MTX User-Club Deutschland

Info 31
08.03.1989

Zweck: Zusammentragen und Austausch von Tips & Tricks u.s.w., Hilfestellung bei allen möglichen Problemen, Aufbau einer Programmbibliothek und Basteln von Hardware-Erweiterungen.

Programme (nur **Selbstgeschriebenes**): Tausch von kurzen und einfachen Routinen. Gute Programme (mit Dokumentation) können über den Club an alle Mitglieder verkauft werden. Wer solche Programme an uns schickt erhält ggf. Verbesserungshinweise und eine Besprechung im Info.

Mitglied kann jeder werden! Keine Beitragsgebühr! Anmeldung kostet DM 1.-.

Verpflichtungen: Einsendung unseres Anmeldeformulars.

Bitte: Einsendung von Tips & Tricks, Fragen, Antworten, kurzen Routinen, Programmen, Beiträgen zum Info, Hinweisen auf preiswerte Hard- und Software, und was noch so zusammenkommt und andere interessieren könnte.

Club-Info, unser Blatt, verschicken wir ca. 8-wöchentlich. Inhalt ist alles was uns über den MTX/FDX (ohne Copyright) in die Hände fällt. Es kostet nicht über DM 12.- (75 Seiten) je Exemplar. Jeder kann dazu Beiträge liefern und hier gratis Kleinanzeigen veröffentlichen.

Kosten: Wir berechnen ausschließlich Selbstkosten und verschicken **nichts**, wenn Ihr persönliches Guthaben nicht reicht! (s.u.)
Schüler, Studenten, Auszubildende, Grundwehrdienstleistende, Rentner und Arbeitslose erhalten einen Nachlaß von 40% auf die zukünftigen Infos nach Einsendung einer entsprechenden Bescheinigung für deren Gültigkeitzeitraum.

Geld/Konto: Für jedes Mitglied führt Herbert zur Nedden ein Konto, von dem die jeweils enstehenden Kosten abgehen. Der Kontostand wird bei **jeder** Sendung mitgeteilt (**er steht über der Anschrift**) und kann selbstverständlich jederzeit erfragt werden! Wir verschicken nur gegen Vorkasse!

Einzahlungen bitte auf's Club-Konto: (oder V-Scheck)
(**Absender**! incl Name **und** Anschrift bitte nicht vergessen!)
Postgiroamt Hamburg, BLZ 200 100 20,
Herbert zur Nedden, Sonderkonto C, Nr. 3480 00-200

Kontaktadressen: (nach PLZ geordnet)

Herbert zur Nedden	Christian Löhrmann	Thomas Wulf	Hans Gras
Sonnenau 2	Grevenbleck 24	Roritzer Str. 8	Statenhoek 49
2000 Hamburg 76	3005 Hemmingen 1	8500 Nürnberg 90	NL 1506 VM Zaandam
(040) 200 87 04	(0511) 41 78 77	(0911) 33 52 52	(0031-75) 17 49 91

Telefon-Sprechzeiten
Herbert zur Nedden: Do 18 - 22 Uhr, Sa 10 - 14.30 Uhr

Inhaltsverzeichnis

Preis für dieses Info: DM 13,70

(d.L. Abo-Preis: 13,02 !)

Liebe Leserin, lieber Leser,

wieso kostet dieses Info mehr als die im Deckblatt genannten DM 12.- fragst Du Dich sicherlich. Nun ja. Innerhalb kürzester Zeit trudelte hier dermaßen viel an Beiträgen zum Info ein, daß ich mich gefragt habe, was ich tun soll. Artikel bis zum nächsten Info zurückstellen, das Inhaltsverzeichnis mit seinen 25 Seiten weglassen oder unter Ausnutzung der noch günstigen Porto-Gebühren das Info teurer machen, als es geplant war. Ich hielt letzteres für die Euch am meisten zusagenden Lösung. Selbstverständlich haben all diejenigen das Info erhalten, die ein Zwölfmarks-Info erhalten hätten. Bitte kontrolliere daher Deinen Kontostand. Es könnte sein, daß Du zu den dreien gehörst, die hierdruch das Konto überzogen haben.

Hast Du das Info gar als Päckchen erhalten, so liegt das an den Public-Domain-Disketten, die Du abonniert hast, bei denen es sich immerhin um zwei (evtl. mittlerweile drei) CLUB-PD's und eine KLICK·handelt. Bist Du jedoch Abonnent und findest die Plastikscheiben nicht, langte Dein Konto nicht aus, und der Umschlag mit den Disketten liegt hier für Dich bereit. Die KLICK.005 ist von vielen von Euch angemahnt worden: sie enthält nämlich die KLICK-Overlays als .KLX - naja, fast alle. HARD3 und ASCII3 folgen nach; NOTE wird (vorerst) nicht umgestellt, da dieses Programm anders ist, als die anderen KLICK-Overlays, und die Umstellung auf .KLX bedeutet den Turbo-Pascal-Teil in Assembler neu zu schreiben, oder zwei Programme draus zu machen. Obendrein gibt's ganze 13 neue SIG/M-PD's, nämlich das, was wir so zu ZCPR3 fanden.

Die Mitgliederliste im Info 30 mit den vielen 'I' links hat doch einiges Erstaunen und viele Anfragen nach der Zukunft der MTX User-Club Deutschland hervorgerufen. Es ist nicht zu leugnen, daß eine gewisse Fluktuation statfindet, aber die vollzieht sich recht langsam. Einige der 'I' gelten schon seit Mitte 1988!, hinter anderen verbergen sich Mitglieder die eh kein Info haben wollen. Aber deshalb geht's mit dem Club noch lange nicht zu Ende. Mir ist bewußt, daß das Info einigen Mitgliedern zu teuer oder zu uninteressant ist. Nun ja - daher gibt es das Angebot, nur die Angebotsliste des Clubs zu abonnieren. Diese kostet dann DM 1.50 frei haus je Exemplar, setzt allerdings voraus, daß der Kontostand zum Erscheinungstermin des Infos mindestens diese DM 1.50 enthält. Wer es verschlust, sein Konto auf mindestens DM 1.50 zu halten: PECH! (Die selbe Angebotsliste ist natürlich auch Bestandteil des Info's!)

In Bezug auf Horst Kupka's Erweiterung der 80Zeichen-Karte muß ich Euch leider noch etwas vertrösten. Horst hat mittlerweile den Schaltplan der 80Zeichen-Karte vollständig per Hand gezeichnet, und Holger Göbel testet die Aufrüstanleitung. Jedefalls soll der Speicher der 80Zeichen-Karte auf 64kB erhöht werden: 16k CP/M-Text, 16k KLICK-Text und 32k Grafik. Damit entfällt das Retten des Bildschirmes bei Anzeige-Wechsel und KLICK-Aufruf.

Seit einiger Zeit erscheint die Zeitschrift c't in gebundener Form, was recht unpraktisch ist, wenn es darum geht, die Jahres-Inhaltsverzeichnisse herauszunehmen. Wer hat Lust, zu versuchen das Gesamt-Inhaltsverzeichnis von c't aus deren Mailbox (CosmoNet, 300 Baud, 8 Bit, keine Parity, 1 StopBit. 0511 - 555392 und 555398) zu entladen. Ich glaube, daß das ein gutes Werk wäre.

Wer hat Lust sich mal zu folgenden Themen im Info auszulassen:
1. Harddisc (auch Festplatte genannt) und 2. WD-FloppyController-IC's (wär evtl. etwas für Michael Keßler).

Fröhliche Ostereier wünscht Euch

Euer Hubert

C l u b t r e f f e n: 6+2 Teilnahmemelungen

Hemmningen, April/Mai 1989 (Herbert zur Nedden, 2000)

Es ist schon bemerkenswert, wie groß das Interesse am Clubtreffen ist. Da kannst Du Terminenschränkungen, VIELLEICHT, ... angeben - und einige, denen ich durchaus Interesse am Treffen unterstellen möchte

M E L D E N S I C H E I N F A C H N I C H T

Bislang sind lumpige 6 Teilnamhe-Meldungen auf dem Formular aus Info 30 sowie zwei weitere in anderer Form eingegangen. Und ich bin sicher, daß da noch weitere von Euch Lust und Interesse haben - und gar kommen wollen.

Vielleicht ist Dir nicht klar, daß es wichtig ist, mögl. im voraus Bescheid zu wissen, wer WANN Zeit hat, kommen und vor allem übernachten will. Wie sollen wir sonst Räumlichkeiten und Betten reservieren können ?? Ob wir nur 10 oder 30 Leute sind, macht schon etwas aus!

Klar, die Lösung:
'Treffen am xx.xx.19xx um yy:yy Uhr Da-und-Da, Übernachtung ist Dein Problem' ginge auch.

WOLLT IHR DIESE ART DER LÖSUNG ???

Dann ist aber auch nix Vorbereitung und Einplanung/Sicherstellung, daß bestimmte Dinge vorgeführt werden (können) gegeben.

Außerdem lohnt sich Übernachtung nur dann, wenn auch genügend viele da bleiben - alleine muß ich daß nicht haben! Außerdem spart sich leicht Geld, indem zwei sich nicht unsympatische Mitglieder in einem Raum übernachten. Und sicherlich ist es vernünftig, wenn jemand, der vor Ort wohnt die Zimmer reserviert - da er in der Gegend Bescheid weiß.

Da hier im Club anscheinend nichts ohne **TERMINZWANG** funktioniert, greife ich also zu eben diesem Mittel, in der Hoffnung, daß es dann klappt. Jedenfalls will ich hier ein für alle Mal klarstellen, daß dies das letzte derartige Clubtreffen sein wird, wenn Euer Interesse an einem so gestalteten Treffen so mau ist, daß nicht einmal Zeit übrig ist, einen Zettel an mich abzuschicken, auf dem ggf. draufsteht, daß Du kommen möchtest, ggf. unter gewissen Einschränkungen.

FÜR DIESE EINSTELLUNG IST MIR MEINE ZEIT NÄMLICH ZU SCHADE!!!!

Genug genörgelt - aber es mußte raus.

Zum geplanten Clubtreffen:

Termin: Sa/So, frühestens 29./30. April, vermutlich jeodch Mai
(die meisten wollen übernachten.)

WICHTIG: Wenn Du auch kommen willst, melde Dich bitte bis zum

14. APRIL 1989

bei mir an. Meldest Du Dich später, dann hast Du evtl. ein Problem! Ich weiß nicht, wie es dann mit Übernachtung und/oder Essen aussieht. Ggf. müssen wir, um den Raum billig zu bekommen, mit so-und-soviel Mittagessen und Abendbrot bestellen. Ob die dann flexibel in der Menge sind, weiß ich nicht.

C l u b: Korrektur & Nachtrag / Fragen & Antworten

Korrektur und Nachtrag

XOVL (Programm zum Boot-EPROM von Michael Keßler) (Michael Keßler, 5600)
In XOVL.MAC ist die Zeile, die mit DEFB 94, ... beginnt, durch
DEFB 94,28,0,3,7,0,0B8,0,3F,0,0C0,0,10,0,3,0,3,7,18,0,4,1,8,28,1,1,5
zu ersetzen und mittels eines Z80-Assemblers neu zu assemblieren.

Boot-EPROM von Michael Keßler
Die Einseitigen Disketten-Formate scheinen bei älteren Boot-EPROMs nicht richtig zu funktionieren. Michael ist der Fehler schon bekannt.

Info 30, Seite 11: NSWEEP-Patch (Holger Hansen, 3300)
Vielen Dank für die Korrektur vom NSWEEP-Patch. Für Leute die nichts von Assembler ahnen noch eine kleine Korrektur:
Adresse 2990 : JR 2993 in JR 2995 ändern
im Hex-Dump war es richtig angegeben (war ja auch ein relativer Sprung)
Die Restaurierung des alten Calls sollte an den Anfang gestellt werden, weil CALL 70Ah nicht wieder zurückkommt.

Fragen und Antworten

F: (Holger Hansen, 3300)
Kann man (ähnlich wie bei MSDOS) unter RAM 4.3 ohne DOS bzw BIOS direkt in den Bildschirmspeicher schreiben ?
(schneller als mit CALL $FFD3 oder CALL $F400)

A: (Herbert zur Nedden, 2000)
Ja, indem Du direkt auf die Ports ab 30 Hex gehst. Wie das funktioniert ist im Info u.a. 3/14, 4/44 zu finden. Dabei mußt Du jedoch unter RAM 4.x die im Speicher abgelegten Kopien der Registerinhalte, insbesondere der Adresse im Bildschirmspeicher, die der Ecke oben links entspricht beachten. Und vergiß nicht die Interrupts!

F: (Holger Hansen, 3300)
Ist es Absicht, daß bei RAM43 M der Status von Alpha Lock mit abgespeichert wird?

A: (Herbert zur Nedden, 2000)
Falls RAM42 M das schon machte, hat sich Bernd Preusing vielleicht etwas dabei gedacht.
Falls das erst ab RAM43 passiert: Nein!

F: (Holger Hansen, 3300)
Der BIOS-Jump 0 (Kaltstart) an E100h zeigt auf einen Sektorpuffer (Bei mir steht dort ein Sektor des DIR).
Was hat dies für einen Sinn???

A: (Herbert zur Nedden, 2000)
Vermutlich garkeinen - scheint aber nichts auszumachen, da der Kaltstart nur bei RESET benutzt wird, und dann der BIOS-Jump 0 offensichtlich nicht ins Nirwana zeigt. Bei RAM 5.x werden Olaf und ich diesem Umstand abhelfen - und sei es mit einem NOP NOP NOP, um so auf den BIOS-Jump 1 (Warmstart) zu laufen.

A: (Wolfgang Dexheimer, 6719)
Anton Reisers Frage zu dBase
dBase macht keinen Seitenvorschub vor einem Report-Befehl, wenn dieser vorher mit SET EJECT OFF ausgeschaltet wird. Damit wird wirklich nur dieser erste Seitenvorschub unterdrückt, und Report hält sich an die mit 1 eingestellte Seitenlänge.

F: Wer hat Unterlagen über/Erfahrungen mit Sprachausgabe und -IC's ??

L e s e r b r i e f: Hartmut Traber, 5270

Muß noch ins Januar-Info!!

Hartmut Traber

Hohbeulstr. 8, 01.01.89
5270 Gummersbach
Tel.: 02261/65399

Lieber Herbert,

das war die Überraschung!!!

Drei Disketten am letzten Tag des alten Jahres (club.035, club.036 und Klick.004)!

Nach erster Durchsicht hat mich ganz besonders Dijey und das zugehörige Handbuch beeindruckt! Die Arbeit, die drin steckt, kann man unschwer erahnen!

Aber das Ergebnis hat den Aufwand gelohnt!
Das ist genau das, was ich schon immer haben wollte!
Das ist ja 90%-Sidekick!!!
Toll,toll,toll,toll,toll,toll,toll,toll,toll,toll,toll,toll,toll,toll!

Vielen Dank, Olaf!!!

(Dein Rechtschreib-Programm kann aber so gut nicht sein: ein Groß-/Klein-Fehler steckt im Manual, die Leerer unter uns mögen mich korrigieren!)

Leute, das müßt ihr haben!

Die vielen Ausrufezeichen haben wirklich ihren Sinn!

Ist das eine Besprechung?

(Ich bin allerdings der Meinung, daß keine Meldung über ein Programm positiv zu werten ist; jeder hat seine Anwendungen laufen und ist darüberhinaus wohl genau so faul wie ich, was allerdings negativ ist und mir bewußt! Solange kein echter Verriß kommt, sollte jeder annehmen, daß das Angebotene was taugt!).

Kauft Klick!
Rüstet den Rechner auf! (Mindestens Uhr, 80-Zeichen-Karte und Speicher!)
Mit Klick und RAMxx wird er besser und schneller als diese neumodischen Kompatiblen mit den 16-/32-Bit-Bussen!

Herbert, das Vorstehende klingt vielleicht euphorisch, Du darfst redigieren.
Es ist aber wirklich meine Meinung!

Bis bald,

Hartmut

Anm.d.HzN.: Text habe ich nicht redigiert! Sorry Hartmut, aber Dein Brief kam einen Tag zu spät bei mir an - daher halt erst im Oster-Info.

L e s e r b r i e f: Hartmut Traber, 5270

Hartmut Traber
Hohbeulstr. 8, 21.01.89
5270 Gummersbach
Tel.: 02261/65399

Lieber Herbert,

heute erhielt ich das neue Info mit Deinem Formblatt zur Anmeldung für das Club-Treffen.
Und wie das so ist und ich auch sonst schon mal Formulare ausfüllen muß (Steuer !) und der MTX immer bereitsteht, die Schreibmaschine dagegen (zuhause !) nicht, kam mir folgende BLENDENDE IDEE !

Wir brauchen für sowas ein Prográmmchen !

MTX als Schreibmaschine !

Alles was auf der Tastatur eingegeben wird, geht ohne jede Änderung
unter Umgehung des Spoolers auf den Drucker !

In Basic würde ich sogar mir zutrauen, sowas zu schreiben, aber ich mag das nicht: Basic laden, Programm laden, arbeiten.
Und kompiliertes Basic ? Ba, viel zu lang.

I/O-Redirection, hah, was ist das ? An dieser Stelle habe ich mir Device noch mal angesehen. So geht das also nicht.
Und mit ^P wird die Antwort des Systems auch immer noch mal gedruckt.
PIP LST:=CON: hat den Nachteil, daß man den Druckkopf nicht vorab positionieren kann, die Leertaste ist solange wirkungslos, als nicht mindestens ein Buchstabe und das abschließende Return kommt.
Die Esc-Sequenzen an den Drucker schicken geht, ist aber sehr umständlich.

Du schüttelst das Kaninchen sicher in 5 Minuten aus dem Zylinder ?
Oder diese Idee ins Info ?
Oder gibt es das schon und ich habe es nur vergessen ?

Und als ich diesen Schrieb von der RAM-Disc auf eine genehme Floppy-Disc schreiben wollte, kam das Ergebnis der Anlage heraus !
Alles war gelöscht gekennzeichnet, aber kursive "e" sind mir noch nicht untergekommen, mal abgesehen davon, daß die übrigen Dateien garnicht erscheinen !
(Sonst keine Fehler, der Effekt ist nicht reproduzierbar).

Doch noch nicht Schluß!
Ich wollte gerade einen Überblick über diesen Text gewinnen. Natürlich ganz einfach mit nwschirm, hatte es aber vorher nicht aufgerufen: Fehlanzeige!
Neu begonnen, mit vorherigem Aufruf, was merke ich?
Das Normalformat läßt sich nicht mehr einstellen!
Meine Formate: 80 x 28, 96 x 28, 96 x 56, die beiden letzteren gehen. (?)
Bei den Letzteren ist der Cursor da, wo er vorher war.
Beim Normalformat erscheint ein anderer Textausschnitt und Weitermachen mit Schreiben ist nicht, kein Befehl mehr möglich, Tastatur für nw tot. Nochmals Shift-Esc, anderes, größeres Format, alles paletti!
Siehste, und jetzt, nach mehrmaligen Versuchen geht es wieder!
Auch nicht mehr reproduzierbar in dieser Sitzung.

Ob das Viren sind?
Anm.d.HzN.: Vermutlich - oder haben die Programme 'nen Fehler ? Nein, VIRUS!

Viele Grüße

Hartmut

L e s e r b r i e f: Andreas Fischer, CH 4303

Andreas Fischer CH-4303

Leserbrief

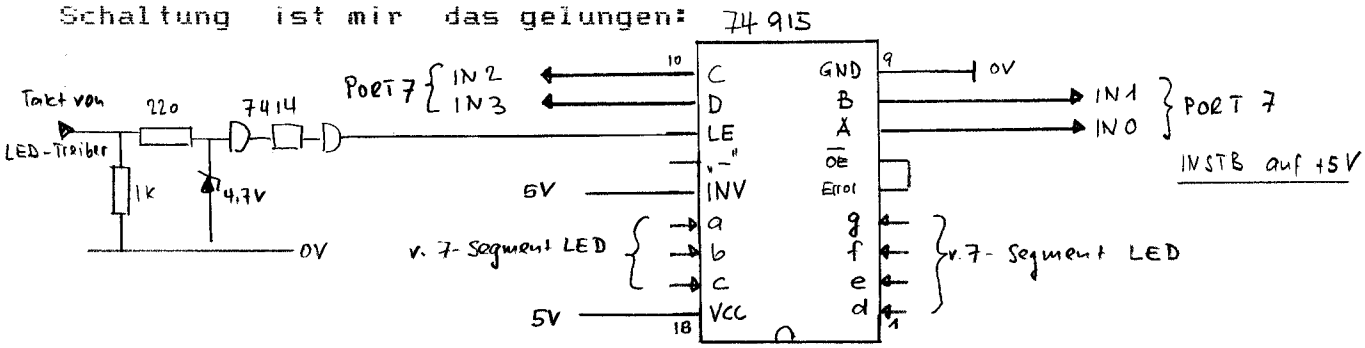
recht hat er, der Peter Lang.. ran an die Tasten!

Pocket Modem an MTX mit SDX-80-Zeichen

Seit kurzer Zeit bin ich im Besitze eines Modems. Von der PTT (in der Schweiz) zugelassen sind neu Pocket-Modems vom Typ Discovery 1200 P. Als Bindeglied zwischen MTX und Modem ist ein GENDER CHANGER M/M (Stecker-Stecker) dazwischenzuschalten. Als Ausgang nehme ich den RS323-1 Port.
Damit die automatische Nummernwahl möglich ist, habe ich das Modem-Programm M1.COM auf der PD 001 folgendermassen abgeändert:
0104H neu FF
0105H neu 50 (siehe MDM712.UPD auf PD 001 mit DDT M1.COM)
Mit dieser Aenderung wird der NUM-Befehl ungültig und der Befehl CAL wird möglich. Somit können Telefonnummern gemäss der angezeigten Liste automatisch angewählt werden.
Da die Liste auf M1 lediglich (amerikanische?) Telfonnummern enthält, musste sie noch abgeändert werden. Das wurde etwas mühsam, da ich auch das mit dem DDT ausführte:
D 0D10, dann ist das bestehende Verzeichnis ersichtlich
mit S z.B.0D70 kann man dann etwas mühsam mit HEX-Eingaben das eigene Telefonverzeichnis kreieren.
In diesem Zusammenhang habe ich übrigens herausgefunden, wie der Befehl SAVE xx M1.COM zu handhaben ist. Die Speicherzahl xx kann gemäss (B. Pol:Vom Umgang mit CP/M) berechnet werden:
Bei der Eingabe DDT M1.COM erscheint unter dem Wort "NEXT" eine Zahl. Bsp 4600. Die beiden höherwertigen Stellen werden in dezimale Form gebracht 46: 4*16=64 +6=70. Falls die beiden niederwertigen Stellen 00 betragen, ist noch 1 abzuziehen:70-1=69. Also muss nach der Aenderung mit DDT der Prozess mit "." abgeschlossen werden, dann mit ^C zum CP/M und dann die Aenderungen mit SAVE 69 M1.COM speichern. So einfach ist das!
Das Modem läuft zur vollen Zufriedenheit.
Vielleicht habe ich einen alten Zopf geschrieben..für mich war es jedenfalls neu. Sonst schadet ja eine Repetition auch nicht!

Abgiff von externen Digital-Anzeigen

In der Info 27-40 habe ich die Anwendung des MTX als Wettercomputer dargestellt. Für die Erfassung der Regenmenge griff ich auf eine elektronische Waage zurück. Mit folgender Schaltung ist mir das gelungen:



Damit werden wieder einmal mehr ungeahnte Möglichkeiten auf unserer schwarzen Kiste eröffnet. Ansteuerung mit PORT 7.

L e s e r b r i e f: Kurt-Bernd Rohloff, 8000 / Hans Gras, NL 1506

8000 München, den 11.02.89

Lieber Herbert!

In der Datei **INFO07.DOC** findest Du wieder ein bißchen "Futter" für's Info. Im übrigen finde ich es reichlich unverständlich, daß so viele Leute das letzte Info wegen mangelnder Kontodeckung nicht bekommen haben. Entweder steht bei denen die schwarze Kiste nur noch in der Ecke 'rum oder aber sie sparen hier am falschen Ende. Denn teuer kann man unser Info wirklich nicht nennen, wenn man es einmal am Informationsgehalt mißt. Man muß zudem berücksichtigen, daß unser Info sich ausschließlich mit dem MTX Computer befaßt, während die käuflichen Computerzeitschriften, wenngleich sie vielleicht auf die Seite bezogen billiger sind, sich mit allen möglichen Computern befassen, nur nicht mit dem MTX. Aber wem sage ich das!

Das nächste bayerische **Clubtreffen** soll im April stattfinden. Der genaue Termin steht noch nicht fest, aber es wurde der 8. oder 15. April ins Auge gefaßt.

Mit herzlichen Grüßen
Kurt-Bernd

Wieder etwas aus Holland.

Bitte mich vermerken als MBASIC Specialist (das bin ich leider nicht, aber vielleicht helft es, mehrere Leute an die Arbeit zu kriegen).

Wer intressiert ist in verbesserte Catalog Programmen (mehrere Usern ins 1 Schlag, User ins Catalog File, Angabe zB. "File MENU.COM steht ins User 4") bei mir melden. Beim ausreichende interesse wird alles ins PD gegeben. d.h. mir anrufen oder schreiben reicht um es aufs eine -CLUB.03? oder -CLUB.04? zu kriegen.

Die bereits ins Info 30 (Seite 22) genannte WordStar 4.0 ist auch für CP/M 2.2 lieferbar, und ist auch ZCPR 3 fähig. Ich brauch es mit eine NEC-P6+ und eine Canon LBP 8A1 printer.
Wer benutzt WS 4.0 mit Speller usw., statt von NW 2.xx (außer mir)?

Anm.d.HzN.: Ich habe hier eine interessante Unterhaltung mit einem WordStar 4.0-User, der NewWord 2.16 sah und darin sein WordStar in wenigen Details wiedererkannte. Da ich keinen Kontakt mehr zu NewStar bekam, ist es durchaus möglich, daß NewWord der Vater von WordStar 4.0 ist!!!!!!!!!!

Frage: ist RAM 4.3 KeyBoard buffering ein zu schränken aufs alte 15 Zeichen?
Wann wir Cursor Down einige Zeit festhalten bis DiskAccess, dann steht die ganse Puffer voll und muß über Shift Esc gelöscht werden. Anders bleibt WS 4.0 (und wahrscheinlich NW 2.xx auch) 40 Zeilen Down scrollen.
Anm.d.HzN.: Ja, Adresse 0F030h im laufenden System jetzt: 80h, darf 54h - 80h

Frage: Welches Format 1A mußen wir als Standart einsetsen?
Format mit 4k oder 2k Blocken? (BP/MK)
Meine Vorschlag: Format des Michael Keßler mit 2k Blocken.
Anm.d.HzN.: Bin Deiner Meinung.

Es tut mir leid das ich ins Beschreibungen von RAM 4.3 gar nichts gefunde habe wer der AlphaLock LED schaltung heraus gefunde hat. Warum?
Anm.d.HzN.: Sorry, Hans - ich habe gepennt.
Also: Die Idee der Lösung der AlphaLock-LED via Port 3 ist von Hans Gras!!

L e s e r b r i e f: Hans Gras, NL 1506

Betrift HiRes-Grafik aufs 80-Zeichen Karte von Horst Kupka.
Ich habe so etwas aufs Televideo TS803H gesehen. Spitze!!
Meine Hilfe ist bereits an Horst angeboten.

MBASIC Beispiel:

BC3.BAS, BC3.MAC, BC3.REL auf CLUB.040 Zum Linken mit compilierte Programmen.

Einsetzen Machine Code in MBASIC.

Ich brauchte für meine BASICODE 3 eine SOUND routine für 1 Kanal. Meine erste Versuch war in reine BASIC, war aber nicht schnell. Dafür eine Bastelung in Machine Code.
Achtung: Das POKE-Teil ist notwendig fürs Interpreter. Fürs Compiler ist dies Überflüsig (auch CLEAR). Die (der, die, das, dem usw.) Compiler (Linker) berechnet selbst alle Adressen. Bitte ins Handbuch mal nachschauen wie alles ans MachineCode überreicht wird.

TeilListing BC3.BAS Unterprogrammen.

```
10 REM BASICODE 3 SUBROUTINES (29/12/88) voor "MEMOTECH FDX"-BasiCode Versie 4.3!
11 REM ------------------------------------------------------------------------------
12 CLEAR ,256*(PEEK(7)-1):REM ..... Ruimte vrij maken voor High Memory Routines.
18 GOSUB 30:GOTO 1000:REM ..... Machine Code Initialiseren.
25 REM ------------------------------------------------------------------------------
30 RESTORE 60000:OJ=256*(PEEK(7)-1):OJPTBL=OJ:REM ..... Jump tabel gaat van beneden naar boven!
31 READ OI:IF OI<256 THEN POKE OJ,OI:OJ=OJ+1:GOTO 31:REM ..... Data inlezen.
34 ORAND=OJPTBL:CALL ORAND(RV%):RANDOMIZE RV%:REM ..... Set willekeurige getallen.
35 OBRKON=OJPTBL+2:OBRKOF=OJPTBL+4:REM ..... Break toets aan/uit routine.
36 OMUSIC=OJPTBL+6:RESTORE:RETURN:REM ..... Toon subroutine.
37 REM ------------------------------------------------------------------------------
400 REM ..... Maak een toon.
401 IF OJPTBL=0 THEN GOSUB 30:REM ..... Initialiseer system routines alsnog.
402 O=SP:IF O<47 THEN O=47:REM ..... Ca. 122 Hz, onze SN 76489 A gaat niet lager op 4 MHz.
403 OPITCH%=15343/2^(O/12):REM ..... Omgezet naar MTXcode (15296).
404 O=(SD-0.73)*30:IF O<1 THEN O=1:REM ..... Altijd een delay.
405 ODELAY%=O:REM ..... In kleine stapjes.
406 OVOLUME%=SV:IF SV<0 OR SV>15 THEN OVOLUME%=15:REM ..... Niet harder dan hardst.
407 CALL OMUSIC(OPITCH%,OVOLUME%,ODELAY%):RETURN:REM ..... Machinetaal routine aanroepen.
60000 REM ------------------------------------------------------------------------------
60010 REM ..... Data van de Jumptable.
60020 DATA &H18,&H06,&H18,&H44,&H18,&H43,&H18,&H07
60030 REM ..... Data voor de Random routine.
60040 DATA &HED,&H5F,&H77,&H23,&H36,&H00,&HC9
60050 REM ..... Data voor de Music routine.
60060 DATA &H7E,&HE6,&H0F,&HF6,&H80,&HD3,&H06,&HDB,&H03,&H23,&H7E,&HE6,&H03,&H2B,&HED,&H67,&H7E
60070 DATA &HD3,&H06,&HDB,&H03,&H1A,&H2F,&HE6,&H0F,&HF6,&H90,&HD3,&H06,&HDB,&H03,&H0A,&H5F,&H03
60080 DATA &H0A,&HC8,&HBF,&H57,&H06,&HFB,&HE3,&HE3,&H10,&HFC,&H1B,&H7A,&HB3,&H20,&HF5,&H3E,&H9F
60090 DATA &HD3,&H06,&HDB,&H03,&HFB,&HC9
60100 REM ..... Data voor de Break toets routine.
60110 DATA &HC9,&HC9,999
```

L e s e r b r i e f: Hans Gras, NL 1506

Listing BC3.PRN mit eigentliche Machine Code.

BASICODE 3 SYSTEM ROUTINE's MACRO-80 3.44 09-Dec-81

```
                                TITLE   BASICODE 3 SYSTEM ROUTINE's
                                .Z80
0000'                           CSEG
                        ;-----------------------------------------------------------------------

                        ;OJPTBL:        .PHASE  0D100h      ; Ruim onder de BDOS beginnen met Sysroutines
0000'  18 06            ORAND::         JR      RANDOM      ; Een echte RANDOMIZE opzetten
0002'  18 44            OBRKON::        JR      STPON       ; BREAK toets enabled
0004'  18 43            OBRKOF::        JR      STPOFF      ; BREAK toets disabled
0006'  18 07            OMUSIC::        JR      MUSIC       ; Maak een toontje

                        ;-----------------------------------------------------------------------

0008'  ED 5F            RANDOM:         LD      A,R         ; Dit is RANDOM genoeg
000A'  77                               LD      (HL),A
000B'  23                               INC     HL
000C'  36 00                            LD      (HL),0
000E'  C9                               RET

000F'                   MUSIC:
000F'  7E               OPITCH:         LD      A,(HL)
0010'  E6 0F                            AND     15          ; Data bits 0...3 maskeren
0012'  F6 80                            OR      80h         ; Frequentie TONE 1
0014'  D3 06                            OUT     (6),A
0016'  DB 03                            IN      A,(3)
0018'  23                               INC     HL
0019'  7E                               LD      A,(HL)
001A'  E6 03                            AND     3           ; Data bits 8 en 9 maskeren
001C'  2B                               DEC     HL
001D'  ED 67                            RRD                 ; Decimaal roteren, nu bits 4...9
001F'  7E                               LD      A,(HL)      ; Naar de Accumulator
0020'  D3 06                            OUT     (6),A
0022'  DB 03                            IN      A,(3)
0024'  1A               OVOLUME:        LD      A,(DE)
0025'  2F                               CPL
0026'  E6 0F                            AND     15          ; Volume 0...15
0028'  F6 90                            OR      90h         ; Volume TOON 1
002A'  D3 06                            OUT     (6),A
002C'  DB 03                            IN      A,(3)
                                                            ; De totale vertraging is (BC) * B * 13 uSeconden
002E'  0A               ODELAY:         LD      A,(BC)      ; Naar low byte van de vertraging
002F'  5F                               LD      E,A
0030'  03                               INC     BC          ; Naar high byte van de vertraging
0031'  0A                               LD      A,(BC)
0032'  CB BF                            RES     7,A         ; Voorkom delay op "negatieve" tijden
0034'  57                               LD      D,A         ; DE bevat de vertraging 1...32768
0035'  06 FB            ODELAY_LOOP1:   LD      B,251       ; Constante op de juiste timing te krijgen (4MHz)
0037'  E3               ODELAY_LOOP2:   EX      (SP),HL     ; Verwisselen, dit kost tijd
0038'  E3                               EX      (SP),HL     ; Verwissel terug, dit kost nog meer tijd
0039'  10 FC                           DJNZ    ODELAY_LOOP2
003B'  1B                               DEC     DE
003C'  7A                               LD      A,D
003D'  B3                               OR      E
003E'  20 F5                            JR      NZ,ODELAY_LOOP1 ; Herhalen tot DE=0
0040'  3E 9F            OSOUND_OFF:     LD      A,9Fh
0042'  D3 06                            OUT     (6),A
0044'  DB 03                            IN      A,(3)
0046'  FB                              EI                  ; Nu mogen ze weer
0047'  C9                              RET

0048'  C9               STPON:         RET
0049'  C9               STPOFF:        RET

                        ;-----------------------------------------------------------------------
                                        END
```

d B A S E: RESEt

dBase und der RESET (Wolfgang Dexheimer, 6719)

Vorweg mein Dank an Olaf, ohne dessen Programm MONI2 die nun folgenden Erklärungen nicht zustande gekommen wären. Meine Konfiguration war also MONI2 und dBase 2.41 von Markt&Technik deutsche Version für den Schneider CPC 6128 angepaßt auf unsere Maschine (RAM43).

Ich habe die Diskussion um diesen Befehl mit Interesse verfolgt, da ich seit einiger Zeit diesen Befehl auch benötige (ab und zu muß ich in einem Datenbestand von 2,7 M-Byte wühlen). Leider konnte mich keine der vorgeschlagenen Lösungen überzeugen, also machte ich mich auf die Suche. Ich fand 'das Ei des Columbus' auch nicht, aber immerhin einiges:

Im Modus 1 von MONI2 kann man das geladene dBase starten und gelangt mit G von 100 bis 3CDA fast bis zum Beginn der dBase Eingabeschleife (keine Angst vor der Datumsabfrage und dem dBase-Logo, beides verschwindet nach betätigen der <RET>-Taste wieder und man kehrt zu MONI zurück; gleichzeitig wurde auch noch das dBase Overlay an seinen Platz geladen). Nun muß man im Einzelschritt mit T oder S weiterarbeiten, da bei Adresse 4700 der Stack durch dBase verändert wird. Doch ab 4703 kommt man mit G von 4703 bis 4793 recht schnell zum dBase Prompt, dem '.' und nach T bleibt der Cursor hinter dem T stehen und erwartet den dBase-Befehl. Bei mir also RESE C<RET>.

Mit G von 4796 bis 4700 wird der Befehl vollständig abgearbeitet und nach einmaligem betätigen von T oder S kann es mit G von 4703 bis 4793 wieder mit der Schleife weitergehen. Doch betrachten wir näher was zwischen 4796 und 4700 geschieht:
Es wird geprüft ob überhaupt etwas eingegeben wurde; bei nichts gehts direkt wieder zu 4700, sonst gehts in die Befehl-Such-Schleife. Die gültigen dBase-Befehle stehen in einer Tabelle ab 5EAB und wurden mit dem Overlay an diese Adresse gebracht. Das ganze sieht leserlich gemacht, etwa so aus:

```
5EAB: STORE FB5B3F00B855
5EB7: ELSE ED52
5EBE: ENDDO 0153 usw.
```

D.h. zunächst der Befehl in ASCII, gefolgt von einem Blank (=Ende Befehl), dann die Adresse (also ELSE wird bei Adresse 52ED abgearbeitet) und manchmal noch weiteren Informationen (siehe STORE).

Bleiben wir beim RESET: Das HL-Register wird mit der Adresse 58BC geladen; ein JP (HL) bringt uns dem Ziel näher. Hier erscheint dann ein echtes dB-Dämmerstündchen, und die Erklärung warum eine Laufwerksangabe bei diesem Befehl als **Großbuchstabe** zu erfolgen hat: Es wird nur geprüft ob etwas nach RESET folgt oder nicht. Wenn etwas folgt, so wird von diesem etwas hex 40 abgezogen und übergibt dies dann als Laufwerksnummer.

Bsp: C,
also hex 43-40=3 damit wird das richtige Laufwerk bearbeitet;
folgt hingegen c,
also hex 63-40=23 damit wird ein unmögliches Laufwerk übernommen.

Anmerkung: Der ':' hinter dem Laufwerksbuchstaben kann entfallen.

Der übergebene Wert wird nun etwas aufbereitet und dann an die BDOS-Funktion Nr. 25h (37dez) übergeben, dies ist der RESET MULTIPLE DRIVE Abschnitt. Damit wird das Login-Flag und die DiskRO-Kennung des entsprechenden Laufwerks (der entsprechenden Laufwerke) zurückgesetzt.
Dies klappt auch (entweder mit MONI im Bereich E029=DISKRO, E02B=LOGIN oder in dBase direkt mit dem KLICK-Monitor nachweisbar).

d B A S E: RESEt

Was bedeutet LOGIN bzw. DiskRO:

In dem Wort(=2Byte) LOGIN steht etwa 0000 0000 0100 0001 = 00 41 hex
dabei steht jedes Bit für ein LW P I HGFE DCBA
eine 1 steht für Laufwerk aktiv, eine 0 für nicht 'eingeloggt'.

Bei mir liegt dBase auf der EPROM-Disk G und ich habe vom Laufwerk A aus gestartet. Wurde etwa ein Befehl USE C:file benutzt, so ergibt sich folgendes Bild:
LOGIN = 0000 0000 0100 0101 = 00 45 hex.
Nach RESE C ergibt sich: 0000 0000 0100 0001 = 00 41 hex,
das Laufwerk wurde also 'ausgeloggt'.
DiskRO ist vom Aufbau her mit LOGIN identisch, nur steht hier 1 für dieses Laufwerk ist RO und 0 für dieses Laufwerk ist RW.

RESET setzt in beiden Kennungen das dem Laufwerk entsprechende Flag auf 0. Doch wie kommt dann die R/O - Fehlermeldung zustande?

Ausgelöst wird sie durch eine 1 beim entsprechenden Laufwerk im DiskRO-Flag. Wie kommt nun eine 1 an diese Stelle ?

Hier beginnt nun meine Vermutung die ich noch nicht nachweisen konnte. dBase besitzt für alle 16 möglichen Laufwerke (von A bis P) je 2 eigene FCB's (noch weiß ich nicht wie diese genau verwaltet werden, dies zu ergründen ist m.E. ein schier unendliches Unterfangen). Die Bereiche hierfür sind etwa 4416-4572 und 9280-989C. Will man nun mit einem Laufwerk nach BDOS 25 wieder arbeiten, so ist es zwingend nötig die 'Directory-Prüfsumme' nicht zu prüfen, sondern neu einzutragen. Dies geschieht zum Beispiel mit dem BDOS call open file. Hier jedoch scheint dBase zuzuschlagen. Es versäumt es das Laufwerk neu auszuwählen, und so wird vom BDOS die Directory-checksum nicht neu gesetzt, sondern verglichen. Da meist das Directory von zwei Disketten verschieden ist, wird automatisch das DiskRO-Flag gestzt, um ein versehentliches Einschreiben in diese Diskette zu verhindern. Dies geschieht immer dann, wenn die 'würg' R/O Meldung kommt (sie hat auch Ihr Gutes, denn sonst könnte wirklich versehentlich etwas auf eine beliebige Stelle auf der neuen Diskette geschrieben werden. Auf der alten Disk würde dies an der entsprchenden Stelle Sinn machen, aber auf der neuen ... Murphy läßt grüssen!). Immer wenn diese Meldung nicht erscheint ging dBase über den BDOS - call 0F (OPEN FILE), aber wieso dies nach dem RESET nicht immer geschieht ist mir noch schleierhaft.

Ausblick auf Patches: Selbst wenn der Fehler gefunden wird, halte ich sie nicht für sinnvoll. dBase läuft ja auf dem Schneider mit dem einen RESET fehlerfrei. Andererseits laufen Programme wie NewWord mit unserem DOS ohne obiges Problem, wo also sollte ein Patch etwa in der Art nach dem RESET lw sofort lw wieder einloggen stattfinden?

Für alle die bis hierher mitgelesen haben (trotz der hoffentlich nur orthographischen Fehler) nun mein kleines Programm in dBase (leider mit 2 * RESET):

d B A S E: CHR(0) und Variablen

```
* LOGIN.CMD
IF .NOT. ":"$suchf
 STOR disk+":"+suchf TO suchf
ENDI
STOR T TO login
DO WHIL .NOT. FILE("&suchf")
 § 20,0 SAY "Bitte Diskette mit "+suchf+" ins Laufwerk "+disk+"einlegen"
 § 22,0 SAY "-> weiter mit bel. Taste <-"
 WAIT
 RESE &disk                    <- LOGIN und DISKRO auf NULL, damit wird
 STOR T TO login                bei nächster Leseoperation Dir neu ein-
 § 20,0 SAY CHR(5)              gelesen und gegenebenfalls DISKRO des
 § 22,0 SAY CHR(5)              Laufwerks auf 1 gesetzt.
ENDD
IF login
 RESE &disk                    <- LOGIN und DISKRO auf NULL, beim
 STOR F TO login                nächsten LESEN kann DISKRO nicht ge-
ENDI                            setzt werden, da DISK nicht ge-
RETU                            wechselt wurde und somit die Prüf-
                                summe gleich ist!
```

Bemerkungen:
In disk wird der Laufwerksbuchstabe und
in suchf der Name des neu zu bearbeitenden Files übergeben.
CHR(5) löscht bis Ende der Zeile.

Hierzu folgendes Beispiel:

```
*USE falls noch File offen ist
STOR "ADRESS1.DBF" TO suchf
STOR "C" TO disk
DO LOGIN
*USE suchf  und damit bearbeiten und schliessen hoffentlich ohne R/O.
```

dBase: CHR(0) und die Variablen (Wolfgang Dexheimer, 6719)

In einem String muß der CHR(0) vermieden werden, da er in dBase wie z.B. auch in C das Ende eines Strings kennzeichnet.

```
Also         STOR "Dies"+CHR(0)+" ist schon das Ende" TO sonicht,
liefert für  ? sonicht
nur          Dies
```

Bei den Variablen gilt es folgende Beschränkungen zu beachten:

```
max. Anzahl Variablen      =    64
max. Länge einer Variablen =   254 Zeichen
            ergibt 254*64 = 16256 Zeichen,
```

leider steht aber nur ein Bereich von **1536 Bytes** zur verfügung also Vorsicht Vorsicht (immerhin 1536 ist auch schon viel? ODER ...)
Markt&Technik Handbuch dBase Abschnitt AUSDRÜCKE 3/4 letzter Abschnitt (das Kleingedruckte!!!).

T u r b o - P a s c a l: Text-File verlängern / retten Kommandozeile

Text-File verlängern mit TURBO-Pascal (Wolfgang Dexheimer, 6719)

Hier ein kleines TURBO Pascal (3.0) Programm zum Thema Text-File.
Will man in Pascal einen Text-File bearbeiten, so muß er über RESET oder REWRITE geöffnet werden. Mit RESET kann man einen Text-File aber nur lesen und mit REWRITE wird einer neu angelegt, d.h. ein bestehender zerstört! Da hilft nur RENAME weiter. Das folgende Programm erledigt die Aufgabe automatisch. Ein Beispielprogramm habe ich der Diskette beigefügt; es heißt TEXTVERL.PAS und benötigt TEXTVERL.INC (= Prozedur von unten):

```
(* Im Hauptprogramm wird benötigt:
      TYPE
       FName=STR(.14.);
      VAR
       TF :TEXT;     als FileVariable
       name :FName; als FileName;      *)

PROCEDURE Text_File_verL(name :FName);
 VAR
  TFA  :TEXT;
  namealt :FName;
  Satz :STRING(.255.);
 BEGIN
  ASSIGN(TF,name);
  (*$I-*) RESET(TF); (*$I+*)
  IF IOResult=0                                (* existiert File mit 'name' ? *)
   THEN BEGIN
    CLOSE(TF);
    namealt:=COPY(name,1,POS('.',name))+'ALT';    (* mache Extension zu ALT *)
    RENAME(TF,namealt);                  (* Bsp.: aus FILE.TXT wird FILE.ALT *)
    ASSIGN(TF,name);  REWRITE(TF);           (* mache neuen File mit 'name' *)
    ASSIGN(TFA,namealt); RESET(TFA);
    WHILE NOT EOF(TFA) DO                    (* übernimm alles aus xxxx.ALT *)
     BEGIN
      READLN(TFA,Satz);
      WRITELN(TF,Satz);
     END;
    CLOSE(TFA); ERASE(TFA);                          (* lösche xxxx.ALT *)
       END
  ELSE REWRITE(TF);                                (* lege 'name' neu an *)
 END;
(* Achtung: der File 'name' ist nun geöffnet und kann beliebig mit
            WRITE(TF, ...      beschrieben werden.
            Am Ende CLOSE(TF) nicht vergessen.                    *)
```

Anm.d.HzN.: o.g. Programm samt Demo ist auf CLUB.038

Retten der Kommandozeile unter TURBO-PASCAL (II) (Olaf Krumnow, 2050)

Holger Hansen fragte im letzen Info nach einer Möglichkeit, mittels PARAMCOUNT und PARAMSTR auch Kommandozeilen auszuwerten, die länger als nur 31 Zeichen sind. Eine erste Lösung aus ur-ur-ur-alter Zeit hat Herbert ja wieder ausgegraben. Doch die bietet keinen Zugriff mittels ParamStr, der ja doch gewünscht ist. Noch dazu ist sie umständlicher als nötig. Darum jetzt also eine kurze und schmerzlose Lösung. Zu den kleinen Einschränkungen komme ich dann später.

T u r b o - P a s c a l: retten Kommandozeile

Das Prinzip ist gleich geblieben: Die Initialisierung wird um das Retten der Kommandozeile verlängert. Der String wird immer noch hinter FREE kopiert, soweit dort Platz ist.

```
0103 D9             EXX                 ; RETTE REGISTER
0104 3A 80 00       LD   A,(80H)        ; LÄNGE KOMMANDOZEILE
0107 3C             INC  A
0108 4F             LD   C,A
0109 06 00          LD   B,0            ; NACH BC
010B ED 5B 40 00    LD   DE,(40H)       ; START FREE
010F 2A 46 00       LD   HL,(46H)       ; TOAM
0112 ED 52          SBC  HL,DE
0114 ED 42          SBC  HL,BC          ; GENUG PLATZ ?
0114 38 0A          JR   C,0122H        ; NEIN
0114 21 80 00       LD   HL,80H         ; START KOMMANDOZEILE
011B ED B0          LDIR                ; KOPIERE ZEILE
011D D9             EXX
011E 22 D2 00       LD   (0D2H),HL      ; ORIGINAL-BEFEHL NACHHOLEN
0121 C9             RET
0122 AF             XOR  A              ; A=0
0123 12             LD   (DE),A         ; SETZE LÄNGE AUF NULL
0124 18 F7          JR   011DH          ; UND FERTIG
```

und den Einsprung in die Routine

```
0364 CD 03 01       CALL 0103H          ; VERBIEGE AUF EIGENE ROUTINE
```

und letzlich muß noch etwas bei ParamCount und ParamStr gemacht werden:

```
1F9D 2A 40 00       LD   HL,(40H)       ; LADE START KOMMANDOZEILE
1FA0 3E 7F          LD   A,7FH          ; MAXIMALE ZEILENLÄNGE 127 ZEICHEN

1FA6 06 7F          LD   B,7FH

1FD7 54             LD   D,H            ; KLEINE KORREKTUR IM CODE, DAMIT
1FD8 26 00          LD   H,0            ; AUCH ALLE RICHTIG KLAPPT
```

Soweit die Änderungen, die entweder mit MONI als Assemblereingabe oder mit sonstigen Debuggern in HEX-Eingabe durchgeführt werden kann. Was passiert im einzelnen? Bei 0364 wird in der Initialisierung ein LD-Befehl mit dem CALL auf unsere kleine Routine überschrieben (die Adresse kann übrigens auch eine andere als 0103 sein, falls dort schon ein anderer Patch steht). Diese prüft, ob zwischen FREE und TOAM genug Platz ist. Wenn ja, so wird die Kommandozeile dorthin kopiert. Ist nicht genügend Platz, so wird lediglich ein NULL-Byte gesetzt, damit ParamCount und ParamStr annehmen, daß keine Kommandozeile eingegeben wurde.

Mit diesen Änderungen dürfte es ohne Änderungen in den Anwenderprogrammen möglich sein, auch Kommandozeilen mit mehr als 31 Zeichen zu bearbeiten (maximal 127, aber mehr läßt CP/M ohnehin nicht zu). Folgende Einschränkungen sind zu beachten: Die gerettete Kommandozeile wird nicht durch Ändern des FREE-Zeigers geschützt. Daher kann evtl. ein gestartetes KLICK-Programm die Kommandozeile zerstören, wenn es Platz zwischen TOAM und FREE beansprucht. Dieser Fall dürfte zwar selten vorkommen, aber es ist eine Möglichkeit. Daher würde ich empfehlen, sofort nach Programmstart die Kommandozeile komplett auszuwerten und in sichere Variablen zu kopieren.

S o f t w a r e: LISP / NSWEEP und Squeeze

LISP (Wolfgang Dexheimer, 6719)

In den Heften November bis Januar erschien in der Zeitschrift TOOLBOX ein 'Little LISP' - System. Ein LISP in TURBO Pascal 4.0 für AT's geschrieben. Da ich die Sprache mag, habe ich obige Programme für TURBO Pascal 3.0 und unsere 'schwarze Kiste' umgeschrieben. Leider war der Heap damit schnell gefüllt (viel viel Rekursion). Deshalb habe ich einige kleine Veränderungen vorgenommen (ATOM und STRING Längen gekürtzt etc. und eine kleine Heap-Verwaltung angefügt). Immerhin laufen so die gefürchteten Towers of Hanoi bis 5 immerhin in 21 Sekunden (ein AT unter Turbo 4.0 braucht dafür 3,2 sec. doch auch sein Heap ist schon bei 9 am Ende, trotz 640k Speicher und etwas kompakterem Code!). Eine Erhöhung der Zahl um 1 bedeutet etwa den doppelten Speicherverbrauch!
Wenn im Club Interesse besteht (es gibt auch ein kleines PROLOG Prgrämmchen in LISP dazu) und der DMV-Verlag sein Einverständnis erklärt könnte man das Programm samt dem Pascal-Quellcode auf eine Club-PD übernehmen.

NSWEEP und Squeeze (Herbert zur Nedden, 2000)

Kürzlich hatte ich das Problem, daß ich auf der Inhaltsverzeichnis-Diskette für die Public-Domain-Disketten (auch CLUB.000 genannt), die Inhaltsverzeichnis-Dateien squeezen (d.h. quetschen) zu müssen, da die Scheibe zu voll wurde.

Also NSWEEP aufgerufen, kurz getaggt und dann 'Q' 'S' 'F1:', was mir folgendes bescherte:

```
SQ/USQ    --> F0: -CLUB   .001 to F1:(-CLUB.0Q1)
SQ/USQ    --> F0: -CLUB   .002 to F1:(-CLUB.0Q2)
u.s.w.
SQ/USQ    --> F0: -CLUB   .011 to F1:(-CLUB.0Q1)
```

Wer genau hinsieht, wird feststellen, daß -CLUB.001 und -CLUB.011 beide auf die selbe Datei gesqueezed wurden: -CLUB.0Q1. Das liegt offentsichtlich daran, daß das 'Q', welches in der Extension das mittlere Zeichen überschreibt, damit eine Namensänderung stattfindet, eine in diesem Fall sehr wesentliche Ziffer vernichtet.

Das war natürlich kein Zustand, da ich diesen Datenverlust eigentlich nicht in Kauf nehmen wollte - obwohl so zumindest das Platzproblem gelöst würde. Also muß das 'Q' woanders hin! Die Variante, daß ich die Dateien einzeln squeeze und das Ergebnis dann umbenenne gefiel mir nicht. Klar, ich hätte immer alle, die als mittleren Namen der Extension eine '0' haben, squeezen, und das Resultat mit WildCard-Rename in Ordnung bringen können:

```
'W' 'T' '-*.?0?'      alle mit '0' in der Mitte der Extension markieren
'Q' 'S' 'F1:'         quetschen nach F1:
'L' 'F1:'             gehe nach F1:
'R' '*'               Wildcard-Rename, d.h. gleiche viele umbenennen
'-*.?Q?'              und zwar alle '-*.?Q?'
'Q*.?0?'              in            'Q*.?0?'
                      also '-CLUB.0Q1'  in 'QCLUB.001'
                      also '-CLUB.0Q2'  in 'QCLUB.002'
                      u.s.w.
```

Und nun das gleiche von vorne für alle mit einer '1' mitten in der Extension, dann die mit der '2' u.s.w. bis zur '9'.

Aber auch das ist mir zu blöd.

S o f t w a r e: NSWEEP und Squeeze

Fazit:

Olaf's MONI mußte her, um herauszufinden, wo dieses den Dateinamen für das Ergebnis des Squeeze erzeugt, d.h. 1 bis 3 'Q's in den Dateinamen schreibt (3 'Q's werden eingestellt, falls der Dateiname keine Extension hatte). Also mußte ich in MONI suchen, wo ein 'Q' entweder in den Akku oder nach (HL) geladen wird, da mir einer dieser Befehle am wahrscheinlichsten erschien. Flugs fand ich folgenden Code:

```
0C81  LD   DE,2BEAH       ; Lade Adresse Dateinamen-Puffer -1
0C84  LD   B,0CH          ; B=12
0C86  LD   (HL),00H
0C88  CALL 1AF9H          ; Trage Dateinamen in Puffer ein
0C8B  LD   HL,2BF3H       ; Lade Adresse Datei-Extension in Puffer
0C8E  LD   A,(HL)         ; holen
0C8F  AND  7FH            ; Bit 7 weg
0C91  CP   20H            ; ist's ein Blank, d.h. Datei ohne Extension
0C93  JP   NZ,0C9BH       ; nein, hat Extension
0C96  LD   (HL),51H       ; ab hier 3x 'Q' in Extension
0C98  INC  HL
0C99  LD   (HL),51H
0C9B  INC  HL             ; ab hier 1x 'Q' in Mitte der Extension
0C9C  LD   (HL),51H
0C9E  CALL 1CC2H          ; weitermachen
```

Ich habe mich entschlossen, daß ich das 'Q' an die erste Stelle des Dateinamens, d.h. an die Stelle des '-' haben wollte, d.h. meine NSWEEP-Session sollte wie folgt aussehen:

```
SQ/USQ    --> F0: -CLUB   .001 to F1:(QCLUB.001)
SQ/USQ    --> F0: -CLUB   .002 to F1:(QCLUB.002)
u.s.w.
SQ/USQ    --> F0: -CLUB   .011 to F1:(QCLUB.011)
```

Dazu habe ich den o.g. Ausschnitt von NSWEEP wie folgt geändert:

```
0C81  LD   DE,2BEAH       ; Lade Adresse Dateinamen-Puffer
0C84  LD   B,0CH          ; B=12
0C86  LD   (HL),00H
0C88  CALL 1AF9H          ; Trage Dateinamen in Puffer ein
0C8B  LD   HL,2BEBH       ; Lade Adresse Datei-Namen in Puffer
0C8E  NOP                 ; all diese Befehle sind überflüssig
u.s.w.  (alles NOP)
0C9B  NOP                 ; all diese Befehle sind überflüssig
0C9C  LD   (HL),51H       ; 1x 'Q' an erste Stelle des Dateinamens
0C9E  CALL 1CC2H          ; weitermachen
```

Wem eine andere Stelle oder Mimik oder gar ein anderer Buchstabe oder ... oder ... besser gefallen sollte: Platz ist da ja immerhin etwas.

S u p e r C a l c: Wider dem Chaos (/X-Befehl)

SuperCalc Programme: Kampf dem Chaos (Kurt-Bernd Rohloff, 8000)

SuperCalc Programme (die .XQT Dateien, die mit /X ausgeführt werden) sahen bei mir bisher immer recht kryptisch aus. Eine spätere Änderung wurde daher manchmal zum reinen Glücksspiel. Also habe ich mich mal hingesetzt und überlegt, wie man das Aussehen dieser Programme verbessern kann. Was mir dabei eingefallen ist, will ich euch nicht vorenthalten.

Laut Manual kann man die SC Programme so eintippen, wie man die Befehle auch im Dialog eingetippt hätte. Das stimmt, sieht man von der in Programmen nicht möglichen Verwendung der ESC Taste einmal ab. Es ist aber gar nicht so günstig, wie es zunächst scheint. Nehmen wir mal an, ich stehe mit dem Zellcursor in Zelle B2 und möchte die Formel in dieser Zelle mit /R(eplicate) nach B3:B15 kopieren. Dabei soll die erste Zellreferenz in der Formel nicht, die zweite und dritte jedoch angepaßt werden. Danach soll das Spreadsheet neu berechnet werden. Im Dialog würde ich dann eingeben:

```
/R              RET fuer die aktuelle Zelle
B3:B15,ANYY!    hier kein RET, sondern gleich der naechste Befehl
```

Das ist insofern unschön, als daß dadurch der Befehl über mehrere Zeilen "zerfleddert" wird. Es wäre doch wesentlich übersichtlicher, wenn man einen kompletten Befehl pro Zeile schreiben könnte. Das geht auch! Man kann nämlich auch im Dialogmodus nach Eingabe eines Parameters statt mit RET auch mit dem **Komma** fortfahren. Das bezieht sich auf solche Parameter, die aus mehr als einem Buchstaben bestehen, wie z. B. die Angabe einer Zelle oder eines Bereichs. Im Dialogmodus bevorzuge ich weiterhin das RET, aber in Programmen sieht das Komma besser aus. Am Ende des Befehls kann man nicht immer ein RET (also Zeilenwechsel) setzen, weil dadurch auch der Zellcursor in die augenblickliche Richtung weiterhüpft, und dies könnte nachteilig sein. Das läßt sich jedoch einfach abstellen, indem wir am Anfang des Programms den **NEXT Modus** ausschalten (/GN) und am Schluß wieder einschalten (ebenso). Dabei gehe ich davon aus, daß das Spreadsheet zunächst im NEXT-ON Modus ist. Dies ist die Voreinstellung von SC und daran erkenntlich, daß links unten vor der Nummer der aktuellen Zelle ein kleines Pfeilsymbol zu sehen ist ("<", ">", "v" oder "^"). Wenn wir den Zellcursor im Programm relativ zur aktuellen Zelle weiterbewegen wollen, können wir eines der vier eben genannten Symbole eingeben (z. B. drei Zellen nach unten und dann eine nach rechts: vvv>). Leider kann man den NEXT Modus nur **um**schalten, nicht jedoch definitiv ein oder aus. Unser obiges Beispiel könnte nun so aussehen:

```
/GN             ab hier schadet ein RET nichts mehr
/R,B3:B15,ANYY  Replicate und alles, was dazu gehoert
!               Rechne
/GN             Originalzustand wiederherstellen, eventuell kein RET
```

Ob man die letzte Zeile auch noch mit einem RET abschließen kann oder nicht, hängt von dem jeweiligen Anwendungsfall ab. Ich habe ein Spreadsheet, bei dem der Cursor am Schluß des Programms dort stehenbleiben muß, weil er beim nächsten Programmaufruf genau dort wieder erwartet wird. In vielen anderen Fällen mag aber ein Weiterspringen des Zellcursors nicht schaden, und dann sollte man die letzte Zeile auch mit RET abschließen.

Normalerweise wird ein Spreadsheet ja viele leere Zellen enthalten. Nehmen wir mal an, B2 sei eine solche. Nehmen wir weiterhin an, es sei problemlos möglich, den Cursor dorthin zu setzen. (Dies ist nicht immer möglich, weil dann ja seine ursprüngliche Position verloren geht.) Jetzt können wir in diese frei Zelle den Text "MTX Computer" schreiben. "Wozu?" wirst du nun frage, "ich weiß auch so, an welchem Rechner ich sitze". Gewiß doch, aber der Clou ist eben, daß wir hier **irgendeinen** Text eintragen können. Damit haben wir uns ein Hintertürchen eröffnet, um die SC Programme mit **Kommentaren** zu versehen! Am Schluß des Programms kann man die Kommentarzelle mit /B wieder putzen.

S u p e r C a l c: Wider dem Chaos (/X-Befehl)

Diesen Kommentar sieht der Benutzer am Bildschirm in B2 und, solange der Cursor dort steht, auch unten, wo der Inhalt der aktuellen Zelle angezeigt wird. Dies kann man ausnutzen, um ihn über einzelne Verarbeitungsschritte zu informieren. Falls das Programm zu schnell abläuft, um die Texte zu lesen, kann man es mit **CTRL-D** um ca. eine halbe Sekunde verzögern. Obiges Beispiel könnte dann so aussehen:

```
/GN              ab hier schadet ein RET nichts mehr
=B2
"SC Demoprogramm
<CTRL-D>         warte ein Weilchen
/R,B3:B15,ANYY   Replicate und alles, was dazu gehoert
!                Rechne
"Das war's
<CTRL-D><CTRL-D>
/BB2
/GN              Originalzustand wiederherstellen, eventuell kein RET
```

Nach den oben dargelegten Methoden habe ich mein Programm zur Verbrauchsermittlung aus Info 27-54 noch einmal überarbeitet. Es sieht nun so aus:

```
/GN
=A2
"eine neue Zeile in das VERBR Spreadsheet einfügen (muß im TAB Mode sein)
/GT
=A10
/IR
/FR,TR
>NA
>NA
>NA
>NA
>
"jetzt sind wir in F10
/RF11,F10
>
/RG11,G10,AYN
>NA
/RI11,I10
/RJ11,J10
/BA2
=A10
/GT
/GN
```

Hierbei wird nun auch die letzte Zeile mit RET abgeschlossen. Zum Ausgleich dafür gehen wir davor nach A10 (statt B10) und verwerten die Tatsache, daß die aktuelle Richtung "nach rechts" ist. Es ist zwar länger als die alte Fassung (vergl Info 27-54), dafür aber erheblich übersichtlicher.

Übrigens, was passiert, wenn ein SC Programm wieder den /X Befehl enthält? Nun, das gleiche wie wenn in einer SUBMIT Datei wieder ein SUBMIT vorkommt. Das Programm wird zwar ausgeführt, aber es kehrt (leider) nicht zurück. Man kann aber auch aus dieser Not eine Tugend machen. Etwa beim Testen eines längeren Programms. Dann ist es oft günstig, nicht gleich das ganze Programm ablaufen zu lassen (Politiker sprechen in solchen Fällen von "Schadensbegrenzung"). Man kann dann nach einem logischen Abschnitt einen /X Befehl setzen, um ein Dummy-Programm auszuführen und damit das Hauptprogramm an dieser Stelle beenden. Das Dummy-Programm enthält dann nur die "Aufräumbefehle" wie etwa /GN.

S u p e r C a l c: Nullstellenberechnung

Nullstellenberechnung mit SuperCalc

(Kurt-Bernd Rohloff, 8000)

Aufmerksame Leser dieser Blätter mögen bei der Überschrift vielleicht gedacht haben, hier wird versehentlich mein Artikel aus dem vorigen Info nochmal abgedruckt. Mitnichten! Vielmehr möchte ich hier ein anderes (und besseres) Verfahren zu demselben Zweck vorstellen. Dieses Verfahren hat sich ein gewisser Sir Isaac ausgeheckt, weshalb es meistens "**Newton Verfahren**" oder auch die "**Tangentenmethode**" genannt wird.

Auch hier wird davon ausgegangen, daß die fragliche Funktion eine einfache Nullstelle hat, also die x-Achse schneidet. Ebenso wie bei der "Regula Falsi" muß man auch bei diesem Verfahren die ungefähre Lage der NST kennen, denn man muß einen Anfangswert vorgeben. Dieser sollte der wahren NST bereits möglichst nahe liegen, sonst kann das Newton Verfahren versagen (die genauen Bedingungen für eine Konvergenz möge der geneigte Leser in der mathematischen Literatur nachschlagen). Gegenüber der Regula Falsi führt das Newton Verfahren dann aber schneller, d. h. mit weniger Iterationen, zum Ziel. Ein Nachteil des Newton Verfahrens ist jedoch, daß man außer der Funktion f(x) selbst auch noch ihre Ableitung f'(x) benötigt.

Der Grundgedanke des Newton Verfahrens ist folgender:

a) Zu dem Anfangswert x wird der zugehörige Funktionswert f(x) berechnet. Durch den Punkt (x, f(x)) wird die Tangente an die Funktion gelegt.
b) Die Tangente schneidet die x-Achse an einer Stelle x_1, die sich leicht berechnen läßt.
c) Man wiederholt die Schritte a und b, wobei das x_1 die Rolle des Anfangswerts übernimmt. Sobald der Funktionswert f(x) genügend nahe bei Null liegt, bricht man ab und nimmt das zuletzt berechnete x_1 als Näherung für die gesuchte Nullstelle.

Die Differenz zwischen den letzten beiden x-Werten kann wieder als Schätzung für den begangenen Fehler dienen. Das Spreadsheet für das Verfahren sieht so aus:

```
   1   A   11     B    11         C         1
  11Newton-Verfahren
  21
  31         Start/Stop                     0
  41         Anfangswert:                 1.3
  51
  61         input                        1.3
  71         f(x)                      -1.173
  81         f'(x)                      -3.73
  91         output         .9855227882037534
```

Die Zelle C3 übernimmt wieder die Funktion des Starters. Eine Null dort bringt das Spreadsheet in den Ausgangszustand, d. h. der Anfangswert wird nach C6 kopiert. Damit wird f(x) und f'(x) berechnet und daraus der Tangentenschnittpunkt (output). Jeder andere Wert in C3 führt dann zu der Iterationsschleife, d. h. bei jeder Neuberechnung mit "!" wird der output oben wieder als input hineingesteckt. Die Zellen C6 und C9 sollte man schützen. In C7 wird die Formel für die Funktion eingetragen und darunter ihre Ableitung. Diese müssen von Fall zu Fall geändert werden. Ich habe als Beispiel eine sehr einfache Funktion mit den NST -1, 1 und 3 genommen. Die Formeln dazu sehen wie folgt aus:

S u p e r C a l c: Nullstellenberechnung

```
   1   A   11     B    11         C        1
 11Newton-Verfahren
 21
 31          Start/Stop   0
 41          Anfangswert: 1.3
 51
 61          input        IF(C3=0,C4,C9)
 71          f(x)         (C6*C6-1)*(C6-3)
 81          f'(x)        3*C6*(C6-2)-1
 91          output       C6-C7/C8
```

Die Formeln sind eigentlich einfacher als bei der Regula Falsi, aber dafür muß man hier mehr Vorarbeit leisten. Außerdem kann beim Newton Verfahren der output bei ungünstigen Bedingungen auch von der NST weglaufen, anstatt dorthin zu konvergieren. Wer Lust hat, kann ja mal ein Non-Plus-Ultra Spreadsheet entwerfen, das beide Verfahren kombiniert. Mit der Regula Falsi könnte man sich zunächst grob an die NST herantasten und dann mit dem letzten Näherungswert in das Newton Verfahren 'reingehen, das dann mit einer oder zwei Berechnungen eine sehr gute Näherung liefert. Mal sehen, ob sich einer aufrafft.

A s s e m b l e r: Arrays und Records

Komplexe Datenstrukturen mit MACRO-80/SLRASM (Olaf Krumnow, 2050)

Einer der Vorteile der höheren Programmiersprachen sind komplexe Datenstrukturen. Das sind insbesondere Arrays und Records (entspricht den STRUCTures in C). Diese Datentypen lassen sich mit guten MACRO-Assemblern aber auch auf Assemblerebene nachbilden, wenn auch mit Einschränkungen. Nichtsdestotrotz ist es gelegentlich eine gute Arbeitserleichterung, wenn diese Datenstrukturen zur Verfügung stehen.

Arrays:

Ein Array ist lediglich eine lineare Anordnung gleicher Datenstrukturen. Das ist noch sehr einfach zu definieren. Das dazugehörige MACRO sieht so aus:

```
01:  array     macro     idx1,idx2,type,name
02:  name:
03:  local     idx
04:  idx       aset      idx1
05:            rept      (idx2-idx1)+1
06:            setlab    name,%(idx)
07:            type
08:  idx       aset      idx+1
09:            endm
10:            endm
```

Zur Erklärung: Die Zahlen am Anfang einer Zeile dienen nur zum einfacheren Verweis auf die Zeile, die ich hier in der Beschreibung meine. Sie dürfen NICHT mit abgetippt werden.
Zeile 01 enthält den MACRO-Kopf mit Namen und Parameterdefinition. Das array-MACRO hat vier Parameter: Den Feld-Namen (name), Start- und Endindex (idx1,idx2) sowie den Datentyp (type) des Feldes. In Zeile 02 wird einmal der Grundname des Feldes definiert, um auf das Feld im ganzen zugreifen zu können. In Zeile 03 wird eine lokale Variable deklariert, die zum Fortzählen der Indexnummer benötigt wird. In Zeile 04 wird dieser lokalen Variable der Startwert zugewiesen (Startindex). In Zeile 05 wird ein sogenanntes REPEAT-MACRO definiert, das entsprechend des übergebenen Parameters wiederholt wird. In diesem Fall wird es über jedes Feldelement wiederholt. In Zeile 06 wird ein Label erzeugt, das zusammengesetzt ist aus dem Feldnamen und der Indexnummer. Dazu ist ein eigenes MACRO erforderlich, das gleich beschrieben wird. In Zeile 07 wird der Datentyp definiert. Das können einfache DB-Anweisungen oder auch DS-Anweisungen sein, also alles, was mit dem Assembler in einer Zeile unterzubringen ist. In Zeile 08 wird die lokale Variable um 1 auf den nächsten Index erhöht. Die Zeilen 09 und 10 beenden letztlich das REPEAT-MACRO und das array-MACRO. Das MACRO, das in Zeile 06 aufgerufen wird, sieht so aus:

```
11:  setlab    macro     lab,num
12:  &lab&_&num&_:
13:            endm
```

Es sieht nicht gerade sehr kompliziert aus, läßt sich aber leider nicht in das array-MACRO integrieren. Das hängt mit der Parameterauswertung zusammen. Zur Bildung des Labelnamens brauchen wir die Indexnummer. Diese ist in idx definiert. Der Ausdruck %(idx) ergibt den Zahlenwert der lokalen Variable, aber leider nur bei der Parameterübergabe an ein MACRO. Daher also der Umweg über ein zusätzliches MACRO.

A s s e m b l e r: Arrays und Records

Jetzt können wir uns beliebige Felder definieren. Ein einfaches Byte-Feld sieht so aus:

```
        array    0,10,<db 0>,Feld
```

Dabei sind die spitzen Klammern um das <db 0> absolut erforderlich. Auf keinen Fall dürfen Stattdessen etwas andere Klammern oder Anführungszeichen verwendet werden. Mit diesem Aufruf wird jetzt ein 11-elementiges Feld erzeugt. Zum Zugriff werden folgende Labels erzeugt:

```
Feld:              ; ist der Feldname
Feld_0_:           ; erstes Element
Feld_1_:           ; zweites Element
   :
   :
Feld_10_:          ; elftes Element
```

Leider ist es nicht möglich, den Index mit den eckigen Klammern zu umgeben, wie es in höheren Programmiersprachen der Fall ist. Deshalb habe ich die Unterstriche gewählt. Probiere einmal das obige Macro aus und erzeuge mit dem Assembler ein PRN-File. Dort kannst Du Dir genau angucken, wie welche Labels erzeugt werden und wie das MACRO genau arbeitet. Weiter Beispiel-Aufrufe sind:

```
     array     1,5,<ds 25,0>,Strings
     array     100,130,<dw 100>,Tabelle
usw.
```

Records bzw. Structures:

Ein weiterer beliebter und praktischer Datentyp sind die Records. Hier sind mehrere Elemente unterschiedlicher Datentypen zusammengefaßt, die alle über einen globalen Namen erreichbar und bearbeitbar sind. Ein typischer Record ist ein Datensatz mit Daten über eine Person, z.B. in Vereinen:

```
Person = record
     Name      : String;
     Vorname   : String;
     GebDatum  : Datum;
     Nummer    : integer;
end;
```

Als MACRO läßt sich das jetzt so definieren:

```
01:  Person          macro     n
02:  n:
03:  n&.Name:        ds    30
04:  n&.Vorname:     ds    20
05:  n&.GebDatum:    ds    3
06:  n&.nummer:      dw    0
07:  n&.len          equ   $-&n
08:                  endm
```

In Zeile 1 steht wieder der MACRO-Kopf. Der Parameter n stellt den Namen dar, unter dem im Programm auf die Variable zugegriffen wird. Das entsprechende Label wird in Zeile 02 definiert. In den Zeilen 03 bis 06 werden die benutzten Datentypen definiert. In Zeile 07 wird noch ein Label definiert, das die Länge der Struktur wiederspiegelt, damit einfach ganze Strukturen verschoben werden können. Zeile 08 beendet die MACRO-Definition.

A s s e m b l e r: Arrays und Records

Durch den Aufruf

```
        Person    Neu
```

enstehen jetzt folgende Labels, mit denen auf die einzelnen Elemente zugegriffen werden kann:

```
Neu:                ; für die gesamte Struktur
Neu.Name:
Neu.Vorname:
Neu.GebDatum:
Neu.Nummer:
Neu.len             ; für die Länge
```

Gegeben den Fall, daß für eine Datumsangabe die folgende Struktur definiert ist:

```
Datum     macro     n
n:
n&.Tag:   db   0
n&.Monat: db   0
n&.Jahr:  db   0
          endm
```

so kann Zeile 05 ersetzt werden durch

```
          Datum     n&.GebDatum
```

In diesem Fall wird das Label also als Parameter an die nachfolgende Struktur übergeben. Die enstehenden Labels (nach wie vor mit obigem Aufruf) wären dann

```
Neu.GebDatum.Tag:
Neu.GebDatum.Monat:
Neu.GebDatum.Jahr:
```

Hier muß man dann schon sehr auf die maximal mögliche Labellänge des Assemblers achten, um keine doppelt definierten Labels zu erhalten. Falls es doch mal kritisch zu werden droht, können als erstes die Punkte in den Definitionen weggelassen werden. Sie sind absolut nicht erforderlich und dienen eigentlich nur der optischen Trennung der einzelnen Namenselemente.

Kombinationen der obigen Datentypen:

Es ist leicht möglich, Records zu schachteln, wie das ja auch schon gezeigt wurde. Es ist auch sehr leicht möglich, Arrays als Datenstruktur in einem Record zu definieren. Der Aufruf sieht dann ähnlich wie bei geschachtelten Records aus:

```
     array     1,5,<db 0>,n&.Feld
```

Das würde dann in dem Person-Record die zusätzlichen Labels

```
Neu.Feld_1_
   :
   :
Neu.Feld_5_
```

ergeben.

A s s e m b l e r: Arrays und Records

Andersrum wird es schon schwieriger. Mit der obigen array-MACRO-Definition ist es nicht möglich, Felder von Records zu definieren, denn der Name des Feldelements ist der Basisname des Records. Aber ein leicht geändertes Macro ermöglicht auch dieses. Es muß lediglich die Zeile 07 ersetzt werden durch

```
    Xtype     type,name,%(idx)
```

das ganze MACRO umbenannt werden auf Xarray (schließlich ist das andere ja nicht überflüssig geworden) und das neue MACRO Xtype definiert werden. Es sieht so aus:

```
01: Xtype     macro     name,lab,num
02:           name      &lab&_&num&_
03:           endm
```

Durch dieses MACRO wird der Labelname als Name des Records übergeben. Die Ähnlichkeit zum MACRO setlab ist nicht zu übersehen. Ein Feld von Personen sieht also jetzt so aus:

```
    Xarray    1,5,<Person>,Liste
```

und hat u.a. folgende Labels:

```
Liste_1_.Name
Liste_1_.Vorname
Liste_1_.GebDatum.Tag
Liste_1_.GebDatum.Monat
Liste_1_.GebDatum.Jahr
Liste_1_.Nummer
Liste_1_.len
Liste_2_.Name
   :
   :
Liste_5_.Nummer
Liste_5_.len
```

Was will man mehr?

Einschränkungen:

Diese sind schnell erwähnt, beeinträchtigen die von den Hochsprachen gewohnte Arbeit aber doch ziemlich. Alle mit obigen MACROs definierten Variablen sind statisch (und müssen es sein). Dynamische Variablen sind damit nicht zu bearbeiten. Das Problem ist aber leicht zu umgehen, indem die dynamische Variable kurzfristig in eine statische kopiert wird, was mit LDIR ja sehr schnell geht (dafür ist dann auch die Definition einer Länge ganz praktisch). Wenn der Datensatz verändert wird, sollte man das Zurückkopieren nicht vergessen!
Bei Arrays kann natürlich nicht auf einen berechneten Index zugegriffen werden,

```
    ld   a,(Feld_HL_)
```

ist also leider nicht möglich (logisch, oder?). Hierfür muß wieder auf althergebrachte (und einem jedem Assembler-Programmierer bekannte) Routinen zurückgegriffen werden.

Assemblerkurs

4.4. Rechnen mit dem Z80 (Andreas Nickel)

4.4.1. Binärarithmetik natürlicher Zahlen

Um das Prinzip der binären Arithmetik zu verstehen, beginnen wir mit dem einfachsten Teil, der Addition. Den Rechenbereich grenzen wir für den Anfang auf 0 bis 255 ein, also den, der sich mit einem Byte darstellen läßt (Darstellung ggfs. in Kapitel 2.2. nachlesen).

Als erstes legen wir fest, daß
1.) 0+0=0,
2.) 0+1=1,
3.) 1+0=1 und
4.) 1+1=0 mit Übertrag ist.

Diese Verknüpfung ist den meisten sicher als "Plus"-Verknüpfung bekannt. Die ersten drei Vorgaben sind sicherlich unmittelbar einleuchtend, Die vierte wird einleuchtend, wenn man sich die dezimale Addition "5+5" vor Augen führt. Auch hier ist das Ergebniss auf der Einerstelle "Null", mit einem Übertrag (engl.: Carry) auf die nächste, die Zehnerstelle. Entsprechendes gilt im Binärsystem: 2, die Summe von 1+1, ist binär "10", 4 als Summe aus 2+2 ist binär "100" usw.

Konkretisieren wir das ganze an einem Beispiel, der Addition von 154 und 58 (binär: 1001 1010 und 0011 1010) etwa:

```
  1001 1010 (154) ; 1. Summand
+ 0011 1010 ( 58) ; 2. Summand
   '''  '         ; Überträge
-----------------
  1101 0100 (212)
```

An der zweiten Stelle (alle Stellenangaben von **rechts**) tritt ein Übertrag auf die dritte auf (dort als "'" markiert), die Summe der 2. Stelle ist "0". Auf der 3. Stelle steht als Summe (aus "0+0+Carry") eine "1". Auf der fünften Stelle, wo zu den beiden originär vorhandenen Einsen noch der Übertrag aus der vierten Stelle addiert werden muß, ergibt sich das Ergebniss "1+Carry" auf die sechste Stelle. Dieser und die dort vorhandene Eins addieren sich wiederum zu "0"(+Carry). In den Spalten 7+8 entstehen keine Überträge.

Hier sollte der geneigte Leser seinem unbändigen Drang, das ganze durch Ausprobieren zu vertiefen, mit gutem Gewissen nachgeben. Als Aufgaben empfehlen sich:

1.) 170+85 2.) 79+45 3.) 254+2 u.v.a.m.

Das letzte Beispiel zeigt ein Problem bei der Übertragung auf den Computer: Da dieser nur mit acht Stellen rechnet, fällt der letzte Übertrag einfach weg, das Ergebniss wird verfälscht. (Modernere Systeme arbeiten mit 16 oder sogar 32 Bit und verschieben dadurch diese Schwelle weit nach oben; das prinzipielle Problem bleibt jedoch bestehen. Auf die Lösungen, die der Z80 dazu bietet, wird in einem späteren Kapitel eingegangen werden.)

A s s e m b l e r: K u r s

Und nun zur **Subtraktion.** Verdeutlichen wir uns dazu zunächst noch einmal die Subtraktion im vertrauten Dezimalsystem, 128-67 zum Beispiel:

```
  128
- 67
  '
 ----
   61
```

Wie geht man dabei eigentlich vor? An der letzten Stelle ist es sicherlich recht einfach: 7 wird von 8 abgezogen, das Ergebniss ist 1. An der zweiten Stelle ist die 6 größer als die 2. Bei einer Subtraktion erhielte man ein negatives Ergebniss. Deshalb ziehen wir zunächst 6 von 12 ab, addieren also praktisch 10 zur oberen Zahl ("borgen" uns 10). Das Ergebniss von "12-6" ist 6. Um die Zehn, die unser Ergebniss eben zu groß gemacht wurde (warum?) wieder zu korrigieren, machen wir einen Übertrag auf die nächste Stelle, ziehen also wieder 10 ab. Binär funktioniert das ganz entsprechend :

1.) 1-1=0, 2.) 1-0=1, 3.) 0-1=1 und Übert., 4.)0-0=0

Ein Beispiel möge es wieder verdeutlichen:

```
  1001 1101  (157)
- 0101 0110  ( 86)
-------------------
  0100 0111  ( 71)
```

Erste Stelle: 1-0=1. Klar. Zweite Stelle: 0-1=1 und Übertrag. Warum? Da man von null prinzipiell nichts abziehen kann, ziehen wir die eins von zwei (bin: 10) ab (borgen uns also 2). Das Ergebniss ist eins. Da wir jetzt die obere Zahl vergrößert haben, müssen wir die untere natürlich auch entsprechend vergrößern. Dies geschieht durch Übertrag auf die nächsthöhere Stelle. Diesen Übertrag ziehen wir vom Ergebniss der dritten Spalte (1-1=0) ab und erhalten wiederum eine eins und einen Übertrag. Vierte Stelle: 1-0=1, 1-1(Carry von der 3. Stelle)=0 , usw, usf.

So, damit sind wir mit der Theorie am Ende. Wer das beim ersten Durchlesen verstanden hat ist mindestens ein Genie; herzlichen Glückwunsch. Für nur normal Überintelligente: nochmal lesen und vor allem noch einige Beispiele durchrechnen.

A s s e m b l e r: K u r s

Assemblerkurs

4.4.2 Addier- und Subtrahierbefehle auf dem Z80

Die Sonderstellung des Akkumulators

In Abschnitt 3.1. (Info 24 - Seite 28) wurde bereits auf die besondere Bedeutung des Registers **A** wie Akku eingegangen. Trotzdem fangen wir hier noch einmal ziemlich von vorn an; das erleichtert mir das Schreiben und euch (hoffentlich) das Verständniss.
Der Akku ist das "Herz" unseres Rechners, die meisten (Rechen-) Befehle benutzen ihn als Quelle des ersten Operanden und zum Ablegen des Ergebnisses. Etliche Operationen (logische zum Beispiel, oder die Subtraktion wie unten) können nur über den Akku ausgeführt werden, so daß die Befehle in den meisten Assemblern ohne die Adresse A auskommen. (Ich hatte allerdings einen älteren und mir deshalb die Schreibweise "SUB A,n" angewöhnt. Laut Kurt-Bernd reicht für unseren BASIC-Assembler "SUB n") Für einige Operationen (Addition zum Beispiel) läßt sich das Doppelregister HL als eingeschränkter 16-Bit Akku verwenden. Als zweiten Operanden bei arithmetischen Operationen lassen wir zunächst Konstanten und andere Register zu. Damit erhalten wir für Addition und Subtraktion je zwei unterschiedliche Befehle.

Die einfache Addition

Der Maschinenbefehl **ADD A,n** zum Beispiel, der das gute alte **LET A=A+n** bewirkt. D.h.: zum Inhalt des Akkus wird die Konstante "n" addiert, das Ergebnis wird wieder im Akku abgelegt. Als Beispiel dazu ein kleines Maschinenprogramm, in Klammern die HEX-Befehle, hinter dem Semikolon stehen eventuelle Erläuterungen.

```
LD  A,16 ; Lade den Akku mit 16 (#3E,#10)
ADD A,32 ; Addiere 32 (#C6,#20)
```

Im Akku steht anschließend das Ergebiss "48" (#30), von wo man es zum Beispiel irgendwo im Speicher ablegen und mit PEEK auch dem Basic zugänglich machen kann.

Entsprechende Subtraktion

Entsprechendes gilt für die Subtraktion. Der Befehl heißt **SUB n** (nicht SUB A,n, obwohl manche Assembler auch diese Schreibweise akzeptieren). Wiederum wird "n" vom Inhalt des Akkus abgezogen, dort landet auch das Ergebniss. Fügen wir unserem Zwei-Zeiler von gerade noch die dritte

```
SUB 7 ;                (#D6,#07)
```

hinzu, so wird der Rechner bei der Abarbeitung des Programms vom Akkuinhalt "7" abziehen und das Ergebniss wiederum im Akku ablegen.

Bei der Addition/Subtraktion von Konstanten verwenden wir also Befehle von 2-Byte Länge. Im ersten steht der eigentliche Befehl (also ADD/SUB A,n), im zweiten die Konstante, mit der verknüpft werden soll. Im Gegensatz dazu kann man auch den Inhalt anderer Register addieren/Subtrahieren (ADD/SUB A,r). Dazu werden 1-Byte Befehle verwendet, in denen 3 Bit eines der acht Register adressieren

Etwas besonderes bewirkt der Befehl **ADD A,A**: Zunächst einmal ist das natürlich die Addition A+A=2*A. Im Binärsystem funktioniert die Multiplikation mit zwei so, wie im Dezimalsystem die Multiplikation mit 10: Die Zahl wird eine Stelle nach links geschoben, rechts wird eine Null angehängt:
1234 x 10 = 12340, #1010 x #10 = #10100

A s s e m b l e r: K u r s

Dazu gibt es diverse Schiebe- und Rotierbefehle, die hier allerdings den Rahmen sprengen würden und deshalb in einer weiteren Folge unserer beliebten...

Das C-Flag

In 4.4.1. habe ich das Problem, daß das Ergebniss einer Addition durchaus größer als 255 sein kann, bereits angesprochen. Tatsächlich kann das Ergebniss einer 8Bit- Addition **9** Bit lang sein. Dieses 9. Bit stellt uns der Z80 mit dem **Carry-Bit** im Flag-Register zur Verfügung. Es wird auf eins gesetzt, wenn bei einer Addition ein Übertrag auftritt, sonst auf null. Das Ergebniss der Addition "128+128" (binär 1000 000) ergibt damit zwar immer noch das acht-Bit-breite Ergebniss "0000 0000" aber mit einem gesetzten Carry-Flag. Dieses kann man nun entweder abfragen und gegebenenfalls in eine Fehlerroutine verzweigen oder zum Weiterrechnen verwenden. Wie, das wollen wir uns im nächsten Abschnitt ansehen.

Der Befehl ADC

Mit ADC, vielleicht so etwas wie "ADd mit Carry", kann man bei der Addition zweier Zahlen das Übertragsbit aus einer vorherigen Rechnung berücksichtigen; d.h.: ADC A,s bewirkt etwa ein BASIC'sches "LET Akku=Akku+s+Carry", wobei s für eine Konstante oder ein Register stehen kann. Das schönste Beispiel ist sicher die Programmierung einer 16-Bit Addition als "zweimalachtBit".

Eine 16-Bit-Addition

Als Voraussetzung nehmen wir an, daß unsere zwei Summanden an den Adressen "Adr1" und "Adr2" im Speicher stehen. Das Ergebnis soll an "Adr3" abgelegt werden. Dabei stehen die jeweils unteren acht Bit an den Adressen, die oberen acht eine Speicherstelle darüber.

```
+---------+
! Adr1+1  ! (Obere 8 Bit von Summand1)
+---------+
! Adr1    ! (Untere " " " )
+---------+
! Adr2+1  ! (Obere 8 Bit von Summand1)
+---------+
! Adr2    ! (Untere " " " )
+---------+
! Adr3+1  ! (Obere 8 Bit von Summand1)
+---------+
! Adr3    ! (Untere " " " )
+---------+
```

Unser Programm beginnt ähnlich der 8 Bit Addition:

```
LD A,(Adr1)     Lade A mit dem Inhalt der Speicherzelle Adr1
LD E,A
LD A,(Adr2)     In Adr2 steht die untere Hälfte des zweiten
ADD A,E         Summanden und der wird zu A addiert
LD (Adr3),A     Schreibe Akku nach Adr3.
```

Was ist bis jetzt passiert? Das gleiche wie bei der 8 Bit Addition. Der Unterschied liegt lediglich darin, daß wir statt der absoluten Werte ("LD A,16") mit Adressen arbeiten, in denen die Werte hinterlegt sind.

Die Adressen müssen dem Assembler natürlich bekannt sein. Dann hat man aber den Vorteil einer erhöhten Flexibilität. Man kann dieses Programm dann als Routine nutzen, indem man entweder die zu addierenden Werte in Adr1 und Adr2 hinterlegt oder die Adressen der jeweils aktuellen Summanden übergibt.

Assembler: Kurs

Den Zwischenschritt "LD E,A" benötigen wir, weil der gute alte Z80 den zweiten Summanden nicht direkt adressieren kann, den Befehl "ADD A,(Adr2)" also nicht versteht, ebensowenig wie "LD E,(Adr2). Diesen bilden wir mit den zwei Befehlen nach. Nach der Addition stehen die unteren acht Bit des Ergebnisses im Akku, ein eventueller Übertrag (als neuntes Bit der 8 Bit Addition) im C-Flag. Dieses halbe Ergebniss wird weggeschrieben und wir kümmern uns um die obere Hälfte:

```
LD A,(Adr1+1)[1] ; Obere Hälfte von Summand 1 in Akku
LD E,A           ; von dort nach E oder ein anderes Reg.
LD A,(Adr2+1)    ; Obere Hälfte von Summand 2 in Akku und
ADC A,E          ; Addition der oberen Hälften und des Übertrags
LD (Adr3+1),A    ; Und weck den Dreck
```

Und das war's schon. Wenn man die letzten 4 Befehle noch einmal anhängt, kann man so auch 24- und 32-Bit Additionen realisieren.

Und um das ganze in aller Ruhe etwas sacken zu lassen, gehen wir mal einen Beispiellauf mit konkreten Zahlen durch. Wir gehen davon aus, daß unsere kleine Routine irgendwo im Speicher steht und die Variablen Adr1-3 mit den entsprechenden Werten belegt sind.

Die Zahlen, die wir addieren möchten, mögen "9124" und "41395" heißen, was #23A4 und #A1B3 meint. Im Speicher steht also

```
an Adr1  : #A4
an Adr1+1: #23
an Adr2  : #B3
an Adr2+1: #A1
```

An Adr3 und Adr3+1 soll irgendwann das Ergebniss stehen.

```
LD A,(Adr1)     Im Akku steht #A4 zur Bearbeitung bereit.
LD E,A          jetzt in E
LD A,(Adr2)     Da steht der zweite Summand, #B3, den wir, wie schon gesagt,
                nicht direkt adressieren können.
ADD A,E         Das Ergebnis von #A4 + #B3 heißt #157. Davon werden die #57 als
                Ergebnis im Akku gespeichert, die #1 als gesetztes C-Bit.
LD (Adr3),A     Die #57 sind der untere Teil unseres Ergebnisses und werden
                deshalb nach Adr3 geschrieben.
LD A,(Adr1+1)   Da steht #23 als oberer Teil des ersten Summanden.
LD E,A          Das wieder rein in E.
LD A,(Adr2+1)   Die obere Hälfte des zweiten.
ADC A,E         Addieren, aber jetzt mit dem Carry-Bit! Ergebnis ist #C5.
LD (Adr3+1),A   An seinen Platz zurückschreiben.
```

An Adr3+1 steht jetzt also #C5, an Adr3 #57. Eine kleine BASIC-Zeile, in der Art "LET SUMME=PEEK(Adr3+1)*256 + PEEK(Adr3)" liefert uns jetzt mit 50.519 das gewünschte Ergebniss.

1) Die Adressaddition muß man im Kopf machen und das Ergebnis dort angeben. Praktischer ist es jedoch, ein neues Label zu vergeben. (Andere Assember, z.B. ASM, führen jedoch solche Adressberechnungen automatisch durch.)

Assembler: Kurs

4.4.3 Binärarithmetik ganzer Zahlen (Kurt-Bernd Rohloff, 8000)

Bisher haben wir uns immer um die negativen Zahlen herumgedrückt. Dies geschah ganz bewußt, weil sie nämlich auch ein schwieriges Kapitel sind, das wir nun anpacken wollen. Die Schwierigkeit besteht hauptsächlich darin, daß negative Zahlen keine gesonderte Darstellung im Computer haben, die sie klipp und klar als negativ kennzeichnen. Der Computer kennt nur Bitmuster, deren Bits er nach festgelegten Regeln verknüpft. Erst durch unsere Interpretation wird aus einem solchen Bitmuster eine Zahl, und erst durch eine besondere Interpretation dieser Zahl wird daraus eine negative Zahl. Wegen dieser Schwierigkeiten, die erfahrungsgemäß beim Anfänger für viel Verwirrung sorgen kann, werden wir in kleinen Schritten vorgehen und uns die Sache erstmal in dem uns besser vertrautem Dezimalsystem klarmachen.

4.4.3.1 Komplemente im Dezimalsystem

Dazu greifen wir wieder auf die exotische I2DITVM zurück. Wir hatten bereits erwähnt, daß man mit ihr 100 Zahlen darstellen kann. Diese hatten wir bislang naheliegenderweise mit den positiven Zahlen 0 .. 99 gleichgesetzt, denn so werden sie ja auch angezeigt. (Man beachte, daß hier die Null zu den positiven Zahlen gerechnet wird!) Da wir nun auch negative Zahlen darstellen wollen, werden wir uns dafür mit nur 50 positiven Zahlen begnügen müssen. Die anderen 50 sind dann für die negativen reserviert. Da auch die I2DITVM keine negativen Zahlen kennt, ist es von nun an wichtig, zwischen den Zahlen der I2DITVM und ihrer Interpretation durch uns zu unterscheiden. Ich werde daher die ersteren Maschinenzahlen nennen und die letzteren Menschenzahlen oder einfach Zahlen. Es ist naheliegend, die ersten 50 Maschinenzahlen 0 .. 49 als positive Menschenzahlen 0 .. 49 anzusehen. Die höheren Maschinenzahlen 50 .. 99 werden wir nun als negative Menschenzahlen -1 .. -50 ansehen. (Man beachte die Asymmetrie!) Man könnte nun hergehen und die Maschinenzahl 50 als Menschenzahl -1, 51 als -2 usw. interpretieren. Um aber möglichst einfache Rechenregeln zu erhalten, werden wir dies **nicht** so tun. Wir hatten ja bereits früher gesehen, daß die I2DITVM etwas "merkwürdig" rechnet, indem die Addition von 99 und 1 die Maschinenzahl 0 ergab. Welche Zahl ergibt, um 1 erhöht, den Wert Null? Natürlich -1. Von daher liegt nahe, die Maschinenzahl 99 als -1 zu interpretieren, 98 als -2, 97 als -3 ... 50 als -50. In dieser Interpretation rechnet die I2DITVM nämlich gerade wieder richtig. Probiere als Beispiel die Rechnung 5 - 7 aus (Befehlsliste s. Info 20-29):

```
LOAD  05
SUB   07
```

Das angezeigte Ergebnis ist 98 (als Maschinenzahl), was wir nach dem oben gesagten als -2 interpretieren. Statt die 7 zu subtrahieren, müssen wir dann ja auch -7 addieren können. Auch das geht auf, wenn man bedenkt, daß -7 als Maschinenzahl 93 ist:

```
LOAD  05
ADD   93
```

Das Ergebnis ist das gleiche. Die letzte Rechnung können wir aber auch als Addition der beiden Menschenzahlen 5 und 93 auffassen. Beides ist richtig. Allein **unsere Interpretation** der Maschinenzahlen entscheidet darüber, ob wir die Maschinenzahl 93 als Zahl 93 oder als Zahl -7 auffassen. Für den Computer macht das überhaupt keinen Unterschied!

Wenn wir nun bei unserer negativen Interpretation bleiben (das ist nichts Schlechtes), können wir uns leicht überlegen, wie wir von der Maschinenzahl, sofern sie über 49 liegt, die "richtige" Menschenzahl bekommen:

```
Menschenzahl = -(100 - Maschinenzahl)
```

Den Absolutbetrag davon nennt man übrigens das 100ter Komplement der zugehörigen Maschinenzahl (die Ergänzung zu 100). Also wäre beispielsweise 1 das 100ter Komplement von 99. Das gilt auch umgekehrt. Die Maschinenzahl zu einer negativen Menschenzahl läßt sich ebenso leicht bestimmen:

```
Maschinenzahl = 100 - ABS(Menschenzahl)
```

Um unsere negativen Ergebnisse von der I2DITVM in die zugehörigen Menschenzahlen umrechnen zu lassen, eignet sich obige Formel allerdings nicht, da ja

100 keine Maschinenzahl mehr ist. Wir formen also ein wenig um:

```
Menschenzahl = -((99 - Maschinenzahl) +1)
```

Nun, den inneren Klammerausdruck kann die I2DITVM aber mit Leichtigkeit berechnen. Dafür hat sie nämlich den Befehl **CPL**. Der CPL Befehl liefert uns also gerade das 99er Komplement der Zahl, die vorher im Akku stand. Wir können unser Ergebnis also recht einfach in eine lesbare Form bringen. Das obige Subtraktionsbeispiel sieht dann so aus:

```
LOAD  05
SUB   07
; da Ergebnis > 49 umformen in Menschenzahl:
CPL
INC   A
EXIT
```

Als Ergebnis haben wir nun den "richtigen" Absolutbetrag, wir müssen uns nur im Köpfchen das Minuszeichen dazudenken.

4.4.3.2 Komplemente im Binärsystem

So, nach diesem Vorgeplänkel werden wir nun wieder in die Niederungen der binären Welt herabsteigen. Bei unserer 8-bit CPU stehen uns dort 256 Maschinenzahlen zur Verfügung, die wir bisher als die positiven Zahlen 0 .. 255 angesehen haben. Die Darstellung der negativen Zahlen geschieht hier analog, indem wir nur noch die untere Hälfte 0 .. 127 als positive Menschenzahlen interpretieren, die obere Hälfte der Maschinenzahlen jedoch negativen Menschenzahlen zuordnen, u. z.

```
255   -->   -1
254   -->   -2
253   -->   -3
...
128   -->   -128
```

Die Umrechnung geht wieder ganz analog vonstatten, wobei die Rolle der 100 der I2DITVM nun von der (in der Z80 CPU nicht mehr darstellbaren) Zahl 256 gespielt wird:

```
Menschenzahl = -((255 - Maschinenzahl) +1)
Maschinenzahl = 255 - ABS(Menschenzahl) +1
```

Zahlen, die in dieser Relation zueinander stehen, bezeichnet man als Zweierkomplement voneinander. Also ist 255 das Zweierkomplement zu 1 und umgekehrt. Den inneren Klammerausdruck der oberen Formel bezeichnet man als Einerkomplement, der gerade um 1 kleiner als das Zweierkomplement ist. Dementsprechend nennt man die von uns gewählte Darstellung der negativen Zahlen auch Zweierkomplementdarstellung. Das Einerkomplement, also die Ergänzung zu 255 = FFh = 11111111b kann die Maschine wieder ganz einfach berechnen: man braucht dazu nur jedes Bit umzukippen. Beispiel:

```
46 =  00101110
209=  11010001 Einerkomplement zu 46 (= 255-46)
```

Um das Zweierkomplement zu erhalten, muß noch um 1 erhöht werden, was die Z80 CPU ebenfalls mit links macht. Wir wollen nun obige Beispielrechnung 5 - 7 = -2 noch einmal im Binärsystem durchführen (das übt!). Dazu gibt es zwei Möglichkeiten:

1.) Die binäre Subtraktion, wie in 4.4.1 von Andreas vorgeführt:

```
    5  =  00000101
-   7  =  00000111
         ,,,,,,,
          --------
          11111110 = 254
```

Hierbei haben wir uns von der (nicht in der Maschine vorhandenen) neunten Stelle eine 2 geborgt. Diesen Vorgang "merkt" sich die CPU wieder im Carry Flag.

2.) Die Subtraktion durch Addition des Zweierkomplements. Dies ist meist für uns (und auch für die CPU) einfacher. Wir bilden zuerst die Summe mit dem Einerkomplement und addieren danach die fehlende 1 wieder drauf:

```
   5  =  00000101
+248  =  11111000 (Einerkomplement von 7)
         --------
         11111101 (Zwischenergebnis)
+  1  =  00000001
         --------
         11111110 = 254
```

Wie man sieht, liefern beide Verfahren die gleiche Maschinenzahl. Im Gegensatz zur "echten" Subtraktion tritt hier jedoch kein Übertrag auf! Wir **können**, müssen jedoch nicht, dies Ergebnis noch die zugehörige Menschenzahl umwandeln:

```
         11111110 = 254
         00000001 (Einerkomplement davon)
+  1     00000001 (um Zweierkomplement zu erhalten)
         --------
         00000010
```

Dies interpretieren wir als -2.

Übungsaufgabe 4.4.3.2-1: Bestimme das 8-Bit Zweierkomplement von
a) 00010001 b) 01011100 c) 11101001 d) 11110000
Übungsaufgabe 4.4.3.2-2: Bestimme die 8-Bit Zweierkomplementdarstellung von
a) -1 b) -37 c) 60 d) -100
Übungsaufgabe 4.4.3.2-3: Welche Zahlen werden durch die folgenden Zweierkomplementzahlen dargestellt:
a) 01000110 b) 10000000 c) 11110000 d) 10011001

4.4.3.3 Vorzeichen, Übertrag und Überlauf

Den Übertrag kennen wir ja schon. Er hat beim Rechnen im Zweierkomplement keine Bedeutung. Wir vermerken hier nur nochmal, daß er bei einer "echten" Subtraktion genau andersherum ausfällt als bei der Addition des Zweierkomplements. Die Z80-CPU setzt ihn bei Subtraktionsbefehlen so wie bei einer "echten" Subtraktion, obwohl sie intern eine Zweierkomplementaddition durchführt. Wenn wir nicht im Zweierkomplement, sondern vorzeichenlos rechnen, zeigt ein Übertrag bei einer Addition an, daß das Ergebnis größer als 255 ist. Ob dies einen Fehler signalisiert, hängt von unserem Programm ab (s. das Beispiel der 16-bit Addition weiter oben). Bei einer Subtraktion (wieder vorzeichenlos) zeigt ein Übertrag an, daß das Ergebnis negativ ist, d. h. die zweite Zahl (man nennt sie wohl auch Subtrahend) war größer als die erste. Auch hier kann das einen Fehler signalisieren, muß aber nicht. Wir werden im nächsten Abschnitt ein Programm für eine vorzeichenlose 16-bit Subtraktion angeben, bei dem ein Übertrag in der unteren Hälfte erlaubt ist.

In der Zeierkomplementrechnung kann aber ein anderer Fehler auftreten, der Überlauf. Was ist jetzt das? Nun, dazu ein Beispiel:

```
   64  =  01000000
+  64  =  01000000
         --------
         10000000 = -128 !!!
```

Wir haben zwei positive Zahlen addiert und (immer in unserer Zweierkomplementdarstellung betrachtet) ein **negatives** Ergebnis erhalten! Das kann doch nicht wahr sein! Genau das ist ein (arithmetischer) Überlauf. In vorzeichenloser Rechnung wäre das Ergebnis +128, also richtig. Da wir in der Zweierkomplementdarstellung jedoch den Maschinenzahlen oberhalb von 127 negative Zahlen zugeordnet haben, ist das Ergebnis als Zweierkomplementzahl betrachtet falsch. Wir haben den Darstellungsbereich unserer Zweierkomplementzahlen gesprengt! Da nun die Zweierkomplementrechnung doch ziemlich häufig vorkommt, hat uns der Hersteller Zilog in weiser Voraussicht ein Flag zur Verfügung gestellt, um diesen Fehlerfall zu erkennen. Es handelt sich um das V-Flag (von overflow, engl. für Überlauf). Wenn es nach einem arithmetischen Befehl auf 1 gesetzt ist, zeigt das an, daß das Ergebnis als Zweierkomplementzahl betrachtet falsch ist.

Dem höchsten Bit kommt in der Zweierkomplementdarstellung eine besondere Bedeutung zu. An ihm können wir nämlich bereits das Vorzeichen der Zahl ablesen. Eine 1 bedeutet negativ, eine 0 dann wohl positiv. Häufig hängt in einem Programm die weitere Verarbeitung vom Vorzeichen eines Rechenergebnisses ab (z. B. könnten wir in Abhängigkeit davon einmal "Gewinn" und sonst "Verlust" ausgeben wollen). Obwohl nun der Z80 Befehle zum Testen einzelner Bits besitzt, hat uns Zilog auch für diesen wichtigen Test ein extra Flag spendiert, das S-Flag (von sign = Vorzeichen). Es ist einfach eine Kopie des höchsten Bits im Akku.

4.4.4 Verarbeitung ganzer Zahlen mit dem Z80

So, nach diesem Ausflug in die Gefilde der Mathematik wenden wir uns der Programmierung zu. Das folgende Programm soll eine vorzeichenlose 16-bit Subtraktion durchführen. Hierbei muß das Endergebnis allerdings positiv sein, sonst kommt Mist heraus. Da wir (noch) keinen Maschinenbefehl zur 16-bit Subtraktion kennen, machen wir eine iterierte 8-bit Subtraktion daraus. Achte bei dem Programm insbesondere auf folgendes: die erste Subtraktion für die unteren 8 Bit erfolgt ohne Beachtung des Übertrags. Sie darf auch ein negatives Ergebnis liefern! Denn bei der nächsten Subtraktion für die oberen 8 Bit ziehen wir den (eventuell entstandenen) Übertrag mit ab. Der Befehl dazu ist SBC (subtract with carry). Wenn allerdings bei dieser zweiten Subtraktion etwas negatives herauskommt, ist das Ergebnis falsch. Da wir noch keine Befehle zur Flag-Abfrage und Verzweigung kennen, habe ich die Behandlung dieses Fehlers ins BASIC verlegt. Noch ein kleiner Pferdefuß: Beim einfachen Subtraktionsbefehl SUB geben wir nur den Subtrahenden an. Daß vom Akku abgezogen wird, versteht sich von selbst. **ABER:** beim SBC Befehl müssen wir den Minuenden A explizit angeben, also etwa SBC A,B im Gegensatz zu SUB B. (Der Grund ist, daß man bei SBC und nur bei SBC auch andere Register als A angeben kann. Wir kommen später darauf zurück.) Statt eines Registers können wir auch eine konstante Zahl angeben, etwa SBC A,16.

Programm 4-4-4.1: Vorzeichenlose 16-bit Subtraktion (MTX500 Version)

```
0 GOTO 100
1 CODE

8010          JP BEGIN
8013 XL:      DB 0
8014 XH:      DB 0
8015 YL:      DB 0
8016 YH:      DB 0
8017 ZL:      DB 0
8018 ZH:      DB 0
8019 BEGIN:   LD A,(YL)   ;y_low nach C bringen
801C          LD C,A
801D          LD A,(XL)   ;x_low in den Akku
8020          SUB C       ; = x_low - y_low im Akku
8021          LD (ZL),A   ;in z_low abspeichern
8024          LD A,(YH)   ;jetzt mit den high Bytes
8027          LD C,A      ;ebenso verfahren
8028          LD A,(XH)
802B          SBC A,C     ;jetzt aber mit Carry
802C          LD (ZH),A
802F          RET

Symbols:

BEGIN   8019     XL      8013
XH      8014     YL      8015
YH      8016     ZL      8017
ZH      8018
```

A s s e m b l e r: K u r s

```
2 RETURN
100 REM Vorzeichenlose 16-Bit Subtraktion
110 REM Beispielprogramm zu Abschnitt 4.4.4
120 GOSUB 1000
130 PRINT "********** Vorzeichenlose 16-Bit Subtraktion **********"
140 PRINT "Berechnet wird x - y, wobei 0<=x,y<=65535 gelten muss"
150 INPUT "X:";X
160 INPUT "Y:";Y
170 REM x und y in oberes und unteres Byte aufspalten
180 REM und in XL,XH bzw. YL,YH abspeichern:
190 POKE XLOW,MOD(X,256)
200 POKE XHIGH,INT(X/256)
210 POKE YLOW,MOD(Y,256)
220 POKE YHIGH,INT(Y/256)
230 REM Assemblerprogramm ausfuehren:
240 GOSUB 1
250 PRINT "Ergebnis von ";X;" - ";Y;" ist: ";256*PEEK(ZHIGH)+PEEK(ZLOW)
260 IF X<Y THEN PRINT "Das ist F A L S C H"
270 STOP
1000 REM Adresse von XL: 32787 (MTX500) bzw. 16403 (MTX512)
1010 LET XLOW=32787
1020 LET XHIGH=XLOW+1
1030 LET YLOW=XHIGH+1
1040 LET YHIGH=YLOW+1
1050 LET ZLOW=YHIGH+1
1060 LET ZHIGH=ZLOW+1
1070 RETURN
```

Mit dem nächsten Programm wollen wir mal das anwenden, was wir im vorigen Abschnitt über das Zweierkomplement gelernt haben. Es soll eine Zweierkomplement-Addition durchführen, u. z. natürlich auch gleich mit 16-Bit Zahlen. Der Bereich der positiven Zahlen geht hier von 0 bis 32767, die negativen gehen von -1 bis -32768. Allgemein kann man folgendes festhalten: Bei Zweierkomplementrechnung mit n Bits ist die "negativste" Zahl gerade 2^{n-1}, die "positivste" Zahl ist dann eins kleiner. Das folgende Programm prüft allerdings nicht, ob die eingegebenen Zahlen im erlaubten Bereich liegen. Der Überlauffehler wird wieder im BASIC abgefangen, indem das erwartete Vorzeichen (Variable EXPS) mit dem tatsächlichen (Variable REALS) verglichen wird.

Die Assemblerroutine auf Zeile 1 wandelt die in LO und HI abgelegte 16-Bit Zahl in ihr Zweierkomplement um und legt dieses dann ebendort wieder ab. Das 16-Bit Zweierkomplement ist die Ergänzung zu 65536. Es wird ganz analog zum 8-Bit Fall gebildet, indem wir erst alle Bits umkippen und dann 1 draufaddieren. Programmtechnisch gehen wir wieder in zwei 8-Bit Schritten vor. Für die unteren 8 Bit verwenden wir den Befehl **NEG**, der uns gleich das Zweierkomplement vom Akku liefert. Genau gesagt, liefert er uns das Ergebnis der Rechnung 0 - A. Das Aufaddieren der 1 kann also entfallen. Die Sache hat allerdings einen kleinen Haken: das ist das Carry-Flag. Es wird so wie bei einer Subtraktion gesetzt, während wir es hier gerade wie bei einer Addition des Zweierkomplements bräuchten. Wir sahen schon weiter oben, daß das Carry-Flag in den beiden Fällen gerade umgekehrt steht. Damit wir für die oberen 8 Bit den richtigen Übertrag berücksichtigen, müssen wir das Carry-Flag also gerade umkippen. Und - als hätte Zilog unseren Assemblerkurs schon vorhergesehen - genau dafür hat uns Zilog den Befehl **CCF** eingebaut (complement carry flag). Die obere Hälfte geht dann ganz "normal". Wir kippen zuerst mit dem Befehl **CPL** (complement) alle Bits im Akku um. Den Übertrag von der unteren Hälfte berücksichtigen wir, indem wir eine 0 inklusive Carry-Flag addieren (mit ADC also).

In Zeile 3 steht die eigentliche Additionsroutine. Sie ist weitgehend analog zu der Subtraktion im vorigen Programm gestaltet. Nur wird jetzt das Ergebnis wieder nach XL und XH zurückgespeichert (was wir beim vorigen Programm auch hätten machen können, vielleicht versuchst du es mal).

A s s e m b l e r: K u r s

Im BASIC Teil ist auf Zeile 1000 wieder eine Routine, die Variablen mit den Adressen belegt und die du gegebenenfalls an deine Verhältnisse anpassen mußt. In Zeile 2000 beginnt ein Unterprogramm, daß die Speicher in der Additionsroutine vorbelegt und gegebenenfalls vorher eine negative Zahl in ihre Zweierkomplementdarstellung umwandelt. Dazu wird dann die Routine auf Zeile 1 aufgerufen. Das Ergebnis der Addition wird in der Variablen Z gespeichert, die oberen 8 Bit davon in der Variablen T. Wenn das Ergebnis negativ ist (T ist dann >= 128, d. h. das höchstwertige Bit ist gesetzt), wird der Absolutbetrag davon gebildet, indem wir es nochmal "zweierkomplementieren". Zur Ausgabe wird es noch mit seinem Vorzeichen (REALS) multipliziert, damit wir es auch kapieren (und kontrollieren können).

Programm 4-4-4.2: 16-Bit Addition im Zweierkomplement

```
0 GOTO 100
1 CODE

8010          JP BEGIN
8013 LO:      DB 0
8014 HI:      DB 0
8015 BEGIN:   LD A,(LO)
8018          NEG          ;beeinflusst Carry
801A          CCF          ;aber gerade falsch herum
801B          LD (LO),A    ;Zweierkomplement Low-Byte
801E          LD A,(HI)
8021          CPL          ;beeinflusst Carry NICHT
8022          ADC A,0      ;Carry von NEG dazu
8024          LD (HI),A
8027          RET

Symbols:

BEGIN   8015    LO      8013
HI      8014

2 RETURN
3 CODE

80F2          JP ADD16
80F5 XL:      DB 0
80F6 XH:      DB 0
80F7 YL:      DB 0
80F8 YH:      DB 0
80F9 ADD16:   LD A,(YL)    ;low-bytes verarbeiten
80FC          LD B,A
80FD          LD A,(XL)
8100          ADD A,B      ;ohne Carry addieren
8101          LD (XL),A    ;Ergebnis zurueck
8104          LD A,(YH)    ;jetzt high-bytes verarbeiten
8107          LD B,A
8108          LD A,(XH)
810B          ADC A,B      ;mit Carry von ADD
810C          LD (XH),A
810F          RET

Symbols:

ADD16   80F9    XL      80F5
XH      80F6    YL      80F7
YH      80F8

4 RETURN
```

A s s e m b l e r: K u r s

```
100 REM Programm zur 16-bit Addition im Zweierkomplement
110 REM zu Abschnitt 4.4.4
120 REM 30.11.88/Rohloff
130 GOSUB 1000: REM Adressen initialisieren
140 PRINT "*** 16-Bit Addition im Zweierkomplement ***"
150 INPUT "Zahl 1:";X
160 LET T=X: REM variablen fuer UP 2000 bereitstellen
170 LET ADRL=XL: LET ADRH=XH
180 GOSUB 2000
190 INPUT "Zahl 2:";Y
195 LET EXPS=SGN(X+Y)
200 LET T=Y: REM wie eben
210 LET ADRL=YL: LET ADRH=YH
220 GOSUB 2000
230 GOSUB 3: REM jetzt endlich addieren
240 LET T=PEEK(XH): REM Ergebnisse abholen
250 LET Z=256*T+PEEK(XL)
260 LET REALS=SGN(Z): REM tatsaechliches VZ des Ergebnisses
270 IF T<128 THEN GOTO 350
280 REM wenn bit 15 gesetzt war (T>=128), ist das Ergebnis negativ
290 REM Dann bilden wir vom Ergebnis wieder das Zweierkomplement
300 POKE LO,PEEK(XL)
310 POKE HI,PEEK(XH)
320 GOSUB 1
330 LET Z=256*PEEK(HI)+PEEK(LO)
340 LET REALS=-1
350 REM jetzt ist das Ergebnis (Absolutbetrag) in Z, sein VZ in REALS
360 PRINT X;" + ";Y;" = ";REALS*Z
370 IF EXPS<>REALS THEN PRINT "Ueberlauffehler"
380 STOP
1000 REM Adressen fuer MTX500:
1010 LET LO=32787: REM fuer MTX512 muesste es 16403 sein
1020 LET HI=LO+1
1030 LET XL=33013
1040 LET XH=XL+1
1050 LET YL=XH+1
1060 LET YH=YL+1
1070 RETURN
2000 REM Diese Routine rechnet negative Zahlen in ihre Zweierkomplementwerte
2010 REM um und speichert sie in der ADD16 Routine ab.
2030 REM INPUT: T (Zahl, die umgerechnet und gespeichert werden soll)
2040 REM        ADRL (Adresse zum Abspeichern des Low-Bytes)
2050 REM        ADRH (dito fuer High-Byte)
2070 REM OUTPUT: nichts
2100 POKE LO,MOD(ABS(T),256): REM T wird immer in LO/HI abgespeichert
2110 POKE HI,INT(ABS(T)/256)
2120 IF T<0 THEN GOSUB 1
2130 POKE ADRL,PEEK(LO): REM umspeichern fuer ADD16
2140 POKE ADRH,PEEK(HI)
2150 RETURN
```

A s s e m b l e r: Z80-Befehlssatz

(Peter Würfel, 7262)

CODE	Assembler-Befehl	
8E	ADC	A,(HL)
DD8E05	ADC	A,(IX+d)
FD8E05	ADC	A,(IY+d)
8F	ADC	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
CE20	ADC	A,n
ED4A	ADC	HL,BC
ED5A	ADC	HL,DE
ED6A	ADC	HL,HL
ED7A	ADC	HL,SP
86	ADD	A,(HL)
DD8605	ADD	A,(IX+d)
FD8605	ADD	A,(IY+d)
87	ADD	A,A
80	ADD	A,B
81	ADD	A,C
82	ADD	A,D
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
C620	ADD	A,n
09	ADD	HL,BC
19	ADD	HL,DE
29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD29	ADD	IX,IX
DD39	ADD	IX,SP
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD29	ADD	IY,IY
FD39	ADD	IY,SP
A6	AND	(HL)
DDA605	AND	(IX+d)
FDA605	AND	(IY+d)
A7	AND	A
A0	AND	B
A1	AND	C
A2	AND	D
A3	AND	E
A4	AND	H
A5	AND	L
E620	AND	n
CB46	BIT	0,(HL)
DDCB0546	BIT	0,(IX+d)
FDCB0546	BIT	0,(IY+d)
CB47	BIT	0,A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB4E	BIT	1,(HL)
DDCB054E	BIT	1,(IX+d)
FDCB054E	BIT	1,(IY+d)
CB4F	BIT	1,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB56	BIT	2,(HL)
DDCB0556	BIT	2,(IX+d)
FDCB0556	BIT	2,(IY+d)
CB57	BIT	2,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB5E	BIT	3,(HL)
DDCB055E	BIT	3,(IX+d)
FDCB055E	BIT	3,(IY+d)
CB5F	BIT	3,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB66	BIT	4,(HL)
DDCB0566	BIT	4,(IX+d)
FDCB0566	BIT	4,(IY+d)
CB67	BIT	4,A
CB60	BIT	4,B
CB61	BIT	4,C
CB62	BIT	4,D
CB63	BIT	4,E
CB64	BIT	4,H
CB65	BIT	4,L
CB6E	BIT	5,(HL)
DDCB056E	BIT	5,(IX+d)
FDCB056E	BIT	5,(IY+d)
CB6F	BIT	5,A
CB68	BIT	5,B
CB69	BIT	5,C
C86A	BIT	5,D
CB6B	BIT	5,E
CB6C	BIT	5,H
CB6D	BIT	5,L
CB76	BIT	6,(HL)
DDCB0576	BIT	6,(IX+d)
FDCB0576	BIT	6,(IY+d)
CB77	BIT	6,A
CB70	BIT	6,B
CB71	BIT	6,C
CB72	BIT	6,D
CB73	BIT	6,E
CB74	BIT	6,H
CB75	BIT	6,L
CB7E	BIT	7,(HL)
DDCB057E	BIT	7,(IX+d)
FDCB057E	BIT	7,(IY+d)
CB7F	BIT	7,A
CB78	BIT	7,B
CB79	BIT	7,C
CB7A	BIT	7,D
CB7B	BIT	7,E
CB7C	BIT	7,H
CB7D	BIT	7,L
DC8405	CALL	C,nn
FC8405	CALL	M,nn
D48405	CALL	NC,nn
C48405	CALL	NZ,nn
F48405	CALL	P,nn
EC8405	CALL	PE,nn
E48405	CALL	PO,nn
CC8405	CALL	Z,nn
CD8405	CALL	nn
3F	CCF	
BE	CP	(HL)
DDBE05	CP	(IX+d)
FDBE05	CP	(IY+d)
BF	CP	A
B8	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L
FE20	CP	n
EDA9	CPD	
EDB9	CPDR	
EDB1	CPIR	
EDA1	CPI	
2F	CPL	
27	DAA	
35	DEC	(HL)
DD3505	DEC	(IX+d)
FD3505	DEC	(IY+d)
3D	DEC	A
05	DEC	B
0B	DEC	BC
0D	DEC	C
15	DEC	D
1B	DEC	DE
1D	DEC	E
25	DEC	H
2B	DEC	HL
DD2B	DEC	IX
FD2B	DEC	IY
2D	DEC	L
3B	DEC	SP
F3	DI	
102E	DJNZ	e
FB	EI	
E3	EX	(SP),HL
DDE3	EX	(SP),IX
FDE3	EX	(SP),IY
08	EX	AF,AF'
EB	EX	DE,HL
D9	EXX	
76	HALT	
ED46	IM	0
ED56	IM	1
ED5E	IM	2
ED78	IN	A,(C)
ED40	IN	B,(C)
ED48	IN	C,(C)
ED50	IN	D,(C)
ED58	IN	E,(C)
ED60	IN	H,(C)
ED68	IN	L,(C)
DB20	IN	A,(n)
34	INC	(HL)
DD3405	INC	(IX+d)
FD3405	INC	(IY+d)
3C	INC	A
04	INC	B
03	INC	BC
0C	INC	C
14	INC	D
13	INC	DE
1C	INC	E
24	INC	H
23	INC	HL
DD23	INC	IX
FD23	INC	IY
2C	INC	L
33	INC	SP
EDAA	IND	
EDBA	INDR	
EDA2	INI	
EDB2	INIR	
C38405	JP	nn
E9	JP	(HL)
DDE9	JP	(IX)
FED9	JP	(IY)
DA8405	JP	C,nn
FA8405	JP	M,nn
D28405	JP	NC,nn
C28405	JP	NZ,nn
F28405	JP	P,nn
EA8405	JP	PO,nn
CA8405	JP	Z,nn
382E	JR	C,e
302E	JR	NC,e
202E	JR	NZ,e
282E	JR	Z,e
182E	JR	e
02	LD	(BC),A
12	LD	(DE),A
77	LD	(HL),A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
3620	LD	(HL),n
DD7705	LD	(IX+d),A

A s s e m b l e r: Z80-Befehlssatz

DD7Ø05	LD	(IX+d),B
DD71Ø5	LD	(IX+d),C
DD72Ø5	LD	(IX+d),D
DD73Ø5	LD	(IX+d),E
DD74Ø5	LD	(IX+d),H
DD75Ø5	LD	(IX+d),L
DD36Ø52Ø	LD	(IX+d),n
FD77Ø5	LD	(IY+d),A
FD7Ø05	LD	(IY+d),B
FD71Ø5	LD	(IY+d),C
FD72Ø5	LD	(IY+d),D
FD73Ø5	LD	(IY+d),E
FD74Ø5	LD	(IY+d),H
FD75Ø5	LD	(IY+d),L
FD36Ø52Ø	LD	(IY+d),n
328405	LD	(nn),A
ED438405	LD	(nn),BC
ED538405	LD	(nn),DE
228405	LD	(nn),HL
DD228405	LD	(nn),IX
FD228405	LD	(nn),IY
ED738405	LD	(nn),SP
ØA	LD	A,(BC)
1A	LD	A,(DE)
7E	LD	A,(HL)
DD7EØ5	LD	A,(IX+d)
FD7EØ5	LD	A,(IY+d)
3A8405	LD	A,(nn)
7F	LD	A,A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
7B	LD	A,E
7C	LD	A,H
ED57	LD	A,I
7D	LD	A,L
3E2Ø	LD	A,n
ED5F	LD	A,R
46	LD	B,(HL)
DD46Ø5	LD	B,(IX+d)
FD46Ø5	LD	B,(IY+d)
47	LD	B,A
4Ø	LD	B,B
41	LD	B,C
42	LD	B,D
43	LD	B,E
44	LD	B,H
45	LD	B,L
Ø62Ø	LD	B,n
ED4B8405	LD	BC,(nn)
Ø18405	LD	BC,nn
4E	LD	C,(HL)
DD4EØ5	LD	C,(IX+d)
FD4EØ5	LD	C,(IY+d)
4F	LD	C,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
ØE2Ø	LD	C,n
56	LD	D,(HL)
DD56Ø5	LD	D,(IX+d)
FD56Ø5	LD	D,(IY+d)
57	LD	D,A
5Ø	LD	D,B
51	LD	D,C
52	LD	D,D
53	LD	D,E
54	LD	D,H
55	LD	D,L
162Ø	LD	D,n
ED5B8405	LD	DE,(nn)
118405	LD	DE,nn
5E	LD	E,(HL)
DD5EØ5	LD	E,(IX+d)
FD5EØ5	LD	E,(IY+d)
5F	LD	E,A
58	LD	E,B
59	LD	E,C
5A	LD	E,D
5B	LD	E,E
5C	LD	E,H
5D	LD	E,L
1E2Ø	LD	E,n
66	LD	H,(HL)
DD66Ø5	LD	H,(IX+d)
FD66Ø5	LD	H,(IY+d)
67	LD	H,A
6Ø	LD	H,B
61	LD	H,C
62	LD	H,D
63	LD	H,E
64	LD	H,H
65	LD	H,L
262Ø	LD	H,n
2A8405	LD	HL,(nn)
218405	LD	HL,nn
ED47	LD	I,A
DD2A8405	LD	IX,(nn)
DD218405	LD	IX,nn
FD2A8405	LD	IY,(nn)
FD218405	LD	IY,nn
6E	LD	L,(HL)
DD6EØ5	LD	L,(IX+d)
FD6EØ5	LD	L,(IY+d)
6F	LD	L,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
6B	LD	L,E
6C	LD	L,H
6D	LD	L,L
2E2Ø	LD	L,n
ED4F	LD	R,A
ED7B8405	LD	SP,(nn)
F9	LD	SP,HL
DDF9	LD	SP,IX
FDF9	LD	SP,IY
318405	LD	SP,nn
EDA8	LDD	
EDB8	LDDR	
EDAØ	LDI	
EDBØ	LDIR	
ED44	NEG	
ØØ	NOP	
B6	OR	(HL)
DDB6Ø5	OR	(IX+d)
FDB6Ø5	OR	(IY+d)
B7	OR	A
BØ	OR	B
B1	OR	C
B2	OR	D
B3	OR	E
B4	OR	H
B5	OR	L
F62Ø	OR	n
EDBB	OTDR	
EDB3	OTIR	
ED79	OUT	(C),A
ED41	OUT	(C),B
ED49	OUT	(C),C
ED51	OUT	(C),D
ED59	OUT	(C),E
ED61	OUT	(C),H
ED69	OUT	(C),L
D32Ø	OUT	(n),A
EDAB	OUTD	
EDA3	OUTI	
F1	POP	AF
C1	POP	BC
D1	POP	DE
E1	POP	HL
DDE1	POP	IX
FDE1	POP	IY
F5	PUSH	AF
C5	PUSH	BC
D5	PUSH	DE
E5	PUSH	HL
DDE1	PUSH	IX
FDE5	PUSH	IY
CB86	RES	Ø,(HL)
DDCBØ586	RES	Ø,(IX+d)
FDCBØ586	RES	Ø,(IY+d)
CB87	RES	Ø,A
CB8Ø	RES	Ø,B
CB81	RES	Ø,C
CB82	RES	Ø,D
CB83	RES	Ø,E
CB84	RES	Ø,H
CB85	RES	Ø,L
CB8E	RES	1,(HL)
DDCBØ58E	RES	1,(IX+d)
FDCBØ58E	RES	1,(IY+d)
CB8F	RES	1,A
CB88	RES	1,B
CB89	RES	1,C
CB8A	RES	1,D
CB8B	RES	1,E
CB8C	RES	1,H
CB8D	RES	1,L
CB96	RES	2,(HL)
DDCBØ596	RES	2,(IX+d)
FDCBØ596	RES	2,(IY+d)
CB97	RES	2,A
CB9Ø	RES	2,B
CB91	RES	2,C
CB92	RES	2,D
CB93	RES	2,E
CB94	RES	2,H
CB95	RES	2,L
CB9E	RES	3,(HL)
DDCBØ59E	RES	3,(IX+d)
FDCBØ59E	RES	3,(IY+d)
CB9F	RES	3,A
CB98	RES	3,B
CB99	RES	3,C
CB9A	RES	3,D
CB9B	RES	3,E
CB9C	RES	3,H
CB9D	RES	3,L
CBA6	RES	4,(HL)
DDCBØ5A6	RES	4,(IX+d)
FDCBØ5A6	RES	4,(IY+d)
CBA7	RES	4,A
CBAØ	RES	4,B
CBA1	RES	4,C
CBA2	RES	4,D
CBA3	RES	4,E
CBA4	RES	4,H
CBA5	RES	4,L
CBAE	RES	5,(HL)
DDCBØ5AE	RES	5,(IX+d)
FDCBØ5AE	RES	5,(IY+d)
CBAF	RES	5,A
CBA8	RES	5,B
CBA9	RES	5,C
CBAA	RES	5,D
CBAB	RES	5,E
CBAC	RES	5,H
CBAD	RES	5,L
CBB6	RES	6,(HL)
DDCBØ5B6	RES	6,(IX+d)
FDCBØ5B6	RES	6,(IY+d)
CBB7	RES	6,A
CBBØ	RES	6,B
CBB1	RES	6,C
CBB2	RES	6,D
CBB3	RES	6,E
CBB4	RES	6,H
CBB5	RES	6,L
CBBE	RES	7,(HL)
DDCBØ5BE	RES	7,(IX+d)
FDCBØ5BE	RES	7,(IY+d)
CBBF	RES	7,A
CBB8	RES	7,B
CBB9	RES	7,C
CBBA	RES	7,D
CBBB	RES	7,E
CBBC	RES	7,H
CBBD	RES	7,L
C9	RET	
D8	RET	C
F8	RET	M
DØ	RET	NC
CØ	RET	NZ
FØ	RET	P
E8	RET	PE
EØ	RET	PO
C8	RET	Z
ED4D	RETI	
ED45	RETN	
CB16	RL	(HL)
DDCBØ516	RL	(IX+d)
FDCBØ516	RL	(IY+d)
CB17	RL	A
CB1Ø	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E

A s s e m b l e r: Z80-Befehlssatz

CB14	RL	H
CB15	RL	L
17	RLA	
CBØ6	RLC	(HL)
DDCBØ5Ø6	RLC	(IX+d)
FDCBØ5Ø6	RLC	(IY+d)
CBØ7	RLC	A
CBØØ	RLC	B
CBØ1	RLC	C
CBØ2	RLC	D
CBØ3	RLC	E
CBØ4	RLC	H
CBØ5	RLC	L
Ø7	RLCA	
ED6F	RLD	
CB1E	RR	(HL)
DDCBØ51E	RR	(IX+d)
FDCBØ51E	RR	(IY+d)
CB1F	RR	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
1F	RRC	
CBØE	RRC	(HL)
DDCBØ5ØE	RRC	(IX+d)
FDCBØ5ØE	RRC	(IY+d)
CBØF	RRC	A
CBØ8	RRC	B
CBØ9	RRC	C
CBØA	RRC	D
CBØB	RRC	E
CBØC	RRC	H
CBØD	RRC	L
ØF	RRCA	
ED67	RRD	
C7	RST	ØØH
CF	RST	Ø8H
D7	RST	1ØH
DF	RST	18H
E7	RST	2ØH
EF	RST	28H
F7	RST	3ØH
FF	RST	38H
DE2Ø	SBC	A,n
9E	SBC	A,(HL)
DD9EØ5	SBC	A,(IX+d)
FD9EØ5	SBC	A,(IY+d)
9F	SBC	A,A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
ED42	SBC	HL,BC
ED52	SBC	HL,DE
ED62	SBC	HL,HL
ED72	SBC	HL,SP
37	SCF	
CBC6	SET	Ø,(HL)
DDCBØ5C6	SET	Ø,(IX+d)
FDCBØ5C6	SET	Ø,(IY+d)
CBC7	SET	Ø,A
CBCØ	SET	Ø,B
CBC1	SET	Ø,C
CBC2	SET	Ø,D
CBC3	SET	Ø,E
CBC4	SET	Ø,H
CBC5	SET	Ø,L
CBCE	SET	1,(HL)
DDCBØ5CE	SET	1,(IX+d)
FDCBØ5CE	SET	1,(IY+d)
CBCF	SET	1,A
CBC8	SET	1,B
CBC9	SET	1,C
CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBD6	SET	2,(HL)
DDCBØ5D6	SET	2,(IX+d)
FDCBØ5D6	SET	2,(IY+d)
CBD7	SET	2,A
CBDØ	SET	2,B
CBD1	SET	2,C
CBD2	SET	2,D
CBD3	SET	2,E
CBD4	SET	2,H
CBD5	SET	2,L
CBDE	SET	3,(HL)
DDCBØ5DE	SET	3,(IX+d)
FDCBØ5DE	SET	3,(IY+d)
CBD7	SET	3,A
CBD8	SET	3,B
CBD9	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E
CBDC	SET	3,H
CBDD	SET	3,L
CBD6	SET	4,(HL)
DDCBØ5E6	SET	4,(IX+d)
FDCBØ5E6	SET	4,(IY+d)
CBE7	SET	4,A
CBEØ	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,D
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBEE	SET	5,(HL)
DDCBØ5EE	SET	5,(IX+d)
FDCBØ5EE	SET	5,(IY+d)
CBEF	SET	5,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBF6	SET	6,(HL)
DDCBØ5F6	SET	6,(IX+d)
FDCBØ5F6	SET	6,(IY+d)
CBF7	SET	6,A
CBFØ	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBFE	SET	7,(HL)
DDCBØ5FE	SET	7,(IX+d)
FDCBØ5FE	SET	7,(IY+d)
CBFF	SET	7,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CB26	SLA	(HL)
DDCBØ526	SLA	(IX+d)
FDCBØ526	SLA	(IY+d)
CB27	SLA	A
CB2Ø	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB2E	SRA	(HL)
DDCBØ52E	SRA	(IX+d)
FDCBØ52E	SRA	(IY+d)
CB2F	SRA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB3E	SRL	(HL)
DDCBØ53E	SRL	(IX+d)
FDCBØ53E	SRL	(IY+d)
CB3F	SRL	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
96	SUB	(HL)
DD96Ø5	SUB	(IX+d)
FD96Ø5	SUB	(IY+d)
97	SUB	A
9Ø	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
D62Ø	SUB	n
AE	XOR	(HL)
DDAEØ5	XOR	(IX+d)
FDAEØ5	XOR	(IY+d)
AF	XOR	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
EE2Ø	XOR	n

RAM 4.x: Patches

RAM43 und das Laufwerk G (Wolfgang Dexheimer, 6719)

Das Installieren von RAM43 ging problemlos, die Leuchtdiode für ALPHA LOCK sowie der Bildschirmschoner sind schon eine feine Sache. Auch die anderen Veränderungen erhöhen den Komfort. Doch da gibt es noch das Laufwerk G. Bei mir die c't EPROM-Floppy. Trotz CFIG43 G:57 wurde sie nicht mehr erkannt. Plötzlich schien sie keine Files mehr zu enthalten. PIPEF zeigte mir jedoch, daß sie tatsächlich gefüllt war. Auch mit CFIG4 H:57 war sie ansprechbar (aber leider sind meine Overlays auf G eingestellt). Alles neu brennen? Nein erstmal mit MONI in RAM43 reingeschaut! Auf der Diskette steht an der betreffenden Stelle bei mir folgendes:

```
      LD A,(0FFE8H)   also Laufwerksnummer nach A
12AE  CP 05           mit G vergleichen ?
      JR NZ, ...      nicht G, dann erhält C die richtige Portadresse (A4, A8);
                      ansonsten E4 bzw. E8
```

Da meine EPROM-Disk aber bei A4 liegt, kann sie nicht mehr als Laufwerk G angesprochen werden. Weshalb wurde diese Laufwerksfestlegung gemacht, und wo wurde man darüber informiert ???

Ich habe kurzerhand bei Adresse 12AF die 05 in eine 07 (also Laufwerk I, ich habe kein Laufwerk I) umbenannt; damit arbeitet nun meine EPROM-Floppy wieder unter G.

Anm.d.HzN.: Diese Zugriffslogik auf hat den Grund, den Betrieb von zwei solchen Floppies zu unterstützen, die logischerweise auf unterschiedlichen Port-Adressen liegen müssen. (War von BP so vorgesehen.)
Ich verspreche mich in Sachen RAM 5.x bei solchen Dingen zu bessern (INST5x).

Fehler in MSFORM4 (Michael Keßler, 5600)

Nun bin ich endlich mal dazu gekommen, mir die MSDOS-720k-Diskette anzugucken. Wolfgang Tesch hatte recht: In unserem MSFORM wurde die falsche Version des 80T/DS/9S erwischt. Es gibt dieses Format zwar auch, es findet aber nur auf der Lieferdiskette für PCDOS 3.3 Verwendung. Die korrekte "Normalversion" entspricht derjenigen von Wolfgang. Ich habe MSFORM entsprechend gepatcht (liegt bei).

Es sind lediglich zwei Bytes zu ändern:
- in Adresse 0A1F muß 2 statt 1 stehen
- in Adresse 0A25 muß 3 statt 5 stehen

Anm.d.HzN.: Korrekte Version ist auf KLICK.005.

Macke mit RAM 4.4, 4.5 (Herbert zur Nedden, 2000)

Die Aufrufe 'RAM43 M' (Konfiguration speichern) und 'RAM43 F name' (F-Tabelle laden) funktionieren leider nicht korrekt mit RAM 4.4 und 4.5, d.h. NACH dem Aufruf von P44?DX bzw. P45?DX (?=S oder F). Ursache dafür ist, daß RAM43 bei diesen beiden Aufrufen ein RAM 4.3 vorfinden will, und nichts anderes. Da andererseits nach P44?DX bzw. P45?DX unbedingt die RAM-Versionsnummer verändert werden muß, damit Programme (insbesondere Olafs .KLX-Loader) dieses erkennen können, ist im Speicher kein RAM 4.3 mehr. Willst Du trotzdem RAM43 F oder RAM43 M benutzen, muß daß ohne P44?DX/P45?DX erfolgen. Da das aber eh nur für Installations-Dinge gemacht wird, meine ich, daß man damit leben kann.
Übrigens: patchen von RAM43 auf die Versions-Nummer 4.4 bzw. 4.5 ist nicht empfehlenswert, da P44?DX/P45?DX ein RAM 4.3 vorfinden wollen!

S o f t w a r e: RAM 4.5 / FORMSTAR

RAM 4.5 (Herbert zur Nedden, 2000)

Kaum zu glauben, niwwa ? Ja, es gibt mittlerweile RAM 4.5.

1. Was kann es mehr:
   Das mitgelieferte P2DOS untersützt es, daß die beliebten 'P2DOS Err'-Fehler anders verarbeitet werden können, nämlich statt in einem Warm-Boot zu enden, was bei Olaf's Kopier-Anseh-Druck-u.s.w.-KLICK-Overlay DiJey schon dümmlich war, den Fehlergrund an das aufrufene Programm zurückzumelden, damit eben dieses den Fehler abfängt.
   Warum ? Nun ja, da habe ich unter NewWord einen für mich bedeutungsschwangