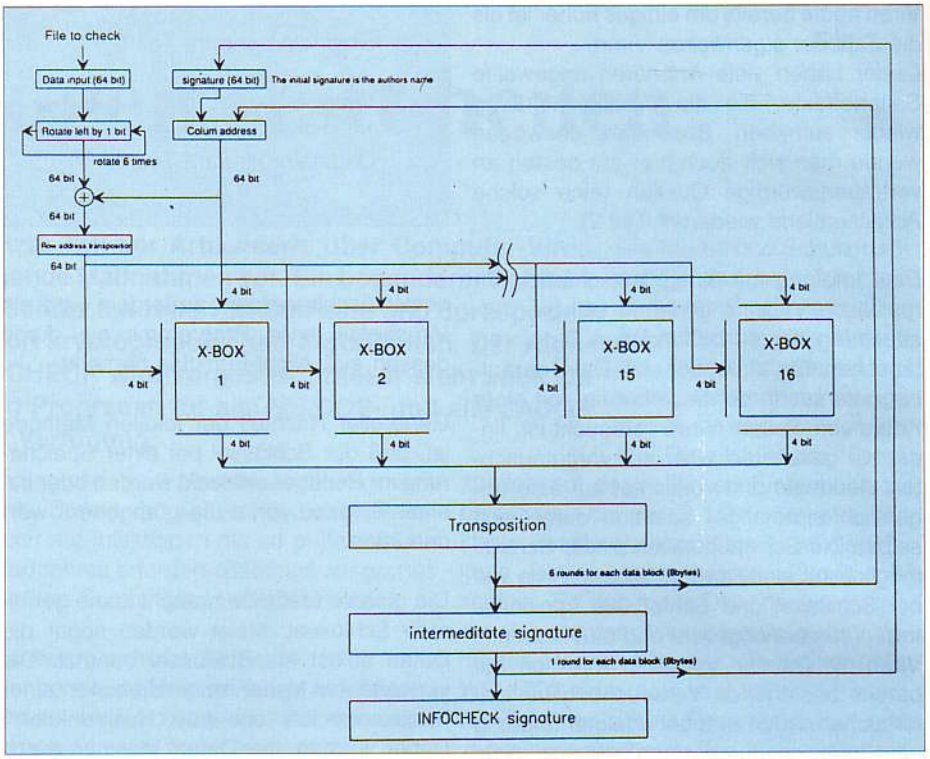


Bild 2: Prinzip "Schaltplan" von INFOCHECK

Jetzt erhältlich: SCSI-Einbindung in CP/M68k

Die Einbindung geschieht über ein neues BIOS für CP/M68k.

Langsam schließt sich der Kreis: die für den NDR-Computer "neue" Festplattenschnittstelle SCSI (siehe auch LOOP 23) ist nun auch mit dem Betriebssystem CP/M68k nutzbar.

Neben der Erweiterung zur SCSI-Platte wurden "nebenbei" noch ein paar weitere Feinheiten integriert:

- SCSI-Anpassung bis 65 MByte
- Hilfsprogramme zum Formatieren der Festplatte
- AUTOEXEC aktiviert den Autostart-Mechanismus nach dem Booten
- Schnelle Diskettenzugriffe bis 3 ms Steprate
- Intelligenter Druckertreiber erledigt endgültig alle lästigen Zeichensatzprobleme.

Die Einbindung des neuen BIOS in das Betriebssystem wird auf der Diskette Schritt für Schritt erklärt.

Falls Sie bereits über das neue Grundprogramm V6.2 verfügen, können Sie auch von der SCSI-Platte booten.

Auch bei älteren GP-Versionen (ab V 4.3) ist das BIOS voll einsatzfähig, bis auf die Einschränkung des Platten-Bootens.

Es könnnen SCSI-Platten von Seagate bis maximal 65 MByte angeschlossen und verwaltet werden.

Die Gesamtkapazität steht "am Stück" als Laufwerk J: zur Verfügung, das heißt die prophiylaktische Anwendung der CP/M-USER-Bereiche (0..15) beugt einem möglichen Dateien-Chaos entgegen.

Das neue BIOS ist automatisch Bestandteil des Lieferumfanges von CP/M68k V1.3-04 geworden (man beachte die Versionsnummern- Anhebung von -03 auf -04).

Für Anwender, die lediglich das SCSI-BIOS (mit obigen Optionen) nachrüsten möchten, bieten wir zu Kopierkosten (DM 30.-) eine entsprechende Diskette mit Anleitung (ReadMe-File) an; einzige Voraussetzung:

Das bisherige CP/M68k trägt die Versionsnummer V1.3-03; die Orginal-Rechnung liegt vor.

Klaus Rumrich

Der JADOS - Linker

Bereits vor fast drei Jahren wurde von Klaus Janßen ein Object-Format für JADOS vorgestellt (siehe LOOP 16). Schon in LOOP 14 war ein Briefwechsel zwischen Martin Husemann und Klaus Janßen vorausgegangen, in dem die Notwendig eines Object-Formates und eines Linkers für JADOS erläutert wurden.
Jetzt wurde ein auf dem alten Format aufbauendes Object-Format definiert. Der entsprechende Linker ist bereits verfügbar. Damit steht nun einer Compiler-Entwicklung für JADOS nichts mehr im Wege.

Was ist ein Linker?

Ein Linker hat die Aufgabe, mehrere Object-Files, die er als Eingabe erhält, zu einem ausführbaren Programm zusammenzufügen. Dabei sollen die einzelnen Object-Files einzeln und unabhängig voneinander - zum Beispiel durch einen Assembler oder Compiler erzeugbar sein.

Daraus folgt, daß zur Zeit des Compilierens der Compiler noch nicht weiß, auf welcher Adresse ein externes Unterprogramm nach dem Linken stehen wird. Daher kann der Compiler einen Sprung auf ein solches Unterprogramm gar nicht selbst übersetzen, sondern legt im Object-File Informationen ab, wo im Programm ein Sprung auf welches Label gewünscht ist.

Erst der Linker weiß, welche Object-Module auf welche Adresse zu liegen kommen und kann anhand der Informationen im Object-File die noch fehlenden Sprungadressen an den richtigen Stellen im Maschinencode einfügen. Analog wird etwa bei extern deklarierten Variablen vorgegegangen.

Wozu ein Linker?

Offensichtlich kann man auch ohne Linker Programme entwickeln, sonst gäbe es für JADOS bisher kein einziges Programm. Bei kleineren Assemblerprogrammen wird sich die Benutzung des Linkers auch in Zukunft erübrigen. Bei umfangreichen Programmen, insbesondere in höheren Programmiersprachen, erleichtert ein Linker dagegen die Arbeit sehr oder macht sie sogar erst möglich.

Compiler für höhere Programmiersprachen haben fertig übersetzte Unterprogramme für häufig benötigte Funktionen oder Prozeduren blicherweise in einer Library (Bibliothek) verfügbar. Bei der Übersetzung eines Programmes, das eine solche Funktion benötigt, setzt der Compiler nur einen entsprechenden Unterprogrammaufruf ein. Der Linker hat dann dafür zu sorgen, daß das benötigte Modul aus der Library zu dem Programm dazugelinkt wird und die richtige Sprungadresse in dem aufrufenden Programmteil eingefügt wird.

Auf diese Weise werden auch bei sehr umfangreichen Bibliotheken nur diejenigen Module wirklich in das Programm übernommen, die auch benötigt werden. Dadurch werden die Programme nicht unvernünftig groß.

Schließlich soll noch auf das Problem der Relokativität hingewiesen werden. Bekanntlich müssen bei JADOS alle Programme relokativ (d.h. auf beliebigen Adressen ablauffähig) sein. Dies wird durch die mögliche Programm Counterrelative Adressierung bei den 680xx-CPUs wesentlich erleichtert. Leider erlaubt diese Adressierungsart (zumindest bei 68008 und 68000) keine Sprünge über Distanzen größer als 32 KByte. Bei Programmen, die größer als 32 KByte sind, muß aber damit gerechnet werden, daß auch Sprünge über so große Distanzen auftreten.

Die Lösung dieses Problems gelang Klaus Janßen mit der Einführung von Hilfslabels. Tritt ein Sprung auf eine externe Marke auf, so wird vom Compiler zunächst ein Sprung auf ein Hilslabel erzeugt. Bei dem Hilfslabel muß nun der Prozessor die Adressdifferenz zu dem externen Label zur Hilfslabeladresse addieren und einen indirekten Sprung auf diese Adresse durchführen.

Hier ein Beispiel für das Programmstück nach dem Hilfslabel print123:

```
print123:                    print123:
    LEA print123(PC),A6          PEA print123(PC)
    ADDA.L #addrdiff,A6          ADD.L #addrdiff,(A7)
    JMP (A6)                     RTS
```

Die linke Möglichkeit ist schneller, zerstört aber ein Adressregister. Die rechte Möglichkeit ist langsamer, verwendet aber statt des Adressregisters den Stack.

Im Modul selbst wird das Unterprogramm mit dem relativen Aufruf

BSR print123

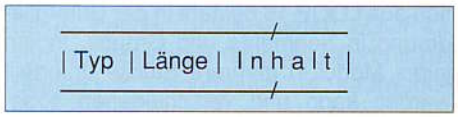
aufgerufen. Dieser verzweigt zum Hilfslabel print123, dort wird der aktuelle Programmzähler an der Marke print123 ermittelt und anschließend der Wert addrdiff aufaddiert.

Damit steht die Adresse der eigentlichen Unterroutine im Register A6 bzw. auf dem Stack. Mit dem indirekten JMP bzw. dem RTS wird schließlich die externe Routine auf der berechneten Adresse angesprungen.
Da das Übersetzerprogramm die Adresse der externen Routine nicht kennt, muß der Linker die Aufgabe übernehmen, den Wert addrdiff zu berechnen und an die richtige Stelle zu setzen.

Durch zusätzliche Informationen an den Linker kann der Umweg über das Hilfslabel sogar vermieden werden, wenn sich die tatsächliche Distanz zum externen Label als kleiner als 32 KByte erweist. Dann kann das externe Label direkt angesprungen werden.

Wie sieht das Object-Format aus?

Grundsätzlich wurde das Object-Format, wie es in LOOP 16 beschrieben wurde, beibehalten.
Allerdings wurden zahlreiche Details geändert, so daß die alte Definition nicht mehr gültig ist. Ein Object-File besteht aus einer Reihe von Records, die den Maschinencode und die Linker-Informationen enthalten. Ein Record besteht formal aus einem Recordtyp, einer Recordlänge und dem Recordinhalt:



Der Typ identifiziert eindeutig die Bedeutung des Records und besteht aus zwei Bytes. Die Länge gibt an, wieviele Bytes der Record hinter der Längenangabe umfaßt.
Die Länge selbst wird mit zwei Bytes angegeben, so daß Records bis zu 64K enthalten können. Jeder Record muß eine ge-

radzahlige Länge haben, damit alle Adressangaben auf Wortgrenze stehen.

Es gibt insgesamt 14 verschiedene Record-Typen. Eine genaue Beschreibung würde diesen Rahmen sprengen, ist aber selbstverständlich in der Dokumentation zum Linker enthalten.

Es gibt folgende Record-Typen:

MHEADER	Modulanfang
MODEND	Modulende
MCODE	Maschinencode
GLOBPROC	Globale Prozedur
EXTPROC	Externe Prozedur
GLOBLAB	Globales Label
EXTSYM	Externes Symbol
LOCPROC	Lokale Prozedur (für Debugger, vom Linker ignoriert)
LOCLAB	Lokales Label (für Debugger, vom Linker ignoriert)
LINNUM	Zeilennummer (für Debugger, vom Linker ignoriert)
COMMENT	Kommentar
LIBEND	Bibliotheksende
GLOBCONST	Globale Konstante
SEGMENT	Segmentanfang

Jedes Objectfile ist nun folgendermaßen aufgebaut:

```
MHEADER
    eventuell GLOBCONST-Records
  SEGMENT
    weitere Records
  SEGMENT
    weitere Records
    . . .
MODEND
```

COMMENT-Records sind an jeder Stelle erlaubt. Ein Bibliotheksfile besteht aus mehreren Modulen hintereinander; hinter dem letzten MODEND-Record muß dann ein LIBEND-Record folgen, um das Ende der Bibliothek anzuzeigen.

Segmente und Groups

Ein wesentlicher Unterschied zu der Definition aus LOOP 16 besteht in der Untergliederung in Segmente und Groups. Indem jedes Modul in mehrere Stücke unterteilt werden kann und verschiedenen sogenannten Segmenten zugeordnet werden kann, ist es möglich, auf die Anordnung dieser Stücke im Speicher Einfluß zu nehmen.

Beim Linken werden die Segmente gleichen Namens hintereinander im Speicher abgelegt. So ist es möglich, in jedem Modul Maschinencode und initialisierte und uninitialisierte Variablen zu haben, die nach dem Linken jeweils in zusammenhängenden Speicherbereichen liegen.
Eine weitere Gruppierung der Segmente ist durch die Angabe von Groups möglich, die jeweils mehrere Segmente zusammenfassen.

Eine sinnvolle Unterteilung in Segmente wäre etwa:

INIT-CODE	Initialisierungs-Programmbereich (CODE-GROUP)
CODE	Allgemeiner Programmbereich (CODE-GROUP)
SMALL-DATA	Datenbereich < 32K (DATA-GROUP)
LARGE-DATA	beliebig großer Datenbereich (DATA-GROUP)
SMALL-BSS	uninitialisierter Datenbereich < 32K (BSS-GROUP)
LARGE-BSS	beliebig großer Datenbereich (BSS-GROUP)

Tatsächlich sind die Segmentnamen für das vorliegende Object-Format völlig frei.

Das obige Beispiel zeigt, daß die Unterteilung in Segmente eine Unterteilung nach der Art der enthaltenen Daten darstellt (im Gegensatz zur Unterteilung in logische Module).
Vorzugsweise steht das eigentliche Programm in einem eigenen Segment (oben CODE). Der Zugriff auf Daten eines anderen Segmentes erfolgt über ein Adressregister, das auf den Beginn des jeweiligen Bereiches zeigt.

Noch ein Wort zu der Unterteilung in INIT-CODE und CODE in unserem Beispiel.
Werden die Segmente in der obigen Reihenfolge deklariert, so startet das Programm mit dem ersten Befehl in INIT-CODE. Dies macht es möglich, ein Initialisierungsprogramm für jedes Modul gewissermaßen automatisch auszuführen, ohne daß sich das Hauptprogramm darum kümmern müßte.
Ein Beispiel wären Unterprogramme für die serielle Schnittstelle. Ein Hauptprogramm, das diese Unterprogramme verwenden möchte, veranlaßt den Linker, die entsprechenden Module einer Library dazuzulinken.
Außerdem werden Initialisierungsroutinen für diese Module vom Linker in dem Segment INIT-CODE abgelegt, die in dem Beispiel die serielle Schnittstelle initialisieren. Das Hauptprogramm muß von der Existenz dieser Initialisierung nichts wissen; es genügt, in den Library-Modulen die entsprechenden Programmteile in dem Segment INIT-CODE unterzubringen. Damit dies Verfahren auch für mehrere Initialisierungen funktioniert und das Hauptprogramm im Segment CODE anschließend ausgeführt wird, dürfen die einzelnen Initialisierungen nicht mit einem RTS beendet werden.

Diese kurze Erläuterung stellt nur ein Beispiel zur Demonstration der Leistungsfähigkeit des Segment-Konzeptes dar. Es ließen sich noch beliebige andere Anwendungsmöglichkeiten für die Strukturierung in Segmente finden. Da die Namensgebung und Verwendung der Segmente völlig frei ist, ist viel Spielraum für künftige Anwendungen gegeben.

Arbeitsweise des Linkers

Der Linker arbeitet in vier Durchläufen.
Im ersten Durchlauf werden die Benutzereingaben interpretiert und die benötigten Files geladen. Diejenigen Module, die benötigt werden, werden markiert.
Im zweiten Durchlauf werden alle Segmente, Groups und globalen Symbole definiert.
Im dritten Durchlauf wird der Maschinencode der benötigten Module den Segmenten entsprechend auf die endgültige Adresse geschoben.
Im vierten Durchlauf schließlich werden alle Fixups der externen Referenzen durchgeführt.
Die Symboltabelle für die globalen Symbole ist durch eine Hashtabelle realisiert. Zur Zwischenspeicherung aller noch undefinierten externen Symbole ist ein Stack eingerichtet.

Der Ablauf des ersten Linker-Durchlaufes stellt sich folgendermaßen dar:

```
Externes Label '@start' —> Stack
WHILE NOT Stack leer DO
  Stack —> Extsymb
  WHILE NOT Extsymb in Hashtabelle definiert DO
    Lade nächstes OBJ oder LIB File
    Schreibe alle globalen Symbole in die
    Hashtabelle
  END
  IF Modul, in dem Extsymb definiert wird,
    unmarkiert THEN
    Markiere das Objectmodul
    Schreibe alle externen Symbole des Moduls auf
    den Stack
  END
END
```

Wenn der Stack abgearbeitet ist, sind alle externen Symbole definiert und alle benötigten Module sind markiert.

Im zweiten Durchlauf wird zunächst die Hashtabelle wieder gelöscht. Anschließend werden alle im Speicher befindlichen und als benötigt markierten Module durch-

laufen und die globalen Symbole in die Hashtabelle geschrieben. Gleichzeitig werden die Längen aller Segmente und Groups bestimmt.
Am Ende dieses Durchlaufs werden die Startadressen aller Segmente und Groups aus den nun bekannten Längen der Segmente und Groups bestimmt.

Im dritten Durchlauf wird der Maschinencode an seine endgültige Adresse geschoben. Die Zieladressen ergeben sich aus den Startadressen der Segmente und Groups.

Im letzten Durchlauf wird der eigentliche Fixup durchgeführt. Bei jedem EXTPROC oder EXTSYM Record wird mit Hilfe der Hashtabelle die Zieladresse bestimmt und je nach geforderter Referenzart der Fixup vorgenommen. Fixups werden grundsätzlich als Addition durchgeführt.

Im Object-File sind zu jeder externen Referenz auch eine Referenzart und eine Referenzgröße angegeben, die bestimmen um welche Adressierungsart es sich handelt und ob die Adresse bzw. das Datum Byte, Word oder Long groß ist. Paßt der Wert nach der Addition nicht mehr in die vorgegebene Referenzgröße, so wird eine Fehlermeldung gegeben.

Erzeugung von Object-Files
Zur Zeit gibt es noch keine Compiler oder Assembler, die das Object-File erzeugen könnten. Ich hoffe daher, daß sich der eine oder andere engagierte NDR-Programmierer findet, der vielleicht einen Compiler schreiben oder anpassen kann.

In der Zwischenzeit können mit einigen Macrodefinitionen für den Assembler des Grundprogrammes ab Version 6.2 bereits Object-Files erstellt werden. Diese Macros sind gemeinsam mit dem Linker erhältlich. Zugegebenermaßen ist diese Möglichkeit nicht übermäßig komfortabel, aber zum ersten Ausprobieren durchaus brauchbar.

Anmerkung der Redaktion: Für alle diejenigen, die die LOOP 16 (noch) nicht haben und an den Grunddefinitionen des JADOS-Linkers interessiert sind, eine erfreuliche Mitteilung: Die LOOP 16 ist in begrenzten Umfang noch erhältlich.

Klaus Rumrich
Spessartstraße 3
6457 Maintal 2

Günter Renner

Das Benutzerinterface

Das Drumherum

Wie wohl heute noch die meisten Computerprogramme hat auch der hier beschriebene Diskettenmonitor eine bisweilen etwas lästige Eigenschaft: er will bedient werden. Diesmal werden jene Programmteile beschrieben, die die Kommunikation mit dem Benutzer besorgen.

Wie bereits in der ersten Folge beschrieben, beginnt alles damit, daß der Monitor mit der Grundprogrammfunktion 'Floppystart' vom Laufwerk A/0 geladen und angesprungen wird. Der Bootsektor lenkt den Programmverlauf dann zu dem Programm namens 'handler' hin, dem eigentlichen Start des Monitors.
Hier wird lediglich die Bildschirmausgabe vorbereitet sowie die Kopfzeile mitsamt der RAM-Adresse der Nutzdaten ausgegeben. Dann geht es weiter nach 'menue'. Auf Tastendruck hin wird hier zu einer der Menuefunktionen verzweigt oder aber, auf 'M' hin, das Programm beendet.
Es folgt dann noch eine kleine Sammlung jener Meldungen, von denen Menuefunktionen Gebrauch machen, um anzuzeigen, daß z. B. eine zu ladende Datei nicht vorhanden ist, daß beim Speichern bereits eine Datei desselben Namens existiert oder dergleichen.
Das Unterprogramm 'getwert' dient der Eingabe der Dateilänge beim Abspeichern; ein Teil der Routine 'getname' liest die eigentliche Eingabe ein, die dann mit der Grundprogrammfunktion WERT in einen Zahlenwert umgesetzt wird. Die übrigen 'Prográmmchen' erklären sich selbst.

Sehr wichtig ist allerdings der Einsprung namens 'testflg'.
Mit ihm enden alle Menuefunktionen, die die Diskette beschreiben oder von ihr lesen. Ist dabei irgendein Fehler aufgetreten, enthält 'fflag' einen Wert ungleich null, und es wird eine entsprechende Meldung ausgegeben, so daß der Benutzer weiß, daß etwas faul ist.

Ausblicke...
Sicher ist die hier vorgestellte Version nicht das letzte Wort. Vieles könnte verfeinert und komfortabler gemacht werden. Man könnte sich folgendes vorstellen:

- eine Version mit Floppytreiber, die auch mit 80-spurigen 5 1/4"-Laufwerken arbeitet
- freie Wahl der RAM-Adresse, von der gelesen und auf die Dateien geladen werden
- automatische Längenbestimmung von Texten
- Starten von Programmen von der Diskette
- Formatierer, der u. a. defekte Cluster in der FAT kennzeichnet
- Unterstützung von Assemblieren und Ausdrucken von Texten
- andere Formate, z. B. 720 kB auf 3 1/2"
- Führen von Datum und Uhrzeit.

Es ist geplant, interessierten LOOP-Lesern ein assemblierfähiges Listing für den Betrieb von 80spurigen 5 1/4"-Laufwerken käuflich zur Verfügung zu stellen - einschließlich Formatierer. Bei entsprechender Nachfrage wäre auch eine Version für 3 1/2", 80spurig und doppelte Dichte verfügbar.

Günter Renner
Schloßbühlstr. 11
7206 Emmingen-Liptingen

Volker Stahl

Farbe für den NDR-PC

Welcher NDR'ler kennt das nicht: neidisch schaut man auf die Super-Grafik Rechner (Amiga und Co.) mit deren unendlich scheinenden Farbenvielfalt, und selbst tippt man auf seinem s/w- oder grün-Monitor herum?! Doch damit ist jetzt Schluß! Mit Ihrem Umstieg auf die CPU8088 haben Sie nicht nur eine größere Software-Bibliothek (MS/DOS), sondern auch die Möglichkeit, billig eine hochauflösende Computergrafik erster Klasse auf Ihren Bildschirm zu zaubern.

Wollen Sie eine Farbauswahl aus 262144 verschiedenen Farbtönen haben, oder gleichzeitig 256 verschiedene Farben auf dem Bildschirm darstellen? Dann brauchen Sie auch eine VGA-Grafikkarte nebst VGA-Bildschirm. Nach meinem Einkauf ließ ich mich sofort von der Supergrafik und Farbenpracht meines Monitors verzaubern. Da erschienen Bilder in Fotoqualität! Ich war einfach begeistert, da können selbst altbekannte Grafik-Rechner (ich will keine Namen mehr nennen) nicht mehr mithalten, aber überzeugen Sie sich selbst!

Und das brauchen Sie dazu:

* BUSKOPP
* VGA-Grafikkarte
* VGA-Monitor

Das ganze können Sie schon weit unter 2000,- DM erhalten.

Die BUSKOPP-Baugruppe von der Firma GRAF stellt die Brücke zwischen NDR-BUS und IBM PC-BUS her und erlaubt es somit, auch IBM PC-Karten (8 Bit BUS-Breite) mit der CPU8088-Karte anzusteuern.
(Weitere Einzelheiten können Sie aus der LOOP 21 entnehmen). Die Baugruppe ist schnell zu löten und einfach in der Handhabung: einfach mit dem NDR-BUS verbinden und in den PC-Slot auf der BUSKOPP die VGA-Grafikkarte einstecken (eventuell Jumperstellung auf der BUSKOPP ändern, aber normalerweise funktioniert die VGA-Grafikkarte einwandfrei mit der EGA-Jumpereinstellung die im CPU8088 Handbuch erklärt wird!).

Eine VGA-Grafikkarte können Sie sich in jedem Computergeschäft besorgen. Es gibt eine sehr große Auswahl an verschiedene Modellen und Herstellerfirmen. Ich habe mich für eine sog. No-Name VGA-Karte entschieden, und dies hauptsächlich aus zwei Gründen:

1. ist solch eine "markenlose" Grafikkarte sehr viel billiger (zwischen 300,- und 450,- DM) als eine Karte von einem bekannten Hersteller, und

2. ist der VGA-Standard ohnehin genormt, sodaß keine sehr großen Unterschiede zwischen den Karten sind (außer vielleicht die Anzahl der mitgelieferten Treiberprogramme).
Beim Kauf einer solchen VGA-Karte sollten Sie aber unbedingt auf folgende Dinge achten:

- 8 Bit-Karte (kurze Steckleiste)
- VGA Chipsatz für IBM PC/XT/AT und kompatible Systeme
- 100% hardwarekompatibel mit der IBM VGA
- EGA, CGA, MDA und Hercules kompatibel
- 800x600 Auflösung in 16 Farben (oder 640x480)
  320x200 Auflösung in 256 Farben
- mindestens 256 kByte Videospeicher (auf der Karte)
- voll kompatibel zum IBM BIOS

Bei solch einer Karte werden Sie keine Probleme mit den verschiedene Grafikprogrammen bekommen und die Karte funktioniert einwandfrei mit der NDR-PC Konfiguration!

Achten Sie aber darauf, daß Sie ein genügend leistungsfähiges Netzteil besitzen, ansonsten kann Ihr Rechner ab und zu schon einmal abstürzen...

Ebenso können Sie einen VGA-Monitor in jedem Computergeschäft kaufen. Hierbei sollten Sie darauf achten, daß der Bildschirm auch die Auflösung ihrer VGA-Grafikkarte besitzt.

Wenn Sie dann alles korrekt und sorgfältig eingebaut haben, legen Sie die DOS-Systemdiskette ins Laufwerk A: und schalten den Computer ein. Wahrscheinlich bleibt der Bildschirm für kurze Zeit dunkel und danach meldet sich kurz das BIOS der VGA-Grafikkarte. Nun können Sie das Ihnen vertraute Bild des NDR-BIOS von R.D. Klein auf dem Monitor sehen, nur mit dem Unterschied, daß das jetzige Bild über ein wesentlich besseres Schriftbild verfügt!
Sollte Ihr Bildschirm schwarz bleiben, so müssen Sie vielleicht noch die alte GPD64k-Grafikkarte aus dem BUS herausnehmen. Flimmert das Bild, oder ist es sehr schwach zu erkennen, so müssen Sie vielleicht noch die Jumper-Einstellung auf der BUSKOPP ändern!

Sollte aber alles korrekt funktionieren, können Sie mein VGA-Testprogramm abtippen, sich in Ihren Sessel zurücklehnen und von der neuen VGA-Grafik faszinieren lassen.

Das Programm 'Farbwahl' ist in TURBO PASCAL 5.0 geschrieben, läßt sich aber sicherlich leicht für einen NDR'ler in eine andere vergleichbare Programmiersprache umschreiben. Ich empfehle Ihnen aber, falls Sie noch nicht in TURBO PASCAL programmieren, auf diese geniale Sprache umzusteigen. Das wäre wieder einmal ein Thema für sich, vielleicht in einer anderen LOOP-Ausgabe...

Jetzt noch kurz zur Funktion des Programmes: Wenn Sie an der Eingangsfrage (neuer Farbwert?) 'n' eingeben, erscheint auf Ihrem Bildschirm eine Farbabstufung von Grautönen, die ineinander überlaufen, was einen sehr tollen Effekt verursacht. Geben Sie an der Eingangsfrage 'j' ein, so können Sie selbst den Farbton und die Abstufungen wählen. Probieren Sie am besten einmal selbst herum...
Hier ein paar mögliche Werteingaben:

Rot	Grün	Blau
1,5	1,1	0,0
2,10	0,0	0,0
0,0	1,1	0,0
1,1	1,1	2,2
0,0	1,10	2,15
2,10	2,10	3,10
usw.		

