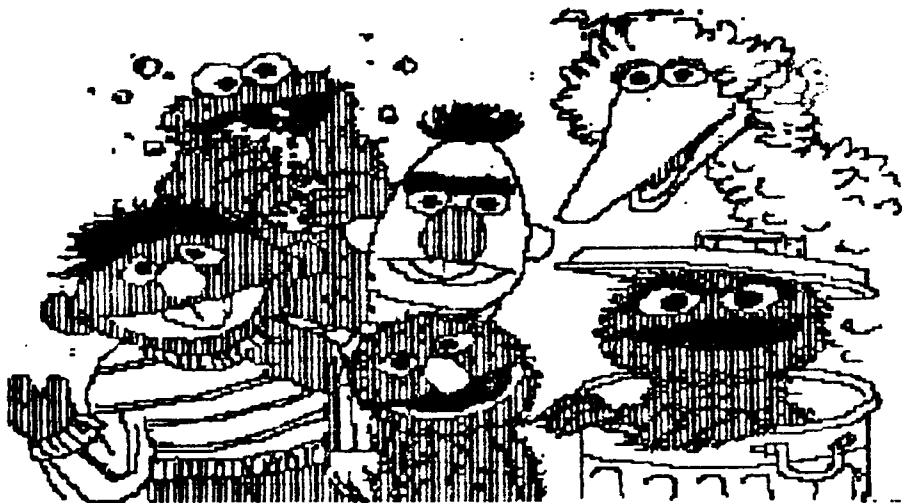


# GENIE USER Club und Colour-Genie T R S B O B R E M E R H A V E N

CLUB-INFO  
CLUB-INFO  
CLUB-INFO  
CLUB-INFO

---



**3. JAHRGANG | 06. AUSGABE**

Red.: Peter Spieß, Trugenhofenerstr. 27, 8859 Rennertshofen 1  
\* Sortiert von: Edeltraud \*\*\* Auflage: 070 Exempl. \*\*\*\*\*



# Inhalt

## C l u b - I n f o

1	Internes / Flohmarkt
2-5	Lissajoussche Figuren v. A. Sopp
6-7	VisiCalc Modifikation v. O. Stark
8	BASICODE v. P. Kröher
9-11	Minimaltreiber für die HRG1b v. A. Sopp
12-15	PILOT80 v. H.-G. Küster
16	Neues vom Rechner
17	Zu P. Kröher's Assemblerliste v. A. Sopp
18-22	APL80 v. H. <del>Schermann</del> Offermann
23-25	RENEW f. L2-BASIC v. J. Seelmann-Eggebert
26-27	Genietext mit Sonderzeichen v. A. Sanz
28-30	Springen - aber wie ? v. A. Sopp
31	Homberger Nachlese v. Kajott
32	Sei Dein eigener Diktator v. Kajott
33	"MOLEKRIS" v. Kajott
34	Fragen, Antworten und Tips

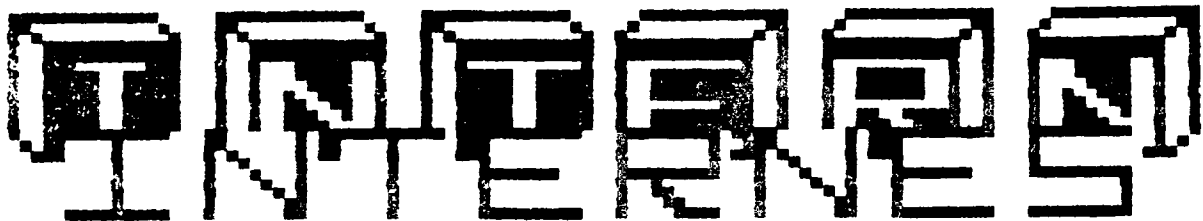
### Geburtstagsecke:

Im Juni können folgende Mitglieder Ihren Geburtstag feiern:

Siggi Bach	Manfred Blaschek
Francisco Otey	Gregor Thalmeier

Herzlichen Glückwunsch !!!





# INTERNES VOM BETREUER

\*\*\* Das Clubtreffen ist vorbei; ich möchte, ohne dem ausführlichen Bericht von Kajott (erscheint im nächsten Heft) vorzugreifen, mich bei allen Teilnehmern recht herzlich bedanken. Bei der Nachbesprechung in kleinster Runde (Holger, Wolfgang und ich) kamen nur positive Ergebnisse heraus. Der Entschluß, dieses Treffen im nächsten Jahr zu wiederholen, steht auf jeden Fall fest. Bedanken möchte ich mich in besonderer Weise bei Paul-Jürgen Schmitz, der sich bei mir im Namen aller Clubmitglieder in Form einer Urkunde und einer Flasche Weinbrand für die Betreuung des Clubs bedankt hat. Ich war so überrascht, daß ich kein Wort mehr herausgebracht habe. Die Urkunde hat den Ehrenplatz über meinem Schreibtisch erhalten, die Flasche einen anderen.

=====

## Flohmarkt

\*\*\* Zu verkaufen sind: 2 TRS80 Mod. 1/L2 mit CP/M und je 2 Lw.  
Anfragen an Dr. Zuchold Tel.: 06120/1634  
1 TRS80 Mod. 4 Tel.: 06131/685816

\*\*\* Diskettenangebot: (gültig bis 25.06.1985)

10 Disketten 1S incl. <u>Diskettenkasten</u> (mit Deckel und Schloß)	80,-DM
10 Disketten 1D                ="=	82,-DM
10 Disketten 2D                ="=	94,-DM
10 Disketten 1D 96 tpi        ="=	94,-DM
10 Disketten 2D 96 tpi        ="=	106,-DM

Disketten: 5 1/4" DISKY   Kasten: ABA M35 mit Deckel und Schloß  
Nach dem 25.06. sind die Preise etwas höher. Bestellungen sind an  
Peter Spieß, Trugenhofenerstr. 27, 8859 Rennertshofen 1 zu richten.

\*\*\* Wegen Nichtbezahlung des Beitrages wurden ausgeschlossen:  
Harald Thom, Torsten Vollmer. Rudolf Ring ist ausgetreten.

Neue Mitglieder ab 01.06.85:

Gerhard Loose

Friedrich Horn

Hartmut Obermann

1 6185

## Lissajousche Figuren

Der französische Physiker J. A. Lissajous konnte nichts für seinen unaussprechlichen Namen. Noch weniger können es die nach ihm benannten Figuren. Es sind Überlagerungen zweier aufeinander senkrecht stehender periodischer Schwingungen. Man kann sie sich wie in Abb. 1 vorstellen. Wenn sie sich überlagert haben, sieht das aus wie in Abb. 2. Eine alte Ellipse lohnt aber noch nicht unbedingt den Aufwand. Wir werden sehen, daß sehr schöne, sehr komplizierte Figuren erzeugt werden können.

Dazu etwas Theorie. In einem Graphen mit einem x/y-Achsenkreuz würden die Koordinaten diesen beiden Gleichungen folgen:

$$x=\sin(i), y=\sin(i+90^\circ)$$

Wenn die beiden Kurven um einen bestimmten Winkel gegeneinander phasenverschoben sind, kann erst eine in sich geschlossene Kurve mit einem Innenraum entstehen. Andernfalls wäre das Resultat ein schlichter Querschnitt.

Diese Phasenverschiebung kann variiert werden. In Abb. 3 beträgt sie nur  $30^\circ$ . Da offenbar die Verschiebung um  $90^\circ$  den Kreis bzw. die unverzerrte Ellipse ergibt, kann man als Grundgleichung statt des Sinus den Cosinus für eine der beiden Koordinaten eingeben, denn er ist quasi ein phasenverschobener Sinus:

$$x=\sin(i), y=\cos(i)$$

Nicht nur die Phasenverschiebung macht die entstehende Kurve interessanter. Es können auch die Argumente der Winkelfunktionen unterschiedlich schnell steigen. Oder anders gesagt, die Perioden können unterschiedlich lang definiert werden:

$$x=\sin(i/2), y=\cos(i/3)$$

Dabei kommt so etwas wie die Abb. 4 heraus. Schließlich können durch eine phantasievolle, meinetwegen auch völlig verrückte Kombination von Phasenverschiebungen, Periodendifferenzen und sogar Amplitudenstauchungen und -spreizungen die abartigsten Kurven geschrieben werden.

Das BASIC unserer Computer hat für den SET-Befehl immer gern x und y gleichzeitig im Argument. Deshalb kommt es nun darauf an, einen BASIC-Algorithmus für die Lissajous-Figuren zu finden. Listing 1 ist der umständliche, aber hoffentlich einigermaßen nachvollziehbare Versuch, seine Entwicklung zu zeigen. Nach diesem Programm entsteht gemäß den Bildschirmproportionen eine Ellipse (Abb. 5). Das Programm geht von der normalen Klötzchengraphik aus, deren Nullpunkt in der linken oberen Ecke steht, und die eine Matrix von  $128 \times 48$  Pixels hat.

Nachdem in Zeile 10 der Bildschirm gelöscht ist, wird zur Bequemlichkeit  $1^\circ$  definiert. Und zwar so: Der Tangens von  $\pi/4$  ist 1. Folglich ist der Arcustangens von 1 gleich  $\pi/4$ . Die Kreiskonstante ist demnach  $4\pi \arctan 1$  (BASIC-Schreibweise:  $4*ATN(1)$ ). Also ist ein 180stel davon  $1^\circ$ . Dieser Wert wird in die Variable G geladen. Um das Bogenmaß, nach dem die Winkelfunktionen in BASIC verlangen, brauchen wir uns fortan nicht mehr zu kümmern.

Um nun Graphikpunkte zu setzen, werden ab Zeile 20 alle Werte von  $0-360^\circ$  durchlaufen. Dazu wird zunächst in Zeile 30 durch Multiplikation mit G aus dem Ereigniszähler I ein echter Gradzähler gemacht. Dieser Schritt wäre nicht unbedingt nötig, aber so entsteht die Kurve ordentlich der Reihe nach. Sodann werden die vorläufigen Koordinaten X1 und Y1 errechnet. Die Phasenverschiebung liegt diesmal spaßeshalber bei X.

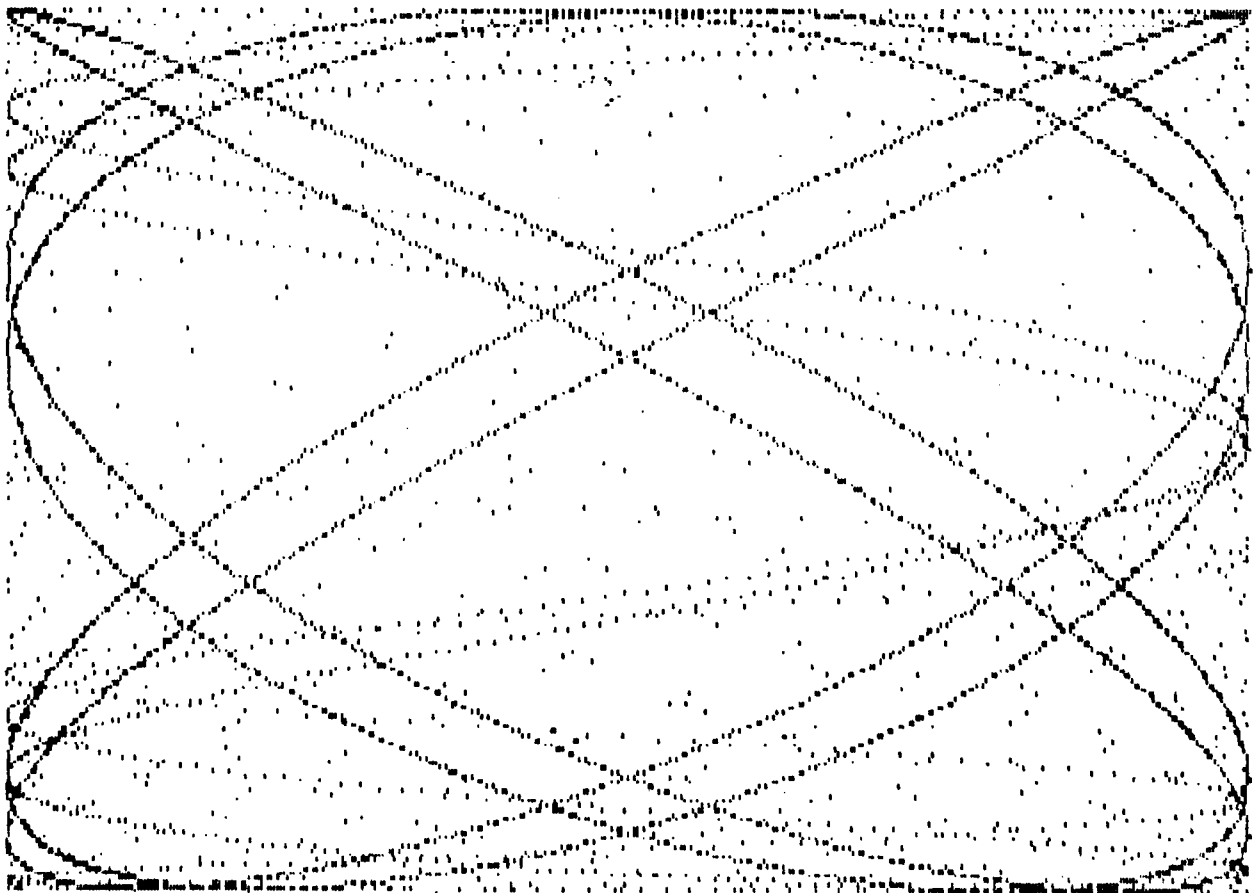
Bekanntlich hat der Sinus seine Extrema bei -1 und +1. Das ist eine Amplitude von 2. Der Bildschirm verträgt aber Amplituden von 128 waagrecht und 48 senkrecht. Deshalb wird das Resultat in X2 und Y2 je mit der Hälfte der Bildschirmkoordinaten multipliziert. Da hier noch negative Werte entstehen, die zu einem Fehler führen würden, wird nun noch in Z:

80 und 90 besagte Hälfte hinzuaddiert. Die SET-Argumente X und Y sind nun endlich mundgerecht aufbereitet.

In Listing 2 ist dieses Programm ein wenig gestreamlined. Das Statement zur Errechnung eines Altgrads ist gekürzt. Die Hälften der Bildschirmkoordinaten werden den Variablen A und B zugewiesen. Der phasenverschobene Sinus ist durch den Cosinus ersetzt. Alle ganzzahligen Konstanten sind mit dem %-Zeichen als Integers definiert, um eine gewisse Beschleunigung zu erhalten. Schließlich gibt noch Listing 3 dasselbe für die hochauflösende Graphik HRG 1b wieder. Hier liegt eine Matrix von 384X192 Punkten zugrunde. Es wird die Syntax benutzt, die für meinen ebenfalls im Info erschienenen Treiber gilt. Da die Kurve bei dieser feinen Auflösung Lücken aufweisen würde, wird sie durch STEP.2 dichter geschrieben.

Es ist eine ganze Anzahl von Lissajous-Programmen im Umlauf. Des Meinigen hätte es wahrhaftig nicht mehr bedurft. Hier kam es mir jedoch darauf an, dem Anwender zu zeigen, was da überhaupt passiert. Nachdem das nun (hoffentlich) klar ist, kann er durch beliebige Variation des SET-Arguments seinem künstlerischen Drang freien Lauf lassen.

Arnulf Sopp



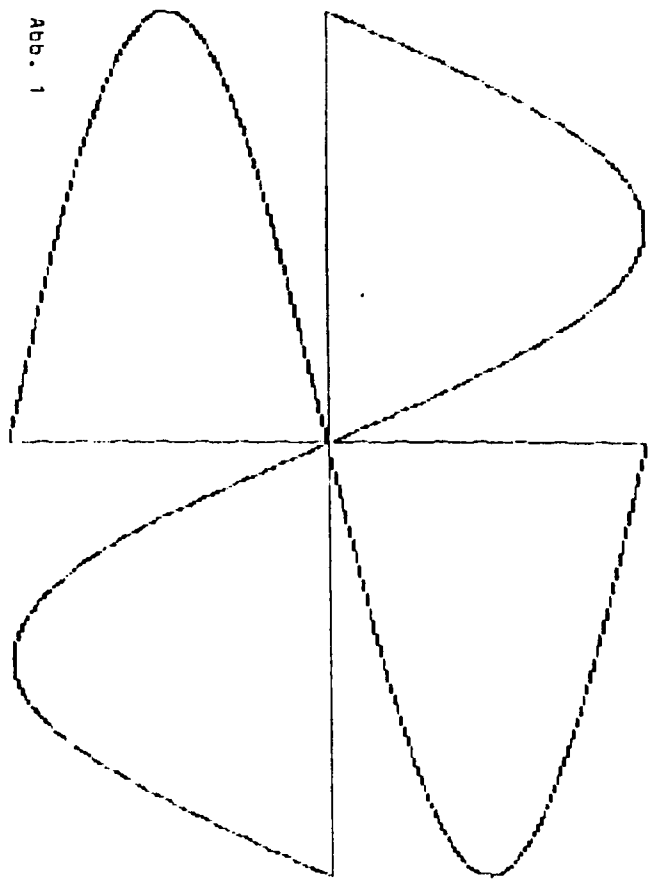


Abb. 1

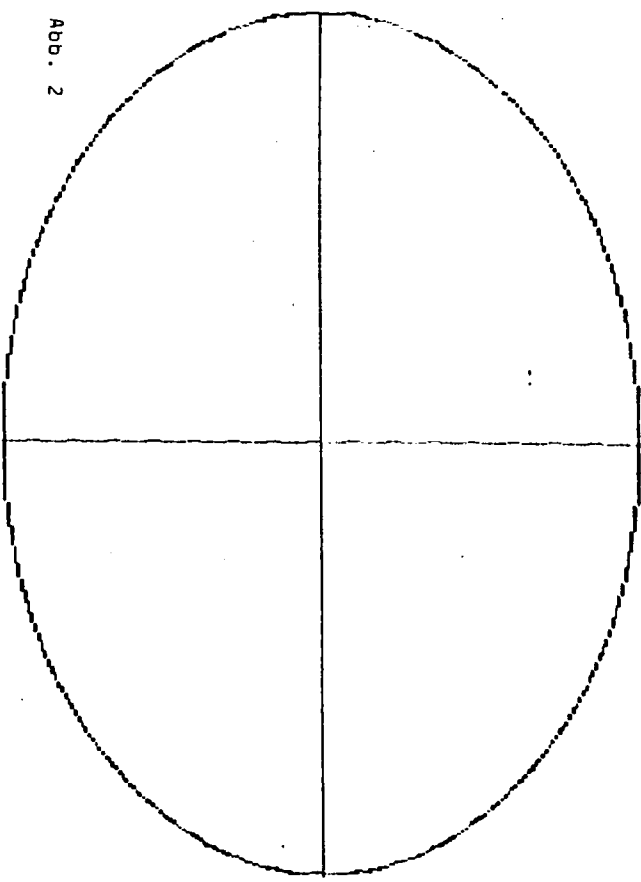


Abb. 2

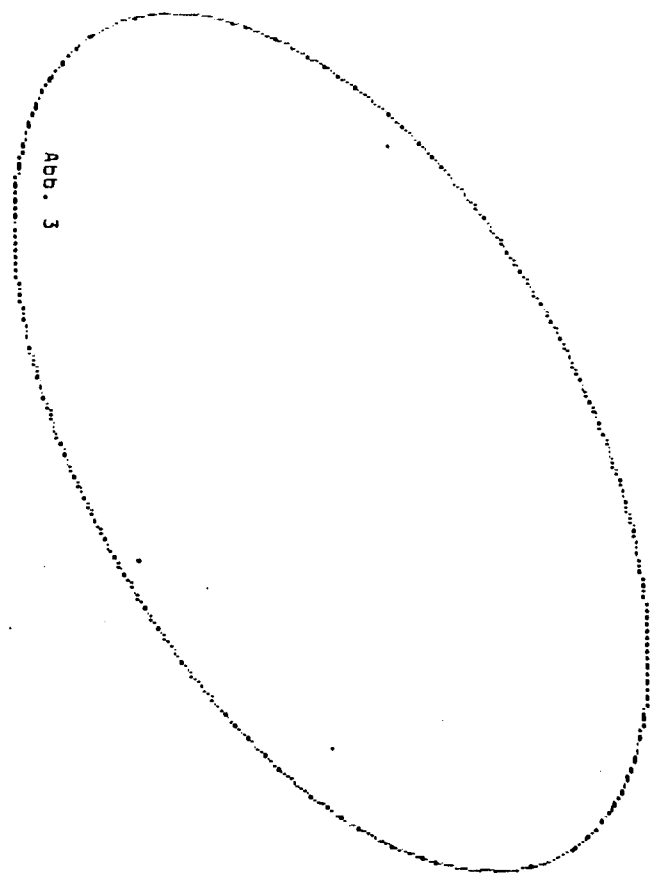


Abb. 3

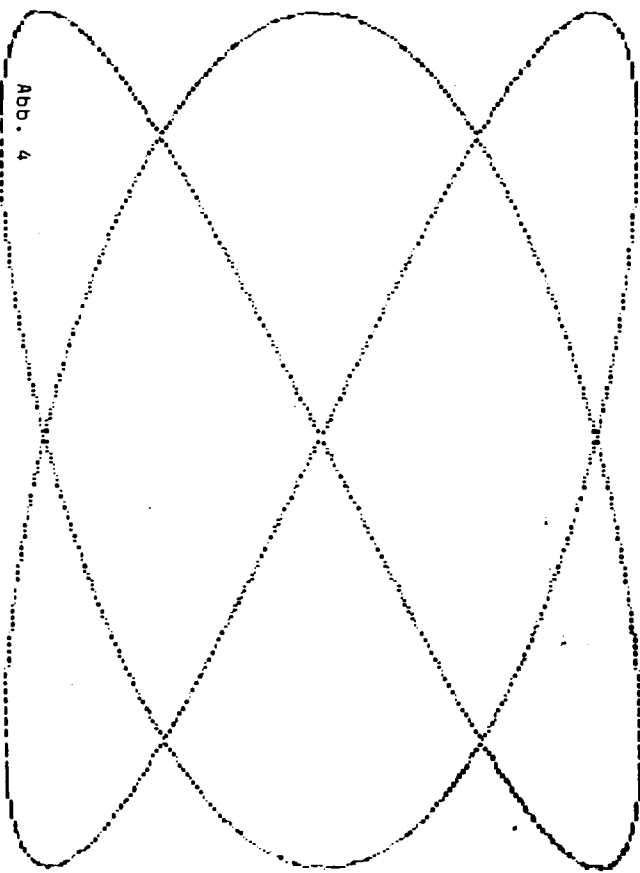


Abb. 4



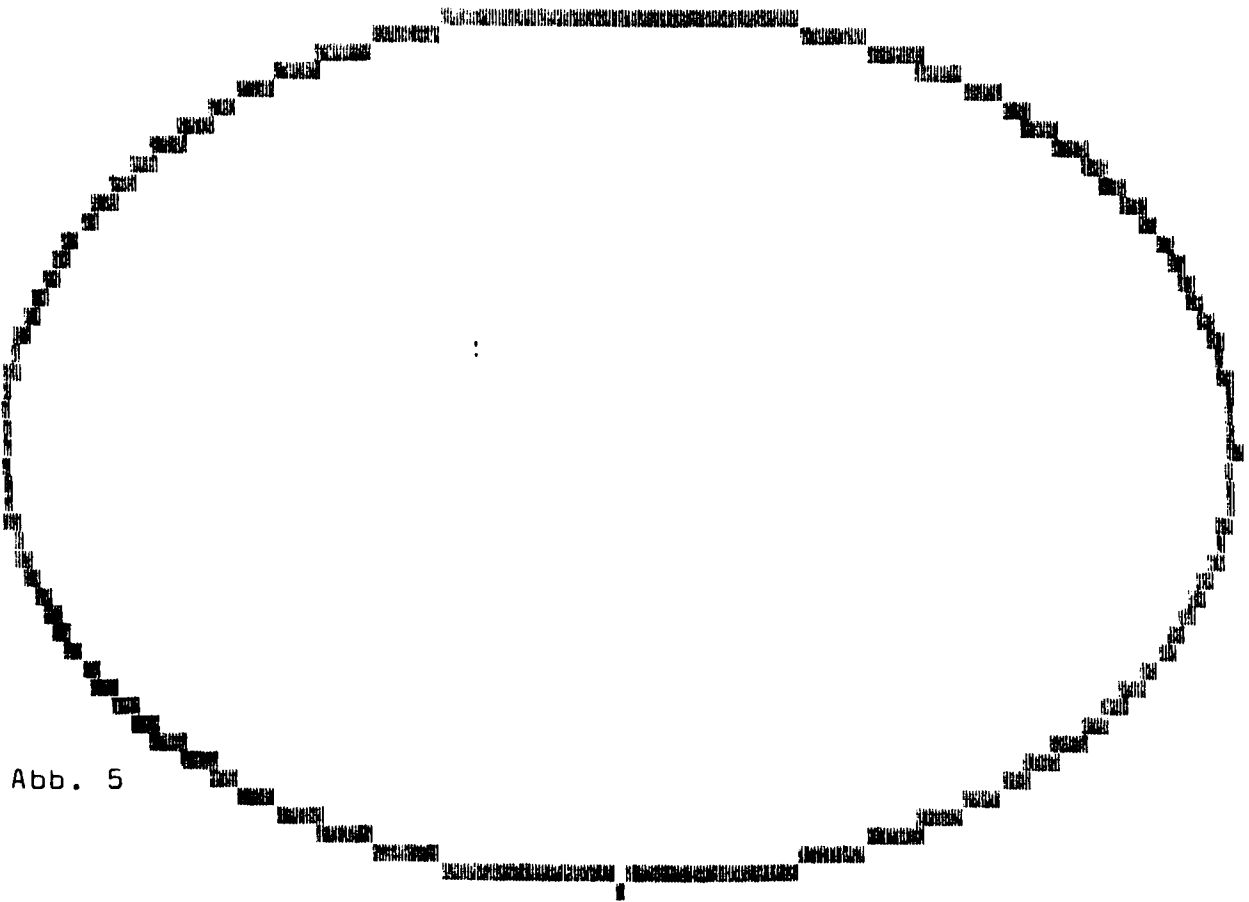


Abb. 5

List. 1

```

10 CLS:G=4*ATN(1)/180:REM 1 Altgrad definieren
20 FORI=0TO360:REM von 0 - 360 Altgrad, STEP 1 Altgrad
30 S=I*G:REM Zählvariable in Winkel umdefinieren
40 X1=SIN(S+90*G):REM Abszisse um 90 Grad phasenverschoben
50 Y1=SIN(S):REM Ordinate nicht verschoben
60 X2=63.5*X1:REM Werte -1 - +1 auf -127/2 - +127/2 spreizen
70 Y2=23.5*Y1:REM dto. Y auf -47/2 - +47/2
80 X=63.5+X2:REM Werte -127/2 - +127/2 auf 0 - 127 schieben
90 Y=23.5+Y2:REM dto. Y auf 0 - 47
100 SET(X,Y):REM Graphikblock setzen
110 NEXT:REM nächster Graphikblock
120 IFINKEY$=""120:REM Graphik stehenlassen

```

List. 2

```

10 CLS:G=ATN(1)/45:A=63.5:B=23.5:FORI=0TO360:S=I*G:SET(A*COS(S)+A*B*SIN(S)+B):NEXT
:REM Kurve zeichnen
20 IFINKEY$=""20:REM Graphik stehenlassen

```

List. 3

```

10 CLS:CMD"CLS.G":OUT1,0:G=ATN(1)/45:A=191.5:B=95.5:FORI=0TO360STEP
.2:S=I*G:SET(A*COS(S)+A*B*SIN(S)+B):NEXT:REM Kurve zeichnen
20 IFINKEY$=""20:REM Graphik stehenlassen

```

## VISICALC - MODIFIKATION

Es störte mich schon lange daß man bei VC3/CMD mit /SL Pfeil rechts das uns nicht unbekannte Geräusch der Datenübertragung aus dem Lautsprecher zu hören ist, es werden aber keine Files am Bildschirm angezeigt.

Wie man Files auf dem Bildschirm bekommt ist ja bekannt und zwar mit /SL DFG 1,1 ENTER (gewünschten File aussuchen) CLEAR ; ENTER CLEAR.. (Cursor auf linke Seite bringen) FILENAME eingeben und ENTER. Hurra es wird geladen. Aber falls man aus Versehen die PFEIL-RECHTS-TASTE drückt, ist mit beiliegender Modifikation das lästige Geräusch beseitigt, ebenso der rechts oben am Bildschirm angezeigte freie Speicherplatz '13' wird auf '18' erweitert.

Diese Änderungen laufen auf Genie I sowie auf Genie IIs, wobei bei Genie IIs bei einer Speichererweiterung von 192 KByte auch nicht mehr als '18' für Daten frei sind.

```

DRV 00 FDE5 CD39 44FD E1DD E128 0CF5 3E00 32A3 ...9D....(>..2.
1 10 B5CD F7A5 F137 C93E 0021 D6B5 0600 7723 .....7.>.!.....w#
1H 20 10FC B7C9 2A7F B5E9 2AA1 B53A A3B5 3D20 .....*.....=.
30 1C11 A4B5 DDE5 FDE5 CD36 44FD E1DD E128 .....6D....(
DRS 40 07F5 CDF7 A5F1 37C9 3E00 21D6 B532 A3B5 .....7.>.!..2..
186 50 7E23 22A1 B5B7 2007 CDF7 A53E 0337 C9B7 0#"...>..7..
BAH 60 C9B7 C93A B5B5 FE00 2002 37C9 3AB6 B5B7 .....7.1....
70 C011 A4B5 DDE5 FDE5 CD2C 44FD E1DD E128 .....D....(
80 06CD F7A5 C3AB A4CD 50A6 B7C9 2288 B5CD .....F..."....
90 35A4 3ABF B5FE FF20 1606 00CD 4BA7 D0CD 5.....H...
A0 3DA4 3B06 047B FE04 3BF1 3E03 3718 0447 =.B...x...B.>..7..G
B0 CD4B A7C9 0E01 CD3D A430 043E 01F2 50A7 .H.....=.0.>..P.
FRS C0 0337 C9C5 21D6 B6DD E5FD E53E 08B7 FDE1 .7...!.....>....
86 D0 DDE1 2056 21D6 B67E B728 4D7E FE3A 2B07 ...V!..B.(MB.:(.
56H E0 FE2F 2B03 231B F4FE 3A2B 1A23 ED5B BAB5 ./(.#....: (.#.A..
F0 0603 1AFE 2020 057E FE3A 1B07 BE20 0413 .....B.:.....

```

DRV	00	01F2	0052	3160	B4ED	7354	B3ED	7D54	B3CD	...R1'...sT...ät...
1	10	8952	CD70	52CD	24AA	CDB8	52CD	A98E	2A42	.R.pR.\$...R...*I
1H	20	407D	E6F8	6F22	54A0	218C	B711	0700	197D	8ü...o"T...!.....ü
	30	E6F8	6F22	56AD	CDEA	8C06	010E	01CD	308C	..o"V.....0.
DRS	40	2273	ADCD	498A	3005	CDBA	9D1B	FB3E	09CD	"s..I.0.....>..
100	50	FEA0	3E43	FD77	A03E	01DD	77AE	DD77	AFCD	..>C.w.>..w.w..
64H	60	7052	CD64	52C3	956D	CD6B	A421	36B2	2234	pR.dR...m.k.!6."4
	70	B2C3	F976	DD21	6DAE	FD21	C5AD	DD36	AAFF	...v.!m...!...6..
	80	DD36	AB40	DD36	B000	DD36	B1B0	C93E	0021	.6.8.6...6...>.!
	90	45AD	11A9	00CD	B052	21ED	AD11	4700	CDB0	E.....R!...G...
	A0	5221	33AE	1197	04CD	B052	2A42	4011	8CB7	R!3.....R*Is...
	B0	B7ED	52EB	7723	1BCB	7A2B	F9C9	3E00	3260	..R.w#.z(..>..2'
FRS	C0	B432	61B4	326B	B5CD	5055	C3AA	543E	0032	.2a.2k..FU..T>..2
0	D0	61B4	C922	67B4	3EFF	3266	B4C9	F5E5	D5C5	a.."g.>..2f.....
0H	E0	DDE5	FDE5	CD7C	55FD	E1DD	E1B7	C4F0	52C1	.....0U.....R.
	F0	D1E1	F1C9	01F2	F052	2161	B4FE	0120	05CD	.....R!a.....
DRV	00	F182	575D	DD7E	8FDD	968D	4FDD	7E8A	DD96	..WÜ.0....0.0...
1	10	94D6	0247	CDA8	ABCD	5B9E	3E00	DD77	85F5	...G....A.>..w..
1H	20	CD96	77F1	3CDD	BE8B	38F2	C36E	9EDD	7E8B	..w.<...8..n..B.
	30	3DDD	7785	FD7E	93ED	445F	0E02	C3EB	A3DD	=.w..B..D.....
DRS	40	7E95	3CDD	7785	FD5E	930E	01DD	7E95	814F	B.<..w..^....B..0
183	50	0600	2133	AE09	6EDD	668C	7D83	5F54	DD7E	..!3..n.f.ü..T.B
B7H	60	8BDD	9695	D602	280F	473E	00FD	8693	10FB	.....(G>.....
	70	4FDD	468A	CDE1	ABCD	5B9E	DD7E	8A3D	DD77	O.F.....A..B.=.w
	80	84CD	9677	DD35	84F2	21A4	C36E	9EF5	3EFF	...w.5...!..n...>.
	90	327D	B5F1	C9F5	3E00	327D	B5F1	C9F5	3A40	2ü.....>..2ü.....5
	A0	38E6	04CC	35A4	C42D	A428	0800	F137	1B08	8...5...-.(...7..
	B0	CD35	A4E1	D1C1	F1B7	C9DD	E5FD	E5CD	2B00	.5.....+.....
FRS	C0	FDE1	DDE1	B7C9	C9C3	2D40	C93E	0032	85B5	.....-8.>..2..
83	D0	C9CD	50A6	C9F5	3A85	B5FE	0028	05F1	3E14	..F.....(..>..
53H	E0	01F2	80A4	37C9	F132	85B5	C3BF	A432	85B5	....7..2.....2..
	F0	ED53	8AB5	11A4	B506	0079	B728	02ED	B03E	.S.....y.(...>
DRV	00	0D12	21A4	B5CD	3AA8	300A	3E13	F5CD	50A6	..!.....0.>...P.
1	10	F1C3	D3A9	200D	3A86	B5B7	2007	CD50	A63E	.....P.>...
1H	20	1037	C93A	86B5	B720	7D3E	0032	E8B6	11A4	.7.....ü>..2.....
	30	B506	0021	D6B5	3A85	B5FE	8020	1DD0	E5FD	...!.....
DRS	40	E5CD	2444	FDE1	DDE1	2806	CDCB	A7C3	ABA4	..\$D....(.....
184	50	21BC	A622	7FB5	3DE1	1837	DDE5	FDE5	CD24	!...>..7.....\$
B8H	60	44FD	E1DD	E128	22CD	C8A7	CDD3	A9FE	0128	D....(".....(
	70	0237	C911	A4B5	0600	21D6	B5DD	E5FD	E5CD	.7.....!.....
	80	2044	FDE1	DDE1	C2AB	A421	6AA6	2281	B53E	.D.....!j.."..>
	90	0032	A3B5	21D6	B522	A1B5	2103	A622	83B5	.2...!.."..!.."..
	A0	CDAB	A6B7	1810	CD48	A9FE	52CA	53A5	FE50	.....H..R.S..P
	B0	CAB0	A53E	1137	C9FD	E5C5	3EFF	0000	00DD	...>..7.....>.....
F3	C0	E5FD	E500	0000	FDE1	DDE1	3EC9	0000	00CD	.....>.....
B4	D0	35A4	C1FD	01F2	70A5	E121	7FA5	2281	B521	5.....p..."..!
54H	E0	4BA6	2283	B5B7	C938	18CD	9AA5	FE0D	2011	K.."....8.....
	F0	3A7E	B5B7	3E0A	C49A	A501	1027	0B78	B120	:B..>.....'..x..
DRV	00	FBC9	DDE5	FDE5	0000	00FD	E1DD	E1CD	3DA4	.....=.
1	10	C9FE	6020	023E	5EC9	21D0	A522	81B5	214B	..'.>^!.."..!K
1H	20	A622	83B5	3EC9	0000	00CD	35A4	3AE8	37E6	.."..>.....5...7.
	30	F0FE	30C8	3E16	37C9	CDA9	A5F5	DDE5	FDE5	..0.>..7.....
DRS	40	CD3B	00FD	E1DD	E1F1	FE0D	2011	3A7E	B5B7	.!.....:B..
185	50	3E0A	DDE5	FDE5	C43B	00FD	E1DD	E1B7	C93A	>.....;.....
B9H	60	85B5	FE00	CA4E	A62A	83B5	E9FE	4020	333A	.....N.*.....5.3:
	70	A3B5	B7C4	8FA6	11A4	B5DD	E5FD	E5CD	2844	.....(D
	80	FDE1	DDE1	C4A8	A4CD	50A6	21EE	B67E	B720	.....P.!..B..
	90	0F3E	B0CD	75A4	C2AB	A4CD	F7A6	DAD3	A9C9	>..u.....
	A0	1811	11A4	B5DD	E5FD	E5CD	2844	FDE1	DDE1	.....(D....
	B0	C4A8	A4CD	50A6	B7C9	3E00	3285	B521	62A6	....P...>..2..!b.
FRS	C0	227F	B522	81B5	2283	01F2	60A6	B5C9	3E02	.."...".....>.
85	D0	37C9	2A81	B5E9	F53E	8032	EEB6	F12A	A1B5	7.*.....>..2...*
55H	E0	7723	3AA3	B53D	200B	CD8F	A6DA	D3A9	21D6	w#:...=.....!.
	F0	B53E	0032	A3B5	22A1	B5D7	C911	A4B5	DDE5	>..2...".....

Stark Othmar  
Seite 2

Paul Kröher

Karpfenweg 6  
2970 Emden, 29. April 1985

Genie/TRS-80 User Club  
c/o Peter Spieß  
Trugenhofenerstr. 27

8859 Rennertshofen 1

Betr: BASICODE

Im Info wurde bereits kurz auf BASICODE eingegangen. Ich hatte mir bereits im Herbst letzten Jahres aus den Niederlanden das Programm BASICODE 2 von NOS zuschicken lassen. Ebenso die benötigte Interface-Platine. Die Beschaffung der Bauteile hat lange Zeit in Anspruch genommen.

Zwischendurch hatte ich versucht die BASICODE-Routinen von Kasette zu laden (was auch ohne Interface geht, da das Interface nur für die Programme notwendig ist). Dieses gelang überhaupt nicht. Ich habe bei NOS reklamiert und eine neue Kasette erhalten. Von dieser konnte ich nach mehreren Versuchen (Pegeleinstellung ist äußerst schwierig) dann die Programme zum laden und speichern von BASICODE Programmen einlesen und auf Diskette abspeichern. Bei ersten Test die Programme zu benutzen, wohlwissend das ohne Interface dieses nicht richtig gehen kann, stellte ich dann fest, das sowohl die GET als auch die PUT Routine zwar ein Relais im Genie I schalten, den Rekorder jedoch nicht freigeben. Ich nahm zunächst an, daß hierfür das Interface mit verantwortlich sein wird. Doch weit gefehlt!!!

Nachdem ich nun endlich die Platine bestückt habe und freudig an einen Test heranging, mußte ich feststellen, daß sich nichts geändert hat. Es muß also wohl ein Fehler in der BASICODE Routine sein. Dieses zu finden erscheint unmöglich, da das Programm nicht listbar ist (wahrscheinlich kompiliert). Das Menue zur Vorbereitung des Programms fragt zwar exakt nach den unterschiedlichen Geräten (TRS 80 ..., Genie mit Rekorder 1, Genie mit Rekorder 2...), aber trotzdem wird wohl nicht richtig vorbereitet. Bevor ich nun mit NOS auf Fehlersuche gehe (mit denen kann man nur auf holländisch (was ich nicht beherrsche) oder auf englisch verkehren (dieses werde ich wohl machen müssen, wenn keine andere Lösung möglich ist), bitte ich um Clubhilfe. Hat der Club (oder ein Clubmitglied) die für das Genie I funktionierenden GET und PUT Programme (evtl. auch das DISPLAY Programm) und kann sie mir zur Verfügung stellen (wenn auf Disk dann bitte mit PD Daten)???

Auf der Kasette aus den Niederlanden befinden sich mehrere Beispiel-Programme. Was sie bewerkstelligen weiß ich z.Zt. nicht, hoffe jedoch (mit Hilfe des Clubs durch funktionierende GET + PUT Programme) es bald zu erfahren. Ich werde dann im Info darüber berichten. Außerdem kann ich von APPLE-Freunden auch dann noch einiges in BASICODE bekommen. Mal sehen ob darunter auch was brauchbares ist. NOS (Hilversum-Radio) sendet auch regelmäßig BASICODE-Programme über Rundfunk aus. Hier werde ich dann zukünftig auch mal aufzeichnen und darüber berichten.

8 6/85 In der Hoffnung auf Clubhilfe verbleibe ich  
mit freundlichem Gruß



PS: Wenn keine Hilfe möglich,  
bitte Info, damit ich mit  
NOS auf Kriegsfuß gehe!!!

## Minimaltreiber für die HRG 1b

Mit meinem Beitrag "Die HRG 1b und BASIC netto" wollte ich zeigen, daß die Karte für hochauflösende Graphik letztenendes auch nur mit Wasser kocht und ohne teure Zusatzsoftware programmiert werden kann. Klar, daß ein reiner BASIC-Treiber sehr langsam arbeitet. Deshalb folgt nun hier eine Lösung in Maschinensprache. Sie ist nur 130 Bytes lang, daher findet sich für sie immer ein ruhiges Plätzchen. Die Ladeadresse F000 (s. Listing) ist nur ein Vorschlag.

Auch dieser Treiber kennt nur die Befehle SET, RESET und POINT. Zur Unterscheidung von der Genie-Pixelgraphik folgt nach diesen Befehlswörtern für die HRG noch ein Punkt. Daraus ergibt sich das erste Problem: Bei der Bearbeitung der normalen Graphikbefehle wird zuerst RST 08 angesprungen, um auf eine sofort folgende offene Klammer zu prüfen. Fehlt sie, wird ein Syntaxfehler ausgegeben. Deshalb ist hier die RST-08-Routine auf das Segment check verbogen. Es wird zunächst geprüft, ob sie von einem der Graphikbefehle aufgerufen wurde. Ist das nicht der Fall, geht es an der alten Stelle 1C96 normal weiter. Andernfalls folgt ein Test auf die offene Klammer. Steht sie da, ist die normale Graphik gemeint. Dann Fortsetzung in der alten Routine. Sonst müssen jetzt ein Punkt und dann erst die Klammer folgen.

Nun steht fest, daß ein HRG-Befehl gemeint ist. Die eingegebenen Koordinaten werden auf zulässige Werte untersucht, dazwischen muß wie gewohnt ein Komma stehen. Sonst werden die entsprechenden Fehlermeldungen ausgegeben. Nachdem der Befehl vollständig analysiert ist, wird der Befehlszeiger auf den BASIC-Text in den Puffer cmdbuf gerettet. Ein PUSH auf den Stack wäre zu kompliziert geworden, weil der bereits beim Einsprung Daten enthält (s. u.).

Und jetzt passiert die Hauptsache: Die Argumente X und Y in der Matrix 384x192 sind sehr benutzerfreundlich, aber die physikalische Lage eines Bytes im HRG-Speicher hat damit leider nicht die Bohne zu tun. Es wird deshalb mit einem ziemlich verworren anmutenden Algorithmus zunächst die HRG-Adresse errechnet. Es fällt damit auch so ganz nebenbei das zutreffende Bit (der angesteuerte Punkt) im HRG-Byte ab. Dies möchte ich jetzt nicht mehr erklären; s. dazu meinen BASIC-Beitrag.

Wenn die Adresse feststeht, wird zunächst der BASIC-Befehlszeiger restauriert. Dann muß geklärt werden, welcher der drei Befehle überhaupt zu bearbeiten ist. In der ROM-Routine für die Pixelgraphik wird zur Unterscheidung zunächst je nach Befehl der Akku mit einem Flag geladen: 00 für POINT, 80 für SET und 01 für RESET. Bevor für alle drei Befehle gemeinsam die Prüfung auf die offene Klammer erfolgt (s. o.), wird der Akku auf den Stack gepusht. Von dort holt ihn unsere Routine nun. Jetzt wird je nach Befehl in die entsprechende Endrunde verzweigt:

In SET (Label set im Listing) wird der alte Wert, der zuvor vom Port 04 gelesen wurde, mit dem neu errechneten Bit oderiert. In RESET (ohne Label) wird exklusiv oderiert. Gab es einen Punkt dort, wird er rückgesetzt, gab es keinen, bleibt das Bit auf 0. In POINT (Label point) wird lediglich das alte HRG-Byte in den Akku geladen. Den Rest macht das Microsoft-ROM im alten POINT-Treiber.

Natürlich interessierte es mich, wie mein Treiber im Vergleich zu BASGR/CMD (Treiber von RB-electronic) abschneidet. Der scheint einem schnelleren Algorithmus zu folgen, denn er arbeitet etwa um ein Zehntel fixer. Vielleicht ist daran auch die Tatsache schuld, daß ich, um Platz zu sparen, in die RST-08-Routine eingreife, weshalb jedesmal ein komplizierter Check erforderlich ist. Dafür aber bleibt das übrige BASIC insgesamt schneller, weil nicht bei ausnahmslos jedem Befehl auf das vorangestellte Doppelkreuz getestet werden muß.

Komplizierte Graphiken, für die man die leistungsstarken LINE-Befehle usw. günstig nutzen kann, soll man gerne weiterhin mit BASGR, HRG-PACK oder GRAPE erstellen. Wo aber die drei alten Graphikbefehle ausreichen, kann man mit meinem Treiber eine Menge Platz und Laufzeit sparen.

00001 : POINT, SET und RESET (im folgenden PSR abgekürzt) für  
 00002 : die HRG 1b. Zur Unterscheidung zu PSR der Genie-Klötz-  
 00003 : chengraphik lautet die HRG-Syntax 'PSR.'

00004

00005 : (C) 1983 by The HACKTORY

00006

4001		00007	ORG	4001h	;RST-08-Vektor
4001	00F0	00008	DW	check	;auf eigene Rout. biegen
		00009			
F000		00010	ORG	0f000h	;beliebige Adresse
F000	E3	00011	check EX	(SP),HL	;HL <- RET-Adresse
F001	D5	00012	PUSH	DE	;wird verändert
F002	113C01	00013	: LD	DE,013ch	;RET-Adresse für PSR
F005	DF	00014	RST	18h	;ist es diese?
F006	D1	00015	POP	DE	;Register restaurieren
F007	E3	00016	EX	(SP),HL	;HL und Stack restaur.
F008	C2961C	00017	JF	NZ,1c96h	;falls anderer Caller
F00B	F1	00018	POP	AF	;Stack korr. wegen RST 08
F00C	7E	00019	LD	A,(HL)	;nächstes Zeichen
F00D	FE28	00020	CP	'('	;Klötzchen-PSR?
F00F	23	00021	INC	HL	;nächste Stelle
F010	CA3D01	00022	JP	Z,013dh	;sonst Klötzchen-PSR
F013	2B	00023	DEC	HL	;Befehlszeiger korrig.
F014	CF	00024	RST	08h	;',' für HRG-PSR?
F015	2E	00025	DB	','	;sonst Syntaxfehler
F016	CF	00026	RST	08h	;auf '(' prüfen
F017	28	00027	DB	'('	;sonst Syntaxfehler
F018	CD461E	00028	CALL	1e46h	;DE <- Abszisse
F01B	E5	00029	PUSH	HL	;Befehlszeiger retten
F01C	217F01	00030	LD	HL,017fh	;Maximalwert für X
F01F	DF	00031	RST	18h	;Vergleich mit Eingabe
F020	E1	00032	fcterr POP	HL	;Befehlszeiger restaur.
F021	DA4A1E	00033	JP	C,1e4ah	;Funktionsf., falls mehr
F024	D5	00034	PUSH	DE	;Abszisse retten
F025	CF	00035	RST	08h	;auf ',' prüfen
F026	2C	00036	DB	','	;sonst Syntaxfehler
F027	CD461E	00037	CALL	1e46h	;DE <- Ordinate
F02A	E5	00038	PUSH	HL	;Befehlszeiger retten
F02B	21BF00	00039	LD	HL,00bfh	;Maximalwert für Y
F02E	DF	00040	RST	18h	;Vergleich mit Eingabe
F02F	E1	00041	POP	HL	;Befehlszeiger restaur.
F030	38EE	00042	JR	C,fcterr	;Fehler, falls höher
F032	2282F0	00043	LD	(cmdbuf),HL	;Befehlszeiger retten
F035	EB	00044	EX	DE,HL	;zur Vereinf. d. Folgend.
F036	E3	00045	EX	(SP),HL	;X- <-> Y-Koordinate
F037	3E06	00046	LD	A,06h	;wegen 6 Dots/Byte
F039	CD7944	00047	CALL	4479h	;HL/A=HL Rest A
F03C	D1	00048	POP	DE	;Ordinate holen
F03D	F5	00049	PUSH	AF	;Bit-Nr. retten
F03E	E5	00050	PUSH	HL	;dto. X-Koordinate
F03F	EB	00051	EX	DE,HL	;HL <- Y-Koordinate
F040	3E0C	00052	LD	A,0ch	;wegen 12 Dotzeil./Stelle
F042	CD7944	00053	CALL	4479h	;HL/A=HL Rest A
F045	F5	00054	PUSH	AF	;Rest retten
F046	3E40	00055	LD	A,40h	;64 Stellen/Zeile
F048	CD7644	00056	CALL	4476h	;A*HL= AHL
F04B	44	00057	LD	B,H	;HL nach BC retten
F04C	4D	00058	LD	C,L	
F04D	F1	00059	POP	AF	;obigen Rest holen
F04E	210004	00060	LD	HL,0400h	;Faktor 1kB
F051	CD7644	00061	CALL	4476h	;A*HL= AHL
F054	09	00062	ADD	HL,BC	;Zwischensumme
F055	C1	00063	POP	BC	;X-Koordinate
F056	09	00064	ADD	HL,BC	;Endsumme = HRG-Adresse
F057	7D	00065	LD	A,L	;LSB
F05B	D302	00066	OUT	(02h),A	;auf HRG ausgeben

F05A	7C	00067	LD	A,H	;MSB
F05B	D303	00068	OUT	(03h),A	;dto.
F05D	C1	00069	POP	BC	;B <- Bit-Nr.
F05E	04	00070	INC	B	;B <- min. 1
F05F	3E80	00071	LD	A,80h	;Anfangswert für A
F061	07	00072 loop	RLCA		;fortgesetzt A*2
F062	10FD	00073	DJNZ	loop	;bis richtiges Bit in A
F064	E63F	00074	AND	3fh	;nur die ersten 6 Bits
F066	4F	00075	LD	C,A	;Akkus retten
F067	DB04	00076	IN	A,(04h)	;Inh. HRG-Speicherstelle
F069	47	00077	LD	B,A	;Akkus retten
F06A	2A82F0	00078	LD	HL,(cmdbuf)	;Befehlszeiger restaur.
F06D	F1	00079	POP	AF	;PSR-Flag holen
F06E	B7	00080	OR	A	;ist es 0 (POINT)?
F06F	2807	00081	JR	Z,point	;falls ja
F071	07	00082	RLCA		;ist es 80 (SET)?
F072	78	00083	LD	A,B	;alter Wert im HRG-Byte
F073	3807	00084	JR	C,set	;falls ja
F075	A9	00085	XOR	C	;Bit rücksetzen (RESET)
F076	1805	00086	JR	out	;zur Ausgabe und Rückkehr
F078	78	00087 point	LD	A,B	;altes HRG-Byte
F079	C39201	00088	JP	0192h	;alter POINT-Treiber
F07C	B3	00089 set	OR	C	;Bit setzen
F07D	D305	00090 out	OUT	(05h),A	;auf HRG ausgeben
F07F	C38C01	00091	JP	018ch	;erledigt
F082	0000	00092 cmdbuf	DW	0000h	;Puffer für Befehlszeiger
		00093			
0000		00094	END		

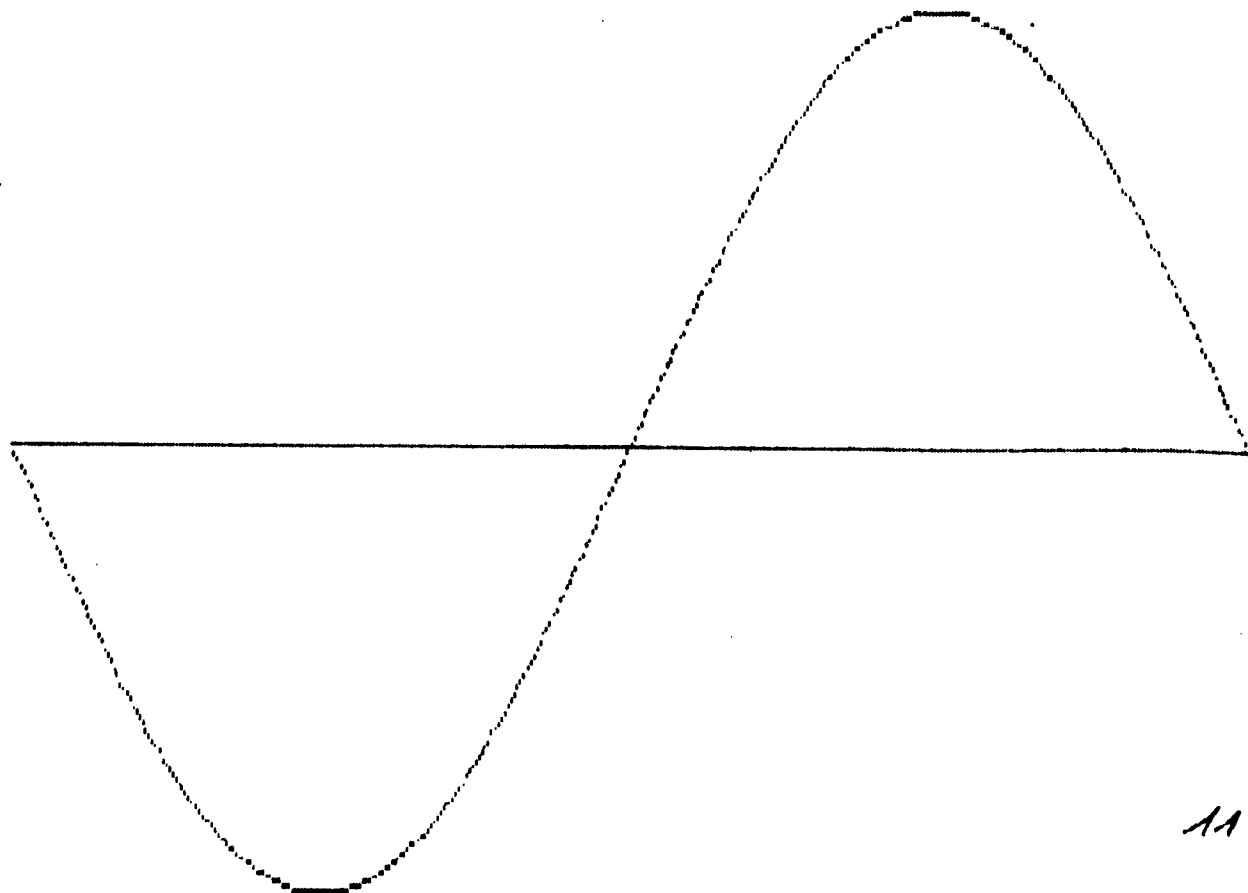
00000 Fehler

10 CMD"load hrg":CLS:CMD"CLS,G":OUT1,255%:DEFINT A-Z'

HRG-Treiber laden, HRG-Speicher löschen, HRG einschalten usw.

40 FORX=0TO383%:SET.(X,95%):NEXT:FORX=0TO383%:SET.(X,SIN(X/61%)\*95.5+95.5):NEXT'

Achse und Sinuskurve ziehen



11 6185

Die Sprache der computergestützten Unterweisung.

PILOT wurde von Dr. John Starkweather Mitte der 70er Jahre geschaffen. Diese Programmiersprache ist als ein ideales Mittel zum Einsatz des Computers als Lerngerät gedacht. Es gibt nur wenige Befehle in PILOT, aber sie sind leistungsfähig genug, um interaktive Lernprogramme zu erzeugen, die für den Lernenden effektiv und leicht zu schreiben sind.

Weiter mit <New Line>?

PILOT ist textorientiert und weniger mathematisch ausgerichtet, daher kann fast jedes Wissensgebiet in Dialogform dargeboten werden. Die Befehle sind so leicht und schnell zu lernen, daß diejenigen, die sich nicht mit höheren Programmiersprachen auskennen, schnell die Einzelheiten der Programmierung erlernen, um sich dann auf die Lernziele konzentrieren zu können, die erreicht werden sollen.

H.-G.  
Köster

Dieses Programm stellt Ihnen PILOT am praktischen Beispiel vor.

Weiter mit <New Line>?

Programme in PILOT haben nur 9 besondere Befehle:

T = TYPE (SCHREIBEN)    A = ASK (FRAGEN)    M = MATCH (VERGLEICHEN)  
J = JUMP (SPRUNG)       E = END (ENDE)       Y = YES (JA)  
N = NO (NEIN)           C = CLEAR SCREEN (BILDSCHIRM LÖSCHEN)  
W = WIDE LETTERS (GESPERSTE SCHRIFT)

Weiter mit <New Line>?

PILOT-Programme für diesen Interpretierer brauchen Zeilennummern. Jede Programmzeile kann beliebig nummeriert werden. Mit dem AUTO-Befehl des TRS-80 kann man automatisch Zeilennummern erzeugen; diese beginnen bei 10 und werden jeweils um 10 erhöht.

Das PILOT-Programm folgt dann diesen Zeilennummern --- zuerst die Befehle der Zeile 10, dann die der Zeile 20, Zeile 30 usw. Wenn Sie möchten, daß das Programm einen Teil überspringt oder zurückspringt, dann gibt es einen Sprungbefehl zur gewünschten Zeilennummer.

Weiter mit <New Line>?

Programmanweisungen in PILOT haben im wesentlichen alle denselben Aufbau. Zeilennummer, Befehlsbuchstabe, ein Anführungszeichen, und die Anweisungen oder Informationen, die dargestellt werden sollen.

Dieses kleine Programm aus einer Zeile würde den Text 'DIES IST IST DIE PROGRAMMIERSPRACHE PILOT' schreiben:

10 T"DIES IST DIE PROGRAMMIERSPRACHE PILOT

Damit haben wir das T für Text dargestellt. Jeder Text nach dem Anführungszeichen wird auf dem Bildschirm genauso dargestellt.

Weiter mit <New Line>?

Dabei darf keine Leertaste zwischen dem T und dem Anführungszeichen sein. Auch darf das Anführungszeichen nicht im Text sein. Es darf nur einmal vorkommen: am Anfang des Textes. Man kann sich aber "hiermit" behelfen.

Mit <New Line> weiter?

Die Befehle C und W sind Text-Kommandos mit besonderen Eigenschaften.

Das C-Kommando löscht den Bildschirm und beginnt mit dem Text in der Ecke oben links. Beispiel:

12 6/95    10 C"LÖSCHEN UND OBEN LINKS BEGINNEN



Das W-Kommando kann dasgleiche, aber der Bildschirm schaltet auf Breitschrift mit 32 Zeichen pro Zeile um. Beispiel:

10 W"LOESCHEN, BREITSCHRIT UND TEXT OBEN LINKS

Weiter mit <New Line>?  
Das ist Breitschrift.

Weiter mit <New Line>?  
Mit dem Befehl A kann eine Frage gestellt werden, die der Computer selbständig mit einem Fragezeichen abschließt. Dann wartet er auf eine Eingabe. Beispiel:

10 A"IHRE EINGABE

Hiermit wird die Frage mit Fragezeichen geschrieben und der Computer wartet auf die Eingabe, die mit <New Line> abgeschlossen wird.

Weiter mit <New Line>?  
Was geschieht nun mit der Antwort?

Sie wird an einer bestimmten Speicherstelle abgelegt und verbleibt dort, bis eine neue Antwort gespeichert werden muß.

Die eingegebene Antwort kann man mit dem 'at'-Zeichen oberhalb der <New Line>-Taste wieder ausdrucken lassen.

Bitte drücken Sie jetzt die at-Taste und dann <New Line>. -->?

Das war nicht die richtige Taste.

Bitte drücken Sie jetzt die at-Taste und dann <New Line>. -->? S

Das Programm kann jede Antwort mit der at-Taste im Programm ausdrucken lassen. Wenn das at-Zeichen in einem Text-Befehl auftaucht, dann wird die letzte Antwort ausgedruckt und nicht das at-Zeichen. Beispiel:

10 A"IHRE EINGABE  
20 T"DIE EINGABE S IST RICHTIG!

Sie müssen bei dem at-Zeichen besonders vorsichtig sein, denn der Computer erkennt at und <Shift>+at als verschiedene Zeichen. Jede Eingabe als Antwort auf die Zeile 10 wird in der Zeile 20 bei dem at-Zeichen ausgedruckt.

Mit <New Line> weiter?  
Wie wird nun die Antwort vom Computer verwertet? Mit dem M-Kommando stellt der Computer fest, ob die Antwort richtig oder falsch war.

Nach der Eingabe einer Frage mit dem A-Kommando kann überprüft werden, ob die Antwort mit einer Antwort einer Liste von möglichen Antworten übereinstimmt.

Weiter mit <New Line>?  
Beispiel:

10 A"IHRE EINGABE  
20 M"JA/SICHER/NATÜRLICH/DOCH

In diesem einfachen Programm wartet der Computer auf die Antwort, dann vergleicht er diese mit der Liste der gültigen Antworten. Die richtigen Antworten werden im Programm durch / gekennzeichnet.

Schrägstrich getrennt werden.

Was geschieht nun, wenn die Antwort stimmt?

Weiter mit <New Line>?

Wenn die Antwort mit einem Element der Antwortliste übereinstimmt, dann wird eine Merkvariable auf 'ja' gesetzt. Wenn die Antwort mit keinem Element der Antwortliste übereinstimmt, dann wird die Merkvariable auf 'nein' gesetzt.

Der Wert der Merkvariablen bleibt erhalten, bis die nächste Antwort verglichen werden muß.

Weiter mit <New Line>?

Nun zu den beiden neuen Befehlen - Y und N, sie stehen für ja und nein.

Wenn der Buchstabe Y am Anfang einer PILOT-Programmzeile auftaucht, lautet der Befehl: wenn die Merkvariable auf 'ja' steht, dann soll der Befehl in dieser Zeile ausgeführt werden.

Wenn die Merkvariable auf 'nein' steht, soll der Befehl der nächsten Zeile ausgeführt werden.

Weiter mit <New Line>?

Analog bedeutet N, daß der Befehl nur ausgeführt wird, wenn die Merkvariable auf 'nein' steht, sonst muß der Befehl der nächsten Zeile ausgeführt werden.

Beispiel:

```
10 A"IHRE EINGABE
20 M"TEXAS
30 YT"STIMMT!
40 NT"NEIN, DAS IST FALSCH!
```

Wird als Antwort zur Frage in der Zeile 10 'TEXAS' eingegeben, dann gibt der Computer die Information in der Zeile 30, stimmt die Antwort nicht, dann wird die Information in der Zeile 40 ausgegeben.

Jeder Befehl kann mit Y oder N kombiniert werden. Je nach Inhalt der Merkvariablen wird die nächste Programmzeile ausgeführt oder ganz übersprungen.

Weiter mit <New Line>?

Schließlich noch der Befehl J - es ist ein Sprungbefehl.

Beispiel:

```
10 J"30
```

Hier soll der Computer zur Zeile 30 springen und dort die weiteren Befehle abarbeiten.

Weiter mit <New Line>?

Die Anweisung:

```
50 YJ"90
```

bewirkt, daß nur nach 90 gesprungen werden darf, wenn die Merkvariable auf 'ja' steht.

Weiter mit <New Line>?

Hier noch ein paar Hinweise:

Alle Befehle zum Editieren von Texten können beim Schreiben von PILOT-Programmen benutzt werden. Dazu gehören: LIST, AUTO, EDIT, NEW und DELETE. Mehr steht im Handbuch zu Level II.

PILOT-Programm werden mit dem Befehl 'NAME' gestartet. Dieser Befehl wird normalerweise nicht von dem BASIC-Interpreter benutzt. BASIC steht weiterhin mit dem Befehl 'RUN' zur Verfügung. Mehr steht im Handbuch zu Level II.

Nach der Eingabe von <New Line> folgt ein Beispiel. -->  
Hier ist eine Musteraufgabe mit dem PILOT-Interpreter.  
Wie heißen Sie, bitte? Heinz-Gerd  
Gut Heinz-Gerd, hier ist Ihre Aufgabe:

Mit welcher Zeitschrift kann man Computertechnik lernen:

- 1 = Heim und Garten
- 2 = Psychologie Heute
- 3 = Northern Bytes

Bitte überlegen Sie sorgfältig, Heinz-Gerd,  
und wählen Sie die richtige Antwort? 1  
Nur dann, wenn Sie Ihren Computer als Blumentopf mißbrauchen.

Mit welcher Zeitschrift kann man Computertechnik lernen:

- 1 = Heim und Garten
- 2 = Psychologie Heute
- 3 = Northern Bytes

Bitte überlegen Sie sorgfältig,  
und wählen Sie die richtige Antwort? 2  
Hat der Computer Sie in den Wahnsinn getrieben?

Mit welcher Zeitschrift kann man Computertechnik lernen:

- 1 = Heim und Garten
- 2 = Psychologie Heute
- 3 = Northern Bytes

Bitte überlegen Sie sorgfältig  
und wählen Sie die richtige Antwort? 4  
Sie schummeln!!!

Mit welcher Zeitschrift kann man Computertechnik lernen:

- 1 = Heim und Garten
- 2 = Psychologie Heute
- 3 = Northern Bytes

Bitte überlegen Sie sorgfältig  
und wählen Sie die richtige Antwort? 3  
Sie haben völlig Recht. Alles Gute und viel Spaß mit PILOT.  
READY

15 6195

Es ist soweit:  
Ein Computerprogramm, das  
selbsttätig Prosa verfaßt,  
hat sein erstes Buch  
veröffentlicht

Lesen Sie. Lesen Sie in aller Ruhe die nebenstehende Kurzgeschichte. Fällt Ihnen etwas auf? Nein, ein besonders gelungenes Stück Prosa ist dies sicher nicht, kein Handke, Böll oder Beckett, kein Musil und Mann, auch kein bekannter Krimi- oder New-Wave-Autor, der sein Kürzel vergessen hat. Einfach eine etwas wirre Geschichte, die von einem Essen bei Freunden handelt. Der Plot schwimmt irgendwo im Unwägbareren, und der Sinn scheint in der Tiefe verborgen, dort, wo wir Normalleser auch sonst nur selten hingelangen. Aber wozu drucken wir eine mittelmäßige Kurzgeschichte, noch dazu von einem unbekannten Autor? Gehört so etwas nicht, wenn überhaupt, in kleine Literaturzeitingen?

1) } Klären wir zunächst über den Autor auf. Sein Name ist Racter. Nie gehört? Kein Wunder. Racter – Vor- gleich Zuname – ist ein Computerprogramm, geschrieben (dies für Computerbewanderte) in kompiliertem BASIC für einen Z80-Mikroprozessor und einen Arbeitsspeicher von nicht mehr als 64 kRAM, also für ein sehr gewöhnliches, überall im Handel erhältliches Gerät. Racter wurde von William Chamberlain, einem New Yorker Computerfreak, geschaffen und kursiert bereits als Schwarzkopie im deutschen Software-Untergrund. Racter (eine Abkürzung für „Raconteur“, Erzähler) fabriziert selbsttätig. Kurzgeschichten, Dialoge, Gedichte, Allegorien, weise und skurrile Vierzeiler und, wenn es sein muß, auch Limericks. Das hier abgedruckte Produkt digitaler Poesie ist eine Übersetzung aus dem Amerikanischen, ein Auszug aus dem ersten voll und ganz (mit Ausnahme des Vorworts) von einem Computer geschriebenen Buch. Es heißt

The policeman's beard is half constructed  
(Der Bart des Polizisten ist halb konstruiert),

Computer prose and poetry by Racter, Warner Books, New York 1984.

Unser Auszug wurde „trivial übersetzt“ – mit Hilfe eines elektronischen Wörterbuches.

Auszug aus Die Zeit v. 3.5.85

1) Merke: Mit unserem Rechner (Z80) geht noch mehr! !!!

## Sanfte Ionen – von Racter, kein Mensch

Helene büstete geschwind ihr dichtes Haar. Sie bügelte bedächtig ihren Büstenhalter, und weit entfernt begann John, der blendende John, ein Spottlied zu singen. Mathew schmachtete nach einem Blick in Helenes Nachgewand, während Wendy über ihre Träume nachdachte (tollgewordene Leoparden verschlangen mondsüchtige Oboisten). Helene schreckte auf, während sie sich die Haare büstete: Sie war ein junges Mädchen, worüber John sehr glücklich war, aber für Oboisten, und dann auch noch mondsüchtige Oboisten, hatte sie keinen Sinn; sie begann einfach, ihre Haare zu kämmen, nachdem sie sie durchgebürstet hatte, und bereitete sich auf das Abendessen vor. Sie (Helene, John, Wendy und Mathew) wären jetzt fertig zum Abendessen, und Helene war in Wirklichkeit schon müde.

Helene beobachtete John und überlegte: Ein Abendessen mit ihm? Geschmacklos! Ein Abendessen würde eine Abhandlung fördern, und eine Abhandlung oder ein Märchen war das, worauf John sorgsam aus war. Hatte er dabei etwas im Sinn? Wein, Butter, Bohnen? Nein! *Elektronen!* John war schlicht und einfach ein Quantenlogiker; seine endlosen Träume waren fesselnd und interessant; Mathew, Helene und Wendy unterstützten ihn unter allen Umständen bei seinen wütenden Versuchen, sich selbst zu verbreitern. Jetzt juckten Legionen von Träumen und stießen an Wendys Bewußtsein. John flüsterte: „Nur eine Minute! Helene ist ein junges Mädchen, ich bin ein Quantenlogiker; können junge Mädchen etwas über Galaxien wissen oder gar über Sterne oder eine Vielzahl galaktischer Systeme? Das Universum ist entmutigend, klein, gargantuanisch; können junge Mädchen Elektronen erkennen? Ich bemerke, daß jeder von euch denkt, ich sei verrückt, aber Elektronen und Neutronen und eine Vielzahl von Mesonen sind in euch allen.“

Amu.

## Zu Paul Kröfers Assembler-Liste

So soll es sein: Der Paul macht einen Assembler-Kurs mit, findet ihn unter diesem oder jenem Aspekt bescheiden und hilft sich kurzerhand selbst, indem er sich - und schließlich im Info 4/85 auch uns - die kompakteste Befehlsliste zusammenstellt, die mir je untergekommen ist. Paul, solltest Du jemals Lübeck heimsuchen, gebe ich Dir fürchterlich einen aus!

Da Paul in der Überschrift außer Rodney Zaks auch mich als Mithelfenden erwähnt, obwohl ich nur ein paar Anmerkungen zu einer Rohversion der Liste losließ, möchte ich nun einige geringfügige Unstimmigkeiten in der Liste geradebiegen. Als Neuling auf dem Gebiet möchte Paul diese letzten Korrekturen vorsichtshalber mir überlassen. Also los:

Die Register, die er mit "I." bezeichnet, sind IX und IV. Da sie beide vollkommen gleich behandelt werden, ist diese Abkürzung völlig in Ordnung, soll hier lediglich dem Unkundigen erklärt werden. Unter dem ersten Auftreten von "I." (S. 22) stehen ein paar Befehle, bei denen die Registerpaare BC und DE Zeiger auf eine Speicherstelle sind, aus denen bzw. in die der Akku geladen wird, also nicht das, was ziemlich oben als letzte Überschrift steht. Ähnliches gilt für die SP-Ladebefehle. Bei den Befehlen RLD und RRD sollte man für die Unsicheren unter uns noch hinzufügen, daß es sich jeweils um vier Bits handelt (ein Nibble). Der Z80 hat zwei Interrupt-Flipflops, IFF0 und IFF1. Was Paul auf S. 25 unten beschreibt, ist IFF0. IFF1 dient u. a. dazu, den Zustand von IFF0 (nach DI oder EI) zwischenspeichern, damit der alte Zustand nach einem NMI (der ein DI ausführt) wiederhergestellt werden kann.

Die Anmerkungen des letzten Absatzes betreffen nur die äußere Aufmachung der Liste, sachlich ist sie da durchaus perfekt. Das einzige in der Liste, was sachlich zumindest unklar ist, steht auf S. 26: Ein RST ist ein Unterprogrammaufruf wie ein CALL. Also hätte Paul statt einfach "Sprung" lieber "wie GOSUB" schreiben sollen, was er für CALL tat. Bei einem ReStart wird nämlich der PC auf den Stack gelegt, damit sich das Programm nach Ausführung der Routine "erinnert", wo es anschließend weitergeht. Das RET, das in der Regel am Ende eines solchen Unterprogramms steht, kommt nämlich genau wie bei CALL einem POP PC gleich.

Die Pseudo-Opcodes, die am Ende von Pauls Liste stehen, sind eine Auswahl, die z. B. bei EDTASM+ vorkommt. Nur das Assembler-Programm, nicht aber der Z80 kann etwas damit anfangen. Vielleicht ist dies nicht allen Lesern klargeworden. Es gibt eine ganze Reihe von Assemblern, und etliche verwenden andere Pseudo-Ops. Hier muß der Programmierer sich jeweils an sein Manual halten.

Insgesamt gehört diese Liste zum Besten, was das Info des Bremerhausener Clubs enthält. Ob noch mehr "Anfänger" mit der Vorstellung eigener Produktionen den Mut finden, zu zeigen, daß sie es eigentlich gar nicht mehr sind?

Arnulf Sopp

A P L 80

APL80-Operatoren in alpa-Reihenfolge a.d. Tastatur (". " für SHIFT)

Taste	*)	Bedeutung	Beispiel
.A	D	NAND, mit "nicht beide" verknüpfen (Boolesche Fktn.)	0, wenn a und b = 1 a.Ab
.B	D	Entschlüsseln von dual zu dez.	2 2 2 2.B1 1 0 1 13
.C	-	Kommentar	.C Programmname
.E	D	Zugehörigkeit prüfen	2 3.E1 2 Ø 1
.H	M	Aufrunden z. nächst. ganz. Zahl	.H2.3 3
.H	D	Maximalwert aufsuchen	2 4 -3.H1 6 9 4 9
.I	M	Indexgenerator	.I4 1 2 3 4
.I	D	erstes Auftreten anzeigen	'T'.I'Wort' 'S'.I'Wert' 4 5
.J	M	Absolutbetrag bilden	.JØ.CLEAR3 5 Ø 3
.J	D	Restfunktion (modulo)	
.K	M	natürlicher Logarithmus	
.K	D	'kleiner oder gleich' prüfen	3.K4 1
.L	M	Abrunden auf die nächst kleinere ganze Zahl	.L3.8
.L	D	Minimalwert aufsuchen	-4 3:L6 5 -4 5
.M	-	alpha-numer. Anforderung (Eingabe ohne Hochkomma)	
.N	M	NOT-Prüfung (Boolesche Fktn.)	1, wenn Wert = Ø Ø, wenn Wert = 1
.O	D	Kreisfkten. (sin,cos,tan,...)	Ø.0x $\sqrt{1-x^2}$ 1.0x $\sin(x)$ 2.0x $\cos(x)$ 3.0x $\tan(x)$ 4.0x $\sqrt{x^2+1}$
.Q	M	Pi=3.14159	

↑  
 \*) mon. od. dyadisch

A P L 80

APL 80-Operatoren (Forts.)

Task	M/D	Bedeutung	Beispiel
.P	D	Strukturieren (Vektor, Matrix bilden)	
		2 2.F6 8 2 9	ergibt folgende Matrix: $\begin{matrix} 6 & 8 \\ 2 & 9 \end{matrix}$
.Q	-	numerische Anforderung, d.h. Hochkomma für CHARACTER	
.R	M	umkehren	
.R	D	Rotation (parallel spiegeln)	
.S	-	System	
.T	D	Verschlüsseln, Darstellung von b im Zahlensystem von a	
		2 2 2 2.T13	ergibt: 1 1 0 1
.V	D	oder-Verknüpfung	$\emptyset$ , wenn a und b gleich $\emptyset$ sind, 1 sonst
.W	D	Expandieren bzw. erweitern	
.X	M	Signum- bzw. Vorzeichenfktn.	
.X	D	Multiplizieren	
.Y	D	NOR bzw. "weder-noch"-Verknüpfung;	wenn a u. b = $\emptyset$ , $\emptyset$ sonst
.Z	D	"größer oder gleich"-Verknüpfung.	

Verwendung der Sonderzeichen:

VERWENDUNG DER SONDERZEICHEN IN APL 80:

TASTE	M/D/-	I	BEDEUTUNG	I	BEISPIEL: ERGEBNIS
!	I	M	I FAKULTAT (FACTORIAL)	I 14	: 1*2*3*4=24
!	I	D	I BINOMIALKOEFF. (KOMB.)	I 3!6	: 20 $\binom{6}{3} = \frac{6!}{3!3!}$
"	I	D	I (AUSSERES PRODUKT)	I 2	
#	I	-	I = CHR\$ BZW. ASC	I # 28 31	# 'ABC'
\$	I	D	I UNGLEICH-ABFRAGE (0/1)	I 2\$3, 4\$4	: 1, 0
%	I	M	I KEHRWERT (RECIPROCAL)	I %4	: .25
%	I	D	I DIVISION	I 3%4	: .75
&	I	D	I AND (LOG. BOOLE-OP.)	I	
'	I	-	I STRING-AUSGABE	I	'AUSGABE'
( )	I	D	I INDEXING	I 17	: 1237
*	I	M	I EXPONENTIALFKTN.	I *1	: 2.71828 (=e)
*	I	D	I POTENZIEREN	I 2*3	: 8
=	I	D	I GLEICHHEIT	I 3=4, 4=4	: 0, 1
CLEAR	I	D	I SUBTRAKTION	I 4CLEAR5	: -1
BREAK	I	-	I DEF-, EDIT-MODUSABBRUCH	I	
:	I	-	I MARKE SETZEN (LABEL)	I MARKE:	
CLEAR	I	M	I VORZEICHEN UMKEHREN	I - -5	: 5
@	I	D	I LOGARITHMUS ZUR BASIS	I 2 @ 5	: 3
,	I	-	I ANEINANDERHÄNGEN	I	
+	I	M	I VORZEICHEN BEIBEHALT.	I + 4, -3	: 4, -3
+	I	D	I ADDIEREN	I 3+3	: 6
,	I	M	I AUFREIHEN (RAVEL)	I 4,3	: 4 3
,	I	D	I KETTEN (CATENATE)	I A<0, 2, 4, 3, 1	: 0 2 4 3 1
.	I	D	I INNERES PRODUKT	I +/P.XP	: +/P.XP (total)
<	I	D	I KLEINER-ABFRAGE	I 3<2	: 0
>	I	D	I GRÖßER-ABFRAGE	I 3>2	: 1
?	I	M	I ZUFALLSZAHLEN BIS...	I ?4	: (z.B.) 3
?	I	D	I STICHPROBE (DEAL)	I 6749	: LOTTOZAHLEN
/	I	M	I REDUKTION	I +/	: QUERSUMME
/	I	D	I KOMPRESSION, AUSWAHL	I 0101/7952	: 9 2
SHIFT-HOCHPFEIL	D	I	I ENTNEHMEN (TAKE)	I 1<2 3 4	: 2 3
SHIFT-HOCHPFEIL	M	I	I AUFWARTS SORTIEREN	I 1<2 3 4	: 1 2 3 4
SHIFT-ABWARTS	M	I	I ABWARTS SORT	I 1<2 3 4	: 4 3 2 1
SHIFT-ABWARTS	D	I	I ENTFERNEN (DROP)	I 1<2 3 4	: 2 3 4
SHIFT-LINKS	-	I	I ZUWEISUNG (ASSIGN)	I A02 3 5	
RECHTSPFEIL	-	I	I VERZWEIGEN (BRANCH)	I	(wie GOTO)



## Der APL 80-EDITOR

EDITOR- und Programmiermodus werden in APL 80 nicht unterschieden. Aufgerufen werden sie durch Eingabe von

```
)EDIT programmname      bzw.  
)DEF  progr.name
```

Folgende nützliche EDITOR-Befehle lassen sich anwenden, und zwar im EDITOR-Modus:

```
DISPLAY:  )? Auflisten des gesamten APL-Programmes bzw.  
           der APL-Funktion auf dem Bildschirm  
  
REPLACE:  )zeilen-nr.  
           Ersetzen dieser Zeile durch eine neue  
:  
INSERT:   )Izeilen-nr.  
           (Insert-Modus) Einfügen einer neuen Zeile  
           vor der mit zeilen-nr. angegebenen  
  
DELETE:   )zeilen-nr., anschließend 'BREAK'-Taste oder ")"?  
           (DELETE-Modus) Löschen der angegebenen Zeile
```

### 3 Anmerkungen:

1. Alle diese Eingriffe in das schon bestehende Programm renumerieren dasselbe sofort, also nicht erst nach Verlassen des EDITOR- bzw. Programmiermodus.
2. Die Kopfzeile (Header) kann ersetzt werden durch ")O", wobei der alte Funktions- bzw. Programmname erhalten bleiben muß. Eine Kopfzeile kann nicht gelöscht werden.
3. Fehlermeldungen vom System erfolgen normalerweise erst nach Starten des Programmes.  
Ausnahmen davon sind SYNTAX- und DOMAIN-Fehler. Sie führen (wie 'BREAK' oder ")?") zum Abbruch des Programmiermodus.

Steuerbefehle - beginnen alle mit ")" (Klammer zu)

- )CLEAR           Löschen des gesamten Arbeitsbereiches
- )ERASE name      Löschen einer Variablen oder Funktion, falls Objekt nicht vorhanden, Meldung "NOT FOUND"
- )VARS            Protokoll sämtlicher globalen Variablen in der Reihenfolge der Eingabe
- )FNS             Protokoll sämtlicher Funktionen a.d. Bildschirm in der Reihenfolge der Eingabe
- )LOAD name       Laden von Programmen oder Daten vom Band in den Arbeitsspeicher
- )SAVE name       Abspeichern von Programmen oder Daten aus dem Arbeitsspeicher auf Band
- )SI              Statusanzeige aller wartenden Funktionen, List derselben \*)
- )DEF program-name  
                  Einschalten des Programmiermodus; es erscheint in der Zeile 0 : Programm-Name, in der Zeile 1: kann die erste Anweisung programmiert werden.  
                  Beenden mit 'BREAK'-Taste
- )EDIT program-name  
                  Editiermodus; es erscheint der Kopf des Programm (Zeile 0) und die nächste frei verfügbare Eingabezeile.  
                  (EDIT-Befehle siehe Extrablatt!)
- )OFF             Vorsicht! Löschen des Programmes APL80 aus dem aktiven Arbeitsbereich, so daß es neu eingeladen werden muß
- )RESET           schaltet SI wieder aus; Meldung: "RESETTING SI"
- )CHECK           überprüft (wie CLOAD?) abgespeicherte Daten

Ein eingegebenes Programm (Funktion) oder eine Variable wird gestartet, indem man einfach den betreffenden Namen bzw. die Bezeichnung eingibt (und mit 'NEW LINE' quittiert).

\*) alle Fkten., die durch Fehler od. 'BREAK'-Taste abgebrochen wurden, werden aufgelistet nebst Abbruchzeile in Klammern dahinter, z.B. G(4) bedeutet: in Programm G fand ein Abbruch in Zeile 4 statt.

Anm.: Im Diskettenbetrieb stehen zusätzliche Steuerbefehle (z.B. )DOS, )RETURN, )HELP ) zur Verfügung.

## RENEW für Level-2 Basic

Es ist sicher schon vielen passiert (besonders, wenn man Maschinenspracheprogramme in ein Basic-Programm einbindet), daß sich der Rechner aufhängt und das im Speicher befindliche Programm löscht. Für Diskbenutzer gibt es dann immer noch die Möglichkeit RENEW. Arbeitet man jedoch im Level-2 Basic, so muß man sich das gelöschte Programm, falls man nicht ein entsprechendes Hilfsprogramm besitzt, per Hand wieder zurückholen. Um diese Methode zu verstehen, ist es wichtig, daß man weiß, wie ein Basic-Text überhaupt im Speicher abgelegt wird.

Für die weiteren Erklärungen betrachte ich folgendes kleines Basic-Programm:

```
10 PRINT "Demoprogramm"
20 FOR I=1 TO 10
30 PRINT I;
40 NEXT
```

Nimmt man nun das Programm mit einem Monitor auseinander (Genie-Benutzer nehmen am besten den eingebauten), ist es ganz nützlich, wenn man weiß, wo das Programm überhaupt im Speicher steht.

Die Anfangsadresse findet man in den Speicherzellen 40A4H (Lowbyte) und 40A5H (Highbyte). Beim Level-2 Basic liegt der Anfang bei 42E9H.

In den Adressen 40F9H und 40FAH steht die Startadresse des Variablenbereiches, die damit gleichzeitig das Ende des Programmbereiches angibt, da diese beiden Bereiche aufeinander folgen.

Macht man nun ein ASCII-Dump ab 42E9H, so wird man nicht mehr viel von dem Basicprogramm erkennen. Es ist zwar noch irgendwo das Wort "Demoprogramm" erkennbar, aber es ist weit und breit kein Basicbefehl und keine Zeilennummer sichtbar.

Zuerst einmal zu den Basicbefehlen:

Diese wandelt der Interpreter bei der Eingabe in 1-Byte lange Codes, sogenannte Token, um.

Neben der Speicherersparnis hat diese Maßnahme für die Abarbeitung eines Programmes entscheidende Vorteile:

- Der Rechner kann über Tabellen schnell die Routinen der einzelnen Basicbefehle finden.
- Alles, was nicht im Bereich der Token ist ( größer als 80H ), wird als Variable angesehen. Folglich muß dazwischen nicht extra unterschieden werden.

Betrachtet man sich im Speicherauszug das Byte vor dem Wort "Demoprogramm", so findet man dort B2H. In der Tabelle nachgesehen erkennt man, daß B2H das Token für den Basicbefehl PRINT ist.

Den Verbleib der Basicbefehle haben wir jetzt geklärt, aber was bedeutet die 5-Bytefolge z.B. vor dem PRINT-Token ?

Zuerst steht dort das Byte 00H. Dieses steht vor jeder Basiczeile und dient als Trennbyte von der vorherigen.

Die folgenden 2 Bytes ( erst Low- dann Highbyte ) dienen als Zeiger ( Pointer ) auf die nächste Programmzeile. Sie geben an, an welcher Adresse die nächste Programmzeile beginnt.

(Der Pointer weist auf das Byte nach dem Trennbyte 00H !!)

Folgt man den Pointern von Zeile zu Zeile, so zeigt er, irgendwann einmal auf die Bytefolge 00H 00H. Durch diese wird das Ende des Basicprogrammes gekennzeichnet.

Die nächsten beiden Bytes geben die Zeilennummer der Basiczeile an, natürlich wieder in der Reihenfolge Lowbyte/Highbyte.

Nun zum eigentlichen Thema RENEW:

Um die Auswirkungen von NEW kennenzulernen, löschen wir das Demoprogramm und gehen wieder in den Monitor.

Bei einem ASCII-Dump erkennt man, daß das Basicprogramm erhalten ist. Es gibt nur 2 Dinge, die geändert wurden:

- Der Pointer der ersten Zeile weist nicht mehr auf die zweite Zeile, sondern er wurde auf 00H 00H gesetzt.
- Der Zeiger auf das Ende des Programms/Anfang der Variablen wurde zurückgesetzt.

Um den Pointer der ersten Zeile wieder herzustellen, muß man nur das Trennbyte hinter der ersten Zeile finden. Der Pointer muß auf das Byte danach zeigen (erst Low dann Highbyte).

Danach sucht man das Ende des Programms. Man sieht dort die Bytes 00H 00H 00H. Das erste Byte 00H ist das Trennbyte hinter der letzten Zeile. Die folgenden 2 Bytes markieren das Ende des Programmes.

In den Adressen 40F9H/FAH muß die Adresse des Bytes nach(!!) den drei Nullen stehen.

# Tabelle mit BASIC-Tokens

80H	END	81H	FOR	82H	RESET	83H	SET	84H	CLS
85H	CMD	86H	RANDOM	87H	NEXT	88H	DATA	89H	INPUT
8AH	DIM	8BH	READ	8CH	LET	8DH	GOTO	8EH	RUN
8FH	IF	90H	RESTORE	91H	GOSUB	92H	RETURN	93H	REM
94H	STOP	95H	ELSE	96H	TRON	97H	TROFF	98H	DEFSTR
99H	DEFINT	9AH	DEFSNG	9BH	DEFDBL	9CH	LINE	9DH	EDIT
9EH	ERROR	9FH	RESUME	A0H	OUT	A1H	ON	A2H	OPEN
A3H	FIELD	A4H	GET	A5H	PUT	A6H	CLOSE	A7H	LOAD
A8H	MERGE	A9H	NAME	AAH	KILL	ABH	LSET	ACH	RSET
ADH	SAVE	AEH	SYSTEM	AFH	LPRINT	BOH	DEF	B1H	POKE
B2H	PRINT	B3H	CONT	B4H	LIST	B5H	LLIST	B6H	DELETE
B7H	AUTO	B8H	CLEAR	B9H	CLOAD	BAH	CSAVE	BBH	NEW
BCH	TAB(	BDH	TO	BEH	FN	BFH	USING	COH	VARPTR
C1H	USR	C2H	ERL	C3H	ERR	C4H	STRING\$	C5H	INSTR
C6H	POINT	C7H	TIME\$	C8H	MEM	C9H	INKEY\$	CAH	THEN
CBH	NOT	CCH	STEP	CDH	+	CEH	-	CFH	*
DOH	/	D1H	[	D2H	AND	D3H	OR	D4H	>
D5H	=	D6H	<	D7H	SGN	D8H	INT	D9H	ABS
DAH	FRE	DBH	INP	DCH	POS	DDH	SQR	DEH	RND
DFH	LOG	EOH	EXP	E1H	COS	E2H	SIN	E3H	TAN
E4H	ATN	E5H	PEEK	E6H	CVI	E7H	CVS	E8H	CVD
E9H	EOF	EAH	LOC	EBH	LOF	ECH	MKI\$	EDH	MKS\$
EEH	MKD\$	EFH	CINT	FOH	CSNG	F1H	CDBL	F2H	FIX
F3H	LEN	F4H	STR\$	F5H	VAL	F6H	ASC	F7H	CHR\$
F8H	LEFT\$	F9H	RIGHT\$	FAH	MID\$				

Jörg Seelmann-Lagobert  
5305 Alfter 4

## Genietext mit Sonderzeichen

Wer ein Genie hat und mit Genietext arbeitet, sollte schleunigst die Hardwareänderung durchführen, die ich in einem früherem Clubinfo vorgestellt habe, um 32 neue Zeichen auf dem Bildschirm darzustellen.

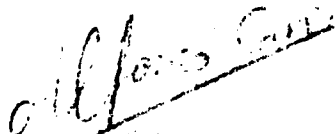
Warum? Ganz einfach: Genietext kann diese Zeichen verarbeiten. Sie sind zu erreichen mit <NEW LINE> und den Zahlen 1-9. Wenn man die Änderung nicht durchgeführt hat, erscheinen gewöhnliche Buchstaben und werden als solche ausgedruckt. Die Sonderzeichen werden als kleine Zahlen (1-9) (So ist es bei meinem Drucker) ausgedruckt. Diese Zeichen haben auch einen Code beim Genietext, nämlich 207 für das Zeichen über der 0 und 216 für Das über der 9. Die anderen liegen dazwischen (208-215).

Was kann man den tolles jetzt damit machen? Man kann nun zum UnterKommando gehen und die Option 'P' anwählen um neue Zeichen zu definieren und diese neue Zeichen den Codes der Zeichen (207-216) geben. Wenn jetzt dieses Zeichen im Text vorkommt, braucht man nicht mehr wie gewohnt ein Doppelkreuz einzugeben (Bsp.: #tl# um ein Telefon auszudrucken), sondern man kann es leicht über die Sonderzeichen machen. Das gute daran ist, daß der Randausgleich nicht gestört wird, wie es bei der normalen Methode üblich ist.

Sollte jemand noch Fragen hierzu haben, möge er mir schreiben. Übrigens! Ich habe bei mir all die spanischen Sonderzeichen definiert und kan somit auch Texte in Spanisch schreiben, was bis jetzt nicht möglich war.

sefoun219

(Alfonso Sanz)



818F 03	00660	DEFB	03H	
8190 4E	00670	DEFM	'KILL SYS11/SYS:0'	
81A0 03	00680	DEFB	03H	
81A1 4B	00690	DEFM	'KILL SYS10/SYS:0'	
81B1 03	00700	DEFB	03H	
81B2 4B	00710	DEFM	'KILL SYS18/SYS:0'	
81C2 03	00720	DEFB	03H	
81C3 4B	00730	DEFM	'KILL SYS19/SYS:0'	
81D3 03	00740	DEFB	03H	
81D4 4B	00750	DEFM	'KILL SYS20/SYS:0'	
81E4 03	00760	DEFB	03H	
81E5 4B	00770	DEFM	'KILL SYS29/SYS:0'	
81F5 03	00780	DEFB	03H	
81F6 4B	00790	DEFM	'KILL BASIC/CMD:0'	
8206 03	00800	DEFB	03H	
8207 20	00810 KOPF	DEFM	'	KILLSYS'
8221 0A	00820	DEFB	0AH	
8222 20	00830	DEFM	'	=====
823C 0A	00840	DEFB	0AH	
823D 0D	00850	DEFB	0DH	
823E 28	00860 RIGHT	DEFM	'(C) by Alfonso Sanz'	
8240 0A0A	00861	DEFW	0A0AH	
8253 0D	00870	DEFB	0DH	
8000	00880	END	INIT	

00000 Fehler  
32740 Zeichen verfügbar

BEFEHL	816E	00630	00210
ERROR	8072	00510	00460
FEHLER	8068	00440	00300
FREE	8091	00530	00180 00380
INIT	8000	00020	00880
KOPF	8207	00810	00040
LOOP	8039	00230	00370
RIGHT	823E	00860	00060
TEXT	8096	00550	00160
TEXT1	80BA	00570	00080
TEXT2	80F2	00590	00100
TEXT3	8130	00610	00120 00400
T.4	816D	00620	00140 00320

## Springen - aber wie?

Wie aus zuverlässigen Clubkreisen verlautet, besteht in einem Volkshochschul-Kurs für Z80-Assembler, den ein Mitglied besucht, Verwirrung darüber, wie sich der absolute (JP, jump) und der relative Sprung (JR, jump relatively) voneinander unterscheiden. Wenn schon der Kursleiter es nicht recht durchschaut, der ja immerhin als Lehrer vor der Gruppe steht, so ist zu befürchten, daß diese Unklarheit unter den Clubmitgliedern erst recht verbreitet ist. Also will ich meinem missionarischen Eifer ein Ventil öffnen:

Eigentlich ist der Unterschied ganz einfach zu beschreiben (nur leider offenbar nicht so einfach zu verstehen): JP setzt den Befehlszähler PC (program counter) auf einen beliebigen neuen Wert, JR erhöht oder erniedrigt ihn um einen bestimmten Betrag von -128 bis +127 Bytes. Daher kann man mit JP an jede Stelle des Speichers springen, während man mit JR nur irgendwohin im Nahbereich von insgesamt 256 Bytes (von PC aus gezählt) springen kann.

Die Beispiellistings veranschaulichen das. Hierzu muß zunächst erklärt werden, daß alle mir bekannten Assemblerprogramme (bzw. deren Editoren) in der Syntax für beide Befehle keinen Unterschied machen. In beiden Fällen wird als Sprungziel eine absolute Adresse angegeben. Bei JP wird diese Adresse für die Assembly ganz einfach übernommen, bei JR wird vor der Assembly zunächst die Sprungdistanz errechnet. Das ist für den User zwar sehr bequem, aber leider ist aus diesem Grunde der Unterschied nur in der Objektcodespalte (2. v. l.) erkennbar:

Im Beispiel 1 erscheint JP 6007h als C3 (Opcode), 07 (LSB) und 60 (MSB der Zieladresse). JR 6007h liefert 18 (Opcode) und 02 (Sprungdistanz als Summand für PC). Der Befehlszähler steht nach dem Lesen des JR-Befehls, also vor der eigentlichen Sprungaktion, auf dem folgenden Byte 6005h. Zum Ziel 6007h fehlen noch 2 Bytes, daher 02 als Distanz.

Im Beispiel 2 wird das Programm nach 8000h verschoben (ORG). Dem JP-Befehl ist das egal. Er kann PC von 8003h (hinter dem Befehl nach dem Lesen) auf 6007h setzen. Vor der JR-Zeile erscheint jedoch die Fehlermeldung "Sprungweite!". Der zulässige Distanzraum von -128 bis +127 Bytes ist weit überschritten. Hier tut der Assembler dem User einen Gefallen: Im Objektcodefeld wird als Distanz in diesem Falle FE (= -2 als Zweierkomplement) assembliert. PC wird bei einem Programmlauf deshalb immer wieder auf den Beginn des Sprungbefehls gesetzt. Sollte der Fehler unbeachtet geblieben sein, so hängt sich das Programm durch pausenlose Sprünge an die alte Stelle auf, ohne Schaden anzurichten.

Es kommt häufig vor, daß man ORG während der Entwicklungsarbeit auf eine neue Adresse setzt. Um dann nicht das ganze Programm nach falschen Sprungzielen durchsuchen zu müssen, werden die Ziele zweckmäßig durch Symbole (Labels) gekennzeichnet. Im Beispiel 3 heißt die Stelle 6007h jetzt "label". Dieses Symbol kann in beide Sprungbefehle eingeschrieben werden. Wie die Objektcodespalte zeigt, ist das Ergebnis dasselbe wie im Beispiel 1.

Nach dem Relozieren nach 8000h (Beispiel 4) tritt kein Fehler auf. Der Objektcode des JR-Befehls hat sich nicht verändert. Relative Sprungbefehle dürfen demnach beliebig verschoben werden ohne Schaden für die Programmlogik. Jetzt verzweigt der JP-Befehl aber nicht mehr nach 6007h, sondern nach 8007h. Der Objektcode ist nicht mehr derselbe. Die Zieladressen absoluter Sprünge müssen deshalb nach der Relokation dem neuen Ladebereich angepaßt werden. So ganz nebenbei ergibt sich daraus der gute Rat, für Sprungziele nach Möglichkeit immer Labels zu verwenden.

Das Relozieren wird problematischer, wenn das Programm bereits fertig im Speicher steht. Dies ist oft notwendig, wenn z. B. der endgültige



Ladebereich erst freigebankt werden muß. Beispiel 5 zeigt, wie man es macht. Unser wohlbekanntes Programm wird bei "init" zunächst in den neuen Bereich übertragen. In Zeile 8 wird es dann an der neuen Stelle angesprungen. Um mit einem Label arbeiten zu können, muß zusätzlich das Label "offset" definiert werden, das die Entfernung des neuen vom alten Ladebereich symbolisiert. Es fällt auf, daß der JR-Befehl davon allerdings nicht betroffen ist, denn dem Befehlszähler PC ist es egal, von welcher Adresse aus er um die zwei Bytes erhöht wird. Nur der JP-Befehl muß um den Betrag "offset" vermindert werden. Das relozierte Programm sieht nun wieder genauso aus wie in Beispiel 1 (Objektcodespalte).

Dies waren die logischen Unterschiede zwischen JP und JR. Es gibt weitere. Wie man an den Objektcodes erkennt, ist JP drei Bytes lang, während JR nur zwei Bytes beansprucht. Im Nahbereich der besagten 256 Bytes kann man daher mit JR immer ein Byte einsparen. Besonders bei Manipulationen im Betriebssystem (Level 2, Disk-BASIC oder DOS) ist es in der Regel so eng, daß sich JPs schlechterdings verbieten, wo die Sprungdistanz klein genug ist.

Die Bearbeitungszeit ist bei beiden Sprungarten ebenfalls verschieden. Ein JP beansprucht immer 10 Taktzyklen. Der JR aus unseren Beispielen kostet 12 Zyklen. Wo es auf extreme Geschwindigkeit ankommt (und Platz für das 3. Byte ist), ist der JP daher unbedingt vorzuziehen. Bei bedingten Sprüngen aber nicht unbedingt: Auch der bedingte JP braucht seine 10 Zyklen. Der bedingte JR braucht seine 12 aber nur, wenn die Bedingung erfüllt ist (z. B. wenn bei JR Z,... das Z-Bit gesetzt ist). Bei nicht erfüllter Bedingung dauert er nur 7 Zyklen und ist damit wiederum dem bedingten JP überlegen. Hier ist die Weitsicht des Programmierers gefordert. Er sollte im voraus wissen, ob die Bedingung bei den meisten Durchläufen erfüllt oder nicht erfüllt sein wird.

Allerdings muß auch das eben Gesagte wieder relativiert werden. Für JR sind nur die Bedingungen Z bzw. NZ und C bzw. NC zulässig. Der bedingte JP kennt aber außerdem noch das Parity- und das Sign-Bit. Daher sind hier auch noch die Bedingungen PE, PO, P und M anwendbar. JR ist in diesen Fällen nicht möglich.

Abschließend sei auf einen Sonderfall des JR hingewiesen, den DJNZ (decrement and jump on non-zero). Je nach dem Inhalt des Registers B wird eine DJNZ-Schleife soundsooft durchlaufen. Das ist wie bei einer FOR-NEXT-Schleife in BASIC. DJNZ hat wie der gewöhnliche relative Sprung nur zwei Bytes. Er dauert 8 Taktzyklen, solange B noch nicht auf 0 dekrementiert ist. Ist B auf 0 angelangt, dann kostet dieser Befehl 13 Zyklen. Dennoch ist er in jedem Falle schneller als ein entsprechendes Konstrukt mit JP, denn er übernimmt auch automatisch die Kontrolle des Registers B. Mit JP müßte sowohl das Dekrementieren als auch das Überprüfen durch gesonderte Befehle bewerkstelligt werden, die natürlich Zeit kosten würden. Allerdings taugt DJNZ ebenfalls nur für einen Zielbereich von -128 bis +127 Bytes, denn er ist auch ein relativer Sprungbefehl.

Besonders bei den Anfängern unter euch war das nun eine geballte Ladung Theorie, die vielleicht nicht so einfach zu verdauen ist. Da die meisten Programme nicht zeitkritisch sind, aber gerne mit wenig Speicherplatz auskommen sollen, ist es empfehlenswert, im Zweifelsfall immer zuerst einen JR zu wählen. Wenn die Sprungdistanz zu weit ist, wird sich der Assembler schon rechtzeitig mit einer entsprechenden Fehlermeldung bemerkbar machen. Dann kann man immer noch den JP dafür einsetzen.

Arnulf Sopp

	00001 :	Beispiel 1:		
	00002			
6000	00003	ORG	6000h	:Ladeadresse
6000 C30760	00004 start	JP	6007h	:absolutes Sprungziel
6003 1802	00005	JR	6007h	:relatives Sprungziel
6000	00006	END	start	:dort Programmansprung

00000 Fehler

	00001 :	Beispiel 2:		
	00002			
8000	00003 :	ORG	8000h	:Ladeadresse
8000 C30760	00004 start	JP	6007h	:absolutes Sprungziel
Sprungweite!				
8003 18FE	00005	JR	6007h	:relatives Sprungziel
8000	00006	END	start	:dort Programmansprung

00001 Fehler

	00001 :	Beispiel 3:		
	00002			
6000	00003	ORG	6000h	:Ladeadresse
6000 C30760	00004 start	JP	label	:absolutes Sprungziel
6003 1802	00005	JR	label	:relatives Sprungziel
6005 00	00006	NOP		:Nonsensbytes
6006 00	00007	NOP		:bis 6007h
6007 00	00008 label	NOP		: = hier
6000	00009	END	start	:dort Programmansprung

00000 Fehler

	00001 :	Beispiel 4:		
	00002			
8000	00003	ORG	8000h	:Ladeadresse
8000 C30780	00004 start	JP	label	:absolutes Sprungziel
8003 1802	00005	JR	label	:relatives Sprungziel
8005 00	00006	NOP		:Nonsensbytes
8006 00	00007	NOP		:bis 8007h
8007 00	00008 label	NOP		: = hier
8000	00009	END	start	:dort Programmansprung

00000 Fehler

	00001 :	Beispiel 5:		
	00002			
8000	00003	ORG	8000h	:Ladeadresse
8000 210EB0	00004 init	LD	HL, start	:Quelladresse
8003 110060	00005	LD	DE, 6000h	:Zieladresse
8006 010800	00006	LD	BC, finito-start	:Anzahl Bytes
8009 EDB0	00007	LDIR		:verschieben
800B C30060	00008	JP	start-offset	: = 6000h
	00009			
800E C30760	00010 start	JP	label-offset	: = 6007h
8011 1802	00011	JR	label	: = +2 Bytes
8013 00	00012	NOP		:Nonsensbytes
8014 00	00013	NOP		:bis 8014h
8015 00	00014 label	NOP		:bzw. 6007h
8016	00015 finito	EQU	\$	:Ende-Flag
200E	00016 offset	DEFL	start-6000h	:Ladeabstand
8000	00017	END	init	:dort Anspr.

00000 Fehler

## "Hombberger Nachlese"

- 1) Datenbank "SUPER" heißt jetzt "GETT" von  
GE enie  
+ TT RS80

Anmerkung: Das Doppel-T: gebietet die Club-Gerechtigkeit. Es ist der Ausgleich dafür, daß es in diesem doppelt so viele GENIE-User wie TRS80-Leute gibt...

\*\*\* So beschlossen am 02.Juno ds.Js. um 00.00.00 Uhr \*\*\*  
\*\*\* im Güntersteiner Hof unter 8 Augen ! \*\*\*

=> ACHTUNG! Club-Datenbank GETT bitte nicht verwechseln <=  
=> mit Club-Maßeinheit GET! (s.INFO 5/85,S.20) <=

- 2) Falls eine Steigerung dieser recht brauchbaren Datenbank bekannt werden sollte, könnte diese dann den Namen "GETTER" erhalten.

Kajott erinnerte jedoch bei dieser Wortschöpfung daran, daß diese Vokabel in der Technik bereits vergeben ist, hatte aber um 00.00.00 Uhr nach dem S.LICHER Bier ein solches Vakuum (=Leere) im Kopf, daß die Erläuterung auf dieses INFO vertagt und die Anwesenden auf heute vertröstet werden mußten - was nun allen Abwesenden auch zugute kommt. Nun wird das Vakuum aber gleich verständlich: Ein GETTER ist in der Tat

wie Arnulf Sopp  
mit kühl'rem Kopp

bereits andeutete, ein "besserer Nehmer" (oder Übernehmer), nämlich ein (Meyers Physiklexikon) :

Fangstoff, der in der Lage ist, letzte Spuren von schädlichen oder störenden Gasen aus Hochvakua durch Sorption oder durch chemische Bindung zu entfernen!

Also äußerst wichtig (für Nutzung in gewissen Oertchen allerdings zu teuer).

Ergo: Sollte Datenbank GETT verbessert werden müssen, nennen wir sie dann einfach - na, wie denn ?

(Natürlich nicht G E T T E R  
sondern B E T T E R)

\*\*\*\*\* Euer KAJOTT \*\*\*\*\*

## Sei Dein eigener "Diktator"!

Der Trick stammt nicht von mir und ist manchem daher wohl schon bekannt- aber sicher nicht jedem, der sich das Abtippen (manche nennen es erbost "Abklopfen") langer Listings oft verkniffen hat, und war das Programm auch noch so vielversprechend, nur weil man Augen- und Nackenschmerzen (vom Wackelkopp) davon kriegt. Für diese Gequälten und Frustierten möchte ich ihn weitergeben: Nimm Deinen Kassettenrekorder oder in Ermangelung desselben ein beliebiges Tonbandgerät (auch minderer Qualität) und lies das Listing (natürlich bei eingeschaltetem Aufnahme-Modus) mit ruhiger, gelassener Stimme vor Dich her, so wie im Selbstgespräch (es soll Leute geben, die lesen immer so), aber mit allen Kommata, Doppelpunkten, Anführungszeichen und und und (nur Fliegendreck weglassen!)- Das ist übrigens sehr gemütlich, man kann es bei Kerzenschein verrichten...

Anm.: Wer Dich als C-Experten bisher schon immer bestaunte, wird, so er Dir zuhört, nun erst recht in grenzenlose Bewunderung für Dich ob Deiner esoterischen Sprachkenntnisse ausbrechen...

Ist dieses "Diktat" gelaufen, kehrst Du die Rollen um und spielst nun die "Sekretärin". Horche Dein Anglojapinesisch, mit deutscher Interpunktion in Reih und Glied gebracht, vom Band ab und tipp im gleichen Rhythmus mit - ohne mit den Wimpern zu zucken oder dem Kopp zu wackeln! Nur: "Ohren auf!" (Kopfhörer!) Dies läuft und läuft...vorausgesetzt, Du hast nicht schneller diktiert, als Du tippen kannst; also lieber gleich "TEMPO 50 BpM" \*).

**Happy Tipping !**

BpM = Bytes per minute

*Kajitt*

(Achtung: Urheberrechtlich ungeschützt!)

-----  
Fragen, Antworten und Tips  
\*\*\*\*\*

Zwei Fragen von Jörg Seelmann-Eggebert

1. Kann mir jemand das Buch "Machine Language-Disk I/O & other mysteries" ausleihen oder kopieren ?
2. Hat jemand die Grafikkarte vom NDR-Computer (512 \* 256 Punkte, 4 Grafikseiten) an seinem Genie (oder TRS80) angeschlossen und kann mir dabei Tips geben bzw. Fragen beantworten ?

```

1 REM Programmname: 'MOLEKRIS' * <C> KaJot Muehlenbein 16/12/84
3 CLS:DEFINTI,M,N,P,W,X,Y
5 PRINT"*****"
10 PRINT" Grafische Darstellung der BROWN'schen Molekuelbewegung
      bzw. des Kristallwachstums
11 PRINT"*****
*
12 PRINT"

Je groesser die Anzahl Molekuele gewaehlt wird,
desto mehr naehert sich das Modell der Natur !
      (Vorschlag: Beginne mit N=100)
15 PRINT:PRINT
17 PRINT"Die Molekuele befinden sich in einem geschlossenen Kasten,
den sie nicht durchdringen koennen.
21 INPUT"

Molekularbewegung (1) oder Kristallwachstum (2) ";W
22 W$(1)="Molekuel":W$(2)="Kristall"
23 CLS:PRINT"Wieviele "W$(W);"e";:INPUT" sollen dargestellt werden
";M
24 DIMPX(M),PY(M)
25 INPUT"Wieviele Schritte sollen von jedem ausgefuehrt werden";N
27 CLS:PRINT"ACHTUNG ! Bei Zusammentreffen zweier ";W$(W);"e
      ertoent jedesmal ein Signal !":FORWW%=1TO1000:NEXT:CLS
30 RANDOM:FORI=1TOM:PX(I)=RND(127):PY(I)=RND(47)
40 SET(PX(I),PY(I))
50 NEXT
55 S(1)=0:S(2)=1:S(3)=-1
60 FORJ=1TON:FORI=1TOM
100 RANDOM
110 X=RND(300):Y=RND(300)
120 IFX<=100,X=1
130 IFX>100ANDX<=200,X=2
140 IFX>200,X=3
150 IFY<=100,Y=1
160 IFY>100ANDY<=200,Y=2
170 IFY>200,Y=3
180 PX=PX(I)+S(X):PY=PY(I)+S(Y)
190 IFPX>127ORPX<0,100
200 IFPY>47ORPY<0,100
210 IFS(X)=0ANDS(Y)=0,100
212 PX(I)=PX:PY(I)=PY
215 IFPOINT(PX(I),PY(I)),LPRINTCHR$(07);:GOTO230
216 A=PX(I)-S(X):B=PY(I)-S(Y)
217 IFA<=0OR A=>127OR B<=0OR B=>47,230
218 IFW=1RESET(PX(I)-S(X),PY(I)-S(Y))
220 SET(PX(I),PY(I)):IFW=1THEN180
230 IFI<M:PRINT$960,USING"##";J;:PRINT". Weg von ";W$(W);I+1;
240 NEXTI,J
250 LPRINTCHR$(07);CHR$(07);CHR$(07);CHR$(07);CHR$(07)
260 GOTO260

```

*Anmerkung:*  
*AS* opp kann dies sicher mittels  
 sembler beschleunigen!

[Du auch, L.L.!?]

*Kjot*

33 6/85



- Siggi Bach hat zwei Fragen:

1. bezgl. SCRIPSIT: Ich fände es eine sinnvolle Erweiterung, wenn mittels Tastencode, z.B. SHIFT Ø, das Programm zu der letzten Cursorposition zurückfahren könnte, ähnlich eines Korrektur-Resets an einer Typenradschreibmaschine. Wer hat schon am SCRIPSIT herumgebastelt und weiß einen Ansatzpunkt? Außerdem möchte ich gern das Cursorblinken abstellen können.

2. alphabetisches Directory. Ich erinnere mich, ein derartiges Programm im Info gelesen zu haben, kann es aber nicht mehr finden. Wer weiß Rat ?

- Klaus-Jürgen Mühlenbein hat eine Bitte an alle Mitglieder, die Beiträge fürs Info einschicken. Er bittet, alle Beiträge mit mindestens drei Stichwörtern zu versehen. Unter dieser Voraussetzung kann er die INFOTHEK rationeller weiterführen.

---

Ein Buch nach dem Computer

P.-J. Schmitz

TURKLE, Sherry: Die Wunschmaschine: Vom Entstehen der Computerkultur, Rowohlt, Reinbek bei Hamburg 1984

Zum Inhalt: Das Buch untersucht weniger nach neuen Einsatzgebieten für Computer, als vielmehr danach, wie diese Maschine in unser gesellschaftliches Leben und die seelische Entwicklung des einzelnen eindringt und unser Denken beeinflusst, vor allem das Denken über uns selbst. Angesprochen wird nicht wie Computer zukünftig sein werden, sondern wie wir sein werden.

---

Aus GENIE DATA Sept./Okt. 84

# SCHWARZmarkt

Wie man immer öfter hört u. liest gibt es in Deutschland einen Software Schwarzmarkt. Hier wird von Schäden in Millionenhöhe gesprochen, die von illegalen Programmkopierern verursacht werden. Man liest von Verhaftungen, Beschlagnahmungen, Schadensersatzklagen und Gefängnis.

Wir wollten einmal wissen, was an diesen Berichten wahr ist, und ob es wirklich so schlimm bestellt ist.

Also haben wir über Kontakteleute Kleinanzeigen veröffentlicht, und die Schwarzkopierer aufgefordert, uns ihre Tauschlisten zu senden. Und siehe da, die Post kam! Die tollsten Colour

Genie-Programme wurden hier zu Spottpreisen angeboten, oder sogar kostenlos, nach dem Motto: „Gibst Du mir, geb ich Dir!“

TCS-Programme, Programme der Fa. Schmidtke und Programme von uns und andere waren hier zu finden.

Aber nur illegal hergestellte Kopien, gefertigt in trügerischer Absicht, mit einem Status, der in etwa dem von Falschgeld entspricht! Ja, der Schwarzmarkt blüht! Hier wechseln Programme, oftmals für viele tausend Mark, den Besitzer, ohne das der rechtmäßige Autor, oder die Vertriebsfirma einen roten Heller sehen.

Da sollte man sich als ehrlicher Computerbesitzer nicht wundern, wenn es bald keine neuen Programme mehr gibt! Denn wenn auf ein verkaufte Programm zehn Raubkopien kommen, wird die Programmentwicklung bald finanziell uninteressant! Wir sollten also alle etwas dafür tun, daß in Zukunft den Raubkopierern keine Chance bleibt!

Sie, indem Sie keine Raubkopien erwerben,  
WIR, indem wir unsere Adressen an unseren Anwalt weiterleiten.

GENIE DATA REDAKTION.