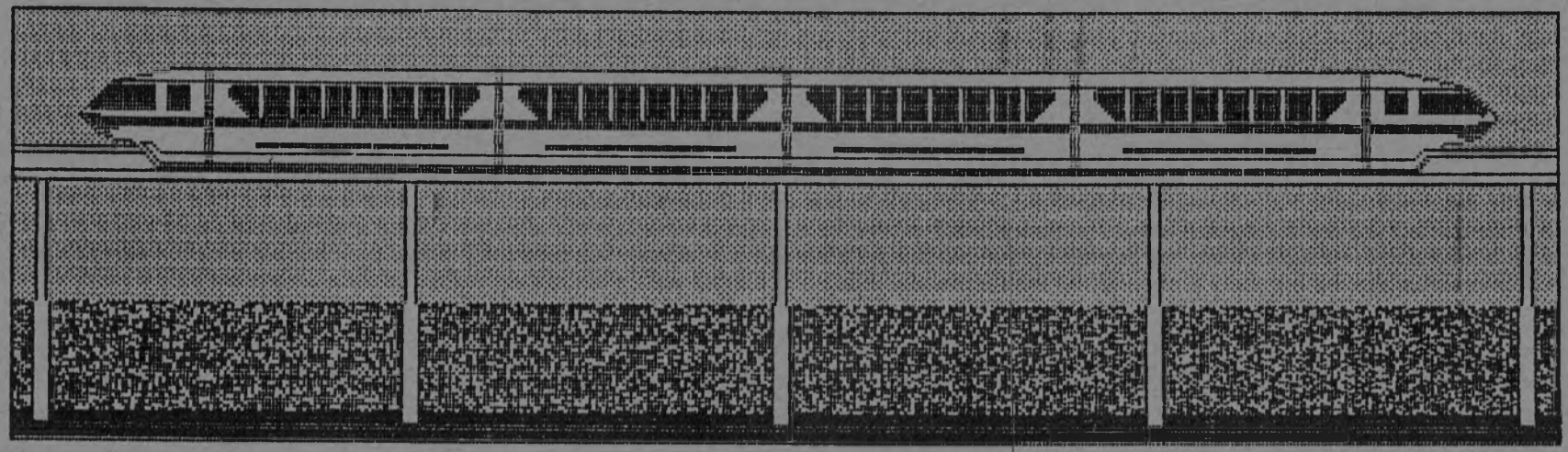
TRS-80 USER CLUB MÜNCHEN



CLUBZEITUNG

26. AUSGABE

INHALT:

ClubKonto: PostscheckAmt Muenchen BLZ: 700 100 40
Kontonr.: 3452 35-800 Gregor Thalmeier
Monatsbeitrag: 4.- DM

Termine fuer Clubtreffen:

Mittwoch	24.4.85
Mittwoch	22.5.85
Mittwoch	19.6.85
Mittwoch	31.7.85

Die Treffen finden jeweils um 19 Uhr statt in der

Gaststätte Kriegersiedlung
Albert-Roßhaupterstr. 61
8000 München 2

Der Termin für das Juli-Treffen kann sich eventuell noch ändern

SOFTWARELISTE
DAS ENDE EINES VERSUCHS

Liebe Clubfreunde.

Wie aus der Titelzeile vielleicht schon zu entnehmen ist, muß ich Euch mitteilen, daß das Echo auf meine zwei Aufrufe zu gering war, um eine vernünftige Liste "auf die Beine zu stellen". Hier danke ich noch denen, die sich bereits die Mühe gemacht haben, Ihre Programme zu katalogisieren. Aber durch die Beteiligung von nur rund 10 % fehlt einfach das Verhältnis zu dem, was geplant war.
Alle Daten über Eure Programme habe ich gelöscht und ausgedruckte Listen vernichtet.

Frohes "computern" weiterhin

Bernd

Hardcopy des kompletten Bildschirms

In der 24. Ausgabe der Clubzeitung stellt Bernard Haible in seinem Epson-Artikel zur Demonstration der Graphikmöglichkeiten des FX-80 ein paar HRG-Hardcopies vor, darunter auch die Demo-Graphik der HRG 1b von RB-Elektronik. Auf den folgenden Seiten tue ich desgleichen. Seine Ausdrucke entstanden vermutlich mit HRGEPSON/CMD und/oder GRAPE/CMD, meine mit JKL.

Die JKL-Funktion unserer diversen DOS'es muß mit dem Dilemma klarkommen, daß kaum ein Drucker dieselben Steuercodes hat wie irgendein anderer. Die logische Konsequenz ist, daß nur ASCII-Zeichen ausgedruckt werden. Die Pixelgraphik und gar die HRG bleiben unberücksichtigt. Sowohl mit dem DOS als auch der HRG 1b werden Treiberprogramme vertrieben, die eine Hardcopy beider Graphikarten für verschiedene gängige Drucker ermöglichen. Der Haken: Die Programme bewältigen nur jeweils eine Graphikart und keine ASCII-Zeichen. Außerdem sind alle langsam und geben die Proportionen des Bildschirms verzerrt wieder.

Daß ich mir selber ein Alternativprogramm schrieb, liegt nicht so sehr an meinem Ehrgeiz als an der bedauerlichen Tatsache, daß es für den Gemini-10X keine Software gibt. Der Ehrgeiz ist nur schuld daran, daß meine Version in mehrfacher Hinsicht besser wurde:

Sie residiert in einem ehemals freien SYS-File (SYS26/SYS, das beim Genie 3 übrigens nicht frei ist) und lädt nach 4D00. Auf diese Weise nimmt sie keinen Platz im Anwenderspeicher ab 5200 weg. Das System hat mit RST28 einfachen Zugriff, so daß JKL keine Verrenkungen machen muß, um die Erweiterung zu erreichen. Der Bildschirm wird zentriert und mit dem genauen Breiten-/Höhenverhältnis ausgedruckt. Es gibt die Möglichkeit, nur ASCII, ASCII mit Pixelgraphik und ASCII mit Pixelgraphik und HRG auszudrucken. Das Ganze geht wahlweise positiv (die dunklen Blanks werden schwarz, die hellen Pixels bleiben weiß) oder negativ (weiße Blanks, schwarze Pixels). ASCII-Zeichen werden immer schwarz auf Weiß gedruckt. Die reine Rechenzeit einer Hardcopy beträgt etwa 6 Sekunden. Zum Vergleich: HRGEPSON/CMD rechnet 76 Sekunden lang.

Aufgerufen wird der Bildschirmausdruck folgendermaßen: Nach JKL sucht das Programm automatisch nach Pixelgraphik. Gibt es keine, wird die alte Routine angesprungen, die nur ASCII ausdruckt und gegf. Punkte statt Graphik setzt. Ist sie aber vorhanden, ertönt vom Drucker ein Piepton. Jetzt hat der User drei Möglichkeiten der Eingabe: Mit A erfolgt ein Ausdruck wie gewohnt (s. o.). Mit P wird es ein Graphikausdruck in positiver Darstellung, mit N ein negativer. Wird JKL mit Shift gedrückt, piept es auch, und dieselben Möglichkeiten stehen offen. Wird jetzt N oder P eingegeben, ist auf der Hardcopy auch die HRG mit drauf.

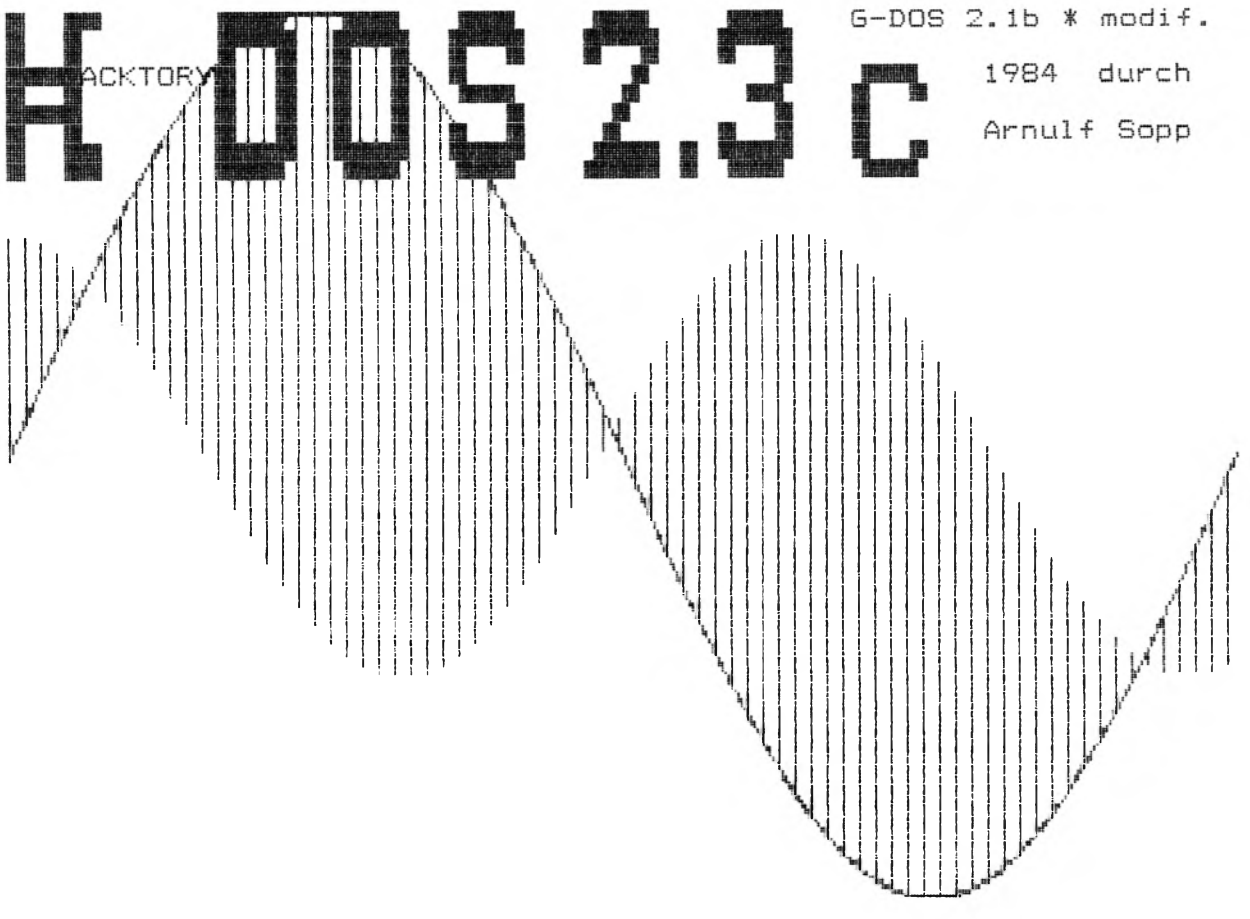
Genug der Prahlerei. Der Grund, weshalb ich das schreibe, ist nicht Reklame (was hätte ich davon?), sondern es ist mehr als Hilferuf zu verstehen. Diese Routine ist Bestandteil meines H-DOS, einer weiterentwikkelten und entwanzten Version von G-DOS. Es soll natürlich nicht nur auf meiner eigenen Maschine mit dem Gemini laufen, sondern mit beliebigen Druckern zusammenarbeiten. Als kleiner lonesome hacker kann ich mir nur den einen leisten; Modifikationen für andere Printer müßte jemand anders vornehmen, gerne in Zusammenarbeit mit mir. Wer sich in der Maschinensprache auskennt, die hochauflösende Graphik seines Druckers beherrscht und überhaupt Interesse an diesem Projekt hat, wende sich bitte an mich (s. Adressenliste der Mitglieder). Dann schicke ich ihr oder ihm eine Diskette mit H-DOS (bitte Spuren, Seiten und Dichte angeben) und ein Listing der Routine sowie des Zaps in SYS3/SYS, der die Erweiterung ansteuert.

Abschließend noch einige Erläuterungen zu den Hardcopies. Auf der folgenden Seite ganz oben steht ein Ausdruck wie gewohnt: ASCII wird wiedergegeben, Pixelgraphik durch Punkte ersetzt, die HRG wird nicht berücksichtigt. Der Ausdruck ist linksbündig und in der Höhe gestaucht. Er entstand mit JKL und der A-Option. Darunter ist eine Hardcopy mit HRGEPSON/CMD. Die betreffenden Steuerzeichen stimmen offenbar mit denen des Gemini überein, so daß ich dieses Programm fahren konnte. Es folgt ein Ausdruck mit JKL und anschließendem Shift-N. No comment.

Die obere Graphik auf der übernächsten Seite ist mit N und ohne Shift entstanden. Die HRG fehlt, der Rest ist da (ebenfalls auf dem Papier zentriert und in den richtigen Proportionen). Der Ausdruck darunter ist das Resultat von JKL und Shift-P. Es fällt auf, daß die ASCII-Zeichen, obgleich doch hell auf dem Bildschirm, schwarz gedruckt wurden. Das sind die gewöhnlichen Zeichen des Druckers. In negativer Darstellung hätte ich jedoch unter Aufwand vieler, vieler Bytes einen negativen (in diesem Sinne eigentlich positiven) Zeichensatz programmieren müssen. Für ein DOS-Feature wäre dieser Aufwand nicht mehr zu rechtfertigen gewesen. Stattdessen hilft sich diese JKL-Erweiterung auf eine andere Weise: Pixel- und hochauflösende Graphik sparen diejenigen Stellen aus, wo Schrift steht. So bleibt sie lesbar.

Und hier straft mich die letzte Abbildung Lügen: Der gesperrte Text "Elektronik GMBH" ist sehr undeutlich. Das liegt an einer Ungenauigkeit übereinanderliegender Spalten. Leider wird die linke Flanke der Buchstaben von der darübergedruckten Graphik angeschnitten. Im unidirektionalen Druck würde das zwar nicht passieren, aber es würde entsprechend länger dauern.

Arnulf Sopp



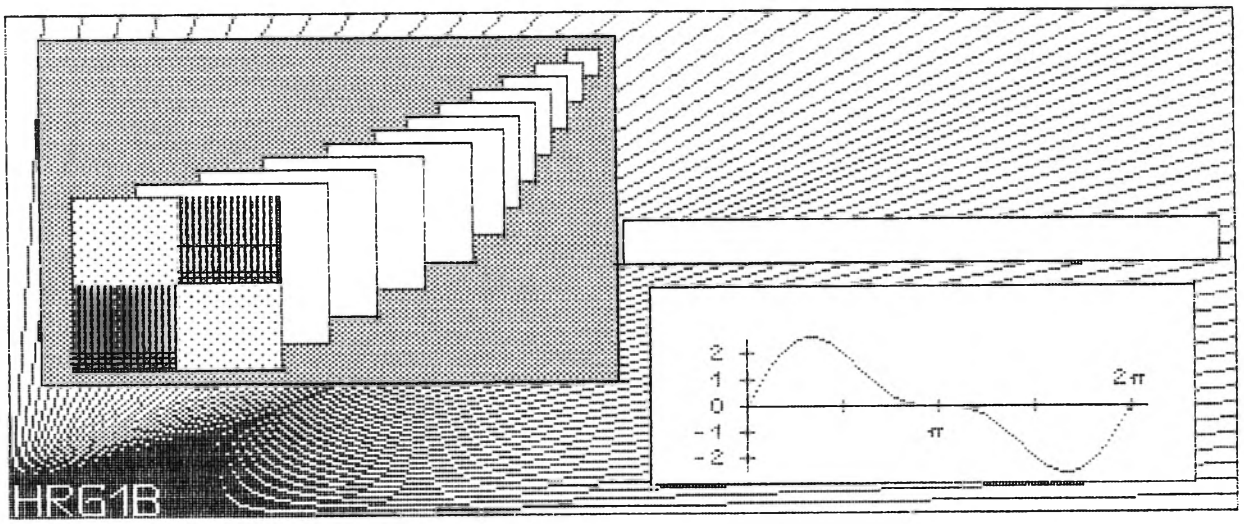
High - Resolution

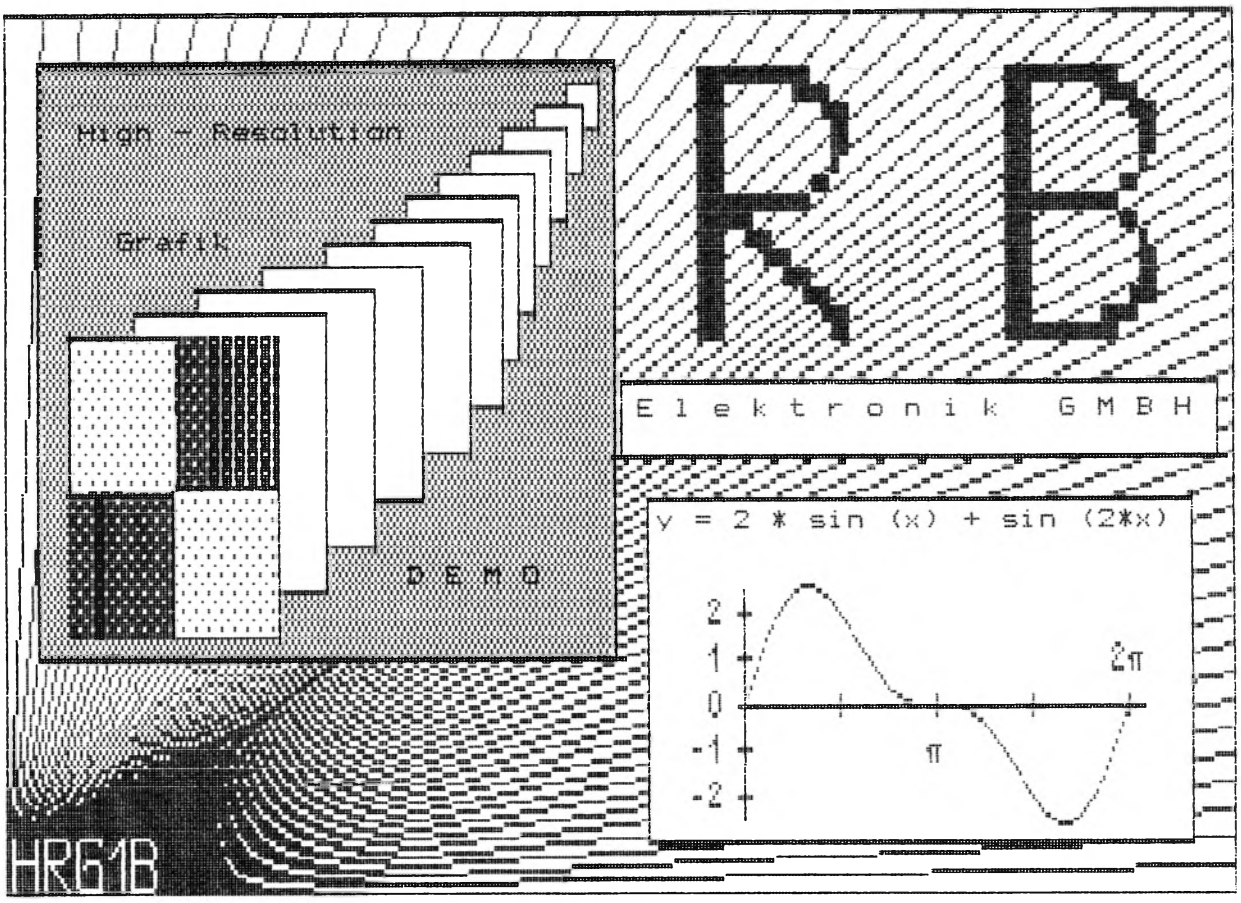
Grafik

E l e k t r o n i k G M B H

y = 2 * sin (x) + sin (2*x)

D E M O





High - Resolution

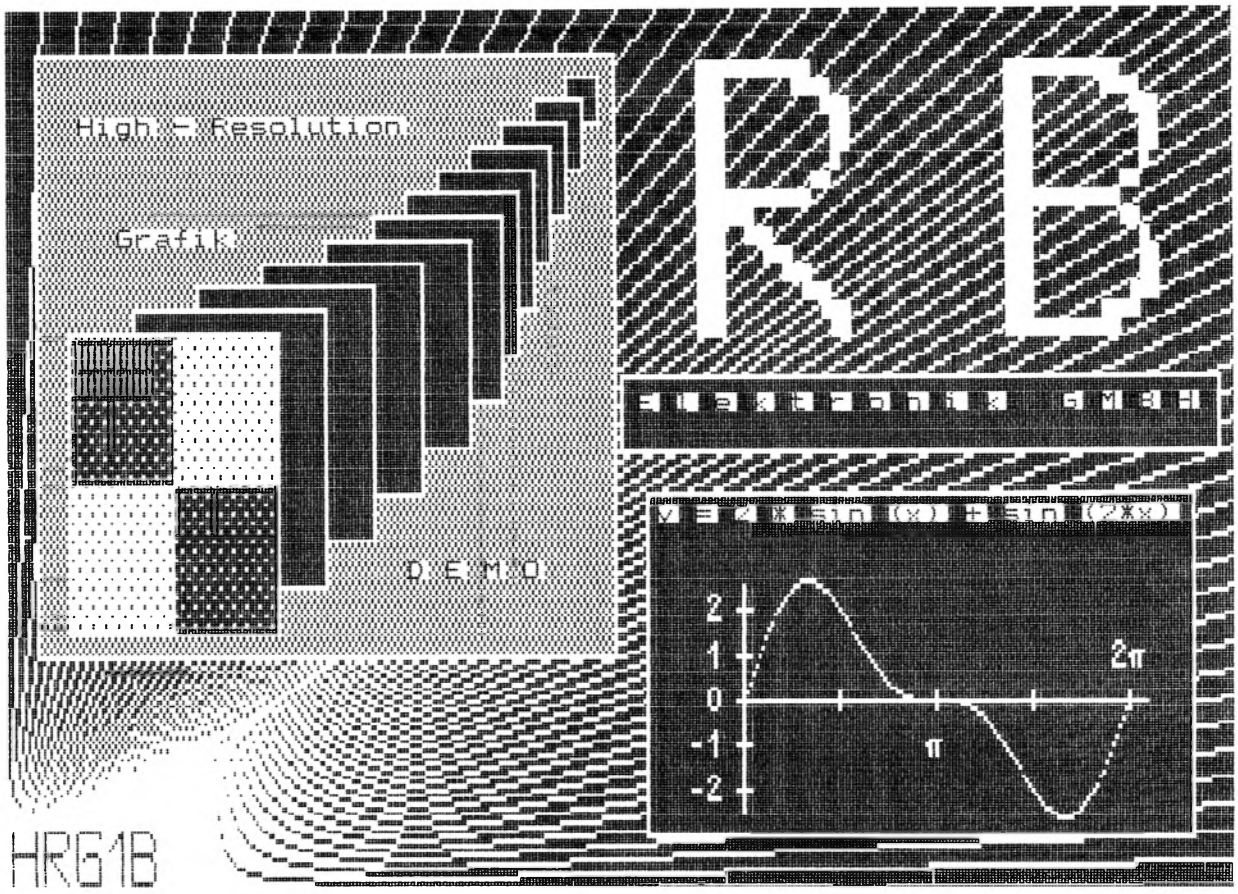
Grafik

R B

E l e k t r o n i k G M B H

y = 2 * sin (x) + sin (2*x)

D E M O

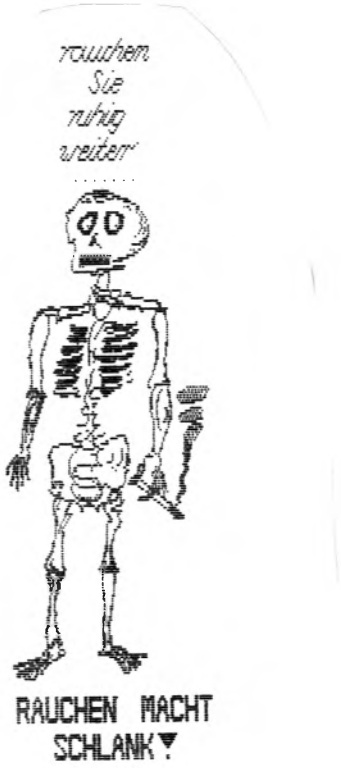


Alfons Kosthorst Klausenhofstraße 106 4236 Haminkeln 2 Dingden

Nach mehreren Jahren der Mitgliedschaft habe ich mich jetzt mal aufgerafft, einen Brief zu schreiben.
Angeregt durch den Hinweis auf das Funkdat Programm in der 25. Ausgabe von Hans Jürgen Miesen habe ich mich veranlasst gesehen, den folgenden Artikel der TRS 80 Clubzeitung zur Verfügung zu stellen. Der Artikel wurde ursprünglich für die CQ-DL geschrieben und zwar vom DL3QP OM Bernhard Unland.
Ich hoffe das ich hiermit einen kleinen Beitrag zur Clubzeitung bringe der sicher für die Amateurfunker unter uns interessant sein dürfte.
Das ich erst jetzt schreibe,hängt damit zusammen das ich seit meinem 21. Lebensjahr durch eine Unfall verursacht, an den Rollstuhl gebunden bin. Das Schreiben macht mir sehr viel Mühe, deshalb will ich auch nicht so viel schreiben.
Ich will jedoch nicht versäumen darauf hinzuweisen, das auch ich an der Funkdatenübertragung mit besten Erfolgen teilnehme.

Mit den besten Grüßen
Alfons Kosthorst dk9ji

Ps. Wer hat schon Erfahrung mit dem Video Schneeschieber für die HRG Grafick !!! fragt Horst Weikamp Tel. 02871/12835
Ich rufe zurück !!!



Funkdat - eine Alternative?

Zugegeben,
Packet Radio ist eine feine Sache, zumal mit Zunahme der Beschreibungen einer Einführung nichts mehr im Wege steht. Aber nach Studium des Artikels in der CQ/DL (Clubzeitung des Deutschen Amateur Radio Clubs) Heft 12/84 haben doch sicherlich einige Amateure an ihren schmalen Geldbeutel gedacht und die Aufnahme der Neuerung in ihr Chack vorläufig zurückgestellt. Aber, warum in die Ferne schweifen, wenn Gutes so nahe und bereits perfekt entwickelt und erfolgreich getestet?

Weiter zugegeben:
Dieser Artikel wendet sich nur an Besitzer von TRS 80 oder kompatiblen Systemen, gleichzeitig soll er aber Besitzer von anderen Systemen auffordern, dieses vom Amateur fuer Amateure geschriebene Programm auch auf anderen Computern lauffähig zu machen.

Zum Thema:
Seit etwa Mitte 1981 hat DC9NG ein Programm zur Übertragung von Computerprogrammen und Texten entwickelt und in der neuesten Version bis zur "Kurzwellentauglichkeit" gebracht.
Das Programm gestattet die fehlerfreie Übertragung aller Programme oder Texte. Ausgedehnte Tests, auch solche auf Kurzwelle unter schwierigen Bedingungen, haben bewiesen, daß das Programm allen Anforderungen gerecht wird.
Es ist sehr bedienungsfreundlich und stellt - zusammen mit den lächerlich geringen Kosten für die Hard - und Software - eine diskussionswürdige Alternative zum Packet Radio dar.

Hardware:
Erforderlich sind: 1 Computer, TRS80 o.ein kompatibler, 64 k,
1 Diskettenlaufwerk
1 Interface
1 Transceiver
1 RTTY Konverter mit AFSK (nur bei erschwerten Bedingungen).

Die Kosten für das Interface betragen in der einfachen Form, das bereits für UKW FM Übertragungen ausreicht, ca. 2 DM, in Worten - zwei Deutsche Mark -, Wobei gesagt werden muß, das man auch hierauf noch verzichten kann. Für Übertragungen unter den genannten schwierigen Bedingungen benötigen Sie ein RTTY Interface, falls Sie nicht schon RTTY über den Computer machen können. Alle Interface, die mit dem M80 , M800, M8000, CQ 2081 oder CWR 80 Programm laufen, können benutzt werden. Die Kosten dafür betragen normalerweise 30 bis 40 DM.

HRG — aber fix!

Einerseits ist es schön, daß mein H-DOS oder auch nur die JKL-Erweiterung zum Ausdruck von hochauflösender und Pixelgraphik von etlichen Clubmitgliedern benutzt werden. Andererseits aber mußte ich zur Demonstration der Hardcopy-Routine jedesmal eine möglichst komplizierte Bildschirmgraphik erstellen und ausdrucken. Es ist reichlich öde, zu warten, bis z. B. RBDEMO fertig auf dem Bildschirm steht, damit ich es drucken kann.

Diese Zeiten sind nun vorbei. Ich kann mir vorstellen, daß viele von Euch ebenfalls gerne eine häufiger gebrauchte Graphik sofort auf dem Bildschirm haben möchten. Die Routinen der verschiedenen HRG-Treiber zum Abspeichern und Einlesen von Graphiken berücksichtigen aber leider nur die HRG, nicht jedoch die ASCII-Anteile und die Pixelgraphik. Zu dumm dafür sind die Autoren mit Sicherheit nicht. Vielleicht zu arrogant, um sich mit ganz gewöhnlichen Buchstaben abzugeben? Auf der letzten Seite dieses Artikels ist ein Assemblerlisting wiedergegeben, mit dem das geht.

Bei mir heißt es GRA/CMD. Um Programmaufwand zu sparen, geht es einen sehr simplen Weg: Unser DOS (welches auch immer) ist in der Lage, einen bestimmten Speicherbereich mit dem DUMP-Befehl auf Diskette zu schreiben. Das Befehlswort, gefolgt vom Filenamen (hier BILD/CMD), der Anfangs-, der End- und der Startadresse sowie dem NEW-LINE-Byte 0Dh wird in den DOS-Eingabepuffer geschrieben. Später wird dieser Befehl mit einem Sprung nach 4405h ausgeführt.

Zuvor soll es aber gerne etwas zu dumpen geben. Dazu wird zunächst ein kleines Programm in den DUMP-Bereich ab 8000h geladen, das die Graphik, wenn man sie mit dem Befehl BILD wieder von der Diskette holen und anzeigen will, in den Bildschirm lädt. Dieses Programm wird gleich mit der Graphik auf die Platte geschrieben, so daß das Bild seine eigene Anzeigeroutine gleich mitführt. Sie beginnt in GRA/CMD beim Label DSPLAY. Es erübrigt sich, sie zu erläutern, denn sie arbeitet fast genauso wie das Hauptprogramm, das später erklärt wird.

Direkt dahinter kommt nun der "normale" Bildschirm, also der Inhalt des Speicherbereichs 3C00-3FFFh. ASCII-Zeichen und Pixelgraphik sind damit in den Puffer gerettet. Die hochauflösende Graphik aus dem eigenen Speicher der HRG 1b muß über die Ports 2 und 3 adressiert und dann über den Port 4 ausgelesen werden. Das jeweils gefundene Byte wird in den Puffer an die aktuelle Stelle gepackt, auf die DE zeigt. Die Zeiger auf die nächste Stelle im HRG-Speicher (Quelle) sowie im Puffer (Ziel), HL und DE, werden nun auf die nächste Stelle erhöht. Jetzt muß geprüft werden, ob die höchste HRG-Speicherstelle 2FFFh bereits überschritten ist. Ist das nicht der Fall, ist also das MSB in H noch kleiner als 30h, dann geht es in LOOP1 weiter wie gehabt. Andernfalls ist es jetzt Zeit für ein DUMP.

Das mitgedumpte Anzeigeprogramm tut genau das Gegenteil der Save-Routine: Nachdem zunächst der Bildschirm gelöscht und die HRG eingeschaltet wurde, werden die ASCII- und gewöhnlichen Graphikzeichen aus dem Puffer in den Bildschirm übertragen. Analog zum Abspeichern werden nun die HRG-Codes über die Ports auf den Bildschirm übertragen. Nach getaner Arbeit wartet das Programm auf irgendeinen Tastendruck. Solange er nicht erfolgt ist, steht die Graphik, ohne daß sie durch ein Prompt oder den

Cursor gestört würde. Jetzt kann ich meine JKL-Demonstration vom Stapel lassen. Wenn der geneigte Leser diesen Bedarf nicht hat, bleibt es ihm unbenommen, den Befehl JP 0049H durch ein schlichtes RET zu ersetzen. Dann kehrt das Programm eben sofort ins DOS oder wohin auch immer zurück.