Na, überzeugt? Falls Sie ein (kleiner) Spieler sind: kaufen Sie sich doch den neuen Flugsimulator FS4 und genießen Sie den Flug in einer Auflösung von 640x480 Bildpunkten...

In absehbarer Zeit werde ich auch ein kleines VGA-Software-Paket für interessierte NDR-PC'ler, die die CPU8088 in Verbindung mit einer VGA-Grafikkarte besitzen herausbringen. Ich werde mich dann nochmal melden. Bis dahin wünsche ich Ihnen sehr viel Freude mit dem NDR-PC und Ihrer neuen VGA-Karte/Bildschirm.

Anmerkung der Redaktion: Auch GES hat preiswerte VGA-Ausrüstungen auf Lager; siehe Umschlagseite 4!

```
{ Programm: Farb-Palette-Auswahl  --> 16 aus 262144 Farben : }
{ (C)Copyright by Volker Stahl April 1990                    }

Program Farbwahl;

Uses
  Graph, Crt;

Var
  graphdriver, graphmode  : integer;
  rot,gruen,blau,zaehler  : integer;
  r1,r2,g1,g2,b1,b2       : integer;
  zeichen                 : char;

Procedure abfrage;
Begin
  Write('Wollen Sie neue Farbwerte eingeben (j/n) ? ');
  zeichen:=ReadKey;
End;

Procedure eingabe;
Begin
  ClrScr;
  Write('Rotmultiplikator  : ');
  ReadLn(r1);
  Write('Rotaddierer       : ');
  ReadLn(r2);
  Writeln;
  Write('Grünmultiplikator : ');
  ReadLn(g1);
  Write('Grünaddierer      : ');
  ReadLn(g2);
  Writeln;
  Write('Blaumultiplikator : ');
  ReadLn(b1);
  Write('Blauaddierer      : ');
  ReadLn(b2);
End;

{ Hauptprogramm }
Begin
  ClrScr;
  GraphDriver:=vga; GraphMode:= vgahi;
  r1:=5; r2:=1; g1:=5; g2:=1; b1:=5; b2:=1;
  abfrage;
  If zeichen='j' Then eingabe;
  InitGraph(graphdriver,graphmode,'');
  ClearDevice;
  SetColor(1);
  For zaehler := 1 to 15 Do
    Begin
      gruen :=zaehler*g1+g2;
      rot   :=zaehler*r1+r2;
      blau  :=zaehler*b1+b2;
      SetRGBPalette(zaehler,rot,gruen,blau);
      SetPalette(zaehler,zaehler);
      SetColor(zaehler);
      Rectangle(zaehler,zaehler,639-zaehler,479-zaehler);
      Rectangle(31-zaehler,31-zaehler,639+zaehler-31,479+zaehler-31);
      SetFillStyle(1,zaehler);
      Circle(319,239,200-zaehler);
      FloodFill(319,239,zaehler);
  End;
  For zaehler :=15 downto 1 Do
    Begin
      SetColor(zaehler);
      SetFillStyle(1,zaehler);
      Circle(319,239,200-31+zaehler);
      FloodFill(319,239,zaehler);
  End;
  zeichen:=ReadKey;
  CloseGraph;
End.
```


Kleinanzeigen/Leserbrief

Zum Stichwort NDR-Produkte, was noch? mein Wunsch wäre eine Baugruppe mit dem Chips aus der 68000er Familie, wie den DUART (68681) und den PI/T (68230). Es handelt sich dabei um 2 serielle- sowie 2-3 parallele Schnittstellen. Die Baugruppe SER ist unbefriedigend. Da diese Chips das DTACK-Signal unterstützen, wäre ein Umweg über das WAIT-Signal nicht sinnvoll. Auf der BUS-Platine sind noch einige Leitungen frei, wie zum beispiel PI, PO. Mit einigen Änderungen auf dem BUS und der CPU-Platine sollte dies möglich sein. Allerdings wird so der Z80 ausgeschlossen.

Kontakte
Das seit Mai '88 bestehende "Europäische Forum für OS-9" (EFFO) ist eine rechnerunabhängige und firmenneutrale Vereinigung von OS-9/68k-Anwendern. Gerade die erste Auseinandersetzung mit so einer komplexen Materie, die ein Multiuser- / Multitasking-Realtime-Betriebssystem nun einmal darstellt, bringt Probleme mit sich, die andere bereits gelöst haben. Die wichtigsten Ziele liegen deshalb in der Beratung und Hilfestellung sowie in der Förderung des Kontakte unter OS-9 Benutzern. Letzteres geschieht durch regelmässigen Diskettenversand. Um möglichst viel Public-Domain-Programme zur Verfügung stellen zu können, ist das Forum auf die aktive Mitarbeit seiner Mitglieder angewiesen. Zur Zeit werden verschiedene Rechner mit CPU 680x0 unterstützt. Dazu gehört natürlich auch der NDR-Klein-Computer!
Weitere Informationen sind erhältlich bei

EFFO
Postfach
CH-8606 Greifensee

Verkaufe betriebsbereiten NDR-Computer, der wahlweise mit dem CP/M68k-(Grundprogramm usw.) oder MS-DOS-Betriebssystem benutzt werden kann. Enthalten sind: 2 Laufwerke (3.5" und 5 1/4"), FLO2, Netzgerät, Bus mit 9 NDR- und 3 IBM-Steckplätzen, 19"-Gehäuse, CPU 68008, CPU 8088, 2*ROA-256k, 1*ROA-512 von H. Krutof, 2*ROA64, BankBoot, GDP 64k, KEY, IOE, PROMER, Hercules-IBM-Karte, V24-IBM-Karte mit Uhr, Cherry-Tastatur, IBM-Tastatur, TTL-Monitor, verschiedene Software und Literatur.
Preis komplett DM 1500,-, Einzelteile 50% unter Listenpreis.
Fritz Ellerweg, Rilkestr. 8, 4802 Halle, Tel.: 05201/4762

Verkaufe Baugruppen, einzeln oder komplett, mit Doku und Zubehör: Tastatur mit Zehnerblock (11072), KEY, CAS, ESKOP, Akustik-Koppler dataphon s21d-2, IOE, POW5 mit Trafo, EBASIC, HEBAS, gegen Gebot. Alle Baugruppen funktionstüchig und einsatzbereit. Jürgen Brummer, St. Martins-Weg 9, 7750 Konstanz, Tel.: 07531/15449

Fortsetzung der Kleinanzeigen auf Seite 30

Volker Stahl

CPU8088 - MS/DOS

Teil 5: AUTOEXEC.BAT

Entlich ist es soweit.....Ja, Sie haben richtig gelesen, Sie halten Teil 5 des Beitrages "CPU8088 - MS/DOS" in Ihren Händen! Vielleicht ist dieser Artikel bereits überflüssig geworden (seit Teil 1 ist schon eine lange Zeit vergangen....), was mich einerseits natürlich auch freut, doch für all' diejenigen, die bis jetzt mit ihrem NDR-PC nur Spiele machen, oder Briefe ausdrucken, habe ich mich entschlossen, diesen Kursus zu schreiben (mit der Hoffnung, das Sie, als `alter NDR-ler' auch mit Ihrem neuerworbenen NDR-PC viel Freude und so manche lange Nacht haben werden).

Für die Profi-User unter Ihnen, wird wohl der Begriff 'AUTOEXEC.BAT' kein Fremdwort mehr sein. Es zahlt sich also auch hier aus, wenn Sie sich schon in früheren Jahren intensiv mit Ihrem NDR-Computer-Betriebssystem befaßt haben! Für mich ist dies wieder mal ein neuer Beweis dafür, daß der NDR-Computer einen sehr guten Nährboden für die zukunftsorientierte Computertechnik ist. Jetzt soll's aber richtig losgehen.....

Die Dateien vom Typ '.BAT' gehören zu den sogenannten Stapeldateien (was auch schon das Kürzel .BAT verrät -> engl. batch).

Also ist unsere AUTOEXEC.BAT eine Stapeldatei. Doch was ist eine Stapeldatei? Allgemein gesagt ist sie eine vom Benutzer angefertigte Datei (ähnlich der CONFIG.SYS, siehe Teil 4!), die nacheinander Batch-Processing-Befehle abarbeitet.

Hier eine kurze Übersicht der BATCH-Befehle:

CALL: Hiermit kann eine andere Batch-Datei aufgerufen werden. Syntax: CALL file

ECHO: 1. Ein- bzw. Ausschalten der Bildschirmausgabe von den Batch-Befehlen. Syntax: ECHO ON/OFF

2. Ausgabe eines Textes während des Abarbeitens der Batch-Datei. Syntax: ECHO text

FOR %% IN DO: Wiederholte Ausführung eines Kommandos.

Syntax: FOR %%x IN(menge) DO befehl

Dabei ist 'x' eine Variable (nur Buchstaben!), 'menge' eine bestimmte Auswahl, z.B *.TXT und 'befehl' ein DOS-Kommando

Beispiel: FOR %%d IN(*.exe) DO dir

GOTO: Sprungbefehl zu einer vordefinierten Marke.

Syntax: GOTO marke (Marken werden durch einen Namen mit einem anschließendem ':' (Doppelpunkt) gekennzeichnet).

IF: Altbekannter Befehl: in Abhängigkeit einer bestimmten Bedingung Ausführung bzw. Unterlassung eines Befehls.

Syntax: IF [NOT] bedingung befehl

PAUSE: Die Abarbeitung der Batch-Befehle wird solange unterbrochen, bis eine Taste gedrückt wird. Wahlweise kann dazu noch ein Text mitausgegeben werden.

Syntax: PAUSE [text]

REM: Altbekannter Befehl: Kommentarzeile, die nur zur Übersichtlichkeit der Batch-Datei dienen soll.

Syntax: REM text

SHIFT: Erweiterung der Batchfiles-Variablen von 9 auf10. Als Variablebezeichner sind folgende zulässig: %0 - %9

Syntax: SHIFT

Die Anwendung bzw. Einsatzmöglichkeiten einer Batch-Datei (allg. Batch-Datei) werden wir anhand eines sehr einfachen Beispieles beantworten. Nehmen wir an, Sie kauften sich ein Textverarbeitungsprogramm mit dem Namen 'SUPERTEXTWRITER'. Wenn Sie das Text-Programm starten wollten, müßten Sie den sehr langen Namen: SUPERTEXTWRITER eingeben. Das kostet Zeit (und für manchen unter uns bestimmt auch kopfzerbrechende Tipparbeit, aber nicht aufgeben wenn Ihnen das 'Buchstaben-Suchen-Tippen' noch schwerfällt, auch mir erging es mal so; zur Übung können Sie ja mal diese LOOP abtippen..... (wenn Sie noch ungebunden sein sollten, und daher genügend Zeit dafür haben sollten!). Doch mit dem Batch-File 'STW.BAT' schaffen wir Abhilfe! In disem Falle würden wir mit EDLIN (siehe Teil 3) folgendenText unter dem Namen STW.BAT abspeichern:

```
REM (C)Copyright by Hugo Selbstschreib
CLS
ECHO SuperTextWriter Version 0.01 wird geladen!
ECHO Bitte warten.....
supertextwriter
CLS
ECHO SuperTextWriter ist vom Benutzer beendet worden!
ECHO Haben Sie Neuänderungen gesichert?
```

Wenn Sie nun im DOS 'STW' <ret> eintippen, so erscheint die nach dem ECHO-Befehl stehende Textpassage und das Programm SUPERTEXTWRITER.EXE wird geladen. Nach dem Beenden erscheint der Text der nach dem zweiten CLS steht, und die Batch-Datei STW.BAT ist vollständig abgearbeitet worden.

Was wir bis hierher besprochen haben gilt für alle Batch-Dateien (auch für die spezielle AUTOEXEC.BAT - Datei)! Doch worin unterscheidet sich die AUTOEXEC.BAT von den anderen Batch-Dateien? - Nun, der Unterschied liegt darin, daß die AUTOEXEC.BAT nach jedem Bootvorgang des Betriebssystems abgearbeitet wird, d.h. jedesmal wenn DOS geladen wird, wird die AUTOEXEC.BAT - Datei zuerst abgearbeitet, noch bevor der Benutzer irgendwelche Eingaben machen kann.

BOOT-VORGANG von MS/DOS:

1. DOS wird konfiguriert mit CONFIG.SYS (siehe Teil 4)
2. DOS-Befehls-Interpreter COMMAND.COM wird geladen
3. AUTOEXEC.BAT wird durchgearbeitet
4. Das Prompt-Zeichen A:> erscheint, und der Anwender kann DOS-Befehle eingeben.

Tja, und damit sind wir auch schon am Ende von unserem Ausflug in die PC - Computerwelt angekommen. Aber zum Abschluß habe ich noch etwas kleines für

Sie, damit Ihnen der Umgang mit Ihrem NDR-PC und dessen DOS erleichtert wird! - Eine kleine AUTOEXEC.BAT - Datei:

```
KEYB GR,437,KEYBOARD.SYS
ECHO OFF
PROMPT $P $G
CLS
VER
```

Anmerkung: Sollten Sie schon stolzer Besitzer einer Festplatte sein, so können Sie als erste Zeile in Ihrer AUTOEXEC.BAT den Befehl 'PATH=C:' einfügen. Wenn Sie jetzt nämlich auf Laufwerk A: arbeiten, und einen externen DOS-Befehl (Befehl, der auf der DOS-Programmdiskette ist, z.B. FORMAT.COM) eingeben, so erscheint nicht mehr die Meldung, es sei ein falscher Befehl, oder diese Datei existiere überhaupt nicht, sondern lädt den DOS-Befehl von C: (hier muß dann natürlich auch das ganze DOS drauf sein!).
Sie haben es geschafft! - Nach diesem letzten Teil meines DOS-Schnell-Kursus, sollten Sie eigentlich in der Lage sein, selbstständig mit dem Betriebssystem arbeiten zu können.
Zuletzt noch ein guter Rat: Ich verstehe ja, daß Sie in letzter Zeit sich weniger mit Ihrer CPU8088 befaßten, als daß Sie sämtliche DOS-Software 'durchspielten', doch denken Sie daran: Selber machen macht schlau! (..... Grundsatz meines ehem. Lehrers, der mich damit 'angestiftet' hat 1986, mit 14 Jahren, den NDR-Computer zu bauen, wofür ich ihm aber sehr dankbar bin).

Software-Unterstützung für Promer2

In der LOOP 24 wurde der PROMER2 bereits einmal vorgestellt, die technischen Daten haben bei vielen Anwendern Interesse hervorgerufen. Da er sehr universell einsetzbar ist, kamen viele Anfragen unserer Kunden nach der Softwareunterstützung, speziell für ihr individuelles System. Hier also eine Übersicht für alle PROMER2-Interessenten:

Z80 - EPROMSYSTEME
Für die Z80-Prozessorreihe wurde ein 8 kByte langes Steuerprogramm entwickelt, das sich relokativ im Adressraum der Z80 einsetzen läßt. Es benötigt lediglich ab der Adresse 8000h einen mindestens 2 kByte großen Arbeitsspeicherbereich. Das Programm-EPROM (2764) heißt "EPRO V1.1" (Best. Nr. 11537) und unterstützt selbstverständlich sämtliche Möglichkeiten des PROMER2. EPRO V1.1 hat als Besonderheit sämtliche Bildschirm- und Tastaturroutinen selbst enthalten und ist damit autark, was Betreibssystem-Hilfsroutinen anbelangt. Es kann also im RDK-Grundprogramm gleichermaßen eingesetzt werden, wie auch in FLOMON/ZEAT-EPROM-Systemen.

RDK-Grundprogramm:
das EPROM "EPRO" kann ab Adr. 2000h eingesetzt werden (ROA 64 oder SBC). Über den programminternen Arbeitsspeicher von 2 kB ab 8000h hinaus, muß natürlich noch genügend RAM vorhanden sein, um EPROM-Inhalte zwischenlagern zu können.

FLOMON V3.2 + ZEAT (A + B):
Da hier die Bank 0 mit RAM vollbestückt sein muß, kann EPRO V1.1 entweder auf der Bank E (Adr. 0) installiert werden, oder auf der SBC3 selbst. In beiden Fällen wird das Programm über das FLOMON-Menü aufgerufen. EPRO kann nur den Speicher der augenblicklichen Bank ansprechen. Inhalte von anderen Speicherbänken müssen mit Verschiebe-Hilfsroutinen transportiert werden. Dazu eignet sich der FLOMON-Befehl "BANK" an Adr. F05Bh.

Z80 - DISKETTEN SYSTEME
Für CP/M 2.2 wurde das Programm "DPRO V1.1" (Best. Nr. 61461) entwickelt. Funktional ist es identisch mit der EPROM-Version. DPRO ist ebenso wie EPRO relokativ (!) geschrieben, benötigt aber den festge-

```
,
F05B    JP BANK         HL=Adresse in der Quellbank 0..EFFF
                        DE=Adresse in der Zielbank  0..EFFF
                        C=Nummer der Quellbank      0..F
                        B=Nummer der Zielbank       0..F
                        Es werden 128 Bytes transportiert.
                        Dabei muß von Adresse F000 bis FFFF
                        auf der jeweiligen Bank auf jeden Fall
                        ein RAM-Speicher sein. Sonst wird
                        ein Carry als Ergebnis geliefert.
                        Der Bereich F000 bis FFFF ist immer
                        für das Transportprogramm reserviert.
```

Bild 1: FLOMON-Befehl BANK zum Verschieben von Speicherinhalten über mehrere Banken hinweg.

legten Speicherbereich von 2200h - 27FFh als internen Arbeitsspeicher. EPROM-Inhalte können als Dateien gelesen und gespeichert werden. Im Lieferumfang von DPRO V1.1 ist auch das Programm EPRO V1.1 enthalten, so daß der Anwender beide Varianten zur Auswahl hat.

680xx - SYSTEME

Für diese Prozessorserie ist die Programmierroutine im Dombrowski-Grundprogramm ab V6.2x enthalten. Aus dem Grundprogramm-Menü können die alte PROMER-BAugruppe oder die PROMER2-Baugruppe bedient werden. Bei Einsatz des PROMER2 werden nach Möglichkeit optimierende Algorithmen angeboten. Auf die System-Routinen "Lesen" und "Programmieren" kann über die TRAPs 15 und 16 auch aus Anwenderprogrammen heraus zugegriffen werden.

IBM-KOMPATIBLE

In dieser Gruppe treffen sich der **mc modular AT** und der NDR-Computer mit CPU8088 wieder.

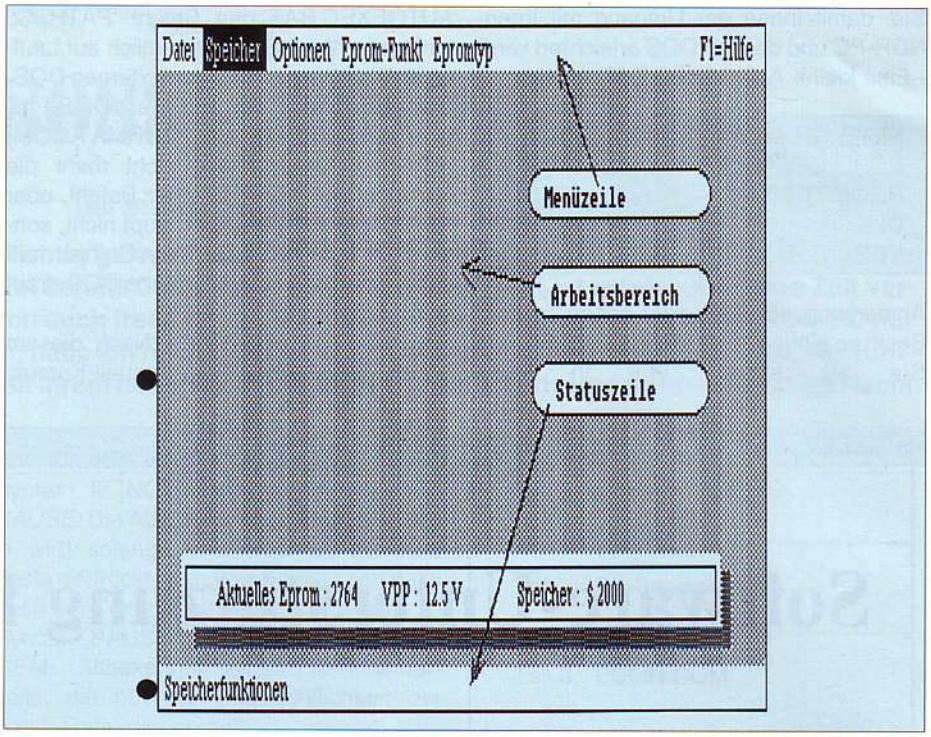


Bild 2: Das Eröffnungs-Fenster der PROMER-Software unter DOS

Hier ist bereits im Lieferumfang des PROMER2-IBM ein sehr leistungsfähiges Hilfsprogramm "PROMER.EXE" (bei separater Bestellung: 61452) enthalten. Die Bedienung wurde nach den neuesten Erkenntnissen der Software-Ergonomie gestaltet. Die Oberfläche ist nach der SAA (Software Application Architecture) gestaltet worden und läßt sich deshalb genauso elegant bedienen, wie die bekannte grafische Oberfläche WINDOWS von Microsoft.

Neben den elementaren EPROM-Funktionen, wie "Lesen", "Programmieren" und "Überprüfung" steht leistungsfähiges Dateihandling, sowie umfangreiche Speichermanipulationen zur Verfügung: Hex-Dump, Füllen und Suchen/Kopieren.

Kleinanzeigen

Mobile Workstation als Parallel-System konzipiert:
68020+FPU / 80386SX - Komplettsystem in einem Gehäuse. Im Gehäuse ist ein Hercules-Monitor integriert. Beide Subsysteme sind einzeln oder parallel einsetzbar, wobei Monitor und Tastatur auf jedes System umschaltbar sind. Evtl. kann auch das 68020-System mit Gehäuse alleine abgegeben werden. NP für das Komplettsystem mit reichhaltiger Software ca. 15000.-DM, VHB 8700.—DM. Christian Karman, Tel.: 089/6126477

Verkaufe wegen Systemwechsels: Monitor (Grün), KEY, GDP64k, CPU68k, DRAM128k, Tastatur + ehäuse, 3 ROA64, FDC, 128k stat. RAM, CENT2, BUS4A/16Bit, Floppy 5 1/4" + 3 1/2" + Kabel, NE3, Druckerkabel, Pascal/S, Grundprogramm 4.3, JADOS, Schach, Jedi, alle LOOP, Literatur. Preis VHB 2600,- DM (neu ca. 4600,- DM).
Kai Ruppert, Rebenring 63 (Zimer 16), 3300 Braunschweig, Tel.: 0531/342001 (lange klingeln lassen!)

Verkaufe: NDR-Computer (Z80) mit BUS3, FLO3, KEY3, ROA256, ROA64, Netzteil NE4, Gehäuse, 1 LW, ZEAT, Literatur u. a. Bauteile (IOE, HEXIO) wegen Systemwechsel meistbietend zu verkaufen. Ab 16.00 Uhr. Rudolf Murer, Sontheimer Landwehr 50, 7100 Heilbronn

---

Verkaufe: Akustik-Koppler 150 DM, Bus-Test 120 DM, Prommer mit POW 22/26/V1 90 DM, POW5V 15 DM, Ein/Aus 150 DM, EPORMs 2764 ESPS2, EGRUND2, EFLOMON je 20 DM, MSPAKET 150 DM, WINDOW 35 DM,. SPS-Comp. 150 DM, (Format 5 1/4", 80 Spuren, Original-Software CP/M-80). Edgar Würth Tel.: 0711/752839

Verkaufe 2xROA64 voll bestückt,KEY, PROMMER, IOE mit CENT, diverse TTL-ICs, stat. RAMs, EPROMs alles komplett VB 300,- DM (geprüfte, einwandfreie Ware). Peter Fleskes Tel.: 08801/2142.

Wegen Systemwechsel zu verkaufen: NDR-Computer, CP/M 2.2, PC-Gehäuse, Floppy 5 1/4", Sanyo Monitor 14 Zoll grün, BUS3, NE2, IOE2, Prommer, 2xROA64k, etc. Literatur und Software, kompl. 700,- DM. Tel.: 05532/5552 ab 18.00 Uhr

---

NKC-User sucht in Österreich Anwender des NKC-68000 und mc 68000. Gernot Shimetta, Rembrandt-Gasse 6, A 8010 Graz

BESTELLKARTE

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

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10061	LOOP-Abo	25,-

Adresse (umseitig) nicht vergessen!

Datum

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

BESTELLKARTE

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

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10061	LOOP-Abo	25,-

Adresse (umseitig) nicht vergessen!

Datum

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

Neue Preise - Neue Produkte

#11566	VGA-Paket Farbe, komplett mit 8 Bit VGA-Karte und VGA-Colormonitor 14"	DM 949,-
#11309	VGA-Profi-Paket Farbe, mit 16 Bit VGA-Karte und NEC Multisync	DM 1798,-
#11303	NDR-BUS-KOPP, Fertiggerät PC/XT-Steckplätze am NDR-BUS	DM 198,-
#11503	PROMER2 Fertiggerät Das Universal-Programmiergerätfür den PC/XT/AT, incl. SW	DM 428,-
#11532	DPROM Z80-Diskettensoftware (5 1/4") zum PROMER2-NDR-Ausführung	DM 39,-
#11537	EPROM Z80-Epromsoftware (2764) zum NDR-PROMER2	DM 35,-
#70236	SCSI-NDR Fertiggerät SCSI-Festplatten-Anschluß (bis 40 MB) für NDR-Systeme (8...32) Bit.	DM 69,-
#11023	BIOSZ80 V2.0 (5 1/4"), CP/M 2.2 lernt den Umgang mit einerSCSI-Platte	DM 49,-
#11376	Festplatte SCSI 20 MB/65 ms Segate ST225-N	DM 769,-
#11377	Festplatte SCSI 40 MB/28 ms Segate ST251-N	DM 1198,-
#11570	Festplatte SCSI 111 MB/15 ms Segate ST1126-N(3 1/2")	DM 1598,-
#10952	Anschlußkabel 0.5m für SCSI-Platte an SCSI-NDR Baugruppe	DM 39,-
#10292	JADOS V3.5 auf 5 1/4"-Disk incl. umfangreicher Handbücher jetzt mit Festplattenunterstützung	DM 159,-
#10690	JADOS V3.5 auf 3 1/2"-Disk	DM 159,-

Sonderangebote - Software - Disketten - EPROMs

NDR-Software zu Schleuderpreisen, teilweise die allerneuesten Versionen:

#70266	Grundprogramm V 6.0 für 68008 (27256er EPROMs) incl. Doku	: DM 89,-
#70268	Grundprogramm V 6.0 für 68000 (27256er EPROMs) incl. Doku	: DM 89,-
#70265	EGOSI V3.1 für 68008 (2764er EPROMs)	: DM 59,-
#70264	EGOSI V3.1 für 68000 (2764er EPROMs)	: DM 59,-
#70263	EPASCAL V0.4 für 68000 (2764er EPROMs)	: DM 59,-
#70262	RL-BASIC V3.4 für 68008 (2764er EPROMs)	: DM 99,-
#70261	RL-BASIC V3.4 für 68000 (2764er EPROMS)	: DM 99,-
#70275	JADOS V3.1-02 5 1/4"-Disk incl. Doku	: DM 99,-
#70276	JADOS V3.1-02 auf 3 1/2"-Disk incl. Doku	: DM 99,-

Bitte Porto nicht vergessen

ANTWORT

GRAF computer

Graf Elektronik Systeme GmbH
Postfach 1610

8960 Kempten

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vorkasse
☐ Bankeinzug

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

BLZ _____ Konto-Nr. _____

Bank: _____

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

Datum _____ Unterschrift _____

Bitte Porto nicht vergessen

ANTWORT

GRAF computer

Graf Elektronik Systeme GmbH
Postfach 1610

8960 Kempten

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vorkasse
☐ Bankeinzug

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

BLZ _____ Konto-Nr. _____

Bank: _____

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

Datum _____ Unterschrift _____