HARDWARE * SOFTWARE

COMPUTER-
TANDY-/ BENIE- ANWENDER
CLUB 80

ERFAHRUNGSAUSTAUSCH

TEXT DATEI

TEXT DATEI

CLUBINFO

27. AUSGABE

KONTAKTADRESSE : CLUB 80 / ALEXANDER SCHMID / ST. CAJETAN STR. 38 VII / 8000 MÜNCHEN 80
TEL.: 089 / 495326

- I N H A L T S V E R Z E I C H N I S -

Neues von der Prostata !

Unser ehrenwerter erster Vorsitzender hat zwar nicht den Löffel, aber zumindest den Vorsitz abgegeben. Trotz heftiger Gegenwehr mußte ich mich schließlich der erdrückenden Mehrheit beugen und so hat man mich auf dem diesjährigen Clubtreffen (vom 28. April - 1. Mai) zur neuen Vorstandsdrüse des Vereins ernannt. Ihr hättet es verhindern können, warum wart Ihr nicht da ?

Wenn man mir die Anwesenheitsliste nicht entführt hätte, könnte ich Euch genau sagen, wieviele Leutchen da waren, aber die Vorstellung war sehr gut besucht. Bei der Sitzung selber waren es 21, dazu kommen noch die, die nur vorher oder danach da waren und, nicht zu vergessen, die Boxenmannschaft, sprich Frauen und Freundinnen. Neben den obligatorischen Tandies und Genies waren natürlich auch wieder einige Kriegsgefangene und Exoten aus dem Atari- und MS-DOS-Lager dabei. Nachdem jede verfügbare horizontale Fläche mit Rechnern, Monitoren oder Handbüchern belegt war wurde es langsam eng, was dem Erfahrungs- und sonstigem Austausch aber keinen Abbruch tat, eher im Gegenteil. Außer der Hackerei, die nach Augenzeugenberichten teilweise bis nach 2 Uhr nachts ging, gab es selbstverständlich auch den offiziellen Teil, über den ich noch etwas mehr sagen möchte.

Im Vorstand hat sich einiges getan, die meisten Posten sind neu besetzt worden, obwohl ich nicht finde, daß die 'Alten' das so schlecht gemacht haben. Von dieser Seite nochmals Danke für die geleistete Arbeit und mögen sie ihren verdienten Ruhestand genießen. Die beiden einzigen Altgedienten sind Jens Neueder, der auch in diesem Jahr wieder als Info-Redakteur tätig werden wird und Jutta Obermann. Jutta hat sich dankenderweise wieder für die Kasse geopfert, die sie im vergangenen Jahr mit wahrer Todesverachtung vor dem Bankrott bewahrt hat. Neuer erster Vorsitzender bin also nun ich, Alexander der Schmid (nein, nicht der Große), zweiter Vorsitzender ist Horst-Dieter Schroers. Neuer Hardware-Koordinator ist Andreas Magnus, der mit seinem Genie IIIs in einem riesigen IBM-Towergehäuse und mit Festplatte für Aufsehen gesorgt hat und die Buch-Bibliothek hat Christian Menk übernommen. Für die Newdos-Programme ist jetzt Oliver Volz zuständig und last, but not least, hat Rüdiger Sörensen die CP/M-Bibliothek vereinnahmt.

Unter anderem wurde auch zum wiederholten mal über MSDOS 'ja' oder 'nein' abgestimmt und glücklicherweise (meine Meinung) wieder abgelehnt. Die Gründe waren im großen und ganzen die, daß wir schon mit unserem eigenen TRS-80, CP/M und sonstigem Z80-Kram genug zu tun haben und mit einem 'echten' MSDOS-Club sowieso nicht konkurrieren könnten. Auch wäre der Aufwand für die Verwaltung der unzähligen PD-Programme viel zu groß. Wir bleiben also ein Z80-kompatibler Club mit bevorzugter Richtung TRS-80. Rüdiger Sörensen hat daraufhin als Alternative angeboten, daß er fast alles, was in der IBM-Welt an Public Domain rumläuft auch besorgen kann. Damit war dieses Thema vom Tisch und wer also ein spezielles (PD !!!) Programm braucht, wende sich bitte an ihn. Außerdem wird niemand standrechtlich erschossen, wenn er mal einen Artikel über Atari oder IBM schreibt. Als Information ist sowas immer herzlich willkommen, wenn es auch nicht ausdrücklich unterstützt wird.

Zur Unterstützung unserer Mitglieder in der DDR wurde beschlossen, daß der Club, wenn jemand seinen alten Compi gerne abgeben möchte und diesen in Richtung Ost verschifft, entweder die Frachtkosten übernimmt, oder selber versucht, das zu organisieren. Bevor er verstaubt und schließlich einfach weggeworfen wird, ist das wohl die eindeutig bessere Lösung. Bei der Gelegenheit kann ich Euch auch gleich ein Buch über Tabellenkalkulation ans Herz legen. Da Frank & Co. verständlicherweise nicht als Bittsteller dastehen möchten, könnten Sie uns interessante Computerbücher aus der DDR besorgen, die dann hier im Club verkauft werden. Von dem Geld können wir ihnen dann Disketten und sonstiges Material schicken, die dort gar nicht oder nur zu extrem hohen Preisen zu bekommen sind. Das Buch heißt 'Tabellenkalkulation mit Personalcomputern' und hat 144 Seiten im A5-Format. Darin werden zwar speziell Programme namens KP und MULTIMAC beschrieben, aber beim Durchblättern haben die mich doch irgendwie an Multiplan o.ä. erinnert. So groß werden die Unterschiede wohl kaum sein und zum ersten Kennenlernen von Spreadsheets dürfte es gar nicht schlecht sein. Wer also was abzugeben hat oder das Buch möchte, soll sich bitte melden. Der Preis richtet sich nach dem, was Ihr geben wollt oder könnt. Neben den unverzagten TRS-Usern gibt es auch noch ein paar C64er, die sich um Gerhard Gruber geschart haben. Wer was für die hat, soll sich bitte auch melden.

Ich hoffe, daß das mit dem Info jetzt besser klappt. Wir haben diesmal mehrere Optionen, wo wir Drucken lassen können und wenn alles schiefgeht, wird doch ein eigener Kopierer angeschafft. Wenn es jetzt nicht hinhaut, dann wohl nie.

Was vielleicht auch noch wichtig ist, ist, daß jetzt endlich mit der Eintragung ins Vereinsregister ernst gemacht werden soll. Das war auch mit einer der Gründe, warum Horst-Dieter zweiter Vorsitzender geworden ist. Um den ganzen notariellen Krempel zu erledigen müssen die beiden Vorsitzenden nämlich zusammen zum Notar gehen und das wäre sonst etwas schwierig geworden.

So, Ihr könnt wieder Luft holen. Das wäre soweit erstmal alles, was ich zum Einstand zu sagen hätte. Schreibt weiter fleißig Eure Artikel und schickt mir bitte noch die letzten Fragebögen zurück, die Ihr irgendwo rumliegen habt.

Meine Anschrift steht zwar im Anhang, aber hier nochmal groß und deutlich wie der Briefkopf aussehen könnte:

An den
hochwohlgeborenen 1. Vorsitzenden
des ehrenwerten Club-80

Alexander Schmid
St. Cajetan Str. 38/VII
8000 München 80

Wenn es sein muß, würden auch die letzten drei Zeilen reichen, der Brief müßte dann rein theoretisch auch noch ankommen.

Ich kann nur hoffen, daß der Club mich überlebt und daß Ihr alle mit mehr oder weniger Begeisterung dabei bleibt. Wenn mir mein Knuddel-Device genug Zeit läßt, verspreche ich, daß ich mein Möglichstes zum Gelingen der Clubarbeit beitragen und immer als Briefkastenonkel zur Verfügung stehen werde.

Viele Grüße, frohes Schaffen, happy Hacking wünscht Euch

Alexander

Die Mitgliederversammlung mit 21
zur Zeit anwesenden Clubmitgliedern
tagt in den Räumen des Idsteiner Hofes.

Der neue CLUB 80 Vorstand:
(von links nach rechts)
Hartmut Übermann.....Kassenwart (Vertretung für Jutta)
Jens Neueder.....Redaktion
Christian Menk.....Club 80 Bücherei
Andreas Magnus.....Hardware-Koordinator
Alexander Schmid.....1. Vorsitzender
Rüdiger Sörensen.....CP/M-DisKothekar
Oliver Volz.....NEWDOS-DisKothekar
Horst-Dieter Schroers...2. Vorsitzender

==========
HAPPY BIRTHDAY !!
==========

Hallo Ihr "Clubfreunde",

in der Hoffnung, daß Ihr das Clubtreffen gut überstanden habt, und nach Überwindung einiger interner Schwierigkeiten, können wir wieder mit Volldampf in das Clubleben einsteigen.

Da es immer noch nicht sicher ist, wann dieses Clubinfo erscheint, beginne ich einfach ganz frech mit dem Monat Juni bei den Geburtstagsgrüßen. Denn es wird Euch bestimmt momentan nicht interessieren, wer in den Monaten Februar bis Mai Geburtstag hat, wenn das Info erst im Juni kommt. Den davon betroffenen Clubmitgliedern sei gesagt, daß Ihr bestimmt nächstes Jahr nicht übergangen werdet.

Wie jedes Mal, lege ich Euch ans Herz, mir zu schreiben, wenn Ihr Euch übergangen fühlt oder das Datum nicht stimmt.

Noch eine Bitte an dieser Stelle. Unterstützt Jens bei der Clubarbeit so gut es geht. Scheut Euch nicht, Beiträge an ihn zu schikken. Ihr wißt ja, nur durch das Clubinfo lebt der Club. Deswegen sollte jeder sein Quentchen dazu beitragen.

Aber nun folgen die genauen Daten.

Im Monat Juni haben Geburtstag:

01.	:	Wolfgang Beckhausen
10.	:	Claus Ruschinski
14.	:	Christian Menk
15.	:	Harald Mand
19.	:	Hans-Otto Wulf
20.	:	Eckehard Kuhn
28.	:	Matthias Homann

Im Monat Juli folgen:

07.	:	Armin Ahlers
10.	:	Frank-Michael Schober
18.	:	Hans-Günther Hartmann
19.	:	Hermann Brandl
30.	:	Bernd Retzlaff

Im Monat August gratulieren wir:

02.	:	Günter Issig
05.	:	Hans-Martin Stephan
		Ulrich Heidenreich
10.	:	Helmut Bernhardt
13.	:	Klauspeter Schmidt
15.	:	Waldemar Misioch
16.	:	Harald Braun
24.	:	Hartmut Obermann

Im Namen des Vorstandes wünsche ich allen genannten und ungenannten Clubfreunde ein fröhliches Feiern und alles Gute auch weiterhin.

Jutta

-- Termine -- Termine -- Termine --

Nächster Redaktionsschluß .....ca. 5 Wochen
nach Erscheinen dieses INFO's
Bitte denkt an kleinere INFO-Artikel (ca. 1 - 4 Seiten) für unser Clubinfo !!

-- Messen '89 --

Internationale Funkausstellung .....Berlin 25. August - 9. September 1989
PRODUKTRONIA .....München 7. - 11. November 1989

Hallo Leute,

ganz verschämt möchte ich mich als (nicht mehr ganz so) neues Clubmitglied endlich (nach jetzt über 8 Monaten) vorstellen. Zu einem ganz kleinen bißchen Entschuldigung möchte ich erwähnen, daß ich von dieser Zeit ca. ein halbes Jahr nicht zu Hause war und mir jetzt mein Genie III nach über 2 Monaten der (Wieder-) Eingewöhnung noch immer recht fremd vorkommt.

Mein Name ist André Schut, geb.am 29.3.1951 und ich wohne in Berlin. Ich kam beruflich (allerdings nur als "Anwender") vor ca. 5 Jahren zu meinem ersten Kontakt mit einem Genie III. Vor ziemlich genau 3 Jahren habe ich mir dann das gleiche Teil gebraucht zugelegt und habe zu meiner Schande aber noch immer nicht viel mehr als ein "fundiertes Halbwissen" über die Hardware und gerade mal "Kentnisse" von Basic (weia!) und Cobol. Anfänge von Pascal, die ich vor meinem "Langzeiturlaub" mal ansatzweise in Angriff genommen habe, sind total verschüttet.

Mein Interesse ist 1. wieder "'reinzukommen" und 2. mal etwas mehr von Hardware (sowie Elektronik überhaupt) mitzubekommen. Ich würde mich freuen, wenn unter den Hardwarebastlern ein Genie III-Besitzer wäre, der bereit wäre, mir etwas "auf die Sprünge" zu helfen.

Auf hoffentlich bis bald

Hallo Clubfreunde,
Ich möchte mich hier kurz als neues Mitglied vorstellen.

Ich heiße Andreas Magnus, bin auch geboren und zwar am 17.10.1965 in Gelsenkirchen wo ich auch wohne.
Von Beruf bin ich Fernmeldeelektroniker welchen ich aber z.Z. nicht ausübe da ich bei der Bundeswehr bin
Ich befasse mich schon einige Jahre mit Computern und habe einen Genie II sowie einen Genie IIIs
Ich habe immer Wissensdurst, helfe aber auch gerne bei Fragen, speziell zu Hardwareproblemen.
Auf gutes Gelingen.

Hallo -80er Fan´s,

als neues Clubmitglied möchte ich mich kurz vorstellen.

Mein Name ist Egbert Schröer, geb. am 31.05.58, demzufolge (noch) 30 Jahre alt. Verheiratet und wohnhaft in Dorsten, einer Stadt am Rande des Münsterlandes.
Beruflich nutze ich einen HP-Vectra ES/12 Model 41 und einen IBM XT.
Angefangen hat alles mit einem SHARP PC 1251 (nur für Leute mit kleinen Fingern) vor ca 8 Jahren.

Mein TRS 80/Model 1 48K mit Single Density Controler (leider) ist mit Kleinschreibung und HRG1B der Firma RB Elektronik ausgerüstet und läuft unter NEWDOS80 Vers. 2.057. Zwei 40 Spur Floppy´s, ein LW SS das andere DS (leider zur Zeit noch SS angeschloßen), vervollständigen das System.

Auf gute Zusammenarbeit
Egbert Schröer

Hallo Clubfreunde,

ich möchte mich als neues Clubmitglied kurz vorstellen.

Mein Name ist Oskar Drechsler und wohne in Bergisch Gladbach. Ich bin 42 Jahre alt, habe Holzkaufmann gelernt, bin anschließend Soldat geworden und geblieben und in der Personalverwaltung tätig.

Ich besitze einen Genie I mit 2 Laufwerken (40 Track SS/DD und 80 Track DS/DD). Den Rechner und die Laufwerke habe ich in ein Alu-Gehäuse eingebaut, mit der HRG 1b-Platine erweitert und das Netzteil gegen ein Schaltnetzteil ausgetauscht. Die Tastatur habe ich gegen eine Tastatur mit Umlauten und 22 Funktionstasten ausgetauscht. Sie liegt vor dem Rechner. Einige Funktionstasten habe ich schon mit Druckersteuerzeichen belegt, für den Rest fehlen mir noch die Ideen. Außerdem habe ich den Rechner an einen ECB-Bus angeschlossen, der zur Zeit nur mit einer Uhr bestückt ist. Ein Tongenerator ist in Vorbereitung.

Als Betriebssystem benutze ich NEWDOS 80 V2.052, habe bisher überwiegend in Basic programmiert und glaube darin ziemlich fit zu sein. Seit einem Jahr programmiere ich auch in Assembler. So habe ich für meine neue Tastatur einen Treiber geschrieben und den von Frank Debatin geschriebenen HRG-Graflix-Editor mit Maschinenroutinen ergänzt. Ich werde das Programm demnächst vorstellen.

Durch meine Clubmitgliedschaft erhoffe ich mir Kontakte zu Hardware-Freaks, weil ich die Speicherkapazität erweitern und die Taktgeschwindigkeit erhöhen möchte. Außerdem bin ich an Kontakten zu Assembler-Programmierern interessiert.

Auf gute Zusammenarbeit

Oskar

Hallo Club 80 !

Als neues "überregionales" Klubmitglied aus der DDR möchte ich mich an dieser Stelle vorstellen.

Mein Name ist Frank-Michael Schober, ich wohne in Spremberg/Lausitz, in der Nähe von Cottbus und bin 37 Jahre alt. Verheiratet bin ich mit dem Computer, ansonsten nicht. Meine Brötchen verdiene ich als Lehrmeister in der Nachrichtentechnik. Dort lasse ich auch meine Nerven.

Angefangen hat das Drama bei mir 1982 mit einem ZX 81-Bausatz. Nachdem ich dann über ZX-Spectrum zu einem ALPHATRONIC PC8 umgestiegen bin, hat es mir die große, weite CP/M-Welt angetan.

Ich bin ein leidenschaftlicher Byte-Wurschtler (Maschinencode-Fan), daher meine Vorliebe für systemnahe Programmierung.

Ich hoffe trotz der Entfernung auf eine gute Zusammenarbeit.

Mit CP/M Kompatiblem Gruß

Frank

Projekt: Erstellung/Dokumentation von Modula-Bibliotheken

Ziel : Voll dokumentierte Modula-Bibliotheken
Betriebssystem: CP/M-80
Sprache : Turbo-Modula

vorhanden:
- Turbo-Modula unter CP/M
- einige Turbo-Modula-Bibliotheken mit englischer Dokumentation
- viele Turbo-Pascal-Bibliotheken mit CHIP-Dokumentation

Anforderungen an die Teilnehmer:
- Kenntnisse von Turbo-Pascal (oder Turbo-Modula)
- Bereitschaft, Turbo-Modula zu lernen

Motivation:

Bei der Programmierung verschiedener Programme stellen sich oft ähnliche Probleme. Wenn jedes Mal "das Rad neu erfunden" werden muß, dauert die Programm-Entwicklung zu lange und wird mühselig. Deshalb werden zunehmend Bibliotheken fertiger Programmstücke angeboten, die in eigene Programme integriert werden können und Standard-Aufgaben wie die Eingabe über Menüs, die Ausgabe auf Windows, das Sortieren von Arrays oder Files usw. übernehmen.

Unter CP/M gibt es viele Programmiersprachen, aber nur wenige stark verbreitete. Die bekannteste dürfte Turbo-Pascal sein, für die viele Programme und einige Bibliotheken vorliegen, letztere vor allem aus der Reihe "CHIP-Special".

Leider bietet Turbo-Pascal keine günstigen Voraussetzungen für die Arbeit mit Bibliotheken: Wenn eine Bibliothek ein File ist, muß in das Programm die ganze Bibliothek übernommen werden (per INCLUDE), auch wenn nur ein Teil benutzt wird. Oder der Programmierer muß sich mühselig die benötigten Teile herauskopieren, wobei er evtl. Hilfsprozeduren oder Variablen-Deklarationen übersieht oder zuviel übernimmt. Eine vorherige Syntaxprüfung der Bibliothek ist nur auf Umwegen möglich, die Schnittstellen zwischen Bibliothek und Anwender-Programm können nicht explizit festgelegt und geprüft werden. Der Bibliotheks-File darf seinerseites nicht auf andere Bibliotheken zurückgreifen, denn es sind keine verschachtelten INCLUDEs erlaubt. Somit können Programme nicht hierarchisch aufgebaut werden.

Deshalb scheint die Arbeit mit den Bibliotheken unter Turbo-Pascal nicht so verbreitet zu sein. Die Sprache ist einfach nicht dafür entworfen worden. Außerdem sind die Bibliotheken nicht standardisiert und oft nicht ausreichend dokumentiert. Oder sie liegen nur in MS-DOS-Versionen vor.

Die Lösung scheint mir deshalb in Turbo-Modula zu liegen, das leider nicht so stark verbreitet ist. Es bietet ähnliche Möglichkeiten wie Turbo-Pascal und noch einige dazu. Turbo-Pascal-Programme und -Bibliotheken können fast unverändert übernommen werden und der Programmierer braucht keine neue Sprache zu lernen, denn Modula ist eine Weiterentwicklung von Pascal. Außerdem sind alle Programmteile des Modula-Systems fast genauso wie das Turbo-Pascal-System zu bedienen.

Welchen Vorteil bietet nun Modula? Die Sprache wurde für die intensive Nutzung von Bibliotheken entworfen, die hier "Module" genannt werden. Die Verwaltung der Module samt Syntaxcheck und Überprüfung der Schnittstellen übernimmt das Modula-System. Ein Modul hat immer eine klare Schnittstelle (Definitionsteil), alles andere (also die Implementation) bleibt dem Benutzer verborgen. Ein Modul kann wieder auf andere Module zurückgreifen, was eine Hierarchisierung der Programmteile ermöglicht. Außerdem kann ein Linker herausfinden, welche Teile einer Bibliothek wirklich gebraucht werden und diese herausfiltern; leider nutzt Turbo-Modula diese Möglichkeit nicht aus.

Standardmäßig sind Modula einige Module beigegeben, die vor allem zur (einfachen) Ein-/Ausgabe und zur Speicherverwaltung dienen. Der Benutzer kann nun eigene Module entwerfen (z.B. zur Daten-Verwaltung) und diese dann in seinen Programmen benutzen. Diesen Entwurf teilweise anhand vorhandener Turbo-Pascal-Bibliotheken, wollen wir in dem Projekt leisten. Außerdem sollen vorhandene und neu erstellte Module einheitlich (neu) dokumentiert werden.

Das Ergebnis werden (hoffentlich) viele gut dokumentierte Module sein, die dann jeder in seinen Programmen einsetzen kann. Dadurch sollte sich später die Programmerstellung vereinfachen.

Auszuführende Arbeiten:

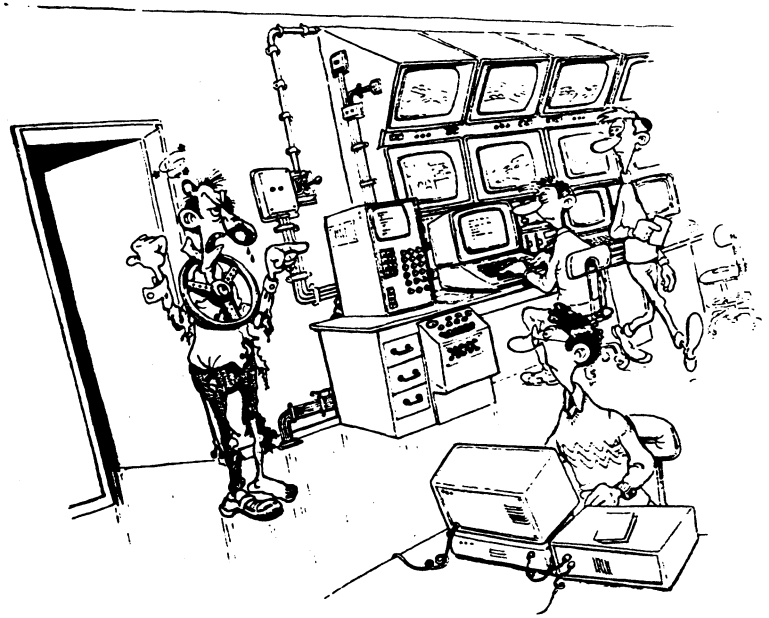
Die vorliegenden Turbo-Modula-Bibliotheken sollten neu dokumentiert werden, denn sie sind bisher nur in Englisch in der Turbo-Modula-Anleitung beschrieben worden. Die Beschreibungen sollten eine einheitliche Form bekommen, damit der Benutzer nicht bei jedem Modul lange nach benötigten Informationen suchen muß. Es geht hier nicht um eine Dokumentation des Source-Codes (der nicht vorliegt), sondern um eine Beschreibung der Schnittstellen und der Benutzung der Module. Dies könnte auch von Anfängern ohne Turbo-Modula-Erfahrung geleistet werden, soweit sie mit der englischen Anleitung zurechtkommen.

Aus der Reihe "CHIP-Special" liegen eine Menge Bibliotheken und Programme für Turbo-Pascal vor. Diese müßten gesichtet und für Turbo-Modula umgeschrieben werden. Teilweise existieren mehrere Versionen einer Bibliothek (z.B. Masken-Verwaltung), so daß eine Zusammenfassung bzw. Auswahl nötig wird. Für diesen Teil werden Programmierer mit Kenntnissen in Turbo-Pascal und möglichst auch in Turbo-Modula benötigt. Wenn Turbo-Modula nicht bekannt ist, müßte der Interessent sich einarbeiten (ausführliche englische Anleitung liegt vor).

Die fertiggestellten Module sollten wie die vorliegenden Module dokumentiert werden, wofür nur Turbo-Modula-Kenntnisse benötigt werden (auch für Anfänger). Außerdem wäre es schön, wenn zu den Modulen Beispielprogramme erstellt werden, um die Benutzung der Module zu demonstrieren.

Wenn aus dem Kreis der Interessenten noch Vorschläge für eigene Bibliotheken (Module) geäußert werden, können wir auch selbst noch welche entwerfen. Vorschläge (auch von Außenstehenden) werden immer gerne angenommen. Außerdem sei jeder Leser aufgerufen, seine eigenen Bibliotheken (seien sie in Turbo-Pascal oder Modula geschrieben) zur Verfügung zu stellen.

Interessenten melden sich bitte bei:
Gerald Schröder
oder Alexander Schmid



Welcher Idiot ist für die Ampeln am Kudamm verantwortlich!?

Ein kleiner Fehler im Z80DOS

Wie bei den CP/M-Benutzern hoffentlich allgemein bekannt, kann man dort die Bildschirmausgabe durch einen Druck auf Control-P auf den Drucker und wieder zurück, nur auf den Bildschirn, umleiten. Normalerweise sollte man davon nichts auf dem Bildschirm sehen, nur der Drucker sollte mitlaufen. Als ich jetzt das antike BDOS durch das Z80DOS ersetzt hatte, erzeugte das Cntl-P plötzlich seltsame Zeichen, die da gar nicht hingehörten. Als übeltäter stellte sich schließlich die Abfrageroutine im Z80DOS heraus. Der Autor hatte einen kleinen Sprung vergessen und so wurde fälschlicherweise in der Ausgaberoutine weitergemacht, statt nur ein Flag zu setzen und stillschweigend auf das nächste Zeichen zu warten. Im Source-File sieht das so aus:

```
; Read buffer
;
rdbuf:  ld      a,(tabcnt)              ; Get current position cursor
        ld      (tabcx1),a              ; Save it
rdbuf0: push    ix                      ; Save start address buffer
        pop     hl                      ; Get it in hl
        ld      c,(hl)                  ; Get maximum line lenght
        inc     hl                      ; Increment to line lenght position
        ld      b,0                     ; Clear line lenght counter
        push    hl                      ; Save start line - 1
rdbuf1: push    hl                      ; Save registers
        push    bc
rdbuf2: call    getch                   ; Get character
        pop     bc                      ; Restore registers
        pop     hl
        and     07fh                    ; Mask character
rdbuf3: cp      ContH                   ; Test backspace
        jr      nz,rdbuf4               ; Not then jump
doback: ld      a,b                     ; Test if deleting char from empty line
        or      a
        jr      z,rdbuf1                ; Yes then get next char
        pop     hl                      ; Get start line
        push    hl                      ; And save it again
        call    delch                   ; Delete character
        jr      rdbuf1                  ; Get next character
rdbuf4: cp      ContP                   ; Test print enable/disable
        jr      nz,rdbufC               ; Not then jump
        ld      a,(fContP)              ; Complement print flag
        cpl
        ld      (fContP),a
        jr      rdbuf1                  ;  Get next character       <= hier
rdbufc: cp      ContX                   ; Test delete line            passierts
        jr      nz,rdbufe               ; Not then jump
rdbufd: pop     hl                      ; Get start line
        ld      a,b                     ; Test if last character deleted
        or      a
        jp      z,RDBUF                 ; Yes start routine again
        push    hl                      ; Save pointer
        call    delch                   ; Delete last character line
        jr      rdbufd                  ; Test last character deleted
```

Wenn jemand es selber benutzt, oder er einen kennt, oder er einen kennt, der einen kennt..., bitte weitersagen. Kleine Ursache, mittelgroße Wirkung.

Alexander Schmid

Fraktal Generatoren für den TRS80 M 1 mit HRG 1B

Teil 1

Spricht man mit einem Computer Freak über Fraktale, so denkt dieser sofort an das bekannte Apfelmännchen.
Entdecker dieser Spezies ist Benoit B. Mandelbrot.
Das Apfelmännchen, oder auch Mandelbrot-Menge, ist aber nur eines von vielen Fraktalen, die man auch mit einem Fraktal Generator auf dem TRS 80 darstellen kann.
Das hier vorgestellte Programm in BASIC erzeugt eine vielzahl von Fraktalen in akzeptabler Zeit.

Bevor es an das Programmieren geht, einiges zur Theorie:

Fraktale (lat. ´fractus´= gebrochen) sind irreguläre, aber recht hübsch anzusehende Gebilde, die eine gemeinsame Eigenschaft haben: Ausschnitte dieser Gebilde sind sich selbstähnlich und Teile dieses Ausschnittes wiederum dem Ausschnitt selbstähnlich (toll).

Dieser geometrischen Prinzip folgen auch natürliche Gebilde, wie z.B. Küstenlandschaften oder Baumrinden um nur zwei zu nennen.
Theoretisch ist dies ein unendlicher Kaskadenprozeß, praktisch gibt es jedoch einen äußeren Abschnitt (Cutoff), der die Größe der Figur bestimmt, sowie einen inneren Cutoff, der theoretisch dann auftritt, wenn man die Größe eines Atoms erreicht. Das läßt sich graphisch naürlich nicht mehr darstellen. Wenn man dem Fraktal Generator Parameter angibt, die graphisch nicht mehr darstellbar sind, führt dies zu einer Fehlermeldung.

Wie sieht nun ein solcher Fraktal-Generator aus.
Man benötigt zunächst einen Initiator, im Programm festgelegt durch ´sides ?´, dann folgt ein ´Wachstumsgenerator´, der den Wachstumsprozeß festlegt, festgelegt durch ´Level ?´.
Der Generator wird dabei so skaliert und gedreht, daß die Endpunkte mit den Endpunkten der Initiator-Strecke zusammenfallen.

Das BASIC-Programm erzeugt Fraktale wie z.B. die Koch´sche Schneeflocke, Drachenflächen und die Serpinsky-Pfeilspitze.
Bei der Option ´Inverse ?´ erfolgt der Wachstumsprozeß nach

innen, begrenzt durch den kleinsten darstellbaren inneren Cutoff
Wird dieser überschritten, erfolgt eine Fehlermeldung.
So, genug geschrieben nun folgt das Programm:

```
10 ´*************************************************************
12 ´*********          Creating Fractals          **************
14 ´*********    with High Resolution Graphik     **************
16 ´*************************************************************
18 ´*********      TRS-80  Computerstation        **************
20 ´*********                Dorsten              **************
22 ´*********      Copyright by Egbert Schröer    **************
24 ´*********        Januar 1988    Vers. 1.3     **************
26 ´*************************************************************
28 ´*********       Original Version by           **************
30 ´*********    Michiel van de Panne Dec.1984    **************
32 ´*********       80 MICRO  Dec.´84 side 114    **************
34 ´*************************************************************
100 ´
150 ´***   All remarks can be left out   ***
160 ´
180 #CLS: #OPEN
200 CLEAR 1000 : CLS
300 CF= 3.1416/180
400 ON ERROR GOTO 9600
500 ´
600 ´*** Define point to start drawing ***
700 ´
800 CX%=241: CY%=154: X2=CX%: Y2=199-CY%
900 INPUT"Number of sides ";S%
1000 INPUT"Inverse";I$: I$=LEFT$(I$,1)
1100 ´
1200 ´*** Define numbers of degrees for right and left turn ***
1300 ´
1400 R=-360/S%: L=R+180
1500 IF I$="Y" THEN C$="A": W=R: R=-L: L=-W ELSE C$="R"
1600 A$=C$
1700 IF I$<>"Y"  THEN 2000
1800 AL$="R": S1%=S%-2: FOR N%=1 TO S1%: AL$=AL$+"L": NEXT:
     AL$=AL$+"R"
1900 GOTO 2200
2000 AL$="L": S1%=S%-2: FOR N%=1 TO S1%: AL$=AL$+"R": NEXT
2100 AL$=AL$+"L"
2200 INPUT"Number of levels ";LE%:IF LE%=1 THEN GOTO 3300
2300 ´
2400 ´*** Build string containing right and left ***
2500 ´*** turns to be done.                       ***
2600 ´
2700 FOR N%=2 TO LE%: LN=LEN(A$)
2800 FOR N1%=1 TO LN: B$=B$+MID$(A$,N1%,1)+AL$: NEXT N1%
2900 A$=B$: B$="": NEXT N%
3000 ´
3100 ´*** Lines 125-130 choose length of line segments ***
3200 ´
3300 CLS: IF S%<5 THEN GOTO 3500
3400 LL=452/(S%*3^LE%): GOTO 4000
```

```
3500 LL=144/(3*LE%)
3600 '
3700 '*** Actual drawing loop begins here ***
3800 '
3900 #CLS:#OPEN
4000 FOR Z%=1 TO S%
4100 FOR N%=1 TO LEN(A$)
4200 B1$=MID$(A$,N%,1)+AL$
4300 FOR N1%=1 TO LEN(B1$):B$=MID$(B1$,N1%,1)
4400 IF B$="A" THEN A=A+W: GOTO 4600
4500 IF B$="L" THEN A=A+L ELSE A=A+R
4600 AL=A*CF
4700 X=X+LL*COS(AL): Y=Y+LL*SIN(AL)
4800 XP=INT(X): YP=INT(Y): X1=XP+CX%: Y1=199-(YP/2+CY%)
     GOSUB 6400
4900 X2=X1: Y2=Y1
5000 NEXT N1%,N%,Z%
5100 '
5200 PRINT@1019,"Done";
5300 '
5400 '*** Press any key to continue ***
5500 '
5600 A$=INKEY$
5700 IF A$="" THEN 5600 ELSE GOSUB 8300
5800 END
5900 GOTO 5700
6000 '
6100 '*** Check to ensure that endpoints of ***
6200 '*** Line are on screen                ***
6300 '
6400 IF X1<0 OR X1>383 THEN RETURN
6500 IF X2<0 OR X2>383 THEN RETURN
6600 IF Y1<0 OR Y1>191 THEN RETURN
6700 IF Y2<0 OR Y2>191 THEN RETURN
6800 IF X1=X2 AND Y1=Y2 THEN #SET(X1,Y1): RETURN
6900 '
7000 A5=ABS(X2-X1): B5=ABS(Y2-Y1): IF A5>B5 THEN 7100 ELSE 7500
7100 IF X2>X1 THEN C5=X1: D5=X2: E5=Y1: F5=Y2: GOTO 7300
7200 C5=X2: D5=X1: E5=Y2: F5=Y1
7300 G5=D5-C5: H5=(F5-E5)/G5: FOR U5=C5 TO D5
7400 #SET(U5,E5):E5=E5+H5: NEXT: GOTO 7900
7500 IF Y2>Y1 THEN C5=Y1: D5=Y2: E5=X1: F5=X2: GOTO 7700
7600 C5=Y2: D5=Y1: E5=X2: F5=X1
7700 G5=D5-C5: H5=(F5-E5)/G5
7800 FOR U5=C5 TO D5: #SET(E5,U5): E5=E5+H5: NEXT
7900 RETURN
8000 '
8100 '*** Graphik store routine ***
8200 '
8300 #CLOSE
8400 CLS:PRINTCHR$(23)
8500 PRINT@8*64,"Graphik ---> Floppy Y/N ";
     INPUT I$ : IF I$="N" OR I$="n" THEN CLS : RETURN
8600 PRINT@640,"Dateiname :   ...........";
     PRINT@665,"";
     LINE INPUT DAT$
8700 CD$="RENAME FRACTAL/HRG "+DAT$
8800 CLS:#OPEN
8900 #SAVE"FRACTAL/HRG"
9000 #CLOSE
9100 CMD CD$
9200 RETURN
9300 '
9400 '*** Error checking Routine ***
9500 '
9600 CLS: #CLS
9700 PRINTTAB(18)"E r r o r  M e s s a g e"
9800 PRINTTAB(15)STRING$(31,45)
9900 PRINT@4*64,"If Level II Error : Errorcode ";ERR/2+1
     PRINT@6*64,"Line number       :";ERL
     PRINT@8*64,"Last DOS-Error    :";CMD"E"
10000 PRINT:PRINT"Program can't work on !"
      PRINT"Please press <ENTER> to restart"
10100 A$=INKEY$: IF A$<>CHR$(13) THEN 10100 ELSE RUN
```

Alle REM's im Programm können entfernt werden. Anschließend sollte man das Programm mit ACCEL3 compilieren, damit man nicht eine Ewigkeit auf ein Fraktal warten muß.
Der Fraktal Generator ist unter NEWDOS80 und GDOS lauffähig.
Wer sich die Mühe des abtippens ersparen will, kann von mir eine compilierte Form anfordern.
Wie bereits im Titel angedroht folgt noch ein zweiter Teil, der sich mit einem Apfelmännchen Generator beschäftigt.

Literatur:

Benoit B. Mandelbrot: The Fractal Geometry of Nature
W.H. Freeman and Company

Michiel van de Panne: Creating Fractals
80 MICRO, Dezember 1984, Seite 114 u.f.

U L R I C H H E I D E N R E I C H

Werderstr. 35
4300 Essen 1
priv. 0201 / 282 770
dienstl. 0203 / 278 64

U.Heidenreich,43 Essen,Werderstr.35

CLUB 80
Herrn Gerald Schröder
Am Schützenplatz 14

2105 Seevetal 1

Ihr Schreiben/Zeichen 31.01.89 | Mein Schreiben/Zeichen | Essen, den 08.01.89

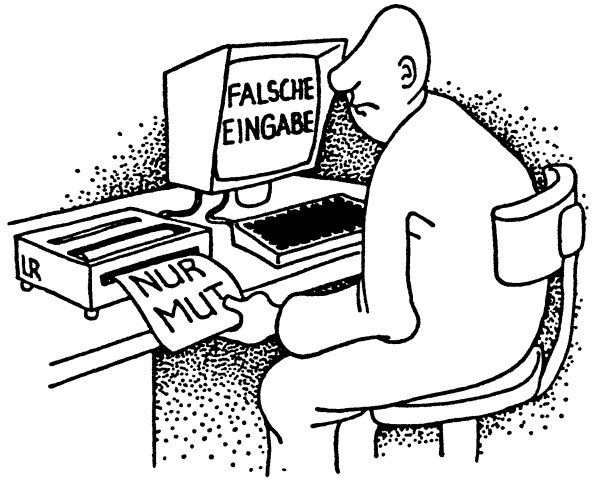
Lieber Gerald, Liebe Clubmitglieder,

meine Pflichten als Clubmitglied habe ich zugegebenermaßen fürchterlich vernachlässigt (vernachlässigen müssen ?). Leider bin ich inzwischen beruflich derart in Anspruch genommen, daß keine Zeit für den Club bleibt. Das ist nicht nur so dahingesagt: Ich war nämlich so dämlich (?), mein Hobby zum Beruf zu machen und habe demzufolge keine Zeit zum Hobby mehr. (Womit auch meine Anwesenheit beim Club-Treffen ausfallen muß, da mein Samstag nun zu den Arbeitstagen zählt!) Nichtsdestowenigertrotz hat mich das Einladungs-Schreiben dazu provoziert, zumindest meinen Senf zum Thema "'Club 80' jetzt auch gleich 'Club 8088' ?" hinzuzugeben:

Um die Allmacht der IBM-Nachbauten - ja gar nicht Mama Blue selbst! - herumzukommen, scheint auch unserm Club kaum möglich zu sein: Sei es, Einsteiger in die Computerwelt einzuführen (das macht heute niemand mit 'nem TRS80 mehr!), sei es aber auch dem zweiten Grundgedanken des Clubs zu folgen, KnowHow zu vermitteln. Unseren Betriebssystem- und Hardwareforschern wird das schier unermesslich undokumentierte Feld MSDOS / IBM auf Jahre hinaus interessanten Stoff bieten.

Auch um auf der 80er Linie zu bleiben, wäre ich der Meinung, die Aktivitäten statt in Richtung Atari, Amiga o.ä. (Nichts gegen die Geräte; Fettnäpfchen läßt grüßen: Ich verkaufe inzwischen sowas!) weiter in Richtung IBM auszudehnen. Wahrscheinlich bin ich nicht der einzige Club-80er, der vom Genie über GDOS/NEWDOS zum - würg! - CP/M-wechselte, dort statt BASIC (in dem wohl manch einer "seit Jahren nicht mehr programmiert") an (Turbo-) Pascal Spaß fand und über diese Sprache nahtlos zum MSDOS wechseln konnte.

Unsere Assembler-Freaks werden darüber hinaus schnell feststellen, daß 8088-Assembler gar nicht mal soooo vom vertrauten Z80 abweicht und sich auch in dieser Materie sofort wohlfühlen.



Es hat sich als sinnvoll erwiesen, wenn der Computer nicht intelligenter ist als der Anwender.

Wobei ich - mit etwas Melancholie! - "unserm" GDOS/NEWDOS keineswegs den Gnadenstoß verpassen möchte. Oft Ray Duncans Kommentare im "mc" zur MSDOS-Szene lesend, kommt mir der Aha-Effekt: Was der Mensch da leise ironisch bemängelt, könnte NEWDOS/GDOS - entsprechend dem Zug der Zeit folgend - allemale! (Was ist eigentlich aus Apparat Inc. geworden?)

Quasi als Produkt meines schlechten Gewissens dem Club gegenüber, als Diskussionsgrundlage für die Öffnung unseres Clubs in Richtung MS-DOS und als abschreckendes Beispiel, wie schlecht ein NEWDOS-Verwöhnter mit dem ach so modernen Dos zufrieden ist, möchte ich doch folgende Seiten für's Club-Info beisteuern.

MfG

Ulrich Heidenreich

Möglichkeiten und Grenzen des MS-DOS -

- Erfahrungsbericht eines Auf(?)steigers

Nach langem Sparen wanderte mit meinem TRS-80 ein NEWDOS/80 und ein noch älteres CP/M auf den Altenteil: Ein AT glänzt nun auf dem Schreibtisch. Die berühmten drei Buchstaben prangen zwar nicht auf dem Typenschild, dennoch gibt sich der Neue alle Mühe, dem großen Vorbild gerecht zu werden: Von der Tastatur - auf der Sie als Cherry-Verwöhnter dank des seitlichen Spiels der Tasten und der frühen Kontaktgabe oft genug mehrere Tasten gleichzeitig treffen - bis hin zur Bildschirmdarstellung steht er Big Blue in nichts nach. So wäre es auch wohl nicht zeitgemäß, den TTL-Monitor mit schmalsten Linien gestochen scharfe Schrift darstellen zu lassen, obwohl's er durchaus auf die Beine bringt. Sans Serif ist halt nur bei Druckern wieder in; auf dem Schirm darf den Augen zuliebe (?) ruhig gekleckst werden.

(Nebenbei: als glücklicher Besitzer einer EGA-Karte könnten Sie eine bessere Darstellung "stylen" . In Kürze: Interrupt 10H, AH=11H, ES:BP auf eigene Zeichenmatrix setzen! - wenn nicht bereits MS-DOS's CLS-Befehl den Original-Zeichensatz wieder aktivieren würde!)

UNIX inklusiv

Aber was soll's. Schließlich hat man sich das Gerät nicht deswegen auf den Schreibtisch gestellt, sondern liebäugelte mit anderen Vorteilen: Neben der obligatorischen Festplatte - endlich keine Assembler-Kaffeepause mehr! - verspricht MS-DOS Möglichkeiten, die CP/M damals noch durch Micro-Shell hinzugefügt werden mußten. Bei MS-DOS dagegen sind mit Piping, Input/Output-Redirecting und seinen Batch-Fähigkeiten diese Unix-Features inklusiv.

So ist MS-DOS in einer seiner neuesten Versionen wohl leistungsfähiger als ein altes CP/M 2.2 mit UNIX-Shell. Schon die einfache Möglichkeit, mehrere Text-Dateien auf den Bildschirm zu holen, begeistert: Kein XPIP, SWEEP oder sonst ein Utility ist nötig, einfach COPY *.TXT CON:! Dies schreibt sogar vor jeder so aufgelisteten Datei als Überschrift deren Namen auf den Bildschirm. Mannigfaltig und interessant sind die Wege, dasselbe auf den Drucker zu bekommen:

COPY *.TXT PRN: kopiert die Datei-Inhalte auf den Drucker, die Dateinamen landen jedoch weiter auf dem Bildschirm. Auch COPY *.TXT CON: >PRN: ist interessant: Jetzt gerät die Liste der Dateinamen auf den Drucker, die Datei-Inhalte erscheinen aber trotz Output-Redirection auf dem Bildschirm. Aber warum einfach, wenn's auch kompliziert geht: COPY *.TXT CON: | MORE bringt Seite für Seite auf dem Schirm, die dann per Taste "PrtScr" zum Drucker geschickt werden könnten. Leider erscheinen so erst alle Datei-Inhalte ungebremst auf dem Schirm, gefolgt von der Liste der Dateinamen. (Letztere, falls genügend lang, anhaltbar!)

Erst die Holzhammer-Methode, alles, was trotz COPY auf PRN: nicht bei PRN: ankommt, zusätzlich zu redirecten, wirkt Wunder:

```
COPY *.TXT PRN: > PRN:
```


Na ja, nach eingehendem Studium des guten DOS-Handbuchs hätte man ja auch sofort auf diese Lösung kommen können! Die Möglichkeiten des MS-DOS sind halt derart mannigfaltig, daß man nicht alles auf Anhieb sieht. So kann man Dank Filter MORE und der Prüfung mit EXIST doch tatsächlich zwei Inhaltsverzeichnisse vergleichen:

Mit DIRCOMP.BAT -

```
echo off
cd %1
echo Files missing in %2
for %%x in (*.*) do if not exist %2\%%x echo %%x
cd %2
echo Files missing in %1
for %%x in (*.*) do if not exist %1\%%x echo %%x
```

- sollten nach DIRCOMP \DIR1 \DIR2 | MORE alle Unterschiede - nämlich die im jeweils anderen Unterverzeichnis fehlenden Dateinamen - Seite für Seite auf dem Schirm gemeldet werden. Bis auf "Seite für Seite" klappt's auch; nur: die Ausgabe geschieht ungebremst. Was habe ich denn nun schon wieder bei der Beschreibung des MORE-Filters mißverstanden? Heißt "MORE" etwa, daß "mehr" als eine Seite ausgegeben wird?

Spanabhebende Datenverarbeitung

Weiter suchte ich - da das Arbeiten mit Disketten bekanntermaßen spanabhebende Datenverarbeitung darstellt - neulich nach einer Möglichkeit, auf meiner Sicherungs-Disk nicht immer wieder alle Dateien abzulegen, sondern nur die noch nicht gesicherten welchen. Mit MS-DOS's Batch-Fähigkeiten war dann auch schnell die entsprechende BATch-Datei kreiert:

```
ECHO OFF
FOR %%X IN (C:*.*) DO IF NOT EXIST A:%%X COPY C:%%X A:
```

Als Krönung dieser Schöpfung hing ich noch ein > NUL: hintenan. Eine Umleitung auf dieses Gerät läßt ja alle Ausgaben im Niemandsland verschwinden, statt sie dort auftauchen zu lassen, wo sie stören könnten - z.B. mitten in Eingabemasken. Mein Erstaunen, als dann doch eine "1 File(s) copied"-Meldung nach der anderen den Bildschirm verzierte, währte aber nicht lange: Seit MS-DOS 3.2 in meinem AT waltet, erledigt REPLACE C:*.* A: /A diese Aufgabe. Und REPLACE's Ausgaben lassen sich widerspruchslos NULlen.

BATch as BATch can!

Vielleicht lag's ja auch daran, daß Output-Redirecting aus .BAT-Dateien ein wenig anders funktioniert als der unbedarfte Anwender meint. Otto-Normalprogrammierer irrt halt, wenn er davon ausgeht, daß alle Ausgaben des Programms links vom ">" an das Gerät rechts desselben gelangen. Auch wenn in TEST.BAT TYPE AUTOEXEC.BAT steht, so heißt das noch lange nicht, daß nach TEST > PRN: eine Kopie der AUTOEXEC.BAT auf dem Drucker erscheint! Aus Stapeldateien heraus redirected MS-DOS nämlich wesentlich einfacher als mit dem schwerverständlichen ">"-Symbol:

Zunächst einmal definiert man je einen Eingabe- sowie Ausgabekanal Namens IDEV und ODEV durch Aufnahme der Zeilen

```
SET ODEV=CON:
SET IDEV=CON:
```

in die AUTOEXEC.BAT. Dann werden ebenso leicht und schnell in allen *.BAT-Dateien alle Ein-/Ausgaben, die Sie zu redirekten wünschen auf <%IDEV% und >%ODEV% umgeleitet. Nach diesen kleinen Vorbereitungen genügt - zum Beispiel - ein einfaches "SET ODEV=PRN:" für mühelose Ausgaben aus diesen Batch-Dateien an den Drucker!

Kanäle

Wem das nun doch nicht so einfach erscheint, sei daran erinnert, daß für solche Spielereien I/O-Redirecting ja auch nicht gedacht war. Die UNIX-Philosophie ist eher, hierdurch Kanäle zur Kommunikation zwischen Programmen bereitzustellen, welche selbst keine eigenen Mechanismen dafür besitzen. Ein äußerst treffendes Beispiel ist hier der bereits zitierte REPLACE-Befehl:

Weder REPLACE kennt einen Kanal, um seine Meldungen - zum Beispiel an ein laufendes BASIC-Programm - weiterzugeben, noch GW-BASIC einen solchen, um Fehlermeldungen eines DOS-Utilities - zum Beispiel REPLACE - einzulesen. I/O-Redirecting kann diesen Kanal nun aber zur Verfügung stellen:

```
10000 REM      Datensicherung
10010 :
10020 SHELL "REPLACE C:*.* A: /A > ERR_CHAN"
10030 OPEN "I",1,"ERR_CHAN"
10040 WHILE NOT EOF(1)
10050  INPUT #1,ERR.MSG$
10060 :
```

Was habe ich bloß falsch gemacht, daß - zum Beispiel bei voller Sicherungsdiskette - REPLACE's Fehlermeldungen trotz allem den Bildschirm ver(un-)zieren? MS-DOS's Piping-Philosophie ist wohl nicht so ganz UNIX-like. Oder liegt's etwa am Denkfehler, daß man zur Fehlermeldung eines Prozesses an den anderen ja auch keine Pipes nutzen darf (warum nicht?), sondern den DOS-Exit-Code - eingefleischten BATchern auch als Errorlevel wohlbekannt - auswertet? Hierüber lassen sich - geeignete Hochsprachen vorausgesetzt - Fehlermeldungen von Child-Prozessen einlesen; TURBO-Pascal 4.0 zählt zu den genannten:

Nach EXEC ('COMMAND.COM','/Ccopy A:DATEN C: > NUL:'); erschiene im Fehlerfalle zunächst die Meldung "A:DATEN not found" und "0 File(s) copied", die sich aber - o Wunder, warum funktioniert's denn jetzt? - in den NUL:-Kanal schicken läßt. Über die Pascal-Funktion DosExitCode kann man dann ja, ohne diese Meldung auf dem Schirm erscheinen zu lassen, die fehlerfreie oder -hafte Ausführung des COPY-Befehls feststellen:

```
If DosExitCode<>0 then Writeln ('Datei "DATEN" fehlerhaft!');
```


Wie sollte es anders sein: Auch hier stellen sich interessante, aber unbrauchbare Effekte ein: Unkritische (?) Fehler - wie zum Beispiel das Fehlen der Datei "DATEN" - erzeugen einen Exit-Code von 0, gaukeln also Fehlerfreiheit vor; bei ernsthafteren Fehlern - z.B. keine Diskette im Laufwerk A: - schlägt Abort, Retry, Ignore zu! (Zu diesem exzellenten Feature später mehr, jetzt erst:)

Nochmal Eingabe-Umleitung

Eingabe-Umleitung - das Holen von Daten statt von der Tastatur aus einer Datei - kann nämlich nicht nur die Tippfaulheit des Anwesenden befriedigen. Nein, auch automatisch ablaufende Prozesse unter Abwesendheit desselben können so bedient werden:

```
:
IF NOT EXIST A:*.* GOTO NODRIVE_A < IGNORE.CTL
COPY *.* A:
GOTO EXIT
:NODRIVE_A
MD C:\SAVECAT
COPY *.* C:\SAVECAT
:EXIT
:
```

In der Eingabe-Datei müssen halt nur die entsprechenden Tastendrücke stehen; IGNORE.CTL - Inhalt "I <cr>" - bedient in diesem Beispiel die Abfrage "Abort, Retry, Ignore ?"

Aber auch hier muß ich wohl die Handbuch-Notiz, das Symbol "<" bedeute, Eingaben für das Programm links desselben aus der Datei rechts desselben zu holen, mißverstanden haben! Trotz Input-Redirecting beharrte MS-DOS hier solange auf dem Tastendruck "I", bis ich am Montag morgen entsetzt feststellen mußte, daß meine automatische Datensicherung über das ganze Wochenende hinweg auf meinen dicken Daumen gewartet hat! Dafür bietet aber MS-DOS mit diesem Abort-Retry-Ignore-Feature Möglichkeiten, die den oft fälligen Warmstart per CTRL-C im alten CP/M bei weitem übertreffen:

Abort, Retry, Ignore über alles!

Kein vergessener {$I-}-Schalter plagt mehr den Turbo-PASCALler, kein BASIC-Programm bedarf mehr großartiger Error-Trap-Routinen, kein ERRORLEVEL braucht umständlich in BATch-Dateien abgefragt zu werden. COMMAND.COM läßt den Fehler - zum Beispiel eine offene Laufwerksklappe - erst gar nicht bis zur Hochsprache durch; Klappe zu und "R" gedrückt und die Sache ist gegessen. Auch ein versehentliches Ansprechen eines nicht vorhandenen Laufwerkes oder des "Off-Linen" Druckers ist kein Problem: Einfach "A"bort gewählt und schon ist das Versehen ungeschehen! Der Abort-Retry-Ignore-Text in der Datenerfassungs-Maske oder mitten in der CAD-Zeichnung - hier meist dekorativ als "Mixed Pixels" - stört ja eh niemanden; und wenn's wirklich mal vollautomatisch gehen soll, stellt man eben einen schweren Gegenstand auf die I-Taste!

Manche Anwenderprogramme nehmen einem den Abort zwar übel und stürzen ins DOS zurück, aber dafür hat man ja die Leerdatei R.COM, um wieder hineinzukommen. Ach nein, Sorry! Das funktionierte ja nur unter CP/M. MS-DOS kennt da ja etwas viel eleganteres: Rufen Sie COMMAND.COM mit dem Schalter /F, so wird kein Anwendungsprogramm mehr abgebrochen, sondern solche Fehler ohne Rückfrage - natürlich (?) dann auch mit Exit-Code 0 für "fehlerfrei"! - "I"gnoriert.

Aber beim Piping ...

Zurück zum Redirecting: CP/M's-WordStar können Sie mit Micro-Shell's Hilfe problemlos per Eingabeumleitung bedienen: Warum WS patchen, eine Steuerdatei des Inhalts "CTRL-OR78 ‹cr› CTRL-JH0" leistete damals gute Dienste; nur unter MS-DOS geht's so nicht mehr. Ob wohl MS-DOS beim "Redirecten" nicht in die "richtigen" I/O-Routinen eingreift? Dafür spricht dieser - zwar nicht unbedingt als Fehler anzusehende aber doch interessante - Effekt: Der GW-BASIC-Interpreter selbst (why not?) läßt sich nicht durch Piping/Redirecting bedienen, wohl aber das BASIC-Programm PIPETEST.BAS:

```
1000  INPUT A$
1010  PRINT A$
1020  END : REM Hier steht END und nicht SYSTEM!!!
```

Wenn Sie dieses Programm per ECHO Hallo ¦ BASIC PIPETEST mit Daten versorgen, funktioniert es zwar; der BASIC-Interpreter kehrt anschließend jedoch wieder ins DOS zurück. Seltsam, nicht wahr, wie ein Piping, das lediglich Daten in eine Pipe schreibt, den BASIC-Interpreter, der sich ebenso "lediglich" diese Daten aus der Pipe holt, derart beinflussen kann?

Da darüber hinaus Routing - also die Möglichkeit, Ausgabekanäle umzudefinieren (von den besch...eidenen Möglichkeiten des MODE LPT mal abgesehen) - MS-DOS völlig unbekannt ist, griff ich jüngst zur Assemblerlösung, um Ausgaben auf die RS232-Schnittstelle mangels geeigneter serieller Peripherie auf dem Schirm zu sichten:

```
          MOV  AX,2514        ; RS-323-Interrupt 14 via Funktion 25 auf
          MOV  DX,NEWINT      ; neue Routine "umbiegen":
          INT  21
          JMP  HERUM
NEWINT:   MOV  DL,AL          ; Statt die RS-232 zu bedienen
          MOV  AH,2           ; Ausgabe auf Bildschirm schreiben
          INT  21
          XOR  AX,AX          ; Fehlerflags löschen
          IRET                ; 'raus ausm neuen INT 14
HERUM:    MOV  DX,HERUM       ; Routing resident
          INT 27              ; verlassen
```

"COPY IRGEND.WAS COM1:" brachte anschließend das Erfolgserlebnis, tatsächlich "IRGEND.WAS" auf dem Bildschirm bewundern zu können. Ein "TYPE IRGEND.WAS › COM1:" sprach dagegen wieder die RS-232-Schnittstelle statt des Schirms an. Es zeugt wohl von meinen schlechten Fähigkeiten als Systemprogrammierer, wenn ich mich (unter der Annahme, ein BIOS stelle hardwarespezifische Routinen zur Verfügung, welche vom DOS unter Unkenntnis der tatsächlichen Hardware über BIOS-Vektoren - und zwar nur über diese! - genutzt werden) verwundert frage: "Wie - zum Teufel - hat dann MS-DOS trotzdem einen Weg zur RS-232 gefunden?!"

... ist MS-DOS konsequent!

Und nun eine Verständnisfrage: Wann wird wohl ein Input-Redirecting beendet: Bei Rückkehr auf System-Ebene oder am EoF der Input-Datei? Die Antwort bekommen Sie spätestens dann, wenn Ihr MS-DOS im siebten Computer-Himmel weilt, weil unerwarteterweise das zu bedienende Programm mehr Eingaben anfordert, als vom Input-File zur Verfügung gestellt werden. Hier ist MS-DOS nämlich konsequent:

§1. Input-Redirecting bleibt solange aktiv, wie das bediente Programm aktiv ist.

§2. Ist Input-Redirecting aktiv, ignoriere man die Tastatur (inklusiv Alt-CTRL-Del!).

§3. Wenn der Input-File leergelesen wurde, tritt §1 in Kraft!

Wer's nicht glaubt, möge einmal mit dem an sich wenig sinnvollen Befehl DEBUG ‹ AUTOEXEC.BAT die Probe auf's Exempel durchführen: Solange DEBUG keinen Q-Befehl bekommt, bleibt DEBUG aktiv. Solange DEBUG aktiv ist, bleibt die Eingabeumleitung aktiv. Solange die Eingabeumleitung aktiv ist, bleibt die Tastatur inaktiv. Solange ... (Hoffentlich verfügt Ihr Rechner über einen Hardware-Reset, sonst hilft nur noch der Griff zum großen roten Schalter hinten rechts!) Ach, wie einfach wär's doch, nach dem Leerlesen der Datei wieder auf die Tastatur zu lauschen!

Absteigende Versionsnummern?

Zu Guter Letzt noch eine MS-DOS-Logik: Wenn Sie davon ausgehen dürfen, daß mit jeder neuen Version eines BASIC-Interpreters bessere Features hinzukommen und ältere Fehler ausgemerzt werden, so scheint Microsoft seine GW-BASIC-Versionsnummern in absteigender Reihenfolge zu vergeben. Wie ich zu dem Schluß komme? Nun, starten Sie einmal dieses BASIC-Prográmmchen

```
100 FOR ZEILE=1 TO 24
110  FOR SPALTE=1 TO 80
120   LOCATE ZEILE,SPALTE
130  NEXT SPALTE
140 NEXT ZEILE
150 GOTO 100
```

so werden Sie als glücklicher Besitzer der BASIC-Version 2.02 den Cursor schön brav über den Bildschirm flitzen sehen; GW-BASIC 3.11 kann dies nicht: Die Cursorposition wird zwar intern nachgeführt, der Cursor selbst bleibt jedoch immer an der Position der letzten Ausgabe hängen. LOCATE mit nachfolgendem PRINT kann so zwar noch sinnvoll programmiert werden; für die Cursor-Nachführung bei bildschirmorientierter Eingabe ist mit dieser Interpreter-Version allerdings Trick 17 nötig:

```
10000 DEF FNCS$=CHR$(SCREEN(CSRLIN,POS(0)))
10010 :
50000 REM
50010 REM    Unterprogramm Cursornachführung
50020 REM
50030 IF SPALTE=1 THEN IF ZEILE>1 THEN CZEILE=ZEILE-1:CSPALTE=80
50040 IF SPALTE>1 THEN CZEILE=ZEILE:CSPALTE=SPALTE-1
50050 LOCATE CZEILE,CSPALTE
50060 PRINT FNCS$;
50070 RETURN
```

Der Editor des GW-BASIC 2.02 versteht diverse CTRL-Sequenzen, die in kaum einem Handbuch erwähnt sind:

```
CTRL ...  B: Cursor ein Wort zurück
          C: Abbruch, wie CTRL-BREAK
          E: Löschen bis Zeilen-Ende
          F: Cursor ein Wort weiter
          G: Beep
          H: Zeichen rechts vom Cursor löschen
          I: TAB
          J: Leerzeile einfügen
          K: Cursor Home
          L: Bildschirm löschen
          M: New Line
          N: Cursor ans Zeilenende
          R: Einfügen An/Aus
          T: Key ON/OFF (nur in Version 3.11)
          U: Eingabe bzw. Zeile löschen
          W: Wort an Cursorposition löschen
          X: Vorhergehende Zeile listen
          Y: Nächste Zeile listen
          Z: Ab Cursorposition bis Bildschirmende
             löschen
```

In der Version 3.2 funktionieren hiervon CTRL T bis CTRL Z nicht; an Stelle dieser Editor-Hilfen verzieren hier die entsprechenden ASCII-Symbole 21 bis 26 den Programmtext! Auch der Abbruch mit CTRL-C ist dort nicht möglich.

SHELL oder nicht SHELL, das ist hier die Frage

Version 2.02 erlaubt ein SHELL "BASIC"; die höheren Versionen 3.11 und 3.20 verbieten dies mit der Meldung "You can not SHELL to BASIC"! Teils zu Recht, da die 3er-Versionen das MS-DOS hierdurch in die ewigen Jagdgründe schicken können; teils zu Unrecht, weil die Version 2.02 beweist, daß ein SHELL-to-BASIC durchaus funktioniert:

```
Von GW-BASIC V2.02 nach Version 3.xx :
geht problemlos!

Von GW-BASIC V2.02 nach Version 2.02 :
geht problemlos!

Von GW-BASIC V3.xx nach Version 3.xx :
"You can not ..." !

Von GW-BASIC V3.xx nach Version 2.02 :
"You can" zwar, aber beim Ausstieg aus dem gerufenen BASIC hängt MS-DOS!
```


Peinlich wird so etwas, wenn Sie BASIC-Module mit SHELL "BASIC MODULxxx" als echte Unterprogramme - mit echten lokalen Variablen! - rufen möchten. Vorgesehen hatte ich dies zum Beispiel für eine On-Line-Hilfe mit der berühmten F1-Taste:

```
10020 ON KEY(1) GOSUB 65000
10030 :

20000 REM      Bearbeitung Kontenstamm
20010 TOPIC$="KONTEN":KEY(1) ON
20020 :
21000 KEY(1) OFF:RETURN
21010 :

40000 REM      Einlesen Mehrwertsteuer-Schlüssel
40010 TOPIC$="MWST":KEY(1) ON
40020 :
41000 KEY(1) OFF:RETURN
41010 :

65000 SHELL "BASIC "+TOPIC$+".HLP"
65010 RETURN
```

was auch auf meiner BASIC-Version zur vollsten Zufriedenheit funktionierte, bis ich feststellen mußte, daß die 3er-BASICs dies nicht mehr erlauben! Zudem führen die 3er-Versionen bei jedem SHELL-Aufruf ein "Cursor-Home" aus, was den ganzen Bildaufbau störend beeinflussen kann; Version 2.02 beläßt dagegen den Cursor schön brav, wo er ist.

Nach solchen Erfahrungen glaube ich nun fest an die absteigende Vergabe der Interpreter-Versionsnummern, benutze seitdem das "neueste" GW-BASIC 2.02, nehme dabei in Kauf, zu den weniger leistungsfähigen Versionen ab 3.xx nicht mehr abwärts- (oder aufwärts-?) kompatibel zu sein und hoffe auf GW-BASIC 1.0: Dies wird vielleicht endlich auch eine EGA-Grafik vernünftig bedienen, nach WIDTH "SCRN:",132 tatsächlich den 132-Spalten-Modus verstehen und wahrscheinlich auch den RESUME-Befehl vernünftig ausführen können. Werfen Sie einmal einen Blick auf diese Zeilen:

```
10000 ON ERROR GOTO 10080
10010 OPEN "R",1,"A:MURPHY":REM A:MURPHY existiere bereits!
10020 FIELD 1,128 AS ERFINDER$
10030 LSET ERFINDER$="IDIOT"
10040 PUT 1,1
10050 CLOSE 1
10060 PRINT "Daten OK gesichert!"
10070 END
10080 IF ERR<>70 THEN END
10090 PRINT "Bitte Schreibschutz entfernen. ";
10100 INPUT "Fertig ";DUMMY$
10110 RESUME
```

Offensichtlich fängt hier ON ERROR die Fehlermeldung "Write protected diskette" ab, bittet Sie, den Schreibschutz zu entfernen und nimmt mit RESUME die Verarbeitung wieder auf. Da aber frei nach Murphy kein Programm idiotensicher ist, weil Idioten so erfinderisch sind, wird, falls dieser sprichwörtliche Erfinder den Schreibschutz trotzdem auf der Scheibe läßt, ... Aber probieren Sie doch selbst!

Zurück zum DOS: Jüngst sah ich mich der Situation konfrontiert, ein Programm, welches seine Daten partout auf der Festplatte C: erwartet, auf einer reinen Diskettenmaschine ablaufen lassen zu müssen. Null Problemo: ASSIGN C=A leitet ja alle Zugriffe auf C: stattdessen nach A: um. Denkste! DOS's Kommentar "Illegal Parameter" - logisch (???), da ja Laufwerk C: gar nicht vorhanden ist, demzufolge auch nicht Parameter eines Befehls sein darf - machte jedwede Hoffnung zu nichte.

Liebe Firma Microsoft: Wie wär's, statt immer nur neue Features in die nächsten MS-DOS-Versionen hineinzustricken, zunächst endlich einmal die Bugs aus den alten Versionen zu beseitigen? So bleibt zu befürchten, daß auch OS/2 zwar weitere Möglichkeiten - Sprich: Multitasking - verspricht, uns arme Irren aber, die mehr wünschen, als nur fertige Programme (Wer entwickelt eigentlich unter solchen Voraussetzungen brauchbare Programme???) laufen zu lassen, durch solche Fehler weiterhin zur Weißglut treibt!

In Richtung Multitasking schwant mir sowieso Böses: Schon MS-DOS verspricht mit seinem Drucker-Spooler "PRINT.COM" einen Touch "Multi". PRINT funktioniert offensichtlich auch; nur möchten Sie natürlich gerne die Spool-Dateien nach erledigtem Druckauftrag wieder löschen können. Kein Problem: Sie befehlen "PRINT spoolfile" und "DEL spoolfile". Jedes vernünftige Multitasking wird ja wohl dem Task "DEL" solange den Zugriff auf "spoolfile" verbieten, bis der Task "PRINT" seine Aufgabe erledigt hat! Oder etwa nicht ... ??? Dabei wären nicht einmal großartige (Semaphor-) Klimmzüge nötig; PRINT.COM bräuchte die Spool-Dateien nur für die Dauer des Ausdrucks mit dem Read-Only-Attribut zu versehen, um das Schlimmste zu verhindern!

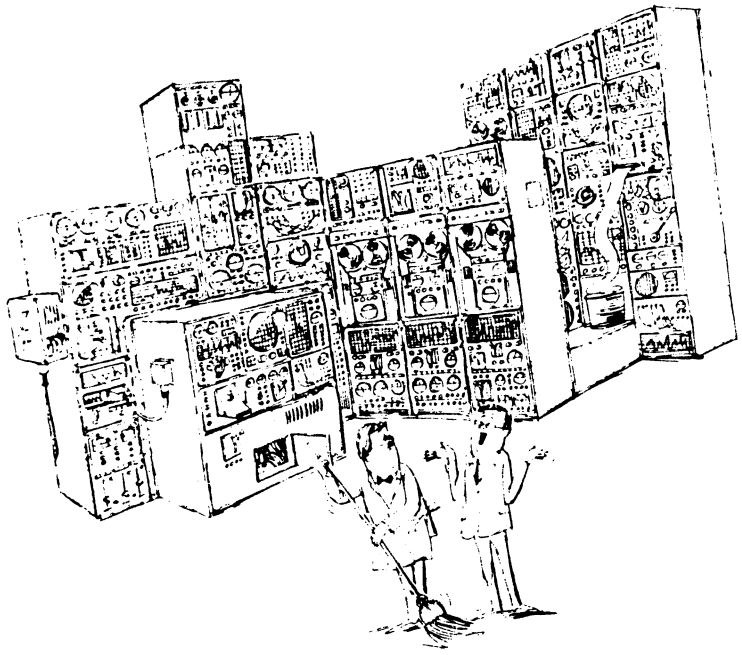
Zum Abschluß noch eine Preisfrage:

Sie befehlen MS-DOS: COPY *.TXT \TEXTE\SUMTEXT

a) Fasst der Befehl alle *.TXT-Dateien in der Datei SUMTEXT im Verzeichnis \TEXTE zusammen?

b) Kopiert dieser Befehl alle *.TXT-Dateien in gleichnamige Dateien ins Verzeichnis \TEXTE\SUMTEXT?

Ach so, was ich am Ende noch bemerken möchte: Falls Sie's etwa als Vorteil ansehen sollten, daß MS-DOS bereits zum Lieferumfang Ihres IBM-Kompatiblen gehört, dann lassen Sie sich eines Besseren belehren. Kaufen Sie Ihr MS-DOS lieber direkt von Microsoft, sonst bekommen Sie bei Rückfragen zum DOS nur den schönen Formbrief, Microsoft unterstütze keine OEM-Versionen seines Betriebssystems!



»Aber Chef, Sie haben doch selbst gesagt, der Computer solle alle Mitarbeiter ihren Fähigkeiten entsprechend einsetzen!«

Wie funktioniert ein Vektorgenerator?

Hochauflösende Grafikkarte für TRS-80 (Teil 2)

Das Listing des Vektorgenerators aus der letzten Ausgabe wird hier genauer erklärt. Zuerst aber soll ein Treiber für die Grafikkarte in Pascal behandelt werden, der mit dem Vektorgenerator in Assembler zusammenarbeitet, anschließend finden Sie den Algorithmus des Vektorgenerators in umgangssprachlicher Form erläutert.

Behandeln wir zunächst kurz den Bildschirmtreiber (Listing 3). Er ist so geschrieben, daß die Parameter der Zeichenbefehle in der Einheit »cm« für einen DIN-A3-Plotter mit einem 36 mal 26 Zentimeter großen Zeichenbereich nicht verändert werden müssen (der Nullpunkt des Koordinatensystems liegt in der Mitte). Trotz »Hochauflösung« ist die Darstellung auf dem Bildschirm im Vergleich zum Plotter recht grob — den 640 Bildpunkten in X-Richtung steht eine Auflösung von 3600 Schritten (10 pro Millimeter) auf einem gewöhnlichen Plotter gegenüber. Die Anpassung dieser Werte an die Anzahl Schritte (das heißt adressierbare Punkte) im Darstellungsbereich der Grafikkarte wird durch Multiplikation mit den Faktoren »cxstepfactor« und »cystepfactor« vorgenommen. Durch die unterschiedlichen Beträge der beiden Konstanten wird auch die etwas unterschiedliche Auflösung in X- und Y-Richtung ausgeglichen. Ein Offset (Verschiebung) für den Nullpunkt ist nicht nötig; dies und das Umdrehen der Y-Achse (siehe weiter unten) erledigt die Assemblerroutine. Die Parameter, welche das Gerät »Bildschirm«, das durch unseren Treiber angesprochen wird, kennzeichnen, werden durch eine Anzahl von Funktionen einem übergeordneten Grafiksystem übermittelt (in einfacheren Anwendungen kann auch darauf verzichtet werden).

Wesentlich sind die beiden Prozeduren »smoveto« und »sdrawto« (das »s« vor dem Namen steht für »screen«). Mit ihnen kann ein imaginärer Zeichenstift an einen neuen Anfangspunkt bewegt (smoveto) oder vom alten Anfangspunkt eine Linie zu einem neuen Zielpunkt (sdrawto) gezogen werden. Die Prozeduren »sdrawtext«, »sdrawreal« und »sdrawinteger« dienen dem Anschreiben von Texten auf dem Bildschirm. Sie sprechen in unserem Fall einfach das normale Video-RAM des Computers an, wobei sie einen Schriftzug in der Zeile und Spalte starten lassen, die der zuvor durch »smoveto« angefahrenen Position am nächsten liegen. Die Möglichkeiten der Schriftausgabe sind daher auf die übliche 80 mal 24-Zeichendarstellung eines Bürocomputers beschränkt.

Die Prozedur »sinitdevice« dient dazu, den Bildschirm zu löschen (weiß oder schwarz) oder das Negativ der Darstellung zu bilden.

Der Bildschirmtreiber ist so ausgelegt, daß er auch Farbbildschirme ansteuern könnte, wovon in unserem Beispiel natürlich kein Gebrauch gemacht werden kann (Prozedur »ssetcolor«).

In den Zeilen 80 und 81 von Listing 3 werden die beiden externen Prozeduren »screenline« und »screeninit« angefordert. Es handelt sich dabei um die schon erwähnten Assemblerroutinen, die nach dem compilieren mit dem Linker eingebunden werden müssen. Sie sind deshalb in Maschinensprache geschrieben, um möglichst schnell zu sein. Bei der Formulierung wurde jedoch in erster Linie Wert auf eine klare Gliederung und Lesbarkeit des Codes gelegt, ein Z80-Freak findet sicher noch Möglichkeiten, den Code geschwindigkeitsmäßig zu optimieren.

Um den Vektorgenerator in Listing 1 (letzte Ausgabe) zu verstehen, wollen wir uns zuerst den zugrundeliegenden Algorithmus überlegen. Sein Auftrag besteht darin, zwischen zwei Punkten eine gerade Linie zu ziehen. Diese Punkte werden durch die Wertepaare (xa, ya) und (xe, ye) bestimmt (siehe Zeile 81 in Listing 3). Da die Punkte völlig beliebig liegen können, kann die geforderte Linie jeden beliebigen Winkel annehmen — also nicht etwa nur waagrecht und senkrecht oder die Vielfachen von 45 Grad. Diese Richtungen wären leicht zu erzeugen — und sie können auch mit hardwaremäßiger Unterstützung der Karte erzeugt werden. Das Raster des Bildschirms läßt nur waagrechte durchgezogene Linien zu. Schon bei senkrechten Linien erkennt man meistens feine Pünktchen, während schräge Linien in mehr oder weniger regelmäßige Treppen aufgelöst werden.

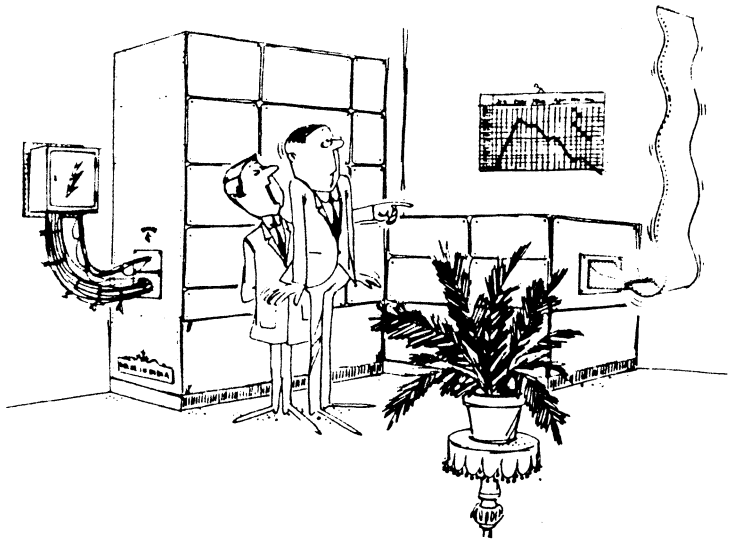
Offenbar erhält man eine Linie, die von links unten im Winkel von 45 Grad nach rechts oben verläuft, durch folgendes Vorgehen:

Man geht einen Schritt (einen Bildpunkt) in X-Richtung und einen Schritt in Y-Richtung, zeichnet einen Punkt und wiederholt dieses Verfahren. Um eine flachere Linie zu erhalten, braucht man bloß die Schritte in X-Richtung seltener auszuführen. Wenn man beispielsweise einen Schritt in Y-Richtung geht, einen Punkt zeichnet, wieder einen Schritt in X-Richtung, einen Punkt zeichnet und erst beim dritten Schritt in X-Richtung auch einen in Y-Richtung macht, so erhält man eine flache Treppe, die der Annäherung einer flachen Linie entspricht. Man kann sagen: Bei einer 45-Grad-Linie entsteht bei jedem Schritt in X-Richtung eine »Schuld« von einem Schritt in Y-Richtung. Bei der flacheren Linie beträgt die »Schuld« in Y-Richtung bei jedem X-Schritt ⅓-Schritte und braucht daher nur alle drei X-Schritte abgetragen zu werden. Nehmen wir an, wir würden vor dem Start des Algorithmus den Betrag der Schuld berechnen — dann könnten wir so verfahren, daß wir bei jedem Schritt in X-Richtung den Schuldbetrag zu einer Schuldsumme hinzuaddieren. Sobald die Schuldsumme den Wert 1 erreicht oder überschreitet, müssen wir einen Schritt in Y-Richtung ausführen — dafür dürfen wir von der Schuldsumme den Betrag 1 abziehen. Dabei gibt es offenbar ein Problem: Wenn die Linie steiler als 45 Grad ist, so wird die »Schuld« bei jedem X-Schritt größer als 1 Schritt in Y-Richtung — das aber kann man leicht dadurch umgehen, daß man zu Beginn entscheidet, ob die Differenz der Start- und Endkoordinaten auf der X-Achse oder auf der Y-Achse größer ist. Falls sie in Y-Richtung größer ist, vertauscht man einfach die Rollen von X und Y und merkt sich beim Fortschreiten in Y-Richtung die Schuld in X-Richtung.

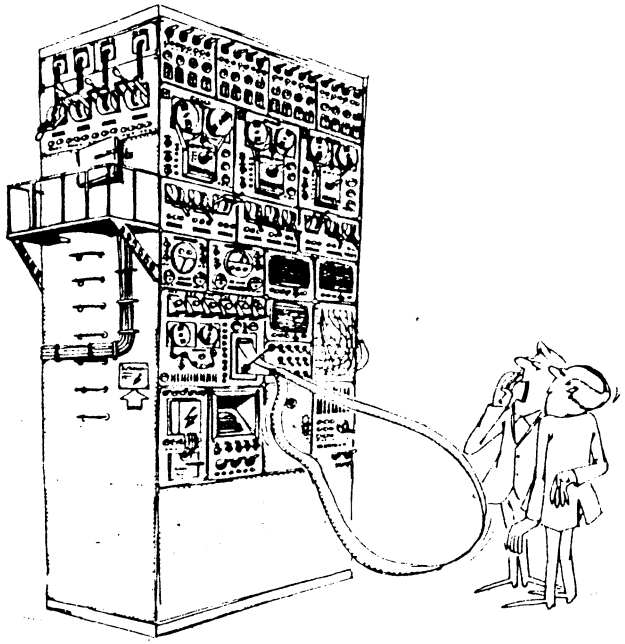
Um zu wissen, wann der Zielpunkt erreicht ist, zählen wir die ausgeführten Iterationen (Wiederholungen des Verfahrens). Dazu wird ein Zähler auf den Betrag von DELTAX (oder DELTAY) gesetzt und bei jeder Iteration um eins heruntergezählt. Sobald der Zähler den Wert Null erreicht hat, brechen wir, nach Zeichnen des Endpunktes, das Verfahren ab.

Offensichtlich treten bei diesem Verfahren, wie es eben geschildert wurde, Bruchzahlen für die Schuldbeträge auf. Das erschwert die Lösung durch eine einfache Assemblerroutine und mindert die geforderte Effizienz. Auch hier gibt es eine einfache Abhilfe: Wir betrachten einfach die gesamte Differenz in Y-Richtung als Schuldbetrag. Als Schwelle für das Einlesen der Schuld nehmen wir die gesamte Differenz in X-Richtung (bei einer Linie, die mehr in X-Richtung geneigt ist). Außerdem addieren wir zu Beginn den halben Schuldbetrag in die Schuldsumme, damit die erste Treppenstufe nur etwa die halbe Länge einer mittleren Treppenstufe beträgt. Das verleiht der approximierten Linie ein ausgewogeneres Aussehen. Wir kommen so mit einfachen Additions- und Subtraktionsbefehlen zur allgemeinen Lösung eines Problems, das nach der Schulmathematik aufwendige Winkelfunktionen erfordern würde (die erwähnte Halbierung des Schuldbetrages, die wir zu Anfang ausführen, erreichen wir durch einfaches binäres Rechtsschieben des fraglichen Differenzbetrages).

Wenden wir uns nun dem Listing 1 zu. Am Anfang findet man die Vereinbarung einiger Konstanten darunter den jeweiligen Offset zur X- beziehungsweise Y-Koordinate, um den Nullpunkt in die Bildmitte zu verschieben. Die Konstanten INITGRAPH und EXITGRAPH dienen



»Wer hatte eigentlich die famose Idee, den Computer aus Indien zu importieren?«



»Um Himmels willen! Jetzt führt er schon Selbstgespräche!«

zum Ein- beziehungsweise Ausschalten der Grafikkarte. Weiterhin wird eine Anzahl von Speicheradressen durch Define-Word-Anweisungen belegt, die uns als Platzhalter für Variable dienen. Dies ist der Start- und Zielpunkt der Linie, die Schreibfarbe (keine, Weiß, Löschen, Invertieren oder Radar), der laufende Wert für X und Y, Merker, die uns sagen, ob wir die Schritte in positiver oder negativer Richtung ausführen müssen (XINC und YINC) sowie die Differenzbeträge DELTAX und DELTAY. Den Sinn der Variablen GOTIT erkläre ich später im Zusammenhang mit der »Schreibfarbe« RADAR. ERROR bezeichnet unseren Schuldbetrag, ZAEHLER zählt die Anzahl ausgeführter Schritte.

Zunächst müssen die Parameter, welche das Pascal-System beim Aufruf der Routine auf dem Stack deponiert hat, von diesem entnommen und aufbewahrt werden. Diese Vorgänge sind im Assemblerlisting durch Kommentare erläutert. Da auf der Grafikkarte die Y-Adressen von oben nach unten wachsen, ein mathematisches Koordinatensystem aber üblicherweise von unten nach oben positivere Werte annimmt, werden die Y-Beträge zwischen dem Holen vom Stack und dem Abspeichern invertiert (Vorzeichenwechsel). Außerdem wird zu den X- und Y-Beträgen der jeweilige Nullpunkt-Offset addiert. Aus bestimmten programmier-technischen Gründen ist die Routine als Funktion geschrieben, daher muß zuletzt noch ein Leerwort vom Stack genommen werden, das den Platz des Funktionsresultates belegt.

Anschließend werden die Differenzbeträge DELTAX und DELTAY berechnet. Da unsere Linie in alle erdenklichen Richtungen laufen kann, können die notwendigen Schritte in X- und Y-Richtung sowohl Plus 1 als auch Minus 1 betragen. Dies wird gleich bei der Berechnung der Differenz festgehalten, indem in die Variablen XINC und YINC jeweils die Zahl 1 oder —1 eingetragen wird.

Bei einer Berechnung von DELTAX wird, falls dieser Wert negativ ist, XSTART und XEND vertauscht. Wir erreichen dadurch, daß alle Linien von links nach rechts geschrieben werden. Damit soll sichergestellt werden, daß Linien, sofern Start- und Endpunkt identisch sind, sich genau decken — besser gesagt, daß ihre treppenförmigen Annäherungen deckungsgleich ausfallen. Das soll sichergestellt werden, damit man eine zuvor geschriebene Linie durch Überschreiben mit der Untergrundfarbe auch tatsächlich wieder auslöschen kann. Wir haben gesagt, daß wir die erste Treppenstufe nach einer mittleren Länge enden lassen (durch Voraufaddierung des halben Schuldbetrages). Es ist aber nicht vorherzusehen, ob die letzte Treppenstufe der Linie auch gerade die halbe mittlere Länge hat oder vielleicht länger oder kürzer ist als die erste. Würden wir daher

```
 1  unit scrndrive;
 2
 3  (* 19. 12. 83 *)
 4
 5  (* Treiber fuer den TRS-80 MODEL II oder 16 mit Graphikkarte unter BGS-M 2.2 *)
 6  (* - Version mit Vektorgenerator in ASSEMBLER *)
 7
 8  interface
 9
10  function swmaxx: real;          (* Maximale X-Koordinate in cm (Window) *)
11  function swmaxy: real;          (* Maximale Y-Koordinate in cm (Window) *)
12  function swminx: real;          (* Minimale X-Koordinate in cm (Window) *)
13  function swminy: real;          (* Minimale Y-Koordinate in cm (Window) *)
14
15  function svmaxx: real;          (* Maximale X-Koordinate in DOTS (Viewport) *)
16  function svmaxy: real;          (* Maximale Y-Koordinate in DOTS (Viewport) *)
17  function svminx: real;          (* Minimale X-Koordinate in DOTS (Viewport) *)
18  function svminy: real;          (* Minimale Y-Koordinate in DOTS (Viewport) *)
19  function sxoffset: real;        (* Null-Punkt-Offset X-Achse *)
20  function syoffset: real;        (* Null-Punkt-Offset Y-Achse *)
21  function sxpositiv: boolean;    (* X-Achse positiv *)
22  function sypositiv: boolean;    (* Y-Achse positiv *)
23  function shascolor: boolean;    (* Verwaltet Zeichenfarbe selbst *)
24  function shasfile: boolean;     (* Memory-Mapped *)
25  function sopenport: boolean;    (* kein Default-Geraet *)
26  function sdevicestatus: integer; (* Geraetestatus abfragen *)
27  procedure sfilename (var filename: string); (* Dummy-Filename *)
28
29  procedure smoveto (x, y: real);
30  procedure sdrawto (x, y, scale: real);
31  procedure sdrawtext (s: string; ascale, richtung: real);
32  procedure sdrawinteger (i, stellen: integer; ascale, richtung: real);
33  procedure sdrawreal (r: real; vkstell, nkstell: integer;
34                       ascale, richtung: real);
35  procedure ssetcolor (color: integer);
36  procedure sinitdevice (stat: integer);
37  procedure sinitfile (name: string);
38  procedure sclosefile;
39
40  implementation
41
42  const   cxdiv =      8;
43          cydiv =     10;
44          clines =    23;
45
46          csxoffset =  40;    (* Versatz fuer Beschriftung *)
47          csyoffset =  12;
```

```
48
49        cwmaxx =      18.0;   (* Maximale X-Koordinate in cm (Window) *)
50        cwmaxy =      13.0;   (* Maximale Y-Koordinate in cm (Window) *)
51        cwminx =     -18.0;   (* Minimale X-Koordinate in cm (Window) *)
52        cwminy =     -13.0;   (* Minimale Y-Koordinate in cm (Window) *)
53
54        cvmaxx =      18.0;   (* Maximale X-Koordinate in DOTS (Viewport) *)
55        cvmaxy =      13.0;   (* Maximale Y-Koordinate in DOTS (Viewport) *)
56        cvminx =     -18.0;   (* Minimale X-Koordinate in DOTS (Viewport) *)
57        cvminy =     -13.0;   (* Minimale Y-Koordinate in DOTS (Viewport) *)
58
59        cxoffset =     0.0;       (* Null-Punkt-Offset X-Achse *)
60        cyoffset =     0.0;       (* Null-Punkt-Offset Y-Achse *)
61
62        cxstepfactor = 17.77778;       (* x * cxstepfactor = XKoordinate *)
63        cystepfactor =  9.230769;      (* y * cystepfactor = YKoordinate *)
64
65        cxpositiv =     true;   (* X-Achse positiv *)
66        cypositiv =     true;   (* Y-Achse positiv *)
67        chascolor =     true;   (* Verwaltet Zeichenfarbe selbst *)
68        chasfile =      false;  (* Memory-Mapped *)
69        copenport =     true;   (* kein Default-Geraet *)
70        cfilenaae =     'rewout:';      (* Dummy-Filenaae *)
71
72  type  tcolor =    (none, white, black, reverse, radar);
73
74  var   (screenfile:     nicht definiert;)
75        penstatus:      tcolor;
76        dummy, status:  integer;
77        newx, newy:     integer;
78        oldx, oldy:     integer;
79
80  procedure screeninit (col: tcolor); external;
81  function screenline (xa, ya, xe, ye: integer; i: integer): integer;
82                       external;
83
84  function swmaxx; begin swmaxx := cwmaxx end;
85  function swmaxy; begin swmaxy := cwmaxy end;
86  function swminx; begin swminx := cwminx end;
87  function swminy; begin swminy := cwminy end;
88
89  function svmaxx; begin svmaxx := cvmaxx end;
90  function svmaxy; begin svmaxy := cvmaxy end;
91  function svminx; begin svminx := cvminx end;
92  function svminy; begin svminy := cvminy end;
93  function sxoffset; begin sxoffset := cxoffset end;
94  function syoffset; begin syoffset := cyoffset end;
95  function sxpositiv; begin sxpositiv := cxpositiv end;
96  function sypositiv; begin sypositiv := cypositiv end;
97  function shascolor; begin shascolor := chascolor end;
98  function shasfile; begin shasfile := chasfile end;
99  function sopenport; begin sopenport := copenport end;
100 function sdevicestatus; begin sdevicestatus := status end;
101 procedure sfilenaae; begin filenaae := cfilenaae end;
102
103 procedure smoveto;
104 begin
105   newx := round (x * cxstepfactor); newy := round (y * cystepfactor);
106   dummy := screenline (oldx, oldy, newx, newy, ord (none));
107   oldx := newx; oldy := newy;
108 end (* smoveto *);
109
110 procedure sdrawto;
111 begin
112   newx := round (x * cxstepfactor); newy := round (y * cystepfactor);
113   dummy := screenline (oldx, oldy, newx, newy, ord (penstatus));
114   oldx := newx; oldy := newy;
115 end (* sdrawto *);
116
117 procedure sdrawtext;
118 var     x, y:   integer;
119 begin
120   if (ascale >= 1 (*cydiv*)) and (round (richtung / 90) = 0) then begin
121     x := ((oldx div cxdiv) + csxoffset);
122     if cypositiv then y := (clines - ((oldy div cydiv) + csyoffset))
123     else y := ((oldy div cydiv) + csyoffset);
124     gotoxy (x, y);
125     write (s);
126   end (* if darstellbar *);
127 end (* sdrawtext *);
128
129 procedure sdrawinteger;
130 var     x, y:   integer;
131 begin
132   if (ascale >= 1 (*cydiv*)) and (round (richtung / 90) = 0) then begin
133     x := ((oldx div cxdiv) + csxoffset);
134     if cypositiv then y := (clines - ((oldy div cydiv) + csyoffset))
135     else y := ((oldy div cydiv) + csyoffset);
136     gotoxy (x, y);
137     write (i:stellen);
138   end (* if darstellbar *);
139 end (* sdrawinteger *);
140
141 procedure sdrawreal;
142 var     x, y:   integer;
143 begin
144   if (ascale >= 1 (*cydiv*)) and (round (richtung / 90) = 0) then begin
145     x := ((oldx div cxdiv) + csxoffset);
146     if cypositiv then y := (clines - ((oldy div cydiv) + csyoffset))
147     else y := ((oldy div cydiv) + csyoffset);
148     gotoxy (x, y);
149     write (r:wkstell:nkstell);
150   end (* if darstellbar *);
151 end (* sdrawreal *);
152
153 procedure ssetcolor;
154 begin
155   if color = 0 then penstatus := none
156   else if color in [2..15] then penstatus := white
157   else if color = 1 then penstatus := black;
158 end (* ssetcolor *);
159
160 procedure sinitdevice;
161 begin
162   status := stat;
163   case status of
164     0:  screeninit (black);     (* Graphikkarte einschalten *)
165     1:  screeninit (white);
166     2:  screeninit (reverse);
167   end (* case *);
168 end (* sinitdevice *);
169
170 procedure sinitfile;
171 begin
172   (* Dummy *)
173 end (* sinitfile *);
174
175 procedure sclosefile;
176 begin
177   (* Dummy *)
178 end (* sclosefile *);
179
180 begin (* Unit-Initialisierung *)
181   penstatus := white;
182   status := 0;
183   sinitdevice (status);
184   ***;
185   screeninit (none);
186 end.
```

eine Linie von links unten nach rechts oben schreiben und von rechts oben nach links unten löschen, dann würden sich sehr wahrscheinlich die Punkte der Löschlinie nicht genau mit den Punkten der Schreiblinie decken, und es blieben »versprengte« Punkte auf dem Bildschirm stehen. Dadurch, daß Linien immer von links nach rechts geschrieben werden, ist sichergestellt, daß gleiche Strecken immer in der gleichen Richtung durchlaufen werden.

Von DELTAY wird der Absolutbetrag gebildet, um es ebenfalls positiv zu machen, es wird aber in YINC festgehalten, ob die Approximationsschritte in positiver oder negativer Richtung auszuführen sind. Schließlich wird entschieden, ob der Betrag von DELTAX oder der von DELTAY größer ist und je nachdem eine der beiden Routinen DOFORX (Linie stärker zur X-Achse parallel) oder DOFORY aufgerufen. Die auskommentierten JP- und RETURN-Anweisungen mit den Vermerken BREAK 1 und so weiter sind Debug-Anweisungen, die in der ausgetesteten Routine entfallen können.

Die beiden Unterroutinen DOFORX und DOFORY sollten aus der Schilderung des Algorithmus und den Kommentaren im Assemblerlisting heraus zu verstehen sein. Jedesmal, wenn ein Bildpunkt zu schreiben ist, wird die Routine DRAWDOT aufgerufen. Sie entscheidet anhand der Schreibfarbe, ob der Punkt weißgeschaltet, schwarzgeschaltet oder invertiert werden soll. Falls die Schreibfarbe 0 ist (keine), braucht natürlich nichts zu geschehen, es folgt ein RETurn-Befehl. Der Wert 4 (RADAR) soll zunächst außer acht gelassen werden. In diesem Fall wird der Label CALLPOINT erreicht, bei dem das Aufrufen der Routine POINT vorbereitet wird. Dazu werden dieser die X- und Y-Koordinate des gegenwärtig erreichten Bildpunktes auf dem Stack übergeben, sowie die Schreibfarbe. POINT gibt den Stack gesäubert zurück, anstelle des zunächst geladenen Leerwortes hinterläßt sie ein Funktionsresultat, das nur bei der Farbe RADAR Bedeutung hat.

Da POINT eine hardwareabhängige Routine ist, wurde sie getrennt geschrieben, sie ist aber in Listing 1 einkopiert und steht an dessen Ende. Wenn man diesen Teil entsprechend umschreibt, kann der Vektorgenerator auch für abweichende Hardware benutzt werden.

Um die Arbeitsweise von POINT zu verstehen, müssen wir uns näher mit der Architektur der Grafikkarte beschäftigen. Sie wird über vier Ports (Ein- und Ausgabekanäle) angesprochen. Die ersten beiden Ports gestatten das Laden je eines X- und Y-Adreßregisters. Die beiden Register adressieren zusammen ein Byte des Grafikspeichers. Dieses Byte kann über den dritten Port geschrieben oder gelesen werden. Der vierte Port dient zum Steuern bestimmter spezieller Arbeitsweisen der Karte.

Wenn ein Byte in den Grafikspeicher geschrieben wird, so sind davon immer acht nebeneinanderliegende Bildpunkte betroffen. Es kann aber sein, daß eine Linie nur durch einen dieser Punkte führt. Dann dürfen die übrigen sieben Bit des Speicherbytes nicht verändert werden. Daher muß die Routine POINT zunächst einmal die Adresse des Speicherbytes berechnen, in dem sich der zu behandelnde Bildpunkt befindet und dieses lesen. Die ersten Bildpunkte links oben gehören zu dem Byte mit der Adresse (0, 0), die letzten rechts unten zu dem Byte (79, 239). Nun muß die Bitadresse des Punktes in dem Byte bestimmt werden, dieses Bit gesetzt, rückgesetzt oder invertiert werden und schließlich das so modifizierte Byte in den Grafikspeicher zurückgeschrieben werden.

Die Y-Adresse des Bytes kann folglich direkt in das zugehörige Register der Grafikkarte geschrieben werden. Aus der X-Adresse sind dagegen zwei unterschiedliche Werte zu berechnen: zum einen die X-Adresse des Speicherbytes, zum anderen die Bit-Adresse des Bildpunktes in diesem Byte. Betrachten wir die X-Koordinate als Zahl, so ist die Bitadresse offenbar X mod 8, das heißt der Rest, der bei der ganzzahligen Division von X durch 8 bleibt. Die Byteadresse erhält man, indem man das ganzzahlige Ergebnis der Division X durch 8 nimmt. Beide Werte können in einem Arbeitsgang gewonnen werden — die Bitadresse steht bereits in den niederwertigsten drei Bit des Doppelregisters DE, in welchem POINT den Wert X abgelegt hat. Schiebt man DE dreimal nach rechts und bringt den Überlauf aus dem CARRY-Flag durch Rotieren in das Register C, so befindet sich in C die abgetrennte Bitadresse (die allerdings noch vollständig durch C nach rechts geschoben werden muß), in DE aber, da dreimal Rechtsschieben einer Division durch 8 entspricht, bereits die fertige X-Byteadresse. Wir haben also nichts weiter zu tun, als C noch fünfmal nach rechts zu schieben.

Die X-Adresse, die in diesem Fall nicht größer als 255 sein kann (tatsächlich höchstens 79 Dezimal) und folglich in ein 8-Bit-Register paßt, kann dem Register E entnommen und direkt in den X-Adreßport geschrieben werden.

Um die Bit-Adresse in Register C in die entsprechende Bitposition umzuwandeln, benutzen wir einfach eine Tabelle, die zu Anfang durch eine Liste von Konstanten bereitgestellt wurde (ab dem Label MUSTER). Wir laden HL mit der Startadresse der Tabelle (der Adresse MUSTER) und addieren die Bitadresse aus (B)C zu HL, anschließend laden wir C mit dem Inhalt der Adresse, die jetzt in HL steht — und haben damit im Register C das gewünschte Bit stehen. Jetzt kann das Video-Byte (Grafikspeicher) gelesen werden. Je nach Schreibfarbe wird das Bit in Register C zum Videobyte ODERiert (weiß), sein Komplement (Maske) UNDiert (schwarz) oder die EXOR-Funktion ausgeführt (invertieren).

Was hat es nun mit der geheimnisvollen »Farbe« RADAR auf sich? Sie werden schon bemerkt haben, daß wir ja den Inhalt des Grafik-RAMs lesen können. In diesem Zusammenhang kann man feststellen, ob auf einer gedachten Linie, die man sich als Radarstrahl vorstellt, irgendein Objekt vorhanden ist — ob also auf dieser Linie ein Bit gesetzt ist. Wenn nun die Prozedur POINT bei der Farbe »4« die Bits im Grafikspeicher nicht verändert, sondern nur testet und eine Null oder Eins zurückgibt, je nachdem, ob das Bit gesetzt ist oder nicht, so kann man anhand eines Iterationszählers im Vektorgenerator feststellen, wo sich — ausgehend vom Startpunkt der Linie — das nächstgelegene Objekt befindet — wie weit es also entfernt ist. Der Vektorgenerator tastet sich also so lange die gedachte Linie (den Radar-Strahl) entlang, bis er ein Objekt (ein gesetztes Bit) entdeckt — dieses Ereignis wird in der Variablen GOTIT durch Setzen auf 1 festgehalten. Allerdings hat das Sourcelisting in dieser Hinsicht einen kleinen Fehler — bei der Schreibfarbe »Radar« müßte das Vertauschen von Start und Ziel der X-Koordinaten unterbunden werden, weil sonst eine falsche Entfernung herauskommt. Nach Beseitigung dieser Schwäche ist diese Funktion beispielsweise für Videospiele recht gut zu gebrauchen.

Der Rat des Computers? – Beten.

Zu unserem kleinen Satz von Grafikroutinen gehört noch die Routine »Screennit« in Listing 2 (letztes Heft), die das Löschen (Weiß- oder Schwarzschreiben) sowie Invertieren (Bilden des Negativs) des gesamten Grafikbildschirms erlaubt.

Ich habe sie ebenfalls abgedruckt, möchte sie aber aus Platzgründen nicht ausführlich erläutern. Der interessierte Leser wird sie zusammen mit der Dokumentation der Grafikkarte sicher leicht verstehen.

Selbstverständlich sind zum Verständnis der hier abgedruckten Programme Erfahrungen in der Assemblerprogrammierung erforderlich, besonders dann, wenn sie an eine abweichende Hardware angepaßt werden sollen. Im Rahmen dieses Artikels kann keine Einführung in dieses Gebiet gegeben werden. Die abgedruckten Routinen wurden auf einem TRS-80 Modell 16 (in der Z80-Betriebsart) entwickelt und erprobt, das sich im Frühjahr 83 zum Test in der Redaktion befand; sie funktionierten auch auf dem Modell II des Autors ohne jede Änderung.

Die Karte ist einerseits die gegebene Erweiterung für TRS-80-Computer, um eine hochauflösende Grafik zu erzielen, wobei vielleicht der Preis (siehe unten) angesichts der mäßigen Hardware-Intelligenz etwas hoch erscheinen mag; andererseits bot sie die Gelegenheit, einmal die Grundlagen der Computergrafik an einem praktischen Beispiel darzustellen. Wir hatten zwei Karten im Test, die erste befand sich vor einigen Monaten in einem Testgerät, die Treibersoftware wurde bereits damals entwickelt. Anläßlich dieses Artikels erhielten wir ein anderes Exemplar, das in einem Modell II eingebaut wurde. In beiden Fällen funktionierte die Karte einwandfrei, ohne die sonstigen Funktionen der Testcomputer in irgendeiner Weise zu beeinflussen. Preise: Für Modell 2, 12, 16 kostet die Karte 1595 Mark inkl. Mehrwertsteuer. Eine ähnliche Karte mit gleicher Auflösung, die sich vom Benutzer aus gesehen gleich verhält, gibt es für Modelle 3 und 4 ab Februar '84 ab 768 Mark.

(le)

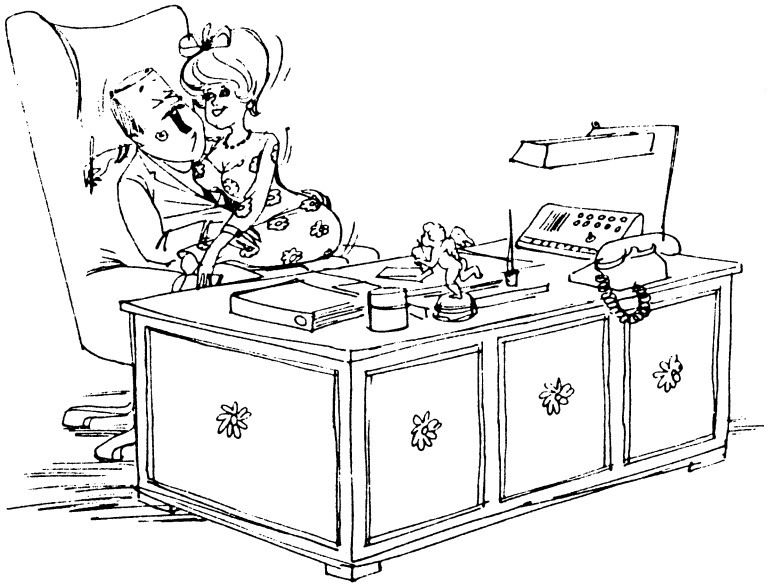
Literaturhinweis: [1] William M. Newman/Robert F. Sproull, Principles of Interactive Computer Graphics, Second Edition, McGraw-Hill.

Z80-Programmtricks in Assembler

Wer den richtigen Assembler besitzt, tippt wesentlich weniger, als der erzeugte Code vermuten läßt. »LD HL,(HL)« zum Beispiel ist da ein gültiger Befehl. Dahinter stecken eingebaute Makros, die dann die richtige Codesequenz generieren. Löst man sie auf, findet man die optimale Lösung für viele Standard-Probleme.

Andere Wege mögen auch nach Rom führen führen, und alle Assembler kann der Autor nicht abklopfen, weshalb in diesem Beitrag ausschließlich die Lösungen aus ALASM (Teil des »Assembler Development System« von Tandy) vorgestellt werden. Je nach Ausrüstung können Sie mit den hier vorgestellten Lösungen Ihre eigene Makro-Lib erweitern, die Routinen als Textfiles einziehen oder schlicht diese CP-Ausgabe als Nachschlagewerk neben den Computer legen. Beginnen wir mit den LD-Befehlen.

Schon »LD HL,BC« mag der Z80 nicht, die Folge »LD H,B:LD L,C« ist nötig. Bei den untrennbaren Pärchen AF, IX und IY schreibt man dann für zum Beispiel »LD HL,IX« die Folge »PUSH IX:POP HL«. Schon lohnender ist das eingangs erwähnte »LD HL,(HL)«, das sehr oft gebraucht wird, und zwar so:



»Haben sie schon mal darüber nachgedacht, weshalb ich Sie noch nicht durch einen Computer ersetzt habe?«

```
PUSH    AF
LD      A,(HL)
INC     HL
LD      H,(HL)
LD      L,A
POP     AF
```

Zählen Sie nach (Bytes und Taktzyklen), der oft gebrauchte Weg via IX kostet mehr.

Und gleich die Umkehr, für »LD (HL),HL« ergibt sich:

```
PUSH    AF
LD      A,H
LD      (HL),L
INC     HL
LD      (HL),A
POP     AF
```

Mit (BC),BC; (BC),DE; (DE),BC und (DE),DE wird's etwas tricky, das geht so:

```
PUSH    1. Operand
EX      (SP),HL
LD      (HL),L-Byte 2. Operand
INC     HL
LD      (HL),H-Byte 2. Operand
EX      (SP),HL
POP     1. Operand
```

Zum Beispiel LD (BC),BC:

```
PUSH    BC
EX      (SP),HL
LD      (HL),C
INC     HL
LD      (HL),B
EX      (SP),HL
POP     BC
```

Hiervon nun wieder die Umkehr geht einfacher, zum Beispiel LD BC,(BC):

```
PUSH    BC
EX      (SP),HL
LD      C,(HL)
INC     HL
LD      B,(HL)
POP     HL
```

Blieben noch so nette Schreibweisen wie »LD (IX+dd),HL. Ganz einfach:

```
LD      (IX+dd),L
INC     IX
LD      (IX+dd),H
```

und ganz trivial: aus LD (IX+dd) wird LD A,(IX+dd):LD (IY+dd),A.

Wie auch immer: Schwierige Probleme waren das alles nicht, nur wenn ein Assembler solche Schreibweisen erlaubt, trägt das zur Lesbarkeit bei, spart einiges an Tipperei, und was das Wichtigste ist: Tippfehlern ist die Chance genommen. Sie wissen, was ein so verwechseltes B mit C für seltsame Fehler und stundenlange Debug-Sessions zur Folge haben kann.

Zwei Makros sollten Sie immer parat haben, nämlich MOVD und MOVI (Move mit Dekrement beziehungsweise Inkrement). Im ALDS gibt es dafür diverse Variationen über »LDIR mit Zubehör«, womit ich Sie nicht langweilen will. Schauen wir uns nur eine an:

```
MOVI nn1,(nn2):   LD      DE,nn1
                  LD      HL,nn2
                  LD      C,(HL)
                  LD      B,0
                  INC     HL
                  LD      A,B
                  OR      C
                  JR      Z,EXIT
                  LDIR
EXIT              EQU     $
```


Ganz im vorgenannten Sinne sollte man auch CMPD und CMPI kreieren. Hier die entsprechende Abart des vorhergehenden Beispiels:

```
CMPI nn1,(nn2):   LD      DE,nn1
                  LD      HL,nn2
                  LD      C,(HL)
                  LD      B,0
                  INC     HL
LOOP              LD      A,B
                  OR      C
                  JR      Z,EXIT
                  LD      A,(DE)
                  CP      (HL)
                  JR      NZ,EXIT
                  LDI
                  JR      LOOP
EXIT              EQU     $
```

Da unsere Subroutinen immer Werte in Registern verlangen, in denen sie beim Aufruf gerade nicht sind, ist Tauschen nicht zu vermeiden. Kombinationen gibt es viele, fünf Lösungen erschlagen alle.

1.) EX AF,BC; EX AF,DE; BC AF,DE an einem Beispiel:

```
EX AF,BC:   PUSH    AF
            PUSH    BC
            POP     AF
            POP     BC
```

2.) EX xx, yy wobei gilt: xx=AF, BC oder DE

```
            yy=IX       oder IY
                        das wird
            PUSH        xx
            EX          (SP),yy
            POP         xx
```

```
3.) EX IX,IY:   PUSH    IX
                EX      (SP),IY
                POP     IX
```

```
4.) EX (SP),BC:   EX      (SP),HL
                  PUSH    BC
                  PUSH    HL
                  POP     BC
                  EX      (SP),HL
```

Mit AF anstelle BC geht das auch

```
5.) EX (SP),DE:   EX      (SP), HL
                  EX      DE,HL
                  EX      (SP),HL
```

Zum Schluß noch ein weiterer Trick:

```
PUSH    AF
EX      (SP),HL
LD      A,L
POP     HL
```

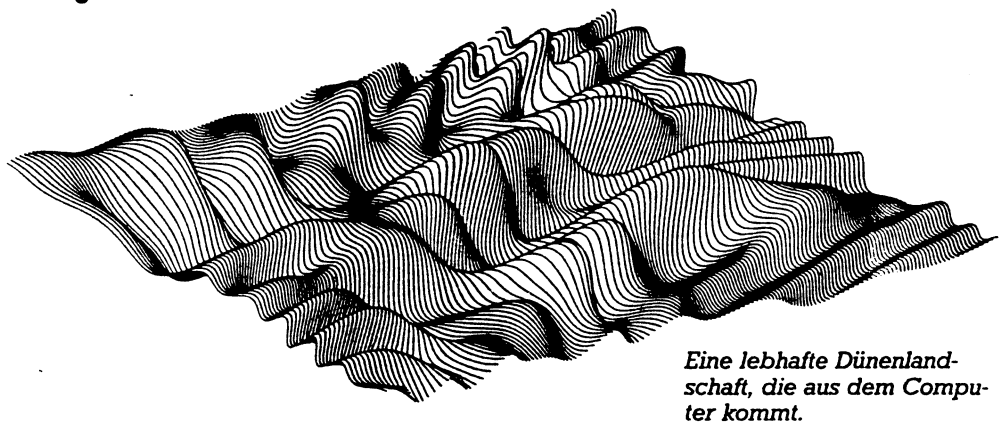
Diese Befehle realisieren schlicht »LD A,F«.

Sind Sie auch schon einmal mit einem Disassembler auf solche Sequenzen gestoßen und dann ins Grübeln geraten? So gesehen ist es doch richtig gemein, daß da einer nur tippt »LD (SP),BC«, und viele dann lange knobeln müssen! Jetzt schreiben wir auch so, und knobeln müssen nur noch die armen Kerlchen, die diese CP nicht gelesen haben.

(Peter Wollschläger;bo)

Traumlandschaften aus dem Computer

Endlich hat auch das deutsche Fernsehen sein vom Computer erzeugtes Markenzeichen. Und was dem Fernsehen recht ist, das ist dem Computerhobbyisten schon lange billig: Grafiken aus dem Mikrocomputer. Grafikprogramme können regelrechte Abenteuerspiele sein — sie schaffen bizarre Mondlandschaften oder sanft gewellte Dünen...



Eine lebhafte Dünenlandschaft, die aus dem Computer kommt.

Was da Anfang Oktober im ersten Programm an unwirklicher Wirklichkeit in die deutschen Wohnstuben flimmerte, ließ selbst dem hartgesottensten Computerignoranten den Atem stocken. Bei einer sanften Dünenlandschaft, die den Hintergrund eines bizarren Balletts aus sich verformenden farbigen Kugeln bildete, fühlte ich mich plötzlich an Computergrafiken erinnert, die ich zwei Tage zuvor mit einem experimentellen Programm angefertigt hatte: »Frequenzy Mountains« habe ich das Programm getauft — weil halt »Frequenzgebirge« so hausbacken klingt. Beides aber bedeutet dasselbe: Die computergenerierten Landschaften sind die dreidimensionale Darstellung mehr oder weniger komplizierter mathematischer Funktionen. Wenn man eine solche komplexe Funktion parametrisiert, kann man eine unerschöpfliche Fülle oft überraschender »Landscapes« erzeugen — es ist ein aufregendes Spiel, bei dem man nach Gefühl oder strategischen Überlegungen möglichst raffinierte Parameter eintippt und dann gebannt die Schreibspur auf dem Grafikbildschirm oder Plotter verfolgt: Was wird diesmal entstehen? Ein simpler Trick, der dafür sorgt, daß Täler, die hinter Gipfeln verborgen sind, unsichtbar bleiben, läßt die Landschaften dreidimensional erscheinen. Mehr darüber in der nächsten Ausgabe.

Das Programm, das diese Traumlandschaften erzeugt, ist mit 366 Zeilen (Turbo-Pascal) gar nicht besonders lang. Wir drucken es in diesem Heft ab, ohne es genau zu erläutern. Das soll in der nächsten Ausgabe geschehen — dort werden Sie auch eine Vielzahl traumhafter Farbfotos finden, die weitere Computerlandschaften zeigen, die von einem Farbgrafiksystem wiedergegeben wurden.

In der hier gezeigten Fassung steuert das Programm direkt den Plotter »MP 1000« an — die beiden Prozeduren »moveto« und »drawto« lassen sich aber leicht auch an andere Plotter anpassen. Ursprünglich wurde das Programm in UCSD-Pascal IV.13 geschrieben und lief auf dem Sage II mit einem universalen Grafiksystem (BGS/M). Für Computer persönlich wurde es in Turbo-Pascal umgeschrieben. Die geänderten Stellen sind in der UCSD-Version in Kommentarklammern im Text verblieben, so daß Sie das Programm auch in UCSD eintippen können.

Nach Beseitigung von zwei kleinen Fehlern akzeptierte der Turbo-Compiler auf einem Z80-Computer das Programm. Erstaunlicherweise erhielt ich aber während der Übersetzung die Meldung »Memory Overflow«. Das liegt wohl daran, daß neben dem Turbo-System die verschiedenen Tabellen des Programms keinen Platz mehr im Speicher hatten. Nach Einschalten der Option (Wahlmöglichkeit), eine COM-Datei zu erzeugen anstatt im Arbeitspeicher zu übersetzen, schaffte es der Compiler jedoch. Nun lief das Programm, blieb aber stecken, wenn es den Plotter ansprechen sollte — ich hatte einfach das Kabel vom Sage auf den Centronics-Port des Modell 12 umgesteckt. Wahrscheinlich fehlte ein Bereitschaftssignal, um das sich der Sage nicht kümmert. Da ich aber mit der Parallelschnittstelle dieses Leihgerätes vorher schon ein anderes Problem hatte, wollte ich mich nicht länger damit aufhalten und überspielte das Programm auf den IBM-PC. Dort lief es auf Anhieb — hier verwendete ich ein Watanabe-Kabel, das den Plotter an den parallelen Druckerport des IBM anschließt.

Es zeigte sich aber, daß der 8088 bei Real-Arithmetik alles andere als ein Wunder an Geschwindigkeit ist: Das Programm arbeitete recht langsam, trotz der Prozeduren »fastsin« und »fastcos«, die — anstelle einer aktuellen Berechnung von Sinus und Cosinus — auf eine Tabelle zugreifen.

Da diese Tabelle nur eine Auflösung von 1 Grad bietet, werden die Kurven auf dem Plotter etwas rauh. Schöner (aber noch langsamer) ist es, wenn man statt »fastsin« die Pascal-Funktionen »sin« (bzw. »cos«) direkt verwendet. Einen Geschwindigkeitsrausch erlebte ich dagegen, als ich das Programm mit dem Arithmetik-Prozessor 8087 laufen ließ — es gibt ja eine Turbo-Version, die ihn unterstützt — und zwar direkt mit den mathematischen Pascal-Funktionen (nicht über die Tabellen). Man muß dabei nur darauf achten, daß alle Winkel durch die Konstante »rad« (Zeile 52) geteilt werden — so wie in Zeile 91! Das deshalb, weil die Winkelfunktionen in der Einheit »Pi-Radian« rechnen.

Die gezeigte Abbildung entstand mit folgenden, über das Menü einzustellenden Parametern:

Gewichtung der Grundfrequenzen A, B und C: Je 0.5.
Grundfrequenzen D: 3.0, E: 5.0, F: 7.0.
Amplituden-Modulations-Frequenzen G: 1.0, H: 2.0, I: 3.0.
Amplituden-Gewichtung J: 1.0, K: 0.7, L: 0.5, M: 0.3, N: 0.5, O: 0.7.
Phasen der Grundfrequenzen P: 0.0, Q: 60.0, R: 120.0.
Phasen der Amplitudenmodulation S: 0.0, T: 30.0, U: 60.0
Punkte pro Kurve und Kurven X: 150, Y: 150.
Gesamtamplitude Z: 1.0.
FM-Frequenzen 1: 0.5, 2: 1.0, 3: 1.5.
FM-Amplituden 4: 0.3, 5: 0.5, 6: 0.7.
FM-Phasen 7: 0.0, 8: 90.0, 9: 270.0.
Bemerkung: FM bedeutet »Frequenz-Modulation«.

Zur Eingabe ist jeweils der Kennbuchstabe (A bis Z) beziehungsweise eine Kennziffer (1 bis 9) anzutippen, dann der entsprechende Zahlenwert und danach die RETURN-Taste.

Wenn Sie keine besonderen mathematischen Ambitionen haben, versuchen Sie lieber nicht, die hochwissenschaftlich wirkenden Einzelheiten des Programms zu verstehen — experimentieren Sie einfach, denn es ist ein Abenteuer-Programm! Ich kann Ihnen versprechen, daß Sie nicht enttäuscht sein werden.

Im nächsten Heft erläutern wir genau, wie das Programm funktioniert und was man alles damit machen kann. (le)

Dieses Programm erzeugt wildbewegte Meere, schroffe Gebirge oder sanfte Hügellandschaften.

```
program gebirge;

(* vereinfachte Version mit Plotteransteuerung *)
(* fuer Computer Persoenlich/Markt & Technik. *)
(* (C) 1984 by Johannes Leckebusch *)
(* Public domain *)
(* Entworfen in: UCSD-Pascal IV.13 unter B65/M *)
(* Version: Turbo-Pascal, stand alone *)
(* Plotter: Watanabe MP-1000 *)
(* Stand: 09. 10. 84 *)

(* UCSD IV.1x: *)
(* uses screenops; *)

const   cmaxslot =      511;
        ckurven =       45;
        cmaxpunkt =     45;
        cf1 =           4.00;
        cf2 =           3.00;
        cf3 =           6.00;
        cg1 =           1.0;
        cg2 =           1.0;
        cg3 =           1.0;
        cph1 =          0.0;
        cph2 =          0.0;
        cph3 =          0.0;
        caph1 =         0.0;
        caph2 =         0.0;
        caph3 =         0.0;
        cgroesse =      1.0;
        caf1 =          1.0;
        caf2 =          1.0;
        caf3 =          1.0;
        cag1x =         1.0;
        cag1y =         1.0;
        cag2x =         1.0;
        cag2y =         1.0;
        cag3x =         1.0;
        cag3y =         1.0;

        esc =           27;
```

Listing »Gebirge«

```
        cr =            13;

        cxoffset =      17.95;  (* Null-Punkt-Offset X-Achse *)
        cyoffset =      12.95;  (* Null-Punkt-Offset Y-Achse *)
        cstepfactor =   100.0;  (* Faktor cm/Plotschritt *)
        (* UCSD Name der Plotschnittstelle: *)
        (* cplotname =      'printer:'; *)
        (* CP/M: *)
        cplotname =     'LST:';

        rad =           57.29578;

var     (* Parametertabelle: *)
        xschritt:       real;
        yschritt:       real;
        kurven:         integer;
        punkte:         integer;
        f1, f2, f3,
        g1, g2, g3,
        ag1x, ag1y, ag2x, ag2y, ag3x, ag3y,
        af1, af2, af3,
        ph1, ph2, ph3, aph1, aph2, aph3,
        faf1, faf2, faf3,
        fag1, fag2, fag3,
        fph1, fph2, fph3,
        groesse:        real;
        bef:            char;
        plotfile:       text;
        sintab:         array [0..90] of real;


(* Die folgenden beiden Prozeduren steuern den Plotter "MP 1000" *)

procedure moveto (x, y: real);
begin
  writeln (plotfile, 'M', round ((x + cxoffset) * cstepfactor), ',',
           round ((y + cyoffset) * cstepfactor));
end (* moveto *);

procedure drawto (x, y: real);
begin
  writeln (plotfile, 'D', round ((x + cxoffset) * cstepfactor), ',',
           round ((y + cyoffset) * cstepfactor));
end (* drawto *);

procedure inisintab;
var     w:      integer;
begin
  for w := 0 to 90 do
    sintab [w] := sin (w/rad);
end (* inisintab *);

function fastsin (w: real): real;
var     winind: integer;
        sign:   real;
begin
  winind := round (w) mod 360;
  if winind > 180 then sign := -1.0
  else sign := 1.0;
  winind := winind mod 180;
  if winind > 90 then winind := 180 - winind;
  fastsin := sintab [winind] * sign;
end (* fastsin *);
```



```

function fastcos (w: real): real;
begin
  fastcos := fastsin (w + 90.0);
end (* fastcos *);

procedure dosinusgebirge;
(* Diese Prozedur zeichnet ein "Gebirgediagramm" einer Sinusfunktion *)

(* Datum: 28. 09. 84 *)

var     z:              real;
        xoffset, yoffset: real;
        zmax:           array [0..cmaxslot] of real;
        (* Dieses Array haelt fest, wo der hoechste Punkt der alten *)
        (* Kurven liegt, eine neue wird nur gezeichnet, wenn sie *)
        (* hoeher liegt. *)
        zmin:           array [0..cmaxslot] of real;
        (* Dieses Array haelt fest, wo der tiefste Punkt der alten *)
        (* Kurven liegt, eine neue wird nur gezeichnet, wenn sie *)
        (* tiefer liegt *)
        zold:           real;
        scaleversatz:   integer;
        (* Haelt den seitlichen Perspektivischen Versatz der Kurven fest *)
        kurve, punkt:   integer;
        hidden:         boolean;        (* war letzter Punkt sichtbar? *)
        up:             boolean;        (* war letzter Punkt oberhalb? *)

function f (x, y: real): real;
(* Darzustellende Funktion *)

var     k, l, m:        real;
        m1, m2, m3:     real;
        r:              real;

begin
  m1 := (1 - fastcos (fph1 + y * faf1 / kurven) * fag1);
  m2 := (1 - fastcos (fph2 + x * faf2 / punkte) * fag2);
  m3 := (1 - fastcos (fph3 + (x + y) * faf3 / (punkte + kurven)) * fag3);
  k := fastsin (ph1 + x * f1 * m1 / punkte) * g1 *
    (1 - fastcos (aph1 + x * af1 / punkte) * ag1x) *
    (1 - fastcos (aph1 + y * af1 / kurven) * ag1y);

  l := fastsin (ph2 + y * f2 * m2 / kurven) * g2 *
    (1 - fastcos (aph2 + x * af2 / punkte) * ag2x) *
    (1 - fastcos (aph2 + y * af2 / kurven) * ag2y);

  m := fastsin (ph3 + (x - y) * f3 * m3 / (punkte + kurven) (* * 2.0 *)) * g3 *
    (1 - fastcos (aph3 + x * af3 / (punkte)) * ag3x) *
    (1 - fastcos (aph3 + y * af3 / (kurven)) * ag3y);

  f := (k + l + m) * groesse;
end (* f *);

begin (* sinusgebirge *)

  for kurve := 0 to cmaxslot do begin (* Initialisiere Hidden-Arrays *)
    zmax [kurve] := -20.0; (*initialisiere zmax*)
    zmin [kurve] := 20.0; (*initialisiere zmin*)
  end (*for*);
  zold := 0.0;

  xoffset := -18.0; yoffset := -9.0;
  (* Beginne mit dem Diagramm links unten *)
  scaleversatz := 0;

```

```
171   {*****************************************************************}
172   for kurve := 0 to kurven do begin (* zeichne Funktionskurven *)
173     z := 0.0; punkt := 0;
174     hidden := false;
175     (* Gehe davon aus, dass bei der ersten zu zeichnenden Linie *)
176     (* der Startpunkt angefahren werden muss. *)
177     moveto (xoffset, yoffset + f (0, kurve));
178     (*****************************************)
179     for punkt := 0 to punkte do begin
180
181       (*********************************************)
182       z := f (punkt, kurve); (*********************)
183       (*********************************************)
184
185       if z + yoffset >= zmax [punkt + scaleversatz] then begin
186         (* Die Kurve ist von oben sichtbar: *)
187         up := true;
188         if hidden then begin moveto (xoffset + (punkt - 0.5) * xschritt,
189                                      (zmax [punkt - 1 + scaleversatz] +
190                                       zmax [punkt + scaleversatz]) / 2.0);
191                             hidden := false;
192                       end (* if hidden *);
193         drawto ((xoffset + punkt * xschritt), (yoffset + z));
194         zold := zmax [punkt + scaleversatz];
195         zmax [punkt + scaleversatz] := z + yoffset;
196         (* Neuer Gipfelpunkt der Kurve *)
197         if z + yoffset < zmin [punkt + scaleversatz] then
198           zmin [punkt + scaleversatz] := z + yoffset;
199       end (* von oben sichtbar *)
200       else if z + yoffset < zmin [punkt + scaleversatz]
201       then begin
202         (* Die Kurve ist von unten sichtbar *)
203         up := false;
204         if hidden then begin moveto (xoffset + (punkt - 0.5) * xschritt,
205                                      (zmin [punkt - 1 + scaleversatz] +
206                                       zmin [punkt + scaleversatz]) / 2.0);
207                             hidden := false;
208                       end (* if hidden *);
209         drawto ((xoffset + punkt * xschritt), (yoffset + z));
210         zold := zmin [punkt + scaleversatz];
211         zmin [punkt + scaleversatz] := z + yoffset;
212         (* Neuer Talpunkt der Kurve *)
213         if z + yoffset > zmax [punkt + scaleversatz] then
214           zmax [punkt + scaleversatz] := z + yoffset;
215       end (* von unten sichtbar *)
216
217       else begin (* unsichtbarer Punkt *)
218         if not hidden and (punkt > 0)
219         then begin (* Der letzte Punkt war sichtbar *)
220           if up then
221             drawto ((xoffset + (punkt - 0.5) * xschritt),
222                     (zold +
223                      zmax [punkt + scaleversatz]) / 2.0)
224           else
225             drawto ((xoffset + (punkt - 0.5) * xschritt),
226                     (zold +
227                      zmin [punkt + scaleversatz]) / 2.0);
228         end (* if not hidden *);
229         hidden := true;
230       end (* Punkt unsichtbar *);
231     end (* for punkt *);
232     (*****************************************)
233
234       xoffset := xoffset + xschritt; yoffset := yoffset + yschritt;
235       scaleversatz := scaleversatz + 1;
236       (* Beginne eine Kurve mit perspektivischem Versatz *)
237       (* Zum neuen Skalenanfang - um eine Position eingerueckt *)
238     end (* for kurve *);
239     (*****************************************************)
240     moveto (17.0, 12.0);
241   end (* sinusgebirge *);
242
243   procedure init;
244   begin
245     kurven := ckurven;
246     punkte := cmaxpunkt;
247     xschritt := 36 / (punkte + kurven);
248     yschritt := 18 / (kurven);
249     f1 := cf1 * 360.0; f2 := cf2 * 360.0; f3 := cf3 * 360.0;
250     g1 := cg1; g2 := cg2; g3 := cg3;
251     ag1x := cag1x; ag1y := cag1y;
252     ag2x := cag2x; ag2y := cag2y;
253     ag3x := cag3x; ag3x := cag3x; ag3y := cag3y;
254     ph1 := cph1; ph2 := cph2; ph3 := cph3;
255     aph1 := caph1; aph2 := caph2; aph3 := caph3;
256     af1 := caf1 * 360.0; af2 := caf2 * 360; af3 := caf3 * 360;
257     faf1 := 0.0; faf2 := 0.0; faf3 := 0.0;
258     fag1 := 0.0; fag2 := 0.0; fag3 := 0.0;
259     fph1 := 0.0; fph2 := 0.0; fph3 := 0.0;
260     groesse := cgroesse;
261     (* UCSD: *)
262     (* rewrite (plotfile, cplotname); *)
263     (* Turbo: *)
264     assign (plotfile, cplotname);
265     rewrite (plotfile);
266     inisintab;
267   end (* init *);
268
269   begin (* Hauptprogramm *)
270     init;
271     bef := ' ';
272     repeat
273       (* UCSD: *)
274       (* sc_clr_screen; *)
275       (* Turbo: *)
276       ClrScr;
277       writeln ('Gebirge-Generator          ESCape fuer Programmende');
278       writeln ('<RETURN> Kurve zeichnen');
279       writeln;
280       writeln ('A, B, C Gewichtung         1: ', g1:3:3, ', 2: ', g2:3:3,
281                ', 3: ', g3:3:3);
282       writeln ('D, E, F Frequenz           1: ', f1/360.0:3:3,
283                ', 2: ', f2/360.0:3:3, ', 3: ', f3/360.0:3:3);
284       writeln ('G, H, I Amplituden-Frequenz 1: ', af1/360.0:3:3,
285                ', 2: ', af2/360.0:3:3, ', 3: ', af3/360.0:3:3);
286       writeln;
287       writeln ('J, K Amplituden-Gewichtung 1x: ', ag1x:3:3,
288                ', 1y: ', ag1y:3:3);
289       writeln ('L, M Amplituden-Gewichtung 2x: ', ag2x:3:3,
290                ', 2y: ', ag2y:3:3);
291       writeln ('N, O Amplituden-Gewichtung 3x: ', ag3x:3:3,
292                ', 3y: ', ag3y:3:3);
293       writeln;
294       writeln ('P, Q, R Phase              1: ', ph1:3:3,
295                ', 2: ', ph2:3:3, ', 3: ', ph3:3:3);
296       writeln ('S, T, U Phase Amplitude    1: ', aph1:3:3,
297                ', 2: ', aph2:3:3, ', 3: ', aph3:3:3);
298       writeln;
299       writeln ('X, Y Punkte pro Kurve:         ', punkte:3,
300                ', Kurven: ', kurven:3);
301       writeln;
302       writeln ('Z Gesamtamplitude:             ', groesse:3:3);
303       writeln;
304       writeln ('1, 2, 3 FM-Frequenz        1: ', faf1/360.0:3:3,
305                ', 2: ', faf2/360.0:3:3, ', 3: ', faf3/360.0:3:3);
306       writeln ('4, 5, 6 FM-Amplitude       1: ', fag1:3:3,
307                ', 2: ', fag2:3:3, ', 3: ', fag3:3:3);
308       writeln ('7, 8, 9 FM-Phase           1: ', fph1:3:3,
309                ', 2: ', fph2:3:3, ', 3: ', fph3:3:3);
310
311       (* UCSD: *)
312       (* read (bef); *)
313       (* Turbo: *)
314       read (kbd, bef); if bef = chr (cr) then bef := ' ';
315       writeln;
316       case bef of
317           'i':    begin writeln ('Generator, bitte warten...');
318                         dosinusgebirge;
319                   end;
320         'y':      begin write ('Kurven: '); readln (kurven);
321                         xschritt := 36 / (punkte + kurven);
322                         yschritt := 18 / (kurven);
323                   end;
324         'x':      begin write ('Punkte: '); readln (punkte);
325                         xschritt := 36 / (punkte + kurven);
326                         yschritt := 18 / (kurven);
327                   end;
```

```
328         '1':    begin write ('FMF1: '); readln (faf1); fmf1 := faf1 * 360.0;
329                 end;
330         '2':    begin write ('FMF2: '); readln (fmf2); fmf2 := faf2 * 360.0
331                 end;
332         '3':    begin write ('FMF3: '); readln (fmf3); fmf3 := faf3 * 360.0;
333                 end;
334         '4':    begin write ('FMG1: '); readln (fmg1); end;
335         '5':    begin write ('FMG2: '); readln (fmg2); end;
336         '6':    begin write ('FMG3: '); readln (fmg3); end;
337         '7':    begin write ('FPH1: '); readln (fph1); end;
338         '8':    begin write ('FPH2: '); readln (fph2); end;
339         '9':    begin write ('FPH3: '); readln (fph3); end;
340         'a':    begin write ('G1: '); readln (g1); end;
341         'b':    begin write ('G2: '); readln (g2); end;
342         'c':    begin write ('G3: '); readln (g3); end;
343         'd':    begin write ('F1: '); readln (f1); f1 := f1 * 360.0 end;
344         'e':    begin write ('F2: '); readln (f2); f2 := f2 * 360.0 end;
345         'f':    begin write ('F3: '); readln (f3); f3 := f3 * 360.0 end;
346         'g':    begin write ('AF1: '); readln (af1); af1 := af1 * 360.0 end;
347         'h':    begin write ('AF2: '); readln (af2); af2 := af2 * 360.0 end;
348         'i':    begin write ('AF3: '); readln (af3); af3 := af3 * 360.0 end;
349         'j':    begin write ('AG1X: '); readln (ag1x); end;
350         'k':    begin write ('AG1Y: '); readln (ag1y); end;
351         'l':    begin write ('AG2X '); readln (ag2x); end;
352         'm':    begin write ('AG2Y '); readln (ag2y); end;
353         'n':    begin write ('AG3X '); readln (ag3x); end;
354         'o':    begin write ('AG3Y '); readln (ag3y); end;
355         'p':    begin write ('PH1: '); readln (ph1); end;
356         'q':    begin write ('PH2: '); readln (ph2); end;
357         'r':    begin write ('PH3: '); readln (ph3); end;
358         's':    begin write ('APH1: '); readln (aph1); end;
359         't':    begin write ('APH2: '); readln (aph2); end;
360         'u':    begin write ('APH3: '); readln (aph3); end;
361
362         'z':    begin write ('Groesse: '); readln (groesse); end;
363       end (* case *);
364
365     until bef = chr (esc);
366   end.
```

Listing »Gebirge« (Schluß)

Die Parallel-schnittstelle

Die marktführende Position des Druckerherstellers Centronics in den Anfangzeiten des Mikrocomputers hat viele Computerhersteller dazu veranlaßt, eine Parallelschnittstelle in ihre Systeme einzubauen, an die ohne Änderungen die Centronics-Drucker anzuschließen waren. Dies hat zu einem Quasi-Standard geführt, den heute fast alle Drucker- und Systemhersteller übernommen haben.

Als Verbindung zum Computer wird ein Kabel mit 36poligen Anschlußsteckern verwendet, bei dem jeweils ein Aderpaar miteinander verdrillt ist. Davon führt eine Leitung Massepotential und dient somit zur Abschirmung der Datenleitung vor äußeren Störeinflüssen oder vor der Beeinflussung nebeneinanderliegender Datenleitungen. Die Signalbedeutungen und die Belegung der Stecker sind weitgehend einheitlich, jedoch treten Unterschiede hinsichtlich von Erweiterungen und der Interpretation von Schnittstellensignalen durch die verschiedenen Hersteller auf, so daß eine Ausnutzung aller Möglichkeiten, welche die Parallelschnittstelle bietet, in den wenigsten Fällen erfolgt. Wie üblicherweise eine Centronics-Schnittstelle aussieht, zeigt Bild 1.

Die einzelnen Signale der Parallelschnittstelle lassen sich in vier Kategorien unterteilen:

— Datenleitungen

Die acht Datenleitungen übertragen die eigentliche Information vom Computer zum Drucker und zwar so, daß jeweils ein ASCII-Zeichen komplett, zu einem Zeitpunkt übertragen wird.

— Handshake-Leitungen

Die Handshake-Leitungen regeln den Datenverkehr. Der Rechner erhält eine genaue Information über den Stand der Übertragungen beziehungsweise der Beschäftigung des Druckers. Es wird dadurch verhindert, daß Daten bei der Übertragung verlorengehen.

Die Handshake-Leitungen haben folgende Bedeutung:

Data-Strobe

Wenn das Rechnerinterface auf dieser Leitung einen kurzzeitigen Impuls liefert, wird die Information, die zuvor auf die Datenleitungen gelegt wurde, in den Drucker übernommen. Bevor dieser Vorgang stattfinden darf, muß sich der Computer von der Bereitschaft des Druckers überzeugt haben.

Acknowledge (Anerkennung, Quittung)

Diese Leitung wird vom Drucker benutzt, um dem Computer mitzuteilen, daß sein Zeichen ordnungsgemäß übernommen wurde. Die Mitteilung ist rein formal, eine Bestätigung, ob es sich tatsächlich um die ausgesandte Information handelt, kann bei dieser Übertragungsart nicht erfolgen.

Busy (beschäftigt)

Ist der Drucker in irgendeiner Weise beschäftigt, beispielsweise beim Ausdruck eines Zeichens, beim Wagenrücklauf oder Formularvorschub, so zeigt er seinen momentanen Status durch das Busy-Signal an. Bevor der Computer Daten an den Drucker senden darf, muß er sich in jedem Fall mittels des Busy-Signals vergewissern, daß der Drucker zur Datenaufnahme bereit ist.

— Zustandsmeldungen

Treten Fehlerzustände während der Übertragung oder beim Drucker auf (zum Beispiel Papierende oder Druckkopf klemmt), so kann dies über spezielle Signalleitungen dem Computer mitgeteilt werden, der daraufhin sein Druckprogramm abbrechen und eine entsprechende Fehlermeldung auf den Bildschirm ausgeben kann.

Im einzelnen sind dies:

PE (Paperend)

Kein Papier mehr im Drucker

SELECT

Der Drucker befindet sich im »Online-Mode« und kann Daten aufnehmen.

FAULT

Im Drucker ist ein Fehlerzustand eingetreten, der eine weitere Benutzung nicht möglich macht, beispielsweise Farbband gerissen, Druckkopf überhitzt, Schlitten klemmt etc.

— ezielle Signale und Stromversorgung

Input-Prime oder Init

Mit diesem Signal kann man vom Computer aus eine Neuinitialisierung des Druckers bewirken. Empfängt der Drucker dieses Signal, so nimmt er eine Ausgangsstellung ein, die meist derjenigen nach dem Einschalten entspricht.

Die Tatsache, daß die Parallelschnittstelle keine richtige Norm ist, läßt natürlich eine weitere Belegung der freibleibenden Leitungen zu. Dies wird deutlich, wenn man beispielsweise die beiden Drucker MX80 F/T von Epson und 8510 von C. Itoh miteinander vergleicht (siehe Bild 2).

Beim Epson kann das AUTO-FEED-XT-Signal dazu benutzt werden, um den Drucker zu veranlassen, nach dem Empfang des Zeilenende-Zeichens »CR«, automatisch einen Zeilenvorschub auszuführen.

Der C. Itoh legt auf Pin 18 seine Stromversorgung von 5 V und liefert somit eine eindeutige Aussage, ob der Computer ein- oder ausgeschaltet ist (vergleiche Bild 3).

Besonders gravierend ist der Unterschied bei Abschluß 36. Er dient beim Epson dazu, den Drucker in den Select-Mode zu schalten und ist dementsprechend ein Eingangssignal, während Input-Busy beim Itoh ein Ausgangssignal ist und eine ähnliche Funktion wie das Busy-Signal, jedoch mit einem anderen Zeitverhalten hat.

In Bild 4 ist der Aufbau einer Rechnerschnittstelle aufgezeigt, die zum Parallelanschluß eines Druckers geeignet ist. Mit den heute zur Verfügung stehenden hochintegrierten Bausteinen ist der Aufbau einer solchen Schnittstelle sehr einfach und bei allen Computerherstellern in ähnlicher Form zu finden. Allgemeine Parallelschnittstellen sind für alle CPU-Familien erhältlich und arbeiten gleichartig. Ein Baustein vereinigt meist zwei oder drei 8 Bit breite »Ports« (Tore), die vom Computer her über bestimmte Adressen angesprochen werden können. Welche Adressen das sind, ist einzig und allein von der Verschaltung der Bausteine innerhalb des Computersystemes abhängig und muß jeweils der Systembeschreibung entnommen werden.

Die Parallelschnittstellenbausteine verhalten sich bezüglich ihrer äußeren Anschlüsse vollkommen neutral, das heißt erst durch einen Grundeinstellungsprozeß (Initialisierung) wird vom Programm her festgelegt, welcher Anschluß eines Ports Eingang und welcher Ausgang sein soll oder anders ausgedrückt, die im Computer eingebaute Parallelschnittstelle wird erst durch ein spezielles Programm, das im Betriebssystem vorhanden sein muß, zur Druckerschnittstelle.

Ganz so flexibel ist die Sache in den meisten Fällen allerdings nicht, denn bevor die Daten und Steuersignale an die Anschlußbuchsen des Computers gelangen, werden sie noch durch sogenannte Leitungstreiber verstärkt. Diese Leitungstreiber sind elektronische Verstärkerbausteine (nicht zu verwechseln mit den Treiberprogrammen), die dafür sorgen, daß die zu übertragenden Signale noch »gut erkennbar« auf der anderen Seite des Kabels, also beim Drucker ankommen. Sie dienen gleichzeitig als Schutz des Schnittstellenbausteines gegen Kurzschlüsse, Fehl- oder Überspannungen im Kabel oder im angeschlossenen Gerät. Meistens werden unidirektionale Treiberbausteine vorgeschaltet, das heißt eine Signalübertragung wird nur in einer Richtung zugelassen. Hardwareseitig ist damit festgelegt, welcher Anschluß der Schnittstelle Eingang und welcher Ausgang ist. Bei der Programmierung der Schnittstelle muß auf diese Gegebenheiten Rücksicht genommen werden.

Bis hierhin können wir von einer physikalischen Anpassung sprechen. Die eigentliche Aufgabenstellung, nämlich die Ausgabefunktion des Computers an den Drucker und seine Überwachung, wird durch ein sogenanntes Treiberprogramm erreicht.

Wie ist ein Treiberprogramm für eine Druckeransteuerung aufgebaut?

Treiberprogramme gehören zum innersten Kern eines Betriebssystems, der die Verbindung der Hardware zum rein logischen Gerät des Betriebssystemes herstellt. Will man Korrekturen oder Erweiterungen daran vornehmen, müssen folgende Voraussetzungen bekannt sein:

1. Kenntnis über die Funktion des verwendeten Schnittstellenbausteines, seine Arbeitsweise und seine Programmierung.
2. Die Einbettung des Bausteines in die Architektur des Computers, also Ansprechadressen und Ansprechtechnik (spezielle Ein-/Ausgabebefehle oder Memory-Map).
3. Das Zusammenwirken des Treiberprogrammes mit dem Betriebssystem — Parameterübergabe, Bufferbereiche und zugehörige Pointer, eventuell auch die Einbindung des Treiberprogrammes in eine Interruptstruktur.

Die Anforderungen, die hier gestellt werden, sind also keineswegs gering und es liegt nahezu auf der Hand, daß eine vernünftige Programmierung hier nur in Assembler oder einer systemnahen Sprache, beispielsweise in C, erfolgen kann. Einfacher geht es, wenn zur Druckerausgabe ein Anwenderprogramm verwendet wird, denn hier entfällt das »Einbauen« des Treiberprogrammes in das Betriebssystem; die neu hinzugekommenen Möglichkeiten des Druckprogrammes können dann allerdings nicht von anderen Programmen genutzt werden.

Schema eines Treiberprogrammes für eine Parallelschnittstelle

Die Software zum Betrieb einer Parallelschnittstelle läßt sich in drei Hauptaufgaben gliedern.

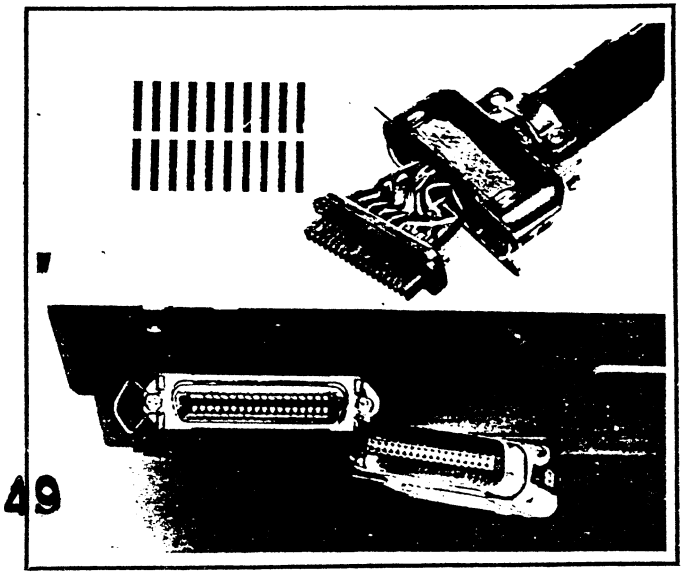


Bild 1. Centronics-Schnittstelle. Druckerkabel mit Cannon-Steckverbindung

	Epson MX80F/T	C. Itoh 8510
Pin 14	AUTO FEED XT	0 V
Pin 18	nc	+5 V
Pin 35	pulled up 5 V	nc
Pin 36	SLCT IN	Input-Busy
nc	heißt not connected = nicht angeschlossen	

Bild 2. Unterschiedliche Pin-Belegung der Centronics-Schnittstelle beim MX80 F/T und 8510

Strobe	1 ! 19	Masse
DATA-1	2 ! 20	Masse
DATA-2	3 ! 21	Masse
DATA-3	4 ! 22	Masse
DATA-4	5 ! 23	Masse
DATA-5	6 ! 24	Masse
DATA-6	7 ! 25	Masse
DATA-7	8 ! 26	Masse
DATA-8	9 ! 27	Masse
ACKNOWLEDGE	10 ! 28	Masse
BUSY	11 ! 29	Masse
PAPER END	12 ! 30	Masse
SELECT	13 ! 31	INIT
0V	14 ! 32	ERROR
nc	15 ! 33	0V
0V	16 ! 34	nc
Chassis GND	17 ! 35	nc
x	18 ! 36	x

Bild 3. Pinbelegung bei der Centronics-Parallelschnittstelle

Bewährtes DFÜ-Protokoll

Wer zwischen Computern Daten austauschen will, muß sich mit dem Partner über die Übertragungsregeln einig sein. Wir erläutern am Beispiel von »XModem« Aufbau und Arbeitsweise eines solchen Regelkataloges.

Das bekannteste Regelwerk — oder Protokoll — für den Austausch binärer Daten ist XModem. Es ist während der Blütezeit des CP/M-Betriebssystems von Ward Christensen entwickelt worden.

Folgendes Prinzip liegt der fehlerfreien Datenübertragung bei XModem zu Grunde: Die Informationen werden in Blöcken zu je 128 Bytes übertragen. Jeder Block wird beim Absenden mit einem Blockzähler und mit einer Prüfsumme versehen, so daß der Empfänger feststellen kann, ob ein Fehler bei der Übertragung aufgetreten ist.

Der Empfänger sendet nach dem Empfang jedes Blockes ein Antwortzeichen zurück, nämlich das Byte <ACK>, wenn der Block fehlerfrei empfangen wurde und die richtige Blocknummer aufwies, oder das Byte <NAK>, wenn irgend etwas nicht stimmte. Bei einem Fehler wird derselbe Block bis zu zehnmal wiederholt. Wenn allerdings alle zehn Versuche fehlschlagen, dann kapituliert XModem.

Die folgenden Zeichen werden zum Steuern der Übertragung verwendet:

Code	Englisch	Deutsch	Hexadezimal
<SOH>	Start of Header	Blockanfang	01
<EOT>	End of Transmission	Ende der Übertragung	04
<ACK>	Acknowledge	Bestätigung	06
<NAK>	Not Acknowledge	Keine Bestätigung	15
<CAN>	Cancel	Abbruch	18

Die Übertragung erfolgt asynchron mit 8 Datenbits, ohne Paritätsbit und mit einem Stopbit.
Ein Übertragungsblock ist folgendermaßen aufgebaut:

<SOH>
<Blocknummer>
<255 — Blocknummer>
<128 Bytes Daten>
<Prüfsumme>

Hierbei bedeutet:
<SOH> = 01 hexadezimal
<Blocknummer> = binäre Zahl, beginnend mit 01, Zählschritt 1, beginnt nach FF wieder mit 00 (nicht 01)
<255 — Blocknummer> = Komplement der Blocknummer nach der 8080 »CMA« Instruktion, das heißt jedes Bit der 8-Bit-Blocknummer wird invertiert. Dies ist das »Einerkomplement«.
<Prüfsumme> = Summe nur der Datenbytes. Überträge werden vernachlässigt (Modulo 256).

1. Initialisierung der Schnittstelle, das heißt Vorbereitung für den Betrieb des Computers mit einem bestimmten Peripheriegerät, hier also einem Drucker.
2. Bedienung einer Ausgabeanforderung des Betriebssystemes (eigentliche Ausgabe)
3. Behandlung von Fehlerereignissen

Punkt 1. haben wir bereits angesprochen. Der Vorgang der Initialisierung wird einmalig beim Start des Computers durchlaufen und stellt die Schnittstelle zum ordnungsgemäßen Zusammenspiel mit einem bestimmten Drucker ein. Zusätzlich können bei diesem Vorgang noch bestimmte Zwischenspeicherbereiche (Puffer) reserviert und systemabhängige Einstellungen vorgenommen werden.

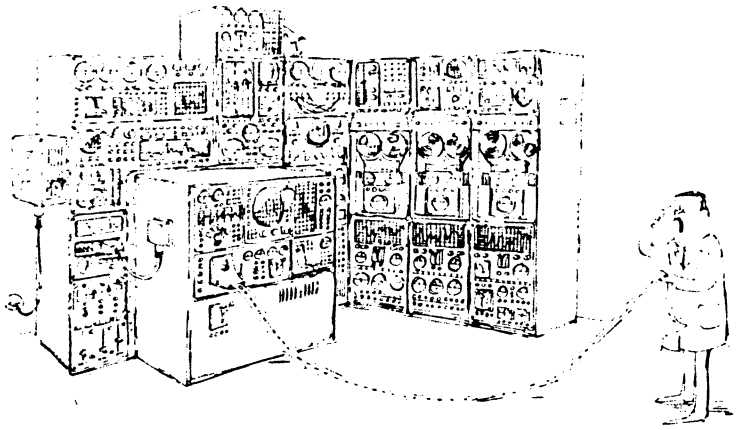
Das eigentliche Ausgabeprogramm wird erst dann angesprochen, wenn ein Anwenderprogramm ein Zeichen auf den Drucker ausgeben möchte. Ist das der Fall, so muß das Treiberprogramm folgendermaßen vorgehen:
1. Ist ein Drucker angeschlossen? nein — Fehlerbehandlung
2. Zeitüberwachung einschalten
3. Ist der Drucker selektiert? nein — Fehlerbehandlung
4. Ist der Drucker beschäftigt? ja — dann warte, gehe zu Punkt 3
5. Daten anlegen
6. Strobesignal erzeugen
7. Auf Quittung (ACK) warten
8. Zeitkontrolle ausschalten
9. Zurück zum aufrufenden Programm.

Im Hintergrund läuft die Zeitkontrolle, die ebenfalls eine Fehlermeldung generiert, wenn der gesamte Vorgang nicht innerhalb einer vorgegebenen Zeitspanne abgewickelt werden kann. Von den hier aufgezählten Schritten sind für eine einfache Ausgabe auch weniger Schritte möglich, und diese werden der Einfachheit halber auch von den meisten Herstellern in einer solchen abgemagerten Form angeboten. Das Ganze sieht dann etwa so aus:
1. Ist der Drucker belegt (entweder beschäftigt, nicht selektiert eventuelle Fehler oder gar nicht angeschlossen)? nein — dann warte
2. Daten anlegen
3. Strobesignal erzeugen
4. Zurück zum aufrufenden Programm.

In der Praxis »wartet« dann meist nicht nur das Ausgabeprogramm, sondern auch der Benutzer, der (wenn er sich nicht gerade auf dem Computersektor auskennt), sich in den wenigsten Fällen zu helfen weiß. Das »Nichtstun« eines Drukkers kann viele Ursachen haben:
1. Der Drucker ist ausgeschaltet
2. Der Drucker ist eingeschaltet, aber nicht angewählt (selektiert)
3. Der Drucker ist eingeschaltet und selektiert, aber das Verbindungskabel steckt nicht im Computer
4. Verbindungskabel steckt, ist aber defekt
5. Verbindungskabel ist in Ordnung, aber der Drucker zeigt einen Fehlerzustand an, zum Beispiel kein Papier, Farbband ist verbraucht, Druckkopf klemmt
6. Drucker oder Schnittstelle sind defekt; hier hilft allerdings auch kein komfortables Treiberprogramm, allenfalls ein Diagnose- und Druckertestprogramm.

Wir haben nun die Parallelschnittstelle in groben Zügen besprochen. Sie ist von der Programmierung einfacher zu handhaben, als die serielle Schnittstelle und erlaubt anhand spezieller Signale wie ERROR, SELECT und ACKNOWLEDGE die Richtigkeit der Übertragung zu prüfen und Fehler- oder Betriebszustände des Druckers abzufragen. Dafür bringt sie einen größeren Aufwand bezüglich der Verbindungskabel mit sich und ist nur für relativ kurze Strecken von 1 bis 5 Meter anwendbar.

(Peter Bonsch)



»Und wenn du diesen Fehler jetzt noch einmal machst, dann schreibst du zur Strafe 5000mal: ICH BIN NUR EINE MASCHINE UND HABE KEINEN FREIEN WILLEN!«

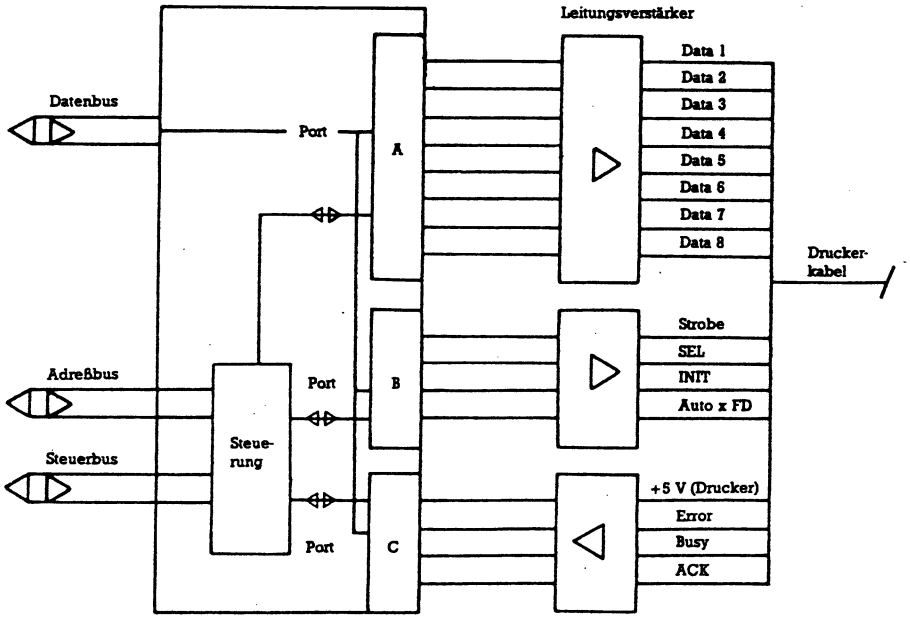


Bild 4. Schaltschema einer Parallel-Schnittstelle für den Anschluß eines Druckers

Das folgende Beispiel zeigt eine typische Übertragung einer drei Blöcke langen Datei. Es enthält die beiden häufigsten Übertragungsfehler — einen falsch übertragenden Block und ein falsch übertragendes <ACK>-Zeichen.

SENDER	EMPFÄNGER (Timeout und Wiederholung nach 10 Sekunden)
Der Sender wartet.	Der Empfänger beginnt mit:
	<—<NAK>
Der Sender sendet den ersten Block.	
<SOH> <01> <FE> Daten <Prüfsumme>—>	
	Der Empfänger hat den Block einwandfrei empfangen und antwortet:
	<—<ACK>
Der Sender sendet den zweiten Block.	
<SOH> <02> <FD> Daten <Prüfsumme>—>	
Eine Leitungsstörung tritt auf.	Der Empfänger hat den Block nicht einwandfrei empfangen und antwortet mit:
	<—<NAK>
Der Sender wiederholt den zweiten Block.	
<SOH> <02> <FD> Daten <Prüfsumme>—>	
	Der Empfänger antwortet mit:
	<—<ACK>
Der Sender sendet den dritten Block.	
<SOH> <03> <FC> Daten <Prüfsumme>—>	
	Der Empfänger antwortet mit:
	<—<ACK>

Eine Leitungsstörung tritt auf. Das <ACK>-Signal wird verfälscht.

Der Sender hat kein einwandfreies <ACK> erhalten und wiederholt daher den dritten Block.	
<SOH> <03> <FC> Daten <Prüfsumme>—>	
	Der Empfänger hatte den dritten Block bereits erhalten und antwortet wieder mit:
	<—<ACK>
Das Ende der Datei ist erreicht. Der Sender sendet ein:	
<EOT>—>	
	Der Empfänger antwortet mit:
	<—<ACK>

Die Übertragung ist beendet. Die Daten sind fehlerfrei beim Empfänger gespeichert.

Christensen (Seine Mailbox-Adresse ist: CompuServe 76703, 302.) empfiehlt, das Zeichen <CAN> (Cancel, Abbruch) nicht mehr wie bei früheren Implementationen zum Abbrechen der ganzen Übertragung zu verwenden, weil es zu leicht vorkommen kann, daß ein <ACK> oder <NAK> durch eine Leitungsstörung in ein <CAN> umgemünzt wird.

Schließlich müssen viele Timeouts für den heutigen Betrieb an weiten Datennetzen und überfüllten Host-Computern länger ausgelegt werden. Die ursprünglichen 10 Sekunden reichen oft nicht mehr.

XModem ist ein einfaches und zuverlässiges Protokoll. Es ist jedoch nicht ohne Schwächen. Insbesondere hat sich die Welt der Telekommunikation seit der Entwicklung von XModem weiterentwickelt. XModem ist nicht an alle Anforderungen, die wir heute haben, optimal angepaßt.

Eine Schwäche des Protokolls ist die Ungenauigkeit in der Dateilänge. Der letzte Datenblock enthält nämlich, genau wie auch alle vorhergehenden, 128 Daten-Bytes. Die Länge der übertragenen Datei muß aber nicht genau in den 128-Byte-Blöcken aufgehen. Der letzte Block ist also meist nicht voll.

XModem schreibt für den letzten Block nur vor, daß er auch 128 Daten-Bytes enthält. Das Übertragungsprogramm muß also gegebenenfalls diesen letzten Block auf die volle Länge auffüllen. Das empfangende Programm hat bei XModem keine Möglichkeit, die ursprüngliche genaue Dateilänge festzustellen.

In den allermeisten Fällen spielt diese Ungenauigkeit keine Rolle. Man wundert sich nur, wenn man eine übertragene Datei mit der ursprünglichen vergleicht (zum Beispiel mit dem COMP-Befehl in MS-DOS), daß dies manchmal mit der Fehlermeldung »Dateien haben unterschiedliche Länge« endet.

Ein zweites Problem ist das an vielen anderen Stellen verwendete X-ON/X-OFF-Protokoll. Dies ist eine Start-Stop-Steuerung, die sich der beiden Zeichen X-ON und X-OFF bedient. Wenn Daten zum Beispiel über Datex-P geschickt werden und das X-ON/X-OFF-Protokoll aktiv ist, dann ist die Wahrscheinlichkeit, daß die Übertragung plötzlich mitten im schönsten Datei-Transfer stehenbleibt, groß.

Probleme mit X-ON/X-OFF

Dies passiert, wenn in der zu übertragenden Datei zufällig ein X-OFF-Zeichen auftritt, aber im gleichen Block später nicht auch noch ein X-ON-Zeichen vorkommt. Die Datenübertragung bleibt dann sofort stehen und läßt sich mit keinem Mittel zum Weiterlaufen bewegen.

Die einzige Lösung dieses Problems besteht darin, das X-ON/X-OFF-Protokoll vor Beginn der XModem-Übertragung abzuschalten. Anders ausgedrückt, der Übertragungskanal muß »transparent« geschaltet werden.

Es gibt noch einige andere, ähnlich gelagerte Probleme, die sich durch Transparent-Schalten lösen lassen, zum Beispiel das Auftreten von Ctrl-P-Zeichen, die das Datex-P-Netz in den Befehlsmodus umschalten. Auch dann ist es aus mit der Datenübertragung.

Wie man den Übertragungskanal transparent schaltet, hängt vom Kanal selbst ab. Eine Telefonleitung ist immer transparent, solange sie nicht zu viele Störungen aufweist. Datex-P hat ein spezielles Kommando zum Umschalten in ein »transparentes Profil«. Wenn Sie Daten mit XModem über Datex-P übertragen wollen, dann geben Sie vor Beginn ein: Ctrl-P prof 3 und drücken Sie dann genau zweimal die Return-Taste. (Für Ctrl-P halten Sie die Ctrl-Taste fest und drücken dann kurz die Buchstabentaste P.)

Datex-P arbeitet nun transparent, das heißt, jedes Bit kommt völlig unverändert auf der Gegenseite an. Das bedeutet zwangsläufig auch, daß das gewohnte Echo verschwindet. Sie tippen also von nun an blind, aber dafür funktioniert XModem zuverlässig selbst auf den längsten interkontinentalen Strecken.

Dies führt uns gleich zum nächsten Problem mit dem XModem-Protokoll. XModem ist ursprünglich für den Betrieb auf Telefonleitungen entwickelt worden. Mit den transkontinentalen Datennetzen war man damals noch nicht so weit.

Heute stellt sich heraus, daß XModem für unsere modernen X.25-Netze, wie Datex-P, nicht besonders gut angepaßt ist. Datex-P verwendet zum Beispiel eine Paketlänge von 128 Daten-Bytes. Ein XModem-Block ist aber unglücklicherweise mit den zusätzlichen Kontrollinformationen gerade 132 Bytes lang. Für die Übertragung von 128 Byte Daten werden also zwei X.25-Pakete benötigt, von denen das zweite fast leer ist, aber voll bezahlt werden muß.

Das nächste Problem mit den X.25-Netzen ist, daß XModem nach jedem Datenblock wartet, bis die Antwort von der Gegenstelle wieder zurückgekommen ist. Bei einer Langstreckenverbindung, zum Beispiel in die USA, kann das leicht eine Sekunde dauern, weil die Datenpakete ja auf dem Hinweg und die Antworten auf dem Rückweg noch einmal einen kleinen Umweg von je 2 x 36000 km machen müssen. Dies ist nämlich der geostationäre Orbit der Kommunikations-Satelliten. Dadurch wird XModem über Datennetze langsam.

X-Modem und Datex-P

Wer glaubt, durch Wechseln von 300 auf 1200 Baud Geschwindigkeit der Datenübertragung auf die Sprünge helfen zu können, der irrt sich hier. Der Paket-Transfer braucht seine Zeit.

Natürlich kann man Daten auch sehr viel schneller über X.25-Netze übertragen. Eine Methode ist, ganz einfach längere Blöcke zu verwenden. Eine noch raffiniertere ist, auf die Antworten nicht erst zu warten, sondern immer schon einige Blöcke »auf Verdacht« im Voraus zu senden. Die beste schließlich ist es, die unsichere asynchrone Verbindung ganz zu vermeiden und sich einen X.25-Anschluß in den PC zu stecken. Alle diese Wege werden in der Tat beschritten, und eine Reihe neuerer Protokolle für die fehlerfreie Datenübertragung benutzen den einen oder anderen hier genannten Weg, um eine bessere Leistung zu erzielen.

Alternativen

Einige andere Protokolle zur fehlerfreien Datenübertragung binärer Dateien sind XModem CRC, Kermit, CompuServe B, MNP und X.PC. XModem CRC, eine Abwandlung des XModem-Protokolls, wurde von John Mahr entwickelt. Der wesentliche Unterschied ist, daß statt einer einfachen Prüfsumme ein »Cyclic Recundancy Check«, ähnlich wie beim Schreiben eines Blockes auf eine Diskette oder Festplatte, verwendet wird. Der CRC verringert die Wahrscheinlichkeit eines unentdeckten Fehlers auf einen verschwindend geringen Wert.

Die letzteren beiden, MNP und X.PC sind leistungsfähige Protokolle, die nicht nur zum Übertragen von Dateien geeignet sind, sondern den gesamten Datenverkehr kontrollieren und sichern. X.PC ist eine Variante des X.25-Protokolls, wie es zum Beispiel innerhalb des Datex-P-Netzes verwendet wird, jedoch abgeändert für den asynchronen Datenverkehr mit Personal Computern.

Auf Seite 107 finden Sie eine Übersicht populärer Kommunikatinsprogramme. Viele davon übertragen mit XModem. Bei der Auswahl eines DÜ-Programmes ist auf folgende Punkte zu achten:

— Hilfstexte sollen jederzeit einblendbar sein.
— Alle Übertragungsparameter sollen veränderbar sein — auch während eines Dialoges.
— Es soll ein Standard-Protokoll für die Übertragung binärer Dateien implementiert sein.

Es sollen Filter für die Umwandlung von 7- und 8-Bit-Zeichensätzen, speziell auch den IBM-Zeichensatz, mitgeliefert werden. Das gilt insbesondere für Programme, die kein Standardprotokoll für Binärfiles können.
— Funktionstasten sollen frei belegbar sein (für Standard-Prozeduren).
— Die Bedienung soll einfach sein (gefragt sind Menüs, Funktionstasten und gängige Tastenkombinationen zum Aufruf einer Funktion).
— Es sollte ein einfacher Editor zum Offline-Editieren von DÜ-Texten zuschaltbar sein. (H. G. Michna/sm)

Fortsetzung des Schwerpunktes »Kommunikation« auf S. 98

Glossar

asynchron auch: Start-Stop-Betrieb, Form der Datenübertragung, bei der jedes Zeichen mit einem »Startbit« beginnt und mit einem oder mehreren »Stopbits« endet. Gegenteil von »synchron«

binär im Zahlensystem mit der Basis 2 verschlüsselt, das heißt nur mit den Ziffern 0 und 1 darstellbar

Byte Gruppe von in der Regel 8 Bit, die ein Zeichen, zum Beispiel einen Buchstaben, darstellt

CP/M-Betriebssystem
CP/M-Betriebssystem Control Program/Microcomputer, am weitesten verbreitetes Betriebssystem für Microcomputer, die auf den älteren 8-Bit-Prozessoren Intel 8080, 8085 und Zilog Z80 basieren, geschrieben von Digital Research Corporation

CRC Cyclic Redundancy Check, zyklische Redundanz-Prüfzahl, die in der Regel durch eine Polynom-Formel aus einem Datenblock errechnet wird und eine sehr hohe Prüfsicherheit gewährleistet, das heißt wenn im Datenblock ein Fehler auftritt, dann wird aus diesem fehlerhaften Datenblock mit sehr hoher Wahrscheinlichkeit nicht dieselbe CRC-Prüfzahl errechnet

Ctrl Control-Taste

Datenbit Bit, aus dem sich im Gegensatz zum Startbit oder Stopbit die eigentlichen Daten zusammensetzen, in der Regel besteht ein asynchron übertragenes Byte aus einem Startbit, 7 oder 8 Datenbits, eventuell einem Paritätsbit und einem Stopbit

Datex-P In der Bundesrepublik Deutschland von der Post betriebener Teil des globalen Datennetzes nach dem X.25 Standard (paketvermittelndes Netz)

Idiot Zurückbeenden der eingetippten Zeichen, damit die eintippende Person auf dem Bildschirm sieht, was sie tippt

geostationärer Orbit Kreisbahn eines Satelliten, auf der der Satellit sich so mit der Erde mitdreht, daß er immer über derselben Stelle scheinbar verharrt, wird besonders für Kommunikationssatelliten verwendet, damit die Datenübertragungen nicht durch Auf- oder Untergehen des Satelliten unterbrochen werden und damit die Antennen nicht nachgeführt werden müssen

Host englisch, »Gastgeber«, hier: Computer, der elektronische Dienstleistungen über Datenfernübertragung bereitstellt

implementieren hier: Realisierung in Form eines Programmes

MS-DOS Microsoft-Disk Operating System, meistverbreitetes Betriebssystem für Microcomputer, die auf einem 16-Bit-Prozessor der Intel 8086-Familie basieren

Parität auch Prüfbit, zu einem Byte gehöriges Bit, das die Anzahl der gesetzten Bits (mit dem Wert 1) je nach Vereinbarung entweder immer gerade oder immer ungerade macht

Protokoll hier: Verfahren zur Datenübertragung, bei dem Fehler automatisch korrigiert werden

Stopbit Bit mit dem Wert 1, das bei asynchroner Übertragung einem Byte nachgestellt wird, bei Übertragungsgeschwindigkeiten bis zu 110 Bit/s werden auch 1 1/2 oder 2 Stopbits verwendet, ab 150 Bit/s in der Regel nur eines

synchron Form der Datenübertragung, bei der mehrere Bytes hintereinander ohne Unterbrechung und ohne Start- oder Stopbits gesendet werden, Gegenteil von »asynchron«

Timeout Zeitüberschreitung

virtueller Kanal unabhängiger Datenübertragungskanal, der scheinbar wie eine eigene Leitung arbeitet, sich jedoch physisch mit mehreren anderen Kanälen dieselbe Leitung teilt

X-ON/X-OFF-Protokoll verbreitetes Datenübertragungs-Protokoll, bei dem der Datenfluß mit einem X-OFF-Zeichen (Ctrl-S, dezimal 19) gestoppt und mit einem X-ON-Zeichen (Ctrl-Q, dezimal 17) wieder gestartet werden kann

X.25 Standard für paketvermittelnde Datennetze, auf dem beispielsweise das Datex-P-Netz beruht, die Daten werden in Paketen, üblicherweise zu je 128 Daten-Bytes, durch das Netz befördert

Verkaufe zum Schrottpreis:
TRS-80
Monitor
2 Laufwerke
diverses Zubehör

Wilfried Knospe
Lesserstr. 126
2000 Hamburg 70
Tel. 040/695 55 76

****** PC-FOUR *** Modell 4 Emulator *** PC-FOUR ****

Wer hat Erfahrung mit dem Modell 4 Emulator PC-FOUR von Hypersoft ?

Gibt es einen weiteren MS-DOS Emulator für das TRS-DOS System ?

Klaus Hermann, Tel.: 07127/71945

****

BÖRSE -- BÖRSE -- BÖRSE

Verkaufe:
ZX-81, 16 KB, Handbücher: 70,-
Genie I, Centronics, Cassetten: 100,-
Sanyo-Monitor, bernstein, 18 Mhz: 150,-
TRS-80 M. I, Exp.-Interface 48 Kb, Banker 256 Kb (ohne RAMs),
HRG-1B, Zerobrenner, 3xBASF 6106 (je 1x40Track): 400,-
außerdem: Software, Handbücher, Anleitungen

Reiner Stober
Nelkenstraße 12
3216 Salzhemmendorf 4
Tel. 05153/1564

Genie IIIs, Z80B-CPU, 7,2/1,77 MHz umschaltbar, 256kB RAM, 12kB ROM-Adreß-raum (ein paar kB belegt; erweiterte ROM-Version auf Wunsch), nach Reset durchgehend RAM (kein ROM mehr. Speicherinhalt veränderbar), 64kB Graphik (Auflösung max. 512X512), beliebig variables Bildschirmformat (max. 2kB), ladbare Zeichensätze 8 Pixels breit (Höhe variabel. max. 16 P.). DOS kom-plett im Speicher, RAM-Disk 64/128kB, 2 Laufwerke 80/DS/DD (andere Formate einstellbar), Echtzeituhr/-kalender batteriegepuffert. G-DOS 2.4 (auf Wunsch erweiterte Version CALVA-DOS 2.4), CP/M 2.2, noch immer voll kompatibel mit TRS-80/Genie I usw., stark erweiterte Tastatur (z. B. 8 Funktionstasten, insg. 96 T.), bisher keine Bernhardt'schen Umbauten (schade!). Neupreis ca. 6000 Mark, Altpreis: Macht mal ein Angebot!
Freut euch nicht zu früh, es handelt sich nicht um meine Maschine, die bleibt euch noch erhalten. Diese gehört einem Freund.
Angebote bitte an Arnulf, Tel. 0451-791926

C compiler
for the model I or 3 using
TRSDOS, LDOS, NEWDOS,
DOSPLUS, or MULTIDOS;
includes full screen text editor and
advanced development package

Wer kennt diesen C-Compiler der Fa. ALCOR für NEWDOS ?
Nachricht bitte an Heinrich Betz 09191/31698

Verkaufe: - altes Shugart-Floppy
- RS 232 C und
- Accustikkoppler-
(dataphon s21d)
- EM 2005 (Banker)

Werner Förster
Christoph-Krebs-Str. 9
8720 Schweinfurt
☎ 09721/21841

BÖRSE -- BÖRSE -- BÖRSE

Sehr geehrter Herr Schröder,

wie ich der C'T 8/88 entnehme, beschäftigt sich Ihr Club mit den guten alten TANDY TRS-80-Rechnern.

Ich bin zwar vor einiger Zeit auf einen ATARI ST umgestiegen, besitze jedoch noch einen TRS-80, den ich nun wegen Platzmangel verkaufen möchte.

Die Anlage besteht aus:

- TRS-80, Mod.I (Bj. 80), 16kB RAM, Level II Basic Groß-/Kleinbuchstaben, eingebauter Joystickanschluß für Standartjoystick (bedient die Pfeiltasten und die Leertaste)

- Tandy-Monitor, grüner Phosphor

- CE-Computer Expansion Interface (Bj. 12/82) mit 32 kB RAM, Floppy Controller und Erweiterungs-Bus interne Erweiterungskarte: DD-Controller für Laufwerke mit doppelter Schreibdichte (Mod. III kompatibel)
- Doppelfloppygehäuse mit 2 * TEAC, 5¼", 40 Spuren, 1-seitig doppelte Schreibdichte (ca. 180 kB/Floppyseite)

- 9-Nadeldrucker Triumpf-Adler, deutscher Zeichensatz, für Einzelblatt, Endlos- und Fernschreiberpapier, 80 cps, (Bj. 83)

- ca. 80 Disketten mit Software (Pascal, Basic, Fortran, M80, FORTH, NEWDOS80, LDOS, Spiele, Anwenderprogramme, SCRIPSIT, Super-SCRIPSIT, etc.)

- ca. 30 kg Dokumentation (Hardware + Software, Zeitschriftenartikel, Schaltpläne, etc.)

Der Anschaffungspreis der Hardware lag bei ca. DM 4000.--, meine Preisvorstellung: DM 450.-- bei Selbstabholung.
Alle Komponenten arbeiten einwandfrei!

Sollten Sie sich für die Anlage interessieren, oder einen Interessenten kennen, bitte ich um Nachricht.

Mit freundlichem Gruß

Peter Büttner
Borkumer Straße 21
8500 Nürnberg 90
0911/ 38 21 36

Angebot

Hallo, Leute!

Möchtet Ihr einen Rechner mit 512 KByte RAM, 20 Megabyte Festplatte, einem Diskettenlaufwerk, mehreren Centronics- und V24-Schnittstellen für 1000 bis 2000 DM haben? Nein, es sind keine IBM-Kompatiblen, sondern Z80-Rechner auf ECB-Bus-Basis (Betriebssysteme CP/M, MP/M, OASIS). Ein Münchner Händler hat die Dinger im Keller stehen und wäre froh, sie los zu werden.

Die Rechner sind unterschiedlich ausgestattet. Voll ausgebaut sehen sie so aus: Z80 B mit 6 Mhz Taktfrequenz, 512 KByte RAM, 2 Floppy-Laufwerke (bis 1.2 MByte), 2 Festplatten (je 15-40 MByte) oder eine Festplatte und ein Streamer (entsprechende Größe), mehrere V24- und Centronics-Schnittstellen, Voll-Ausbau beim Spitzengerät auf 8 Terminals und 8 Drucker möglich. Alle Systeme bauen auf dem ECB-Bus auf und es sind noch diverse Einzel-Karten (RAM, Schnittstellen) vorhanden. Die meisten Systeme haben 15.000 bis 25.000 DM gekostet, sind entweder neu oder liefen in Zahnarztpraxen im Mehrbenutzer-Betrieb. Entsprechend sind mehrere Drucker und Terminals anschließbar und auch noch vorhanden. Als Betriebssysteme stehen CP/M, MP/M (Mehrbenutzer-Variante von CP/M) und OASIS (Mehrbenutzer-Betriebssystem) zur Verfügung, außerdem zumindest ein Textverarbeitungsprogramm mit Datenbankfunktionen und evtl. noch Büro-Software. Außerdem Basic, Cobol und Assembler.

Der Mensch kann uns noch Tandberg-Terminals (neu: ca. 7000 DM) für 200 DM, mehrere V24-Terminals und Typenraddrucker anbieten. Die Dinger können auch als Zugabe zu einem MP/M-/OASIS-Rechner geliefert werden, wenn wir das absprechen.

Es gibt drei Rechner-Varianten (Angaben ohne Gewähr):
Junior mit Z80 B, 256/512 KB, Floppy, 15-20 MB Festplatte, 2-4 V24-Schnittstellen; ca. 1000 DM.
16000 mit Z80B, 256/512 Kb, Floppy, Festplatte 20-40 MB, Streamer, 2-8 Schnittstellen (=Terminals=Benutzer!); unter 2000 DM.
ONIX (wird vielleicht anders geschrieben) mit Floppy, Magnetband, Streamer, 4 Schnittstellen; Preis ???

Ich habe diese Sache nur kurz am Telefon besprochen, leider ist dies inzwischen (2.5.89) über drei Monate her. Eigentlich sollte dieses Angebot mit dem letzten Info erscheinen, aber das klappte wohl nicht.

Es sind scheinbar 8 ONIX-, 10 Junior- und 30 16000-Rechner dort gelagert und verstopfen nur den Platz. Der Mensch möchte nur den Materialwert (Festplatten, Floppys, RAM) haben und danach nichts mehr von den Dingern wissen, versichert aber, daß alles im besten Zustand ist. Wenn sich mehrere von uns finden, die gemeinsam den ganzen Ramsch oder den größten Teil nehmen, will er auf den Preis keine Rücksicht mehr nehmen. Wenn Ihr Interesse habt, meldet Euch bei mir. Ich glaube, so ein Angebot bekommt Ihr nicht noch einmal. Allein der Materialwert ist höher als die Preise. Und im Sechserpack wird's billiger! Ich schätze, bei 15 bis 25 Interessenten je 1000 DM bekommen wir den ganzen Kram. Wer weiß?!

Gerald Schröder
(04105/2602)

Aus einer Mailbox: (hoffentlich kennt zumindest jeder einen, der Englisch kann)

"humor": What's your favourite? (about 100 lines)
--- Mo, 28.11.88 / 00:22:23 von DOPPELKEKS
There are so many programming languages available that it can be very difficult to get to know them all well enough to pick the right one for you. On the other hand most men know what kind of woman appeals to them. So here is a handy guide for many of the popular programming languages that describes what kind of women they would be if programming languages were women.

Assembler - A female track star who holds all the world speed records. She is hard and bumpy, and so is not that pleasant to embrace. She can cook up any meal, but needs a complete and detailed recipe. She is not beautiful or educated, and speaks in monosyllables like "MOV, JUMP, INC". She has a fierce and violent temper that make her the choice of last resort.

FORTRAN - Your grey-haired grandmother. People make fun of her just because she is old, but if you take the time to listen, you can learn from her experiences and her mistakes. During her lifetime she has acquired many useful skills in sewing and cooking (subroutine libraries) that no younger women can match, so be thankful she is still around. She has a notoriously bad temper and when angered will start yelling and throwing dishes. It was mostly her bad temper that made grandad search for another wife.

COBOL - A plump secretary. She talks far too much, and most of what she says can be ignored. She works hard and long hours, but can't handle really complicated jobs. She has a short and unpredictable temper, so no one really likes working with her. She can cook meals for a huge family, but only knows bland recipes.

BASIC - The horny divorcee that lives next door. Her specialty is seducing young boys and it seems she is always readily available for them. She teaches them many amazing things, or at least they seem amazing because it is their first experience. She is not that young herself, but because she was their first lover the boys always remember her fondly. Her cooking and sewing skills are mediocre, but largely irrelevant, its the frolicking that the boys like.
The opinion that adults have of Mrs. BASIC is varied. Shockingly, some fathers actually introduce their own sons to this immoral woman! But generally the more righteous adults try to correct the badly influenced young men by introducing them to well behaved women like Miss Pascal.

PL/I - A bordello madam. She wears silk dresses, diamonds, furs and red high heels. At one time she seemed very attractive, but now she just seems overweight and tacky. Tastes change.

C - A lady executive. An avid jogger, very healthy, and not too talkative. Is an good cook if you like spicy food. Unless you double check everything you say (through LINT) you can unleash her fierce temper. Her daughter C++ is still quite young and prone to tantrums, but it seems that she will grow up into a fine young woman of milder temper and more sophisticated character.

ALGOL 60 - Your fathers wartime sweetheart, petite, well proportioned, and sweet tempered. She disappeared mysteriously during the war, but your dad still talks about her shapely form and their steamy romance. He never actually tasted much of her cooking, but they did exchange simple recipes by mail.

Pascal - A grammar school teacher, and Algol 60's younger sister. Like her sister she is petite and attractive, but very bossy. She is a good cook but only if the recipe requires no more than one pot (module).

Modula II - A high-school teacher and Pascal's daughter. Very much like her mother, but she has learned to cook with more than one pot.

ALGOL 68 - Algol 60's niece. A high-society woman, well educated and terse. Few men can fully understand her when she talks, and her former lovers still discuss her mysterious personality. She is very choosy about her romances and won't take just any man as her lover. She hasn't been seen lately, and rumor has it that she died in a fall from an ivory tower.

LISP - She is an aging beatnik, who lives in a rural commune with her hippie cousins SMALLTALK and FORTH. Many men (mostly college students) who have visited the farmhouse, enthusiastically praise the natural food, and perpetual love-ins that take place there. Others criticize the long cooking times, and the abnormal sexual postures (prefix and postfix). Although these women seldom have full-time jobs, when they do work, their employers praise them for their imagination, but usually not for their efficiency.

APL - A fancy caterer specializing in Greek food. She can cook delicious meals for rows and rows of tables with dozens of people at each table. She doesn't talk much, as that would just slow her work down. Few people can understand her recipes, since they are in a foreign language, and are all recorded in mirror writing.

LOGO - A grade-school art teacher. She is just the kind of teacher that you wish you had when you were young. She is shapely and patient, but not an interesting conversationalist. She can cook up delicious kiddie snacks, but not full-course meals.

LUCID & PROLOG - These clever teenagers show a new kind of cooking skill. They can cook-up fine meals without the use of recipes, working solely from a description of the desired meal (declarative cooking). Many men are fascinated by this and have already proposed marriage. Others complain that the girls work very slowly, and that often the description of the meal must be just as long as a recipe would be. It is hard to predict what these girls will be like when they are fully mature.

Ada - A WAC colonel built like an amazon. She is always setting strict rules, but if you follow them, she keeps her temper. She is quite talkative, always spouting army regulations, and using obscure military

Alexander Schmid

- I M P R E S S U M -

1. Vorsitzender
Alexander SCHMID
St.Cajetan Str. 38 VII
8000 München 80
☎ 089 /495326

2. Vorsitzender
Horst-Dieter SCHROERS
Breslauer Str. 9
8016 Feldkirchen
☎ 089 /9032615

Hardwarekoordinator
Andreas MAGNUS
Pommernstr. 4
4650 Gelsenkirchen
☎ 0209 /870230

NEWDOS-Diskothekar
Oliver VOLZ
Waldburgstr. 73
7000 Stuttgart 80
☎ 0711 /7353817

CP/M -Diskothekar
Rüdiger SöRENSEN
Thomas-Mann-Str. 3a
6500 Mainz 1
☎ 06131 /32860

Club-Bücherei
Christian MENK
Ollsener Str. 52
2116 Hanstedt
☎ 04189 /78251

Redaktion
Jens NEUEDER
Gschlachtenbretzingen
Rudolf-Then-Straße 32
7178 Michelbach /Bilz
☎ 0791 /42877

Autoren
Die Redaktion bedankt sich bei den im INHALTSVERZEICHNIS genannten Autoren für die Mitarbeit an der Club-INFO.

Bankverbindung
des CLUB 80
Postgirokonto
Sonderkonto CLUB 80
Konto-Nummer 285 491 - 465
Postgiroamt Dortmund
BLZ 440 100 46

Eine Zensur oder Kontrolle der Infobeiträge erfolgt nicht. Die Redaktion

Schluss

Hallo Club-80er,

nachdem wir im April/Mai unsere Jahreshauptversammlung hatten, erscheint nun unser neuestes Clubinfo mit neu kreiertem Deckblatt.
Unser Clubemblem ist als neues Erkennungszeichen eingearbeitet.

Auch sonst hat sich einiges getan.
Der Vorstand ist "erneuert" -ihr habt es sicher im Vorwort gelesen- und ein neuer Weg für den Druck des INFO's wurde gefunden. Hoffen wir, daß es diesmal besser funktioniert als in letzter Zeit.

Eine weitere "Gute Nachricht" für Euch wird wohl das Erscheinen von acht Sonderausgaben sein. Die ersten zwei --
NEWDOS-Diskothek's-Verzeichnis
und
CLUB-80-Büchereiliste
-- liegen diesem Info bei.

Nun aber noch einen Aufruf an Euch:
Wie Ihr aus diesem INFO ersehen könnt fehlt es an "kleinen Artikeln". So ca. 1 bis 4 Seiten lange hatten wir bisher immer genügend, nun aber fehlen genau solche! Ich hatte Euch die "Artikelflaute" ja schon im 26. INFO angekündigt.
Sicher habt Ihr irgendwo noch einen "kurzen Artikel" für unser Clubinfo. Ich bitte Euch ihn mir zuzusenden, damit unser Clubinfo auch weiterhin regelmäßig erscheinen kann.
Wie Ihr wißt ist alles zum Thema Computer interessant. Sei es nun über Hard- Software oder irgendwas anderes Computerspezifisches. Ich rechne mit Eurer Mitarbeit.

Bis auf bald, zum 28. INFO und einigen Sonderheften.

Es grüßt Euch Euer

Jens

P.S.: Nicht vergessen - Dein Beitrag zum INFO !!

HEFT 27 Juli 1989