Die Mitdenker unter Euch werden gemerkt haben, daß der DOS-Befehl GRA (den ihr mit dem Namen des Programms nach Laune umbenennen könnt), da er ja schließlich angezeigt wird, zum Bestandteil der Graphik wird. Das sieht nicht gut aus. Die Lösung ist einfach: Das Programm, das die Graphik erzeugt, kriegt an einer passenden Stelle einen verpaßt: Bei RBDEMO z. B. wird in die Zeile 2030 anstelle der INKEY$-Akrobatik der Befehl CMD"GRA" eingeschrieben. Das hält den Bildschirm sauber und die Fingergelenke fit.

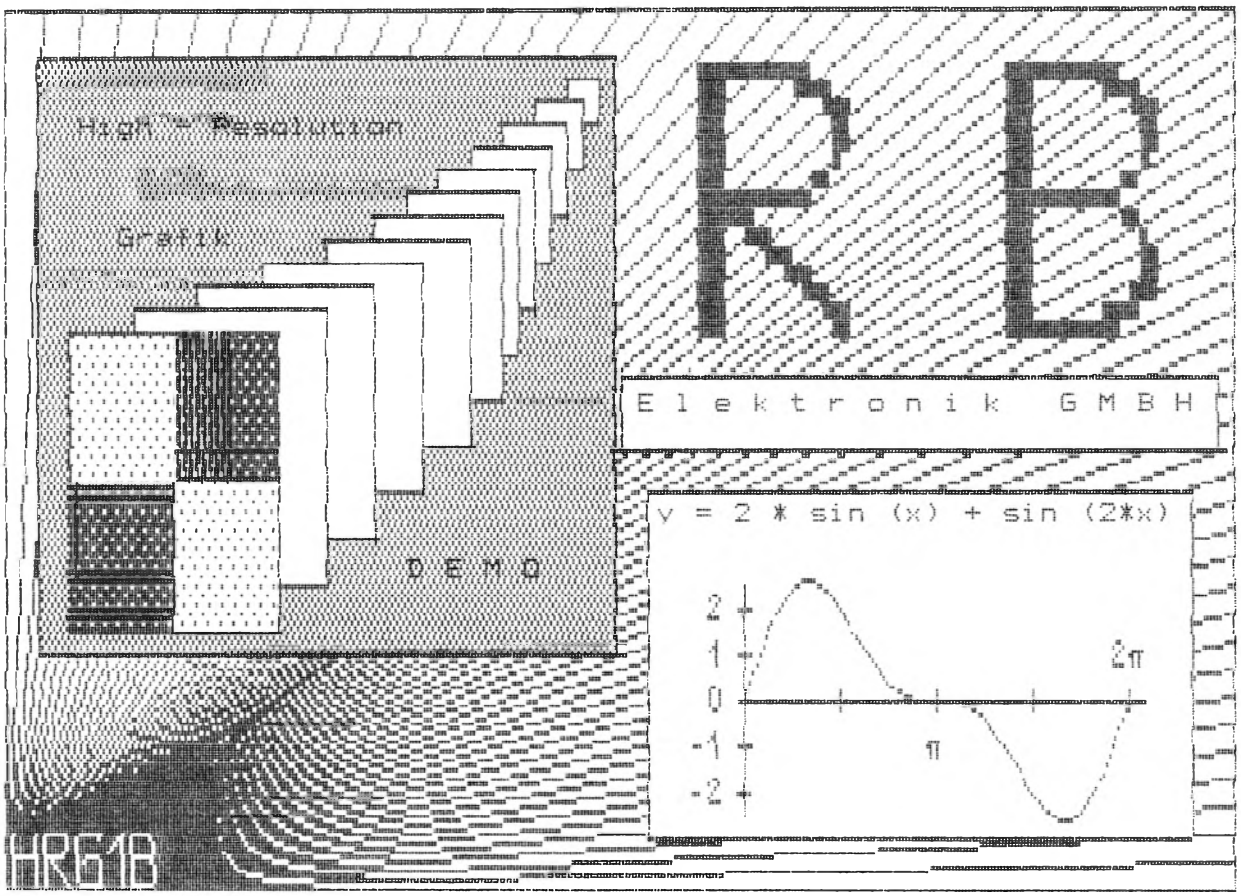
Nun steht da also ein File namens BILD/CMD auf der Platte. Bei der nächsten Graphik wird derselbe Dateiname benutzt. Das bedeutet, daß die vorhergehende Graphik einfach überschrieben wird. Eine Version von GRA/CMD, die zuerst nach einem Filenamen fragt, wäre zwar kein Problem. Es ist aber ebenso wenig eins, BILD/CMD mit dem RENAME-Befehl z. B. RBDEMO/CMD zu taufen. Deshalb habe ich mir das geschenkt. Mit dieser kleinen zusätzlichen Mühe können nun etliche Bildschirminhalte gleichzeitig auf der Diskette stehen. Sie heißen dann eben alle verschieden.

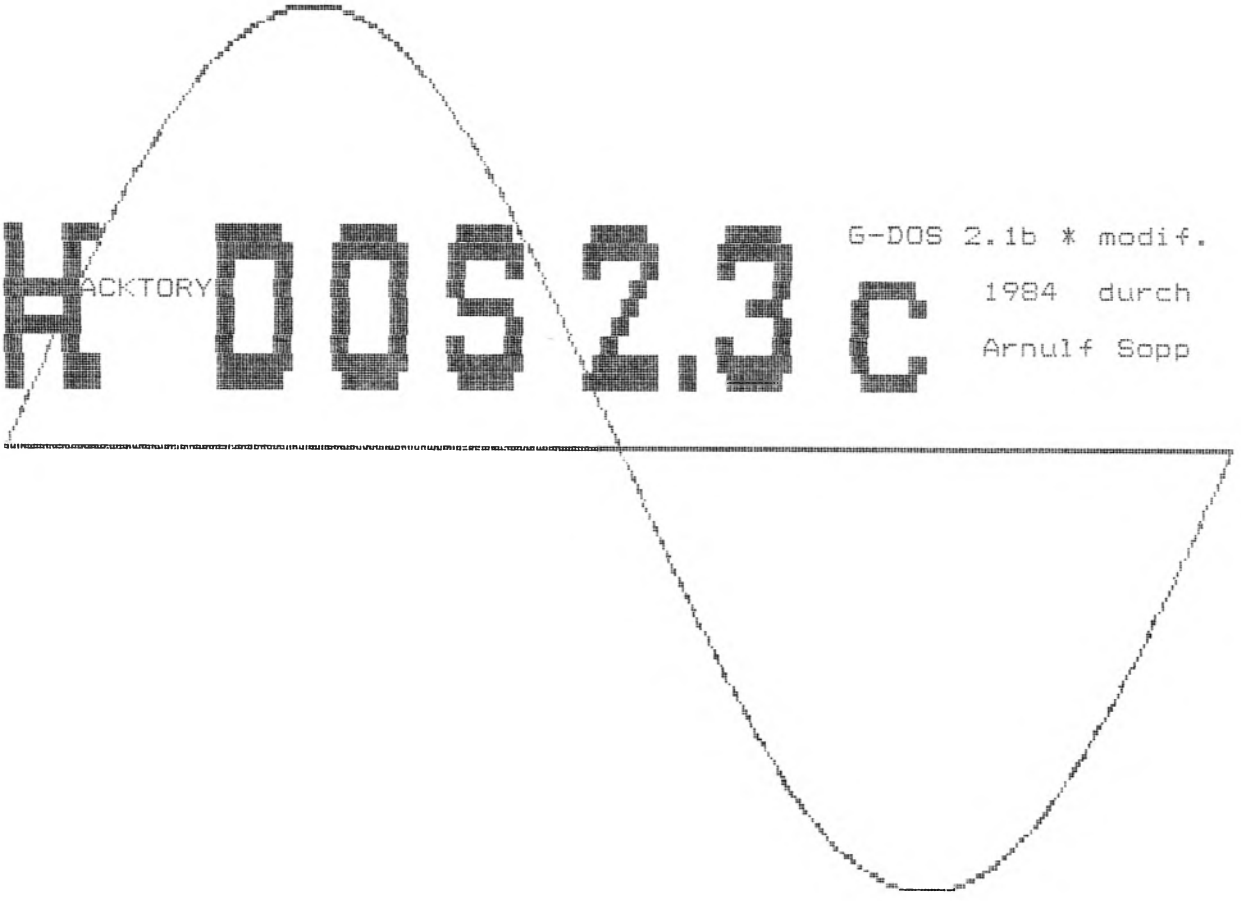
Mit diesem Programm wende ich mich an diejenigen unter euch, die sich nicht damit begnügen, von anderer Leute Arbeit zu profitieren, indem sie fertig raubkopierte Graphiken anzeigen. Wer selber welche erstellt, hat mit dieser Routine ein Werkzeug, mit dem er sie jederzeit schnell wieder auf den Bildschirm zaubern kann. Aber uns Raubkopierern ist die Arbeit natürlich ebenfalls erleichtert, denn die fertigen Graphiken können nach dem Laden genauso gesichert werden. Dann erübrigt sich in Zukunft dieses Laden nach einer äußerst krausen Syntax, wie sie diese HRG-Treiber erfordern. Überhaupt ist es überflüssig, vorher BASGR, GRAPE o. dergl. zu laden. Vor allem können die Bilder nun zusammen mit Texten usw. gesavet werden, die man nun nicht mehr zuerst umständlich dotweise in die HRG übernehmen muß, denn GRA/CMD berücksichtigt alle Zeichenarten.

Überflüssig, zu erwähnen, daß GRA/CMD Bestandteil von H-DOS geworden ist. Aber dort besteht das Programm nur aus den beiden Befehlen LD A,0BCH und RST 28H. Die Bearbeitungsroutine liegt nämlich in SYS26/SYS, einem freien G-DOS-Modul. Wer meint, immer noch ohne den EG 64 MBA auskommen zu können, kann mit GRA den Bildschirm sichern. Wer den MBA aber hat, dem steht der neue Dreitastenbefehl <345> zur Verfügung, der dasselbe leistet, aber den Bildschirm nicht besudelt. So kann sein Inhalt jederzeit und ohne Änderung des erzeugenden Programms gesichert werden.

Nach dieser überschwenglichen Eigenreklame sei eine Schwäche des Programms nicht verschwiegen: Der einfache Weg über den DUMP-Befehl hat den Nachteil, daß der Pufferbereich 8000-B425h gnadenlos zugeschaufelt wird. Zwar kann man durch Änderung der entsprechenden Adressen einen möglicherweise seltener benutzten Bereich ansteuern, aber die 13 kB kostet es allemal. Mit platzsparendem Disk-I/O über einen FCB (oder wie die Dinger heißen) kenne ich mich nun mal noch nicht aus. Es kann daher sinnvoll sein, vor GRA oder <345> Daten oder Programme zu sichern, die ab 8000h residieren.

Arnulf Sopp


High - Resolution
Grafik
D E M O
E K D
E l e k t r o n i k G M B H
y = 2 * sin (x) + sin (2*x)
2
1
0
-1
-2
π
2π
HRG1B


HACKTORY
DOS 2.3 C
G-DOS 2.1b * modif.
1984 durch
Arnulf Sopp

```
7000            00100           ORG     7000H
7000 213170     00110 START     LD      HL,CMD          ;Adresse DUMP-Befehl
7003 111843     00120           LD      DE,4318H        ;DOS-Befehlspuffer
7006 D5         00130           PUSH    DE              ;brauchen wir noch
7007 012000     00140           LD      BC,DSPLAY-CMD   ;Länge des Befehls
700A EDB0       00150           LDIR                    ;dorthin übertragen
700C 110080     00160           LD      DE,8000H        ;DSPLAY dorthin
700F 012500     00170           LD      BC,ENDDSP-DSPLAY        ;Prg.-länge
7012 EDB0       00180           LDIR                    ;Programm übertragen
7014 21003C     00190           LD      HL,3C00H        ;Anfang Bildschirm
7017 0604       00200           LD      B,04H           ;BC=0040h, Länge Bildsch.
7019 EDB0       00210           LDIR                    ;ASCII u. Pixelgr. retten
701B 61         00220           LD      H,C             ;C = 00
701C 69         00230           LD      L,C             ;jetzt HL = 0000
701D 7D         00240 LOOP1     LD      A,L             ;LSB der HRG-Adresse
701E D302       00250           OUT     (2),A           ;ausgeben
7020 7C         00260           LD      A,H             ;dto. MSB
7021 D303       00270           OUT     (3),A           ;ausgeben
7023 DB04       00280           IN      A,(4)           ;was steht da?
7025 12         00290           LD      (DE),A          ;retten
7026 23         00300           INC     HL              ;nächste HRG-Adresse
7027 13         00310           INC     DE              ;nächste Pufferstelle
7028 3E30       00320           LD      A,30H           ;HL höchstens 3000h
702A BC         00330           CP      H               ;schon erreicht?
702B 20F0       00340           JR      NZ,LOOP1        ;falls nein
702D E1         00350           POP     HL              ;sonst CMD-Puffer laden
702E C30544     00360           JP      4405H           ;und Befehl ausführen
7031 44         00370 CMD       DEFM    'DUMP BILD/CMD 8000H B425H 8000H' ;Befehl
7050 0D         00380           DEFB    0DH             ;NEW LINE
7051 D301       00390 DSPLAY    OUT     (1),A           ;HRG einschalten
7053 CDC901     00400           CALL    01C9H           ;Bildschirm löschen
7056 212580     00410           LD      HL,GRABUF       ;Anfang Graphikpuffer
7059 11003C     00420           LD      DE,3C00H        ;Anfang Bildschirm
705C 010004     00430           LD      BC,0400H        ;Länge norm. Bildsch.
705F EDB0       00440           LDIR                    ;ASCII u. Pixelgraphik
7061 51         00450           LD      D,C             ;C = 00
7062 59         00460           LD      E,C             ;jetzt DE = 0000
7063 7B         00470 LOOP2     LD      A,E             ;LSB der HRG-Adresse
7064 D302       00480           OUT     (2),A           ;ausgeben
7066 7A         00490           LD      A,D             ;dto. MSB
7067 D303       00500           OUT     (3),A           ;ausgeben
7069 7E         00510           LD      A,(HL)          ;Byte im Puffer
706A D305       00520           OUT     (5),A           ;anzeigen
706C 23         00530           INC     HL              ;nächste Pufferstelle
706D 13         00540           INC     DE              ;nächste HRG-Adresse
706E 3E30       00550           LD      A,30H           ;höchste HRG-Adresse
7070 BA         00560           CP      D               ;schon erreicht?
7071 20F0       00570           JR      NZ,LOOP2        ;falls noch nicht
7073 C34900     00580           JP      0049H           ;sonst Eingabe abwarten
7076            00590 ENDDSP    EQU     $               ;zum Errechn. v. GRABUF
8025            00600 GRABUF    EQU     8000H+ENDDSP-DSPLAY     ;stimmt genau!
7000            00610           END     START
00000 mal gepennt
33404  Zeichen verfügbar


CMD     7031 00370   00110 00140
DSPLAY  7051 00390   00140 00170 00600
ENDDSP  7076 00590   00170 00600
GRABUF  8025 00600   00410
LOOP1   701D 00240   00340
LOOP2   7063 00470   00570
START   7000 00110   00610
```

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

Hardwarebeschreibung der HRG1B

Die High Resolution Karte HRG1B scheint sich in unserem Club immer weiter zu verbreiten. Deshalb soll an dieser Stelle eine Beschreibung der Hardware erfolgen.
Das mitgelieferte Manual ist hinsichtlich des Einbaus eigentlich sehr ausführlich, stellt die Ansteuerung der Grafik aber etwas unübersichtlich dar. Weiterhin läßt der schlampig gezeichnete Schaltplan hinsichtlich der Lesbarkeit sehr zu wünschen übrig. Darum habe ich einen neuen, wie ich meine übersichtlicheren Schaltplan gezeichnet, anhand dem nun die Funktion der Schaltung und ihre Ansteuerung grundlegend erläutert werden sollen:

Ansteuerung der Karte:

Sechs Ports werden für die Ansteuerung benötigt. Es sind diese Port 0-5. Gibt man auf Port 0 einen beliebigen Wert aus, so wird der Grafikbildschirm ausgeblendet.
Schickt man über Port 1 eine Zahl, so wird der Grafikbildschirm zum normalen Textbildschirm dazugeschaltet.

```
z.B.: OUT 0,0    Ausschalten der Grafik
      OUT 1,0    Einschalten der Grafik
```

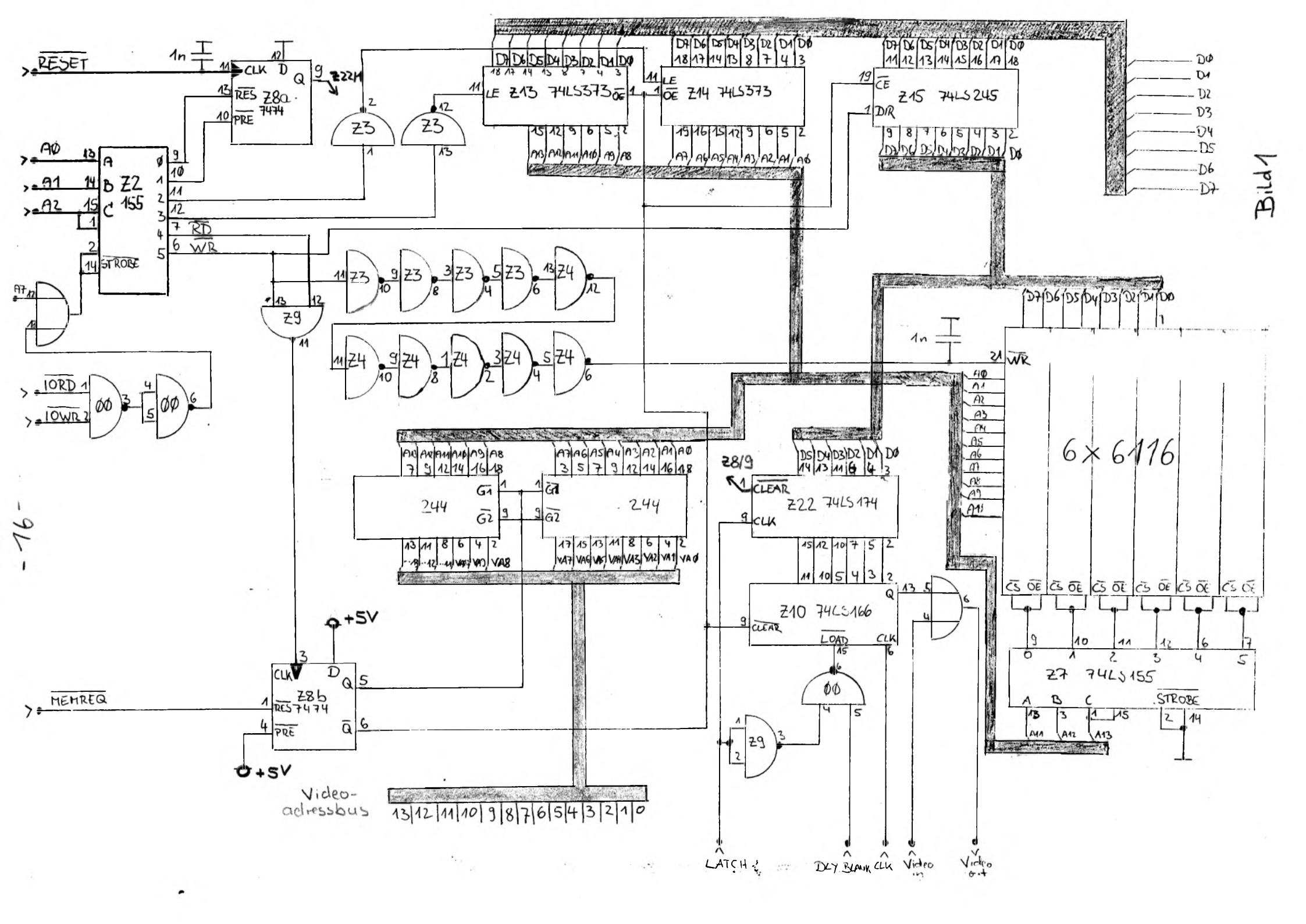
Der Speicherinhalt geht beim Ausschalten nicht verloren, es wird lediglich das Übertragen des Speicherinhalts auf die Videoleitung unterbunden.

Über Port 2 und 3 erhält die Grafikkarte die x/y Koordinate eines zu adressierenden Punktes. Dies ist relativ kompliziert, was die Umrechnung von x,y in ein von der Karte lesbares Format etwas schwierig macht. Die Bits sind wie folgt aufgeteilt:

```
-----------------------------------------------------------
            Port 3                  Port 2
Bit: 7  6  5  4  3  2  1  0  7  6  5  4  3  2  1  0
    >n.b.<> Bereich C<> Bereich B<>   Bereich A   <
 ----------------------------------------------------------
```

Bereich A:	Bit 0-5	Port 2	Position der 64 normalen Spalten
Bereich B:	Bit 6,7 0,1	Port 2+3	Position der 16 normalen Zeilen
Bereich C:	Bit 2-5	Port 3	Position der 12 Zeilen pro Character
n.b. :	Bit 6+7	Port 3	nicht benutzt

Will man also einen Punkt x,y (wobei x von 0-383 und y von 0-191) setzen, so muss man zunächst feststellen, in welcher


RESET
1n
CLK D Q
RES Z8a 7474
PRE
Z22
A0
A1
A2
B Z2 155
C
STROBE
RD
WR
A7
IORD
IOWR
Z3
Z4
Z9
LE Z13 74LS373
OE
Z14 74LS373
CE
Z15 74LS245
DIR
D0
D1
D2
D3
D4
D5
D6
D7
Bild1
6 x 6116
WR
CS OE
Z7 74LS155
A B C
STROBE
244
G1
G2
Z8/9
CLEAR
Z22 74LS174
CLK
Z10 74LS166
LOAD
Z8b 7474
+5V
MEMREQ
Video-adressbus
13 12 11 10 9 8 7 6 5 4 3 2 1 0
LATCH
DLY BLANK CLK
Video in
Video out
- 16 -

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

Spalte zu je sechs waagerechten Punkten sich der gewünschte Punkt befindet.

```
Beispiel:  x     =  0-5   6-11   12-17   18-23   24-29   usw
         Spalte =   0      1       2       3       4
```

Der Wert 'Spalte' kann als TAB-Position bezeichnet werden und wird in seiner Binärdarstellung in den Bits 0-5 von Port 2 abgelegt.

Als Nächstes muß man die Buchstabenzeile (von 0-15) errechnen, in der der gewünschte Punkt zu finden ist, wobei eine Buchstabenzeile aus 12 Grafikzeilen gebildet wird (6x12 Matrix).

```
Beispiel:  y     =  0-11   12-23   24-35   36-47   usw
         Zeile  =   0       1       2       3
```

Der so ermittelte Wert wird in seiner Binärdarstellung in den Bits 6-7 von Port 2 und Bit 0-1 von Port 3 abgelegt.

Schließlich muß man als vorletzten Schritt zur vollkommenen Adressierung eines Punktes noch die Zeile innerhalb eines Characters ermitteln und den Wert dafür in den Bits 2-5 von Port 3 ablegen.

Beispiel für alle drei Werte:

```
 a)  x=50  y=100
     Spalte = 8    Zeile = 8    Z.i.Ch. = 4
 b)  x=10  y=25
     Spalte = 1    Zeile = 2    Z.i.Ch. = 1
                                  (Z.i.Ch = Zeile in Character)
```

Schickt man nun die ermittelten Daten über Port 2 und 3 so zeigt ein Pointer auf eine Reihe mit sechs Dots. Innerhalb dieser Reihe kann man nun über Port 5 die Punkte adressieren. Überträgt man auf Port 5 z.B. die Zahl 5, so sind entsprechend dem Bitmuster von 5 folgende Punkte in der Reihe gesetzt, wobei 1=gesetzt und 0=nicht gesetzt: 000101
Nimmt man z.B. die Zahl 46, so sind die Punkte 101110 gesetzt. Man muß sich also die Stellung (=Binärwert) innerhalb der zuvor errechneten TAB-Position ermitteln, aber evtl. bereits gesetzte Punkte in dieser TAB-Position berücksichtigen, da sonst beim Schreiben des neuen Punktes die anderen Punkte geöscht werden. Dies kann durch eine OR-Verknüpfung verhindert werden.

Über Port 4 kann man eine Reihe von sechs Punkten auf ihren Status prüfen. Ergibt ?INP(4) den Wert 3, so sind die Punkte 000011 gesetzt.

Da nun die softwaremäßige Ansteuerung der Grafikkarte hoffentlich einigermaßen verständlich erklärt ist, möchte ich nun die grundsätzliche Funktionsweise der Schaltung anhand des Schaltplans (Bild 1) erläutern.

Links oben auf dem Schaltplan wird das TRS-80 RESET* Signal dem D-Flip-Flop Z8a auf seinen CLOCK-Eingang an Pin 11 zugeführt. Erfolgt ein RESET, so geht das Signal an Pin 11 zunächst auf LOW und dann wieder auf HIGH, was zur Folge hat, daß die am

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

D-Eingang des Flip-Flops liegende Information auf den Ausgang Q (Pin 9) gegeben wird. Da D (Pin 12) permanent auf Masse liegt, wird ein LOW ausgegeben. Pin 9 von Z8a ist mit Pin 1 von Z22 verbunden. Es handelt sich hier um den CLEAR*-Eingang eines 6-Bit-D-Registers mit Löschen, bei dem die an den Eingängen anliegenden Informationen mit der positiven Taktflanke am CLK-Eingang (Pin 9) intern gespeichert und auf die Ausgänge gelegt werden. Da bei einem RESET* Pin 11 auf LOW geht, gehen auch die Ausgänge auf LOW und es erscheinen keine Grafikpunkte auf dem Bildschirm, da über das Shiftregister Z10 dem regulären Videosignal nur LOWs über das OR-Gatter Z1 zugemischt werden. Zusammenfassend kann gesagt werden, daß bei einem RESET* die HRG weggeschaltet wird.

In Z2 (74155) befinden sich zwei 2-Bit-Binärdekoder, die so zusammengeschaltet sind daß sich ein 1-aus-8 Datenverteiler ergibt, der als Adressdekoder verwendet wird. Er wandelt einen Binärcode, der an den Eingängen ABC liegt um in einen dezimalen und schaltet den entsprechenden Ausgang (0-5) auf LOW. Dieser Baustein fungiert auch als Regelglied dafür, um zu entscheiden, ob die über einen Port ausgegebenen Daten überhaupt für die HRG bestimmt sind. Hier zeigt sich auch schon eine Schwachstelle der Schaltung, die normalerweise nicht auftreten dürfte:
Die HRG soll ja nur angesprochen werden, wenn **nur** auf den Ports 0-5 Daten anliegen. Der Schaltungsentwerfer hat aber nur eine sehr abgemagerte Maßnahme dafür angewendet. Doch zunächst die Problematik:
Die HRG wird jedesmal angesprochen, wenn ein Port angesprochen wird und mindestens eines der ersten 3 Bits (Bit 0-2) auf HIGH liegt. Das bedeutet, daß die HRG nicht nur bei z.B. **OUT0,0**, **OUT1,0** **OUT4,24** angesprochen wird, sondern auch bei allen **OUTn,x**, deren n als Einerziffer die 0-5 hat. Also z.B. **OUT50,0**, **OUT74,3** usw. Das schränkt die Verwendungsmöglichkeiten der an sich noch freien Ports um die Hälfte ein. Lediglich Bit 7 wird berücksichtigt; d.h., daß Ports>127 die HRG nicht mehr ansprechen.
Eine Maßnahme, das Ansprechen der HRG **nur** bei den Ports 0-5 zu erreichen ist in Bild 2 dargestellt. Hier werden die Bits 3-7 geprüft und die HRG nur selektiert, wenn sie alle LOW sind.

Liegt A0 auf LOW (Port 0 selektiert = HRG aus), so wird über das D-Fliflop Z8a das Latch Z22 ähnlich wie bei einem RESET* gesperrt: Es erfolgt keine Zumischung des HRG-Signals.

Liegt A0 auf HIGH (Port 1 selektiert = HRG ein), so geht der Ausgang Q von Z8a Pin9 auf HIGH und befähigt so das Latch Z22, die Daten zur Beimischung freizugeben.

Wird ein **OUT2,n** ausgeführt, also das LSB der Adresse eines Punktes transferiert, so wird durch Z2 der 8-Bit-D-Zwischenspeicher Z14 selektiert, der die Daten als A0-A7 auf den internen Adressbus (für die 12K) legt.

Das selbe in grün passiert bei einem **OUT3,n**, nur daß Z13 selektiert wird und die Daten als A8-A13 auf den internen Adressbus legt.

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

Bei einem **IN4,n** sollen 6 nebeneinanderliegende Grafikpunkte auf ihren Status geprüft werden.
Im Zusammenhang damit soll hier ein weiteres Problemchen der HRG angeschnitten werden: Man sieht, daß bei einem **OUT4,n** Pin 7 von Z2 auf LOW geht, der lediglich mit dem NAND Z9 verbunden ist. Dieses NAND taktet im Falle eines WR* oder RD* das Flipflop Z8b, das nun seinerseits über seinen Ausgang Q (Pin 5) ein HIGH auf die Freigabeeingänge der Leitungstreiber 74244 legt. Dies bewirkt, daß die Videoschaltung des TRS-80 **keinen** Zugriff auf die HRG hat, wenn ein Wert in die HRG geschrieben oder ausgelesen wird. Dies erzeugt zusätzliche kleine weiße Striche auf dem Bildschirm, wie man sie in ähnlicher Weise vom 'Normalbetrieb' des TRS-80 ohne HRG kennt. Für die im Normalbetrieb entstehenden Striche gibt es die 'SNOW SHOVEL', die die Striche unterdrückt (Wirkungsweise und Theorie der Entstehung solcher Striche in 80-Micro, Clubzeitung Nr ,TRS-80 Reference Manual). Hat man diese eingebaut und dann auch die HRG, so schaut man recht dumm aus der Wäsche, da die ganze Modifikation quasi für die Katz' war - Die 'normalen' Striche sind zwar immer noch weg, aber neue von der HRG sind da. Ich muß hier bekennen, daß mir selbst noch nichts geeignetes einfiel, um den Effekt auch bei der HRG zu beheben. Es gibt zwar von RB-Elektronik eine Modifikation (ca 60DM), die mir aber zu teuer ist. Anregungen zur Lösung des Problems nehme ich gerne entgegen.
Wozu nun diese Misere?
Die Videoschaltung des TRS-80 arbeitet quasi unabhängig vom Prozessor. Sie legt also ständig Adressen an die beiden Leitungstreiber 244 an, um die HRG-RAMs auszulesen. Will der Programmierer nun z.B. einen Punkt am Bildschirm setzen, so erzeugt er seinerseits eine Adresse, die er an Z13 und Z14 anlegt. Die beiden Adressen (die des Programmieres und die der Videoschaltung) kommen sich 100prozentig in die Quere: das Chaos ist perfekt.
So weit so gut; dies haben wir also durch 'Wegschalten' des Videoadressbusses in diesem Moment verhindert. Doch nach geschehenem Programmiererzugriff (**OUTm,n** ist ausgeführt) sind die Leitungstreiber noch immer gesperrt und das Schieberegister Z10 immer noch gelöscht, da das D-Flipflop Z8b die Eigenschaft hat, die Informationen zu speichern. Es muß also zurückgesetzt werden.
Nach einem OUT-Befehl folgt logischerweise ein Zugriff auf die 'normalen' RAMs bzw ROMs (Fetch Opcode). Dies bewirkt, daß das Signal MEMREQ* auf LOW geht. Da es mit dem RES*-Eingang des Flipflops verbunden ist, wird es gelöscht und Q geht wieder auf LOW: die Videoschaltung darf also wieder.

Wenn RD* LOW ist, so ist WR* HIGH. Damit ist der Richtungsselecteingang des Leitungstreibers/empfängers Z15 ebenfalls auf HIGH, was die Datenübertragung von der HRG 'nach außen' ermöglicht. Da auch der WR* Eingang der 6 RAMs bei einem RD* auf HIGH liegt, können die adressierten Daten ausgelesen und zum TRS-80 übertragen werden.

Wird ein **OUT5,n** ausgeführt, so gilt im Wesentlichen das selbe wie bei **IN4,n**. Der Unterschied besteht darin, daß Z15

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

nun so geschaltet wird, daß die HRG Daten empfangen kann. Weiterhin liegt nun der Eingang WR* der RAMs ebenfalls auf LOW, so daß Daten in sie geschrieben werden können.

Wie man sieht, ist das Signal WR* über 10 Inverter mit dem WR* Eingang der RAMs verbunden. Diese Inverter verzögern das WR* Signal, so daß erst geschrieben oder gelesen werden kann, wenn alle Daten und Adressen stabil anliegen.

Da die RAMs nur 11 Adressleitungen besitzen (A0-A10), muß noch eine zusätzliche Selectschaltung eingebaut sein, damit die 12K adressierbar sind. A11,A12 und A13 liegen auf den Eingängen von Z7, der wie Z2 als 1 aus 8 Verteiler geschalten ist. Entsprechend der an den Eingängen A,B und C liegenden Binärdaten wird ein RAM selektiert.

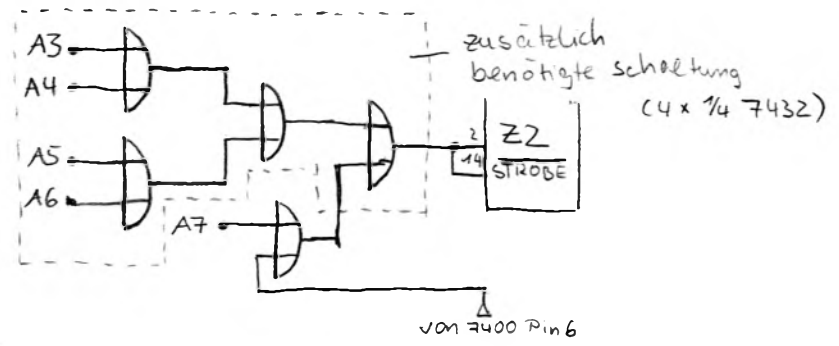
Das LATCH-Signal geht alle sechs (waagerechten) Punkte kurz auf LOW. Das bedeutet: Hat das Schieberegister sämtliche 6 Dots rausgeschoben, so erhält das Latch Z22 mit der steigenden Flanke von LATCH den Befehl, die neuen anliegenden Daten zu speichern und auf die Ausgänge zu legen.

Das CLK Signal ist der Takt der Videoschaltung; das Schieberegister arbeitet also synchron.

LOAD* von Z10 wird deshalb über LATCH und DLYBLANK selektiert, damit das Schieberegister nicht dann geladen wird, wenn der Elektronenstrahl gar nichts schreiben darf (z.B. an den Bildschirmgrenzen). LATCH muß also LOW sein und DLYBLANK HIGH, denn dann darf der Strahl schreiben.

Die Funktionsweise der Schaltung ist nun hoffentlich einigermaßen verständlich erläutert. Nähere Informationen über die Arbeitsweise der Videoschaltung kann man im TRS-80 Reference Handbook nachlesen. Zum Schluß sei noch einmal erwähnt, daß ich sehr dankbar für Vorschläge zur Lösung des 'Stricheproblems' wäre.

Das wär's dann für diesmal, Euer Bernie



Mehrere SYS-Files gleichzeitig

Einige Systemdateien von NEWDOS, G-DOS, H-DOS, TRSDOS usw. arbeiten mit anderen zusammen. So werden beispielsweise von der G-DOS/H-DOS-Funktion DDE in SYS15/SYS weitere SYS-Files aufgerufen, um Sektoren zu laden, zu schreiben, das Inhaltsverzeichnis zu sichten usw.. Da sich diese Dateien nicht überlagern dürfen, hilft sich DDE mit dem sicherlich einfachsten Ausweg: Der Ladebereich von SYS15/SYS beginnt bei 5200.

Es ist kaum anzunehmen, daß man mitten in einem gerade laufenden Programm mit DDE eine Datei modifizieren will. Deshalb ist es unproblematisch, daß SYS15/SYS den unteren Teil des Anwenderspeichers zuschaufelt. Soll jedoch eine Systemdatei im Hintergrund jederzeit, also auch mitten in einem Programm abrufbar sein, ohne anderen Systemdateien den Platz wegzunehmen, muß man sich etwas einfallen lassen.

Bei meinem H-DOS war diese Notwendigkeit gegeben. Darin spielt der EG 64 MBA eine entscheidende Rolle. Er wird von mehreren SYS-Files angesteuert. Da in einigen von ihnen der Platz sehr beschränkt war, sollte ein gemeinsames Unterprogramm für alle da sein. Ich entschied mich für SYS24/SYS, eine freie Systemdatei (die beim Genie 3 übrigens belegt ist). Für den Ladebereich bleibt nicht mehr viel Auswahl: Nur noch der DOS-Eingabepuffer 4318-4367 und der Sektorpuffer 4200-42FF stellen unterhalb 5200 noch eine nennenswerte Lücke dar.

Wenn man wegen der Mutterschaft der Vorsicht über die Porzellankiste alle 5 Sektoren von SYS24/SYS erhalten will, entsteht nun ein gravierendes Problem: In den Sektorpuffer wird alles geladen, auch Records, die mit dem Code 05 (s. Sektordumps) als das gekennzeichnet sind, was in BASIC REM heißt. Das bedeutet, daß der letztenendes aktive Record, der direkt im Sektorpuffer laufen soll, von allem Nachfolgenden verschüttet würde. Folglich darf nichts nachfolgen. Dieser Record muß im unwiderruflich letzten Sektor stehen. Alles Davorliegende muß entweder "REM" (05) heißen, also für den Programmablauf belanglos sein oder in einen anderen Speicherbereich laden, z. B. den Input Buffer.

Es gibt eine weitere Besonderheit, die zu beachten ist: Sollte von einer solchen SYS-Datei aus eine andere aufgerufen werden, so muß die Kontrolle endgültig an diese übergehen, denn ihre Sektoren werden selbstverständlich ebenfalls über 4200 ff. an ihren Bestimmungsort geladen, wobei sie unseren Spezialfile in die ewigen Jagdgründe schicken. Kurz: Während unser SYS24/SYS arbeitet, ist jegliche Disk-I/O ganz einfach verboten.

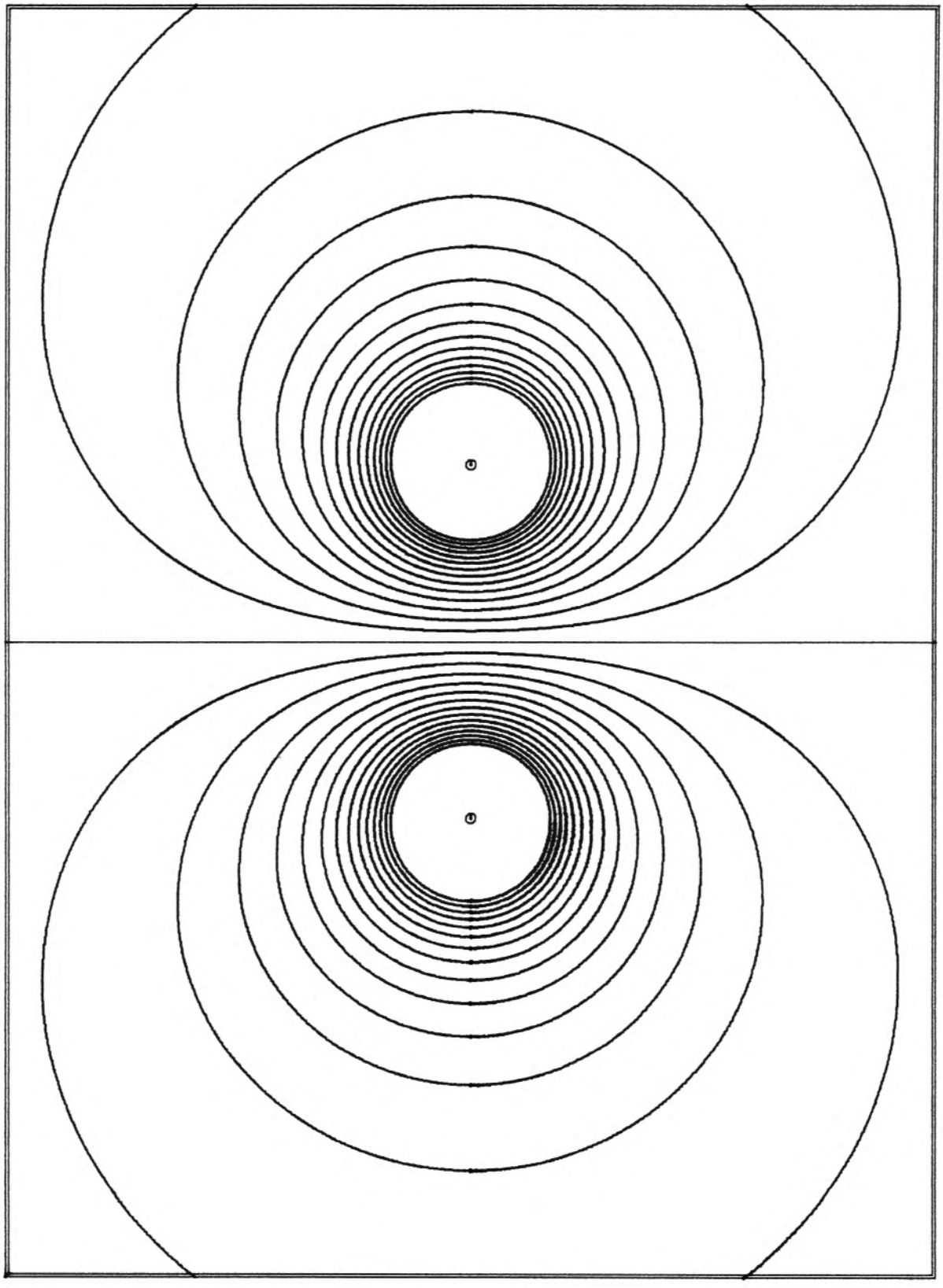
Um SYS24/SYS aufzurufen, gibt es zwei Möglichkeiten. Auf jeden Fall muß der Akku mit xA geladen sein, wobei x eine ungerade Zahl >= 3 ist. Warum, das will ich jetzt nicht erklären, weil es zu weit führen würde. Der Leser wolle sich bitte mit dem Aufruf von SYS-Dateien in der einschlägigen Literatur vertraut machen, z. B. im Clubinfo des Genie/TRS80 User Club Bremerhaven. Die eine Möglichkeit besteht darin, direkt mit dem Befehl RST 28H SYS24/SYS anzuspringen. Das hat dieselbe Wirkung wie ein GOTO in BASIC. Oder man callt eine der vielen Stellen im DOS-Kern 4000-4CFF, wo ein RST 28H steht. Das kommt einem GOSUB gleich, so daß nach Erledigung an der alten Stelle fortgefahren werden kann.

Nanu!? Angeblich ist ein RST doch ein Unterprogrammaufruf wie ein CALL! Der Schlüssel zu dieser Merkwürdigkeit liegt in der Bearbeitungsroutine von RST 28. An 4BC2 wird nämlich der Stackpointer zweimal inkrementiert, wodurch die RET-Adresse vom Stack "verschwindet" (richtiger: Ein RET würde die falsche, die darüberliegende Adresse finden).

Mein MBA-Problem, das auf diese Weise gelöst werden konnte, ist nur ein Beispiel. Es wäre interessant, in einem der kommenden Infos weitere

I)


```
10 GOTO 60:'AEQPOT2/BAS
20 '*E*V*1.2 03/31/83*
30 '*C*Aequipotential-Linien zweier Punktladungen plotten*
40 '*A*W.Gieselmann, Ahrweg 20  5142 Hueckelhoven*
50 '============================================================
60 CLEAR 500:'vor dem Compilieren loeschen
70 DEFINT F,I,J
80 DIM XA(2000),YA(2000)
90 CLS:A$=STRING$(63,"-")
100 ZH=.5:ZV=.25
110 '
120 'PARAMETER : (Default - Werte)
130    DD=.0002     :'Schrittweite fuer RZ  (.0002; .001)
140    VD=.02       :'Schrittweiten-Vorgabe, absolut (.02; .05)
150    L =10        :'Abstand der Ladungen
160    KA=.25       :'Potential  Beginn
170    KD=.25       :'           Schritt
180    KE=3.5       :'           Ende
190    IV=0         :'keine Wertetafel
200    IP=3         :'DIN A4, gedreht
210    IR=1         :'Rahmen zeichnen
220    IG=1         :'O.K.
230 '
240 CLS:PRINT"Aequipotentiallinien zweier Punktladungen"
250 PRINT"----------------------------------------":PRINT
260 PRINT"Abstand der Punktladungen          L=";L;:INPUT L
270 LR=1/L
280 '   Beschraenkung auf die Groesse der Zeichenflaeche
290     A1=18+L*ZH:'    18 ist halbe Breite
300     RG=SQR(A1*A1+13*13)+.2:'   13 ist halbe Hoehe
310 '
320 PRINT"Nummern der Linien mit konstantem Potential"
330 PRINT"    Anfang                         KA=";KA;:INPUT KA
340 PRINT"    Inkrement                      KD=";KD;:INPUT KD
350 PRINT"    Ende                           KE=";KE;:INPUT KE
360 PRINT:K=KA
370 '
380 PRINT"Wertetafel  (Nein=0, Ja=1)         IT=";IT;:INPUT IT
390 PRINT"DIN A3            =1"
400 PRINT"DIN A4            =2"
410 PRINT"DIN A4 (gedreht) =3                IP=";IP;:INPUT IP
420 PRINT"Rahmen      (Ja=1, Nein=0)         IR=";IR;:INPUT IR
430 PRINT
440 PRINT"Alles O.K.  (Ja=1, Nein=0)         IG=";IG;:INPUT IG
450 IF IG=0 THEN 240
460 CLS
470 IF IP=1 THEN PL=1
480 IF IP>1 THEN PL=.6
490 '           PL: Plot-Masstab: .6=DIN A4
500 FX=INT(PL*3600):FY=INT(PL*2600)
510 '
520 '           PLOTTEN
530 '
540 LPRINT"LO"
550 LPRINT"H":'         HOME
560 IF IR=1 THEN GOSUB 1910
570 '           Mittellinie
580 YO=0:       XO=PL*1800
590 GOSUB 1740:'        MOVE
600 YO=PL*2600
```



Niveaulinien des Feldes zweier Punktquellen verschiedener Ladung

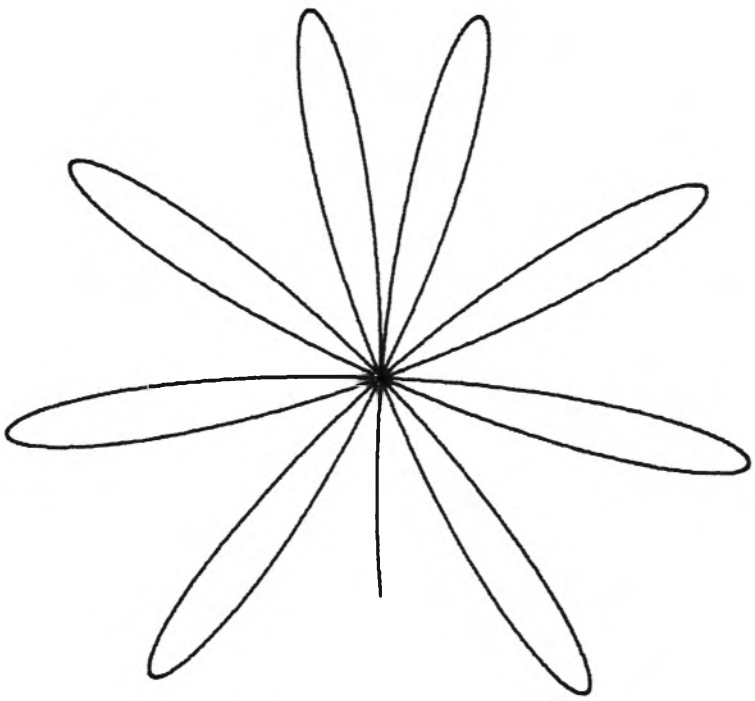
```
610 GOSUB 1650:'        DRAW
620 X0=PL*(1800-L*50)
630 Y0=PL*1300
640 GOSUB 1740:'        MOVE
650 LPRINT"N6":'        Marke
660 'LPRINT"L1":'        Unterbrochene Linie
670 'LPRINT"B20":'       Laenge: 5 mm
680 X0=PL*(1800+L*50)
690 'GOSUB 1560:'        DRAW
700 GOSUB 1740:'        MOVE
710 LPRINT"N6":'        Marke
720 'LPRINT"L0":'        Durchgehende Linie
730 LPRINT"H"
740 GOSUB 1100:'                Wertepaare
750 '
760 '            Symmetrie
770 '
780 JX=1:JY=1:
790 I=I0:GOSUB 1730
800 FOR I=I0 TO 0 STEP-1:GOSUB 1640: NEXT
810 JY=-1
820 FOR I=1 TO I0:GOSUB 1640: NEXT
830 JX=-1:JY=1
840 I=I0:GOSUB 1730
850 FOR I=I0 TO 0 STEP-1:GOSUB 1640: NEXT
860 JY=-1
870 FOR I=1 TO I0:GOSUB 1640: NEXT
880 '
890 GOSUB 1740:'              MOVE
900 K=K+KD
910 IF K<=KE THEN 740
920 '
930 IH=0:INPUT"Bildunterschrift  (Ja=1, Nein=0) ";IH
940 IF IH=0 THEN 1000
950 LPRINT"H"
960 LINEINPUT"Zeichenkopf manuell verschieben, <ENTER>";A$
970 IF IP=3 THEN LPRINT"Q1" ELSE LPRINT"Q0"
980 LPRINT"PNiveaulinien des Feldes zweier Punktquellen verschie
dener Ladung"
990 LPRINT"H"
1000 IH=3:PRINT"Programm wiederholen  = 1"
1010 PRINT"          neu starten  = 2"
1020 PRINT"          beenden      = 3";:INPUT IH
1030 IF IH=1 THEN 440
1040 IF IH=2 THEN 240
1050 END
1060 '-------------------------------------------------------------
1070 '
1080 '-----SBR: WERTEPAARE-----
1090 '
1100 GOSUB 1380:'        Schnittpunkte
1110 CLS:PRINT"K= ";K,"Rmin= ";B1,"Rmax= ";B2
1120 PRINT:PRINT"   I","   X","   Y","   R2"
1130 PRINT A$
1140 '           Anfangswert
1150 '           I: Zaehler fuer Wertepaare
1160 I=0:R2=B1
1170 X1=L*ZH-B1:Y1=0
1180 XA(I)=X1:YA(I)=Y1
1190 PRINT I,X1,Y1,R2
```

```
1200 I=I+1:D=DD
1210 '          Loop
1220 GOSUB 1460
1230 IF JF=0 THEN XA(I)=X:YA(I)=Y:I=I+1
1240 IF JF=0 AND IT=1 THEN PRINT I,X,Y,R2
1250 IF R2>B2 THEN 1290:
1260 IF R2>RG THEN IO=I-1:GOTO 1320
1270 GOTO 1220
1280 '          Endwert
1290 XA(I)=L*ZH-SGN(K)*B2: YA(I)=0
1300 PRINT I,XA(I),YA(I),B2
1310 IO=I
1320 PRINT:PRINT,IO+1;"   Wertepaare":PRINT A$
1330 '          IO: Wertepaare=IO+1
1340 RETURN
1350 '
1360 '-----SBR: SCHNITTPUNKTE MIT DER 1. ACHSE-----
1370 '
1380 A1=1/K:A1=A1*ZH*L:A2=K*K
1390 B1=A1*(K-2+SQR(A2+4))
1400 A3=ABS(A1):KO=ABS(K)
1410 B2=A3*(K+SQR(A2+4*KO))
1420 RETURN
1430 '
1440 '-----SBR: POTENTIALLINIEN MIT SCHRITTWEITENSTEUERUNG-----
1450 '
1460 JF=1:      '   1=nicht drucken,  0=drucken
1470 R1=R2/(1+K*R2*LR):S2=R2*R2
1480 X=ZH*(R1*R1-S2)*LR: S3=L*ZH-X
1490 Y=S2-S3*S3
1500 IF Y<0 THEN PRINT"*";:GOTO 1590
1510 Y=SQR(Y)
1520 '-----Schrittweite-----
1530 X2=X:Y2=Y:'  neue Werte
1540 XB=ABS(X1-X):YB=ABS(Y1-Y)
1550 IF XB<VD AND YB<VD THEN D=D+D:PRINT,D:GOTO 1570
1560 IF XB> 3*VD OR YB> 3*VD THEN R2=R2-D:D=D*ZH:PRINT,D
     :GOTO1590
1570 JF=0
1580 X1=X2:Y1=Y2:' alte Werte durch neue Werte ersetzen
1590 R2=R2+D
1600 RETURN
1610 '
1620 '-----SBR: DRAW-----
1630 '
1640 GOSUB 1850:'**
1650 IX=INT(XO):IY=INT(YO):'**
1660 IF IX<0 OR IX>FX OR IY<0 OR IY>FY THEN GOSUB 1750
          :GOTO 1690
1670 IF IP=3 THEN IW=IX:IX=IY:IY=-IW:'Drehung um +90 Grad
1680 LPRINT"D"IX","IY
1690 RETURN
1700 '
1710 '-----SBR: MOVE-----
1720 '
1730 GOSUB 1850:'**
1740 IX=INT(XO):IY=INT(YO):'**
1750 IF IX<0 THEN IX=0:'**
1760 IF IX>FX THEN IX=FX
1770 IF IY<0  THEN IY=0
```

```
1780 IF IY>FY THEN IY=FY
1790 IF IP=3 THEN IW=IX:IX=IY:IY=-IW
1800 LPRINT"M"IX","IY
1810 RETURN
1820 '
1830 '-----SBR: PLOT-KOORDINATEN-----
1840 '
1850 XO=PL*(XA(I)*100*JX+1800.5)
1860 YO=PL*(YA(I)*100*JY+1300.5)
1870 RETURN
1880 '
1890 '----SBR: RAHMEN-----
1900 '
1910 IA=0:GOSUB 1920:IA=5:LPRINT"D"IA","IA:'**
1920 XO=PL*3600-IA:YO=IA:GOSUB 1650:'**
1930 YO=PL*2600-IA:GOSUB 1650
1940 XO=IA:GOSUB 1650
1950 YO=IA:GOSUB 1650
1960 RETURN
1970 '
1980 '=====Ende==========================================================
```

II)

```
10 GOTO 60:'   ROSETTE/BAS
20 '*E*V*1.0 04/09/83*
30 '*C*Rosetten  R=SIN(N*PH)   plotten*
40 '*A*W.Gieselmann*
50 '============================================================
60 CLEAR 100:DEFINT I,O
70 CLS:PRINT"   ***  R O S E T T E  ***
80 PRINT STRING$(63,"=")
90 OX=880:OY=1130:'Offset fuer DIN A4
100 N=5:'FUENFSTRAHLIGER STERN
110 PL=.5:   DP=.1:   IW=180
120 PRINT"Anzahl der Blaetter                    N =";N;:INPUT N
130 PRINT
140 PRINT"Genauigkeit    (.1 : 3600 Punkte)   DP =";DP;:INPUT DP
150 PRINT"Winkel von 0 Grad bis                IW =";IW;:INPUT IW
160 PRINT"Groesse der Zeichnung (1 : 20 cm)   PL =";PL;:INPUT PL
170 OL=PL*1000
180 PI=3.1415926535:P1=PI/180
190 '
200 LPRINT"H":PH=0
210 GOSUB 360:'             Funktionswerte
220 GOSUB 430:'     MOVE
230 GOSUB 360
240 GOSUB 440:'     DRAW
250 PH=PH +DP
260 IF PH>IW THEN 300
270 PRINT#1000,PH;
280 IF INKEY$="" THEN 290 ELSE 300
290 GOTO 230
300 LPRINT"H"
310 END
320 '============================================================
330 '
340 '-----SUBR: Funktionswerte-----
350 '
360 A1=SIN(N*P1*PH)
370 X=A1*COS(P1*PH)
380 Y=A1*SIN(P1*PH)
390 RETURN
```

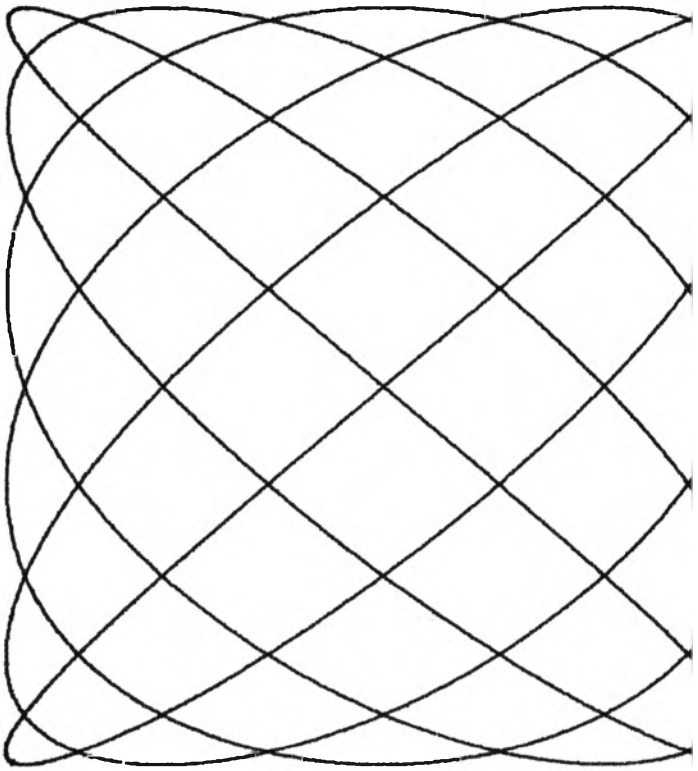


Default-Werte,
ausser N = 8.2

```
400 '
410 '------SUBR: MOVE,DRAW-----
420 '
430 A$="M":GOTO 450:'    MOVE
440 A$="D"
450 IX=INT(X*OL+OX):IY=INT(Y*OL+OY)
460 LPRINTA$IX","IY
470 RETURN
480 '
490 '============================================================
```

III)

```
10 GOTO 170:'  RADION/BAS
20 '*E*V*1.0 04/11/83*
30 '*C*Radionuklide, Zerfall, Graphen plotten*
40 '*A*W.Gieselmann
50 '============================================================
60 LPRINT"M"IX","IY:RETURN:'              MOVE
70 LPRINT"D"IX","IY:RETURN:'              DRAW
80 '
90 FOR II=I0 TO I1 STEP 2*I0:'  Logarithmische Teilung
100      IY=J0+INT(LOG(II+I0)*400):GOSUB 60
110      IX=JX:GOSUB 70
120      IF II=I1 THEN 150
130      IY=J0+INT(LOG(II+2*I0)*400):GOSUB 60
140      IX=J0:GOSUB 70
150 NEXT:RETURN
160 '------------------------------------------------------------
170 CLEAR 100:DEFINT I,J
180 J0=200:'                 Eckpunkte des Feldes
190 JX=1600
200 JY=J0+INT(LOG(300)*400)
210 '
220 CLS:PRINT"            Radioaktiver Zerfall    (fuer WX 4671)"
230 PRINT STRING$(63,"="):PRINT
240 PRINT"--->    Plotter vorbereiten,":PRINT
250 PRINT"--->    Papier  DIN A4  (Hochformat),":PRINT
260 PRINT"--->    dann  <ENTER> eingeben !"
270 LINEINPUT A$:PRINT:PRINT"O.K.
280 '
290 'FELD
300 'Rahmen
310 LPRINT"H"
320 IX=J0:IY=J0:GOSUB 60:'M
330 IX=JX:GOSUB 70:'       D
340 IY=JY:GOSUB 70
350 IX=J0:GOSUB 70
360 IY=J0:GOSUB 70
370 '
380 'TEILUNG DES FELDES
390 'senkrecht
400 FOR I=J0 TO 1400 STEP 200
410      IX=I+100:GOSUB 60
420      IY=JY:GOSUB 70
430      IF IX=1500 THEN 460
440      IX=I+200:GOSUB 60:'     M
450      IY=J0:GOSUB 70
460 NEXT
470 IX=J0:IY=J0:GOSUB 60
480 '
490 'waagerecht
500 I0=1:I1=9:GOSUB 90:'    1.Dekade
```



LISSAJØUS - FIGUR

X = CØS(2*PI*FX*T+PX)
Y = CØS(2*PI*FY*T+PY)

FX = 6
PX = 0 GRAD

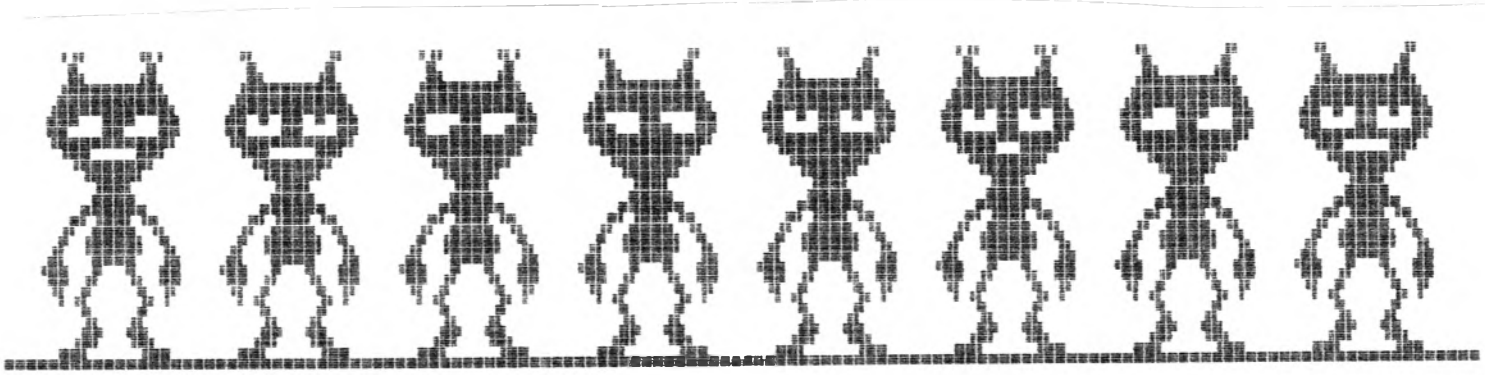
FY = 5
PY = 45 GRAD

PH = 360.013
DP = .1

TRS-80 MØD.I WX4671
27.09.84/GI

S.9

```
840 LPRINT"M"01     ","02-500: LPRINT"PPY = ";P2
850 LPRINT"M"01+350","02-500: LPRINT"PGRAD"
860 LPRINT"M"01     ","02-650: LPRINT"PPH = ";PH
870 LPRINT"M"01     ","02-700: LPRINT"PDP = ";DP
880 LPRINT"M"01     ","02-850: LPRINT"PTRS-80 MOD.I   WX4671"
890 LPRINT"M"01     ","02-900: LPRINT"P"Z$"/GI"
900 '                                 "P"ZD$"/GI"
910 RETURN
920 '
930 '-----SUBR: Datum lesen------
940 '
950 '             Datum in Z$   (tt.mm.jj)
960 I=1:GOSUB 1000:Z$=Z1$+"."
970 I=2:GOSUB 1000:Z$=Z$+Z1$+"."
980 I=0:GOSUB 1000:Z$=Z$+Z1$
990 '
1000 Z1$=STR$(PEEK(16452+I))
1010 IF LEN(Z1$)=2 THEN MID$(Z1$,1,1)="0" ELSE Z1$=RIGHT$(Z1$,2)
1020 RETURN
1030 '
1040 '=============================================================
```



Die Library vergrößern

Die "Bibliothek" (engl. library) des G-DOS, H-DOS und NEWDOS/80 steht in SYS1, verteilt auf zwei Sektoren. In G-DOS 2.1 für das Genie 3 und in H-DOS für die Modelle 1 und 2 ist sie wegen der zusätzlichen Befehle nahezu randvoll. Gleichwohl ist am Ende von SYS1 noch etwas Platz. Der Haken ist aber, daß die Library eben nicht dort, sondern einen Sektor früher zuende ist. Es folgt die Bearbeitungsroutine für den LIB-Befehl nahezu unmittelbar.

Um für weitere Befehle Platz zu schaffen, gibt es ein paar Möglichkeiten. Z. B. ist in G-/H-DOS der Befehl LIB überflüssig, weil das Befehlsverzeichnis auch mit ? ausgegeben werden kann. Dasselbe gilt für mehrere andere Befehle. Aber eine Erweiterung der Library mit einer Verkürzung erkaufen? Dabei ergäbe sich auch das Problem der Kompatibilität meines H-DOS mit G-DOS. Also änderte ich zunächst nur solche Befehlswörter ab, die es nur in H-DOS gibt: INIT heißt jetzt INI, BANK? wurde zu B?. Der neue Befehl ID, mit dem PDRIVE-Parameter von Disketten automatisch erkannt werden, hat nun Platz.

Gut und schön, es soll aber nicht der letzte Mögliche gewesen sein. Mit einer etwas komplizierteren Manipulation gab es noch einmal Platz für einen weiteren Befehl, der genau ein Zeichen lang sein darf. Nach diesem Zeichen müssen noch drei weitere Bytes vorhanden sein für einen Code, der im Register C an das aufgerufene SYS-File übergeben wird, den Requestcode, der in den Akku kommt und ein Schlußbyte, das anzeigt, ob dieser Befehl die Angabe von zusätzlichen Operanden/Parametern erlaubt. Diese insgesamt vier Bytes wurden auf folgende Weise frei:

Die oben erwähnte LIB-Bearbeitungsroutine beginnt mit dem Befehl LD HL,4F58. Damit wird HL als Zeiger mit dem Beginn der Library geladen. Dieser Befehl kann ohne weiteres an eine andere Stelle verschoben werden. Im Prinzip ginge das auch mit den folgenden Maschinencodes, aber aus Gründen, die zu erläutern hier zu weit führen würde, hätte das einen Rattenschwanz von zusätzlichen Änderungen zur Folge. Also nur drei Bytes Gewinn, aber das reicht immerhin für einen neuen Befehl (hat jemand eine Idee, was der bewirken sollte?).

Der LIB-Befehl wird nun auf das Ende von SYS1 umgeleitet. An der Stelle 51E2 wird zuerst der erwähnte Ladebefehl für den Zeiger nachgeholt und sodann an die nächste Stelle in der alten Routine gesprungen. Um dies aber zu ermöglichen, mußte der letzte Record von SYS1 bis ans Ende des letzten Sektors verlängert werden. Das Zählbyte im Record-Header wurde entsprechend geändert, der EOF-Code rutschte ganz nach hinten.

Das war für die Theoretiker, hier nun die Praxis: Auf der letzten Seite dieses Beitrags stehen drei Sektordumps, in denen die modifizierten Bytes unterstrichen sind. Im 0. Sektor wurde der Sprungvektor für den LIB-Befehl auf die neue Adresse gezapt. Im 3. Sektor sind die drei Bytes gekennzeichnet, wo zuvor der Ladebefehl stand. Sie sind nun frei. Im letzten Sektor lautet das Record-Zählbyte jetzt EA, weil noch EAh = 234d Bytes folgen. Am Ende des Records stehen zwei NOPs dort, wo zuvor das EOF war. Es folgen der HL-Ladebefehl und der Sprungbefehl in die alte Routine, schließlich das neue EOF.

Wer gerne an seinem DOS herumzapt, sollte sich unbedingt auf die unterstrichenen Bytes beschränken. Es gibt in H-DOS noch eine ganze Reihe weiterer abgeänderter Bytes, auch in den hier abgebildeten Sektoren. Bitte unbedingt ignorieren, denn diese Änderungen beziehen sich auf wieder andere Stellen im DOS, ohne die eine Änderung das System zuverlässig zum Absturz bringt. Eltern haften für ihre Kinder!

Wem erzähle ich das? Mir ist inzwischen bekannt, daß ich nicht der einzige bin, der DOS-Erweiterungen ausheckt (warum laßt ihr anderen im Info kaum von euch hören?). Wem's in der Library zu eng wird, der kann nun etwas dagegen tun. Das gilt besonders für Benutzer des Genie 3, die in dieser Beziehung ganz schön angemeiert sind.

Arnulf Sopp

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

Selektiertes Restore bei DATA-Statements

Führt man ein normales RESTORE aus, so wird der DATA-Zeiger auf den Anfang der DATA-Liste gesetzt. Will man nun den Pointer auf ein bestimmtes Element im DATA-Feld zurücksetzen, kann man sich zweier Methoden bedienen:
1. Ein Maschinenhilfsprogramm oder
2. wenn man die Elemente von vornherein kennt, auf die der Pointer zurückgesetzt werden soll ein Umsetzen des Pointers von BASIC aus.
Die zweite Methode soll hier erläutert werden. Nehmen wir an, wir hätten ein DATA-Statement wie das folgende:

```
DATA A,B,C,D,E,F
```

Wenn wir ein RESTORE auf das D durchführen wollen, so retten wir einfach den Pointer in eine Variable, bevor wir das D das erste Mal lesen. Hier ein Programm, das demonstriert, was zu tun ist:

```
20 DATA A,B,C,D,E,F
100 CLS:PRINT"GRUPPE 1";TAB(20):FORX=1TO3:READ A$:PRINTA$:NEXT
101 D1=PEEK(&H40FF):D2=PEEK(4100):REM Adr d. nächst. DATAel.
110 PRINT:PRINT"GRUPPE 2";TAB(20):FORX=1TO3:READA$:PRINTA$:NEXT
111 POKE&H40FF,D1:POKE&H4100,D2
120 PRINT"GRUPPE 2 restored";TAB(20)
122 FORX=1TO3:READA$:PRINTA$:NEXT
```

Die Adresse des nächsten zu lesenden DATA-Elements steht in den Speicherstellen 40FFH und 4100H. Wir müssen diese Adresse also vor dem ersten Lesen retten und dann, wenn der RESTORE erfolgen soll, wieder in 40FFH und 4100H poken.

BASIC Overlays

Austauschen von Variablen zwischen verschiedenen Programmen

Jedesmal, wenn ein RUN oder LOAD Kommando ausgeführt wird, werden sämtliche Variablen gelöscht. Aber oft möchte man Variablen einem anderen, folgenden Programm übergeben, ohne daß man diese zuerst auf Diskette sichert und vom anderen Programm wieder einlesen läßt. Vor allem bei Programmpaketen wird so eine Übergabe von Variablen benötigt. So könnte das eine Programm Daten vom Keyboard aufnehmen, das zweite sie auf irgendeine Art verarbeiten und das dritte einen Ausdruck vornehmen. Bei einer großen Datenmenge wäre es nun sehr zeitaufwendig, die Daten in ein Diskfile zu schreiben und anschließend wieder zu lesen.
In besseren, auf den Businessbereich abgestimmten Rechnern hat man die Möglichkeit sogenannte COMMON-Variablen zu definieren, die bei einem RUN oder LOAD nicht zerstört werden. Bei den WANG-Rechnern kann man das z.B. so vornehmen:

```
10 COM A$,D$(30),F,E(50)
20 DIM C(100,1),R(30)
```

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

An dieser Stelle sollen einige Programmiertricks in BASIC beschrieben werden, die größtenteils aus dem Buch von Lewis Rosenfelder 'BASIC FASTER & BETTER' entnommen sind und einge- deutscht auch Clubmitgliedern mit bescheideneren Englisch- kenntnissen nähergebracht werden sollen. Aber genug der Vor- reden - Los geht's :

1. FUNKTIONEN DEFINIEREN

Im BASIC-Manual werden Function Calls so gut wie gar nicht beschrieben. So mancher wird sich fragen, was man mit der geheimnisvollen Anweisung DEFFN alles anfangen kann und läßt sie aufgrund der ungenügenden Aufklärung links liegen. Tatsächlich aber stellt diese Anweisung eine wertvolle Erweiterung dar.

Eine Anwendung, die wohl bekannt sein dürfte, ist folgende:

```
10 DEFFNA(B,C,D)=B+C+D
20 INPUT V,W,X
30 ?FNA(V,W,X):GOTO 20
```

In Zeile 10 wird eine Funktion mit dem Namen 'A' definiert, die drei Zahlen aufaddiert. In Zeile 20 gibt man drei Zahlen ein und in Zeile 30 wird die Funktion mit den Variablen V,W,X aufgerufen. Dies ist nun ein großer Vorteil der Funktionen. Wer sich schon einmal ein bißchen mit PASCAL befasst hat, wird in BASIC die Möglichkeit lokaler Variablen vermissen. Lokale Variablen werden, wenn sie als solche definiert sind, nur in der gerade aufgerufenen Unterroutine geändert. Das bedeutet: Hat man im Hauptprogramm einer Variable X den Wert 10 zugewiesen und operiert nun in einem Unterprogramm mit einer lokalen Variablen X, so wird der Wert von X im Hauptprogramm nicht verändert, obwohl man in der Unterroutine z.B. X=20 ausführt. Auf unser Beispiel übertragen bedeutet das folgendes: Haben wir in BASIC die Variablen B,C und D mit irgendeiner Zahl vorbelegt, so werden nicht die Werte dieser Variablen in der Funktion addiert, sondern es geschieht folgendes: Der lokalen Variablen B wird der Wert der Variablen V aus dem Input zugewiesen, der lokalen Variablen C der Wert der Variablen W usw... Die Inhalte von B,C und D (wie wir sie anfangs definierten) bleiben unberührt !

Weitere Möglichkeiten der 'function calls' sind:

1. Man kann in Funktionen eine andere aufrufen, was z.B. so aussieht:

```
    DEFFNA(B)=B*2
    DEFFNB(C)=C+FNA(C)
```

Das bedeutet, daß man 'function calls' schachteln kann.

2. 'function calls' können ein oder mehrere Maschinenunterroutinen aufrufen

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

Ermittlung des Restes einer Division

Will man den Rest einer Division ermitteln so kann man folgende Funktion definieren:

```
10 DEFFNRE(A1,A2)=A1-INT(A1/A2)*A2
```

Setzt man das Programm fort und will nach der Eingabe von X und Y Den Rest der Division X/Y ermitteln...

```
20 INPUT X,Y
30 PRINT"Der Rest beträgt";FNRE(X,Y)
```

Will man eine Zahl (z.B. den Maschinenprogrammanfang) in zwei Speicherzellen poken, so kann man das einfach durch...

```
10 DEFFNRE(X)=X-INT(X/256)*X
15 A=16524:Z=20456
20 POKEA,FNRE(Z):POKEA+1,Z/256
```

Der mitunter größte Vorteil dieser function calls ist deren Schnelle. Bei den meisten, vor allem komplexeren, ist ein Geschwindigkeitszuwachs zu verzeichnen. Doch nun weitere Funktionen:

IF THEN Logik in Funktionen

Nehmen wir an, wir haben folgendes Programmierproblem:

```
10 IF A>=100 AND A<=300 THEN B=1 ELSE IF A>=301 AND A<=800 THEN
   B=2 ELSE IF A>=801 THEN B=3 ELSE B=0
```

Diese Methode beschert einem viel Tipparbeit und ist relativ langsam. Definieren wir Zeile 10 als Funktion so sieht das so aus:

```
10 DEFFNC(A)=-(A>=100)*-((A>=100)+(A>=301)+(A>=801))
20 INPUTA:B=FNC(B)
```

Natürlich, das Definieren dieser Funktion erfordert einiges an Denkarbeit, macht sich aber durch die höhere Geschwindigkeit bezahlt (vor allem, wenn die Sache noch komplexer wird).
Für Leute, die mit dieser Funktion gar nichts anfangen können, eine kleine Erläuterung: Das Argument (A>=100) liefert -1, wenn A>=100 und 0, wenn A<100. Ist A<100 so ist B=0, da die durch (A>=100) erzeugte 0 mit den Argumenten danach multipliziert wird, was bekanntlich 0 ergibt. Ist A>100, so ergibt sich -1; mit dem Minuszeichen vor der Klammer 1, was dann mit den anderen Argumenten multipliziert wird. Ich glaube, man kann nun selbst nachvollziehen, was in der längeren Klammer noch alles passiert.

Bernd Niedermeier Hirschbergweg 9 8011 Heimstetten ☎ <089> 903 57 31

PEEK_und_POKE_über_32767

Versucht man POKE32868,0 so ergibt das einen OVERFLOW. Man muß bei Zahlen über 32767 die Zahl 65536 abziehen, damit die Sache läuft. Man müßte nun, wenn man verschiedene Adressen benutzt, jedesmal abfragen, ob die Adresse über 32767 ist und gegebenenfalls 65536 abziehen. Dies ist umständlich und kostet Zeit. Schneller geht's mit folgender Funktion:

```
10 DEFFNC(A)=A+(A>32767)*65536
```

Hat man nun in die Adresse AD eine Zahl B zu POKEn, so führt man einfach POKE FNC(AD),B aus. Es kann nichts schiefgehen.
?FN(16000) liefert 16000
?FN(32768) liefert -32768

Die Funktion arbeitet folgendermaßen: Wenn die Adresse grösser als 32767 ist, liefert (A>32767) den Wert -1. Dieser wird mit 65536 multipliziert, ergibt -65536 und wird dadurch von der Adresse abgezogen. Ist die Adresse kleiner oder gleich 32767, so ergibt das Argument nach dem Pluszeichen 0 und die Adresse behält den gleichen Wert.

Um Zahlen im POKE-Format zurückzukonvertieren kann man folgende Funktion definieren:

```
10 DEFFND(A)=A-(A<0)*65536
```

?FND(-1) ergibt 65535
?FND(32000) ergibt 32000

Wollen wir nun Adressen addieren, die wir im POKE-Format haben, so stoßen wir auf Probleme. Z.B. wenn wir zu -32768 die Zahl -1 addieren wollen, erhielten wir normalerweise -32769, was einen OVERFLOW verursacht. Um das richtige Ergebnis 32767 zu erhalten und um grundsätzlich Fehlern dieser Art aus dem Weg zu gehen addieren wir nach Definieren der beiden Funktionen von oben :

```
10 DEFFNC(A)=A+(A>32767)*65536
20 DEFFND(A)=A-(A<0)*65536
30 A1=-32768:A2=-1
40 B=FNA( FND(A1) + FND(A2) )
```

Hex — wozu?

Beim Programmieren in Assembler ist es üblich, nur Hexzahlen zu verwenden. Seit der Grundschule sind wir es aber gewohnt, dezimal zu rechnen. Die Anwendung von sedezimalen Zahlen ist keineswegs eine arrogante Marotte der Maschinensprache-Maniacs, mit der sie sich von den BASIC-Experten unterscheiden wollen. Vielmehr bedeutet sie trotz der Umstellung auf ein ungewohntes Zahlensystem eine echte Erleichterung, auf die ich im folgenden eingehen will.

Zuvor aber ein Wort zur Sprachhygiene: "Hexadezimal" oder "sedezimal" - das ist eigentlich keine Frage. Im Grunde ist "sedezimal" korrekt (lat. sedecim = sechzehn). "Hexadezimal" oder gar die Verballhornung "hex" hat sich im Computerjargon jedoch schon so eingebürgert, daß nur noch Haarspalter Anstoß daran nehmen.

Unser Dezimalsystem arbeitet auf der Basis 10. Nach je 10 fortlaufenden Zahlen wird die nächste Stelle benutzt oder, wenn sie bereits existiert, um 1 erhöht: Nach der einstelligen 9 folgt die zweistellige 10, nach 19 kommt 20. Im Hexsystem, das zur Basis 16 rechnet, ist es ganz ähnlich: Die ersten 10 Ziffern lauten ebenfalls 0-9. Anschließend folgen A, B, C, D, E und F. Dabei entspricht Ah 10d und Fh ist 15d. Danach kommt die Zahl 10h, d. i. 16d. Eine zusätzliche Stelle bzw. die Erhöhung der höherwertigen Stelle um 1 wird demnach erst nach 16 Zahlen fällig.

Hexzahlen werden ziffernweise gelesen. Die Zahl FD spricht sich efde und nicht etwa deundefzig. Wichtig ist das vor allem dann, wenn in einer Hexzahl nur Ziffern von 0-9 vorkommen, da es sonst zu Verwechslungen mit Dezzahlen kommen könnte: 33h liest sich drei-drei und nicht dreiunddreißig. Zwischen 33h (drei-drei) und 33d (dreiunddreißig) besteht ein Unterschied von 12h (eins-zwei) bzw. 18d (achtzehn)!

Nun aber zum praktischen Nutzen dieser Zählweise: Assembler ist eine Sprache, die sich im Gegensatz zu BASIC kaum um den Programmierer, aber weitestgehend um die CPU kümmert. Unser Z80 denkt in Bytes; mit komplexen Aufgaben, wie es etwa der BASIC-Befehl PRINT ist, kann er nichts anfan- b gen. Ein Byte besteht aus 8 Bits. Da ein solches Bit ein- oder ausgeschaltet sein kann, da also in einem übertragenen Sinne die Zustände "0" und "1" gegeben sein können, haben wir es innerhalb eines Bytes mit einem Zahlensystem zur Basis 2 zu tun, mit dem Binärsystem.

Es ist sehr umständlich, größere Zahlen binär zu schreiben. Die RAM-Adresse 402Dh (16429d) sähe dann so aus: 0100000000101101. Das kann kein Mensch lesen. Man faßt daher je 4 Bits zu einer neuen Ziffer zusammen. 4 Bits bzw. eine vierstellige Binärzahl kann höchstens den Wert 15d erreichen. Das ist Fh. Und da sind wir schon im Hexsystem.

Daher kann in Hex jedes Byte mit zwei Ziffern dargestellt werden. Jede Ziffer repräsentiert ein "Nibble", ein Halbbyte. Deshalb hat man bei der Hexdarstellung gleich einen Überblick über jedes einzelne Nibble. Mit etwas Übung überschaut man sogar jedes Bit beim Lesen einer Hexzahl. Schließlich gibt es ja nur 16 verschiedene Ziffern, deren Bitkonfiguration man schnell intus hat.

Und hier liegt der entscheidende Vorteil von Hex gegenüber Dez. Man weiß sofort aufs Bit genau, was man der CPU oder dem RAM antut, wenn man in hex schreibt. Ein alltägliches (?) Beispiel soll dies verdeutlichen:

Nehmen wir an, daß irgendeine kleine Utility benutzt werden soll, um z. B. die Tastatur zu entprellen. Sie startet an der Speicherstelle F123h (61731d). Um sie in Gang zu kriegen, muß ihre Ansprungsadresse in den Tastatur-DCB an der Stelle 4016h (16406d) geschrieben werden. Mit Dezimalzahlen sieht das so aus:

```
POKE16407,61731/256:POKE16406,61731AND255
```

Um Himmels willen! Und so geht es in hex:

```
POKE&H4017,&HF1:POKE&H4016,&H23
```

Gerechnet wird dabei überhaupt nicht mehr. Das erledigt das Disk-BASIC bzw. der benutzte Assembler, falls man gerade in Maschinensprache programmiert.

Dem ist nichts hinzuzufügen, denke ich.

Arnulf Sopp

Erweiterte Library für NEWDOS/80

Hallo Clubfreunde.

Heute kann ich mit einem (hoffentlich) interessantem Artikel meine Aufwartung machen.
Ich habe mich in den letzten Wochen vor meinem Modell I auf den Hosenboden gesetzt und mir das NEWDOS/80 etwas genauer angeschaut. Und tatsächlich habe ich eine Möglichkeit gefunden, meine Pläne zu verwirklichen - nämlich selbstdefinierte Befehle in die NEWDOS-Library mit einzubauen.
Sie können genauso wie die "alten" Befehle (APPEND, COPY, LIST, LCDVR ...) verwendet und mit dem Befehl LIB angezeigt werden.

Die neue Version bleibt 100 % aufwärtskompatibel zur Version 2.0 - alle Programme bleiben unverändert ablauffähig, der bisherige Library-Befehlssatz wird in keiner Weise beeinflußt.

Hier nun die neuen Befehle.

CALL <,adr>	Call stellt einen (Maschinen-)Unterprogrammaufruf dar. Ein Unterprogramm, das im RAM steht, wird aufgerufen (evtl. nach LOAD). Wenn ein RET angetroffen wird, Rücksprung ins DOS. Fehlermeldung, wenn adr nicht angegeben.
CLEAN <,lw>	CLEAN ist das selbe für die Diskette, was CLEAR für den Hauptspeicher. Wenn lw angegeben ist, löscht CLEAN alle unbenutzten Directory-Einträge und unbenutzte Granules auf Drive lw. Sonst wird das Laufwerk aus SYSTEM-option AN (DIR) genommen.
DIRSORT <,lw> <,mod>	Das Directory aus Laufwerk lw wird gelesen und alphabetisch sortiert zurückgeschrieben. Wenn mod mit angegeben, wird zuerst nach Extents (/CMD, /BAS ...) sortiert. Ist lw nicht angegeben wird das Laufwerk aus SYSTEM-Option AN gen.
REPORT <,ON> <,OFF>	Funktion im Prinzip wie DUAL beim TRSDOS. Nach REPORT ON gehen alle Video-Ausgaben sowohl zum Video als auch zum Drucker. Wird mit REPORT OFF wieder abgeschaltet.
REQUEST <,str>	Ist als Schutzfunktion gedacht. Nach REQUEST erscheint auf dem Schirm die Meldung CPU STOPPED AT 00:00:00 es wird gewartet, bis der Benutzer den String <str> eingibt, erst dann kann weitergearbeitet werden. So lange bleibt das System (auch DEBUG) gesperrt. Wird <str> nicht angegeben, wird nur auf <ENTER> gewartet.

RESET	Rücksetzen aller Drucker-Parameter (muß angepasst werden)
UNKILL <,fn> <,:lw>	UNKILL ist das Gegenstück zum Befehl KILL. Es bringt das File <fn> vom Laufwerk <lw> wieder zurück. Dabei werden HIT und GAT vollständig restauriert. Fehler, wenn <fn> nicht gelöscht, nicht vorhanden oder bereits überschrieben. Wenn <lw> nicht angegeben ist, wird das Laufwerk aus dem SYSTEM-Parameter AN (DIR) genommen. Wenn <fn> nicht angegeben ist, werden der Diskettenname und alle gelöschten Files angezeigt.

Alle Befehle werden mit Orginal NEWDOS/80-Overlays geladen und ausgeführt. Sie verändern also den Speicher oberhalb 5200H nicht. Dadurch ist gewährleistet, daß alle neuen Befehle auch z.B. vom BASIC aus mit CMD"doscmd" oder aus Assemblerprogrammen mit CALL 4419H aufgerufen werden können.

Leider ist die Installierung der neuen Befehle nicht ganz so einfach, da dazu die Overlays SYS22/SYS, SYS24/SYS, SYS25/SYS und SYS27/SYS benötigt werden. Systemfiles müssen aber im Directory an einer festgelegten Stelle stehen um vom System gefunden zu werden. Ich muß daher bei Versionen, die diese Overlays nicht haben, diese von Hand anlegen.
Deshalb schickt mir bitte wenn Ihr Interesse an der neuen Library habt, eine Diskette mit Eurem NEWDOS/80 Version 2.X, auf der keine Anwenderprogramme sind. Ihr erhaltet dann umgehend die erweiterte Version zurück.

Wenn Ihr Anregungen für mich habt, welche Befehle noch nützlich wären, schreibt mir bitte. Allerdings sollten es Befehle sein, die für die Masse aller Mitglieder verwendbar sind.

Auf Eure Mitarbeit freue ich mich und verbleibe mit den besten Grüßen

bis zur nächsten "ausgefallenen" Idee

Meine Adresse:
Bernd A. Ruf
Unterflossing 26
8261 Polling 2

Fragen, Antworten und Tips

--> Wer kann erklären, wie der Zeichensatz des ITOH 8510 A und des NEC 8023 B -C im Eprom aufgebaut ist ? Antworten bitte an die Clubleitung

Ich möchte im Laufe des nächsten Jahres ein Grafik-Sonderheft herausbringen. Dazu suche ich Programme, Tips, Lösungen, Vorschläge, Verbesserungen, . . . kurzum alles, was mit Grafik für unsere Computer zu tun hat. Grafik-Freaks bitte melden ! ! !
Peter Spieß

Wer kennt die Grafikkarte der Firma Ingeborg Blank, 8012 Ottobrunn und kann mir ein paar Tips geben ?
Peter Spieß

Paul-Jürgen Schmitz hat ein Problem und bittet um Rat. Bei Disk-BASIC werden die Befehle "CHR$(xx)", besonders mit der Speed-Up falsch gelistet. Es erscheint RIGHT$(x), LEFT$(x), MID$(x) und alle paar Versuche auch mal CHR$(x). Wer weiß Abhilfe ?

Wolfgang Frey fragt, ob jemand im Club die CP/M-Version 2.2c hat und ihm bei Problemen helfen könnte. Tel.: 040/6958854

***** Gesucht wird:
1. MC-Heft 8/83
2. Beschreibung zu PROTEX80, wenn mögl. in deutsch
3. Programm CONVERT/BAS von K. Trappschuh
4. irgend ein einfaches Rechnungsprogramm
5. Programm DISKDAT/BAS

Wie kann man die BREAK-Taste in BASIC ansprechen ?

Gesucht wird eine Möglichkeit, eine Eingabe, die über die Tastatur getätigt wird, auf dem Monitor zu unterdrücken.

Wer helfen kann, wendet sich bitte an Heinrich Thönnißen. *0421/647762*

*** Hans-Otto Langguth hat folgende Frage:
In den vergangenen Monaten kursiert das Gerücht, daß man auf Kosten des Printerbuffers bei den ITOH's softwaremäßig einen eigenen Zeichensatz laden kann. Wer weiß darüber etwas Genaueres ?

*** Werner Grajewski kümmert sich zur Zeit um "Basicode-2" und "Supertape" aus der c't. Mit Hilfe von Supertape lassen sich Programme von anderen Rechnern übernehmen.

Liegt ein Supertape für den TRS8Ø Model 1 oder für den Genie I, II vor? Ist Supertape bei den Clubkameraden unserer Colour Genie Ecke bekannt? Soweit Supertape für die übrigen Computer in unserem Club noch nicht existiert, wäre es interessant zu wissen, ob ein Clubmitglied in der Lage ist, solch ein Programm zu erstellen.

*** Walter Schäfer schreibt: Ich lese immer wieder Anfragen, wie z.B. CHR$(10) erzeugt werden kann. Mit solchen "ungeliebten" ASCII-Werten habe ich mir beim Gemini 10 X damit beholfen, daß ich die Blockgrafikversion dieser Steuerzeichen, d.h. den um 128 höheren Wert (z.B. LF = CHR$(138) usw.) verwende. Damit hat es meistens geklappt.

Für Neueinsteiger mit dem Gemini ist sicher die Gemini-Programmierfibel (für ca. 25,- DM bei Trommeschläger zu erhalten) ganz interessant; für alte Hasen allerdings schade um's Geld. Sollte sich ein Mitglied dafür interessieren, bin ich gegen Portoersatz gerne bereit, sie ihm zur Ansicht mal zuzusenden.

*** Bei Conrad Elektronik, Klaus-Conrad-Str. 1, 8452 Hirschau gibt es Colour Genie Gehäuse mit Tastatur und Netzteil für 79,-DM.

*** *Paul-Jürgen Schmitz hat folgende Frage: Für die Genie-Computer mit dem EG 64 MBA gibt es eine speziell angepasste Version des CP/M. Fall jemand dieses Betriebssystem hat, soll er sich bitte melden.*

*** *Klaus-Jürgen Mühlenbein hat auch noch ein Problem: Warum macht GENIETEXT 3 die rechte Randbegrenzung nicht mehr mit, sobald man ein sog. "umgewandeltes" Zeichen (z.B. "#a") verwendet?*

*** *Heinz-Gerd Küster hat ein spezielles Problem: Ich habe eine Diskettenstation nach dem Trommeschläger-System, d.h. ich brauche für einige selbstbootende Disks einen Singler. Trommeschläger kann keinen mehr liefern, auch diverse andere Händler nicht. Vielleicht hat irgendjemand im Club Schaltungsunterlagen davon, dann könnte ich mir das Ding nachbauen. Noch besser wäre, wenn jemand eins verkaufen würde, aber davon kann ich wohl nur träumen. Außerdem suche ich Scripsit in der Cassettenversion und das "Cassette Portfolio System".*

Die Rubrik "Fragen, Antworten und Tips" wurde aus den letzten Clubinfo's des Bremerhavener Clubs zusammengestellt. Fehlt die Angabe der Adresse bzw. Telefonnummer, wendet Euch bitte an Peter Spieß.

Flohmarkt:

Verkaufe:

TRS-80 Modell 1 m. Speed-Up, Expansions-Interface und Monitor

Andreas Julius

Verkaufe:

Matrixdrucker ITOH 8510 gegen Gebot

Josef Ressel

Suche:

Programme für HRG 1B sowie brauchbare Datenbank für TRS-80

Bernd Niedermeier

Im Club ausleihbare Geräte und Hilfsmittel:

- TRS-80 Mod. 1 L2/16K mit Monitor + Cas.Rec.
- Tandy Printer-Plotter
- CE-Disk Einstelldiskette für 5' Laufwerke
- Reinigungsdiskette (Feuchtreinigung)
- Disklocher
- Werkzeug zum Anbringen von Verstärkungsringen
- SCRIPSIT-Lehrgang deutsch
- Verschiedene Programme und Handbücher

Internes:

Liebe Clubfreunde,

zu Beginn möchte ich gleich zu einem Thema kommen, welches ich schon vor längerer Zeit einmal angesprochen hatte. Ich möchte langfristig mehr Anwender der neuen Rechnertypen (Modell 3, Modell 4, Genie IIS, Genie IIIS usw.) in den Club bekommen, um mit der Marktentwicklung schrittzuhalten. Als Anreiz biete ich jedem Besitzer eines solchen Systems eine Beitragsbefreiung von 6 Monaten. Vielleicht könnte der Eine oder Andere von Euch mal etwas Werbung für unseren Club in seinem Bekanntenkreis machen.

Die früher einmal bestehende Aufnahmesperre habe ich inzwischen wieder aufgehoben. Der Club ist also wieder offen - für jedes neue Mitglied.

Der Artikel von Prof. Dr. Gerstlauer (AMMS-Mitglied) über den Einbau von 64K-Ram's in den TRS-80 und ein dadurch mögliches Speichervolumen von 192 K hat bei den Mitgliedern großes Interesse gefunden. Da bei mir mehrere Anfragen zu diesem Thema eingegangen sind, habe ich Anfang Dezember 1984 Herrn Gerstlauer angeschrieben und um weitere Informationen gebeten. Da mein Schreiben bis heute unbeantwortet blieb, kann man wohl davon ausgehen, daß mit weiteren Informationen seitens Herrn Gerstlauer nicht mehr zu rechnen ist.

Bedingt durch das Versenden der Clubzeitung in unverschlossenen Kuverts, kommt es manchmal vor, daß Post von Personen, die mit dem Club absolut nichts zu tun haben, in die Briefhüllen der Clubzeitungen rutscht. Solche Briefe bitte in den nächsten Briefkasten werfen - und nicht mehr an mich schicken !!!!
Ich bin übrigends gerade dabei, neue Kuverts mit Adhäsionsverschluß einzuführen.

Unsere Clubzeitung erreicht mittlerweile Assembler-Profi-Niveau. Leider kommen dabei die Gelegenheitsprogrammierer, die sich mehr BASIC-Programme wünschen, zu kurz. Es werden also dringend BASIC-Listings für den Abdruck in der Clubzeitung gesucht. Viele Mitglieder haben auch Schwierigkeiten, die Möglichkeiten der Dateibehandlung unter DISK-Basic richtig einzusetzen. Vielleicht könnte mal eines der Mitglieder hierzu so eine Art Kurzlehrgang schreiben, denn die Beschreibung im NEWDOS-Handbuch ist doch etwas unverständlich. Die Beschreibung der beiden, vom TRSDOS unterstützten, Dateiarten könnte ich anfertigen.

Viele Grüße.

Georg

P.S. Sonderangebot - nur solange Vorrat:

Drucker EPSON RX-100 (F/T) + 136/233 Zchn./Zeile
Mit automatischem Einzelblatteinzug nachrüstbar.

DM 1300.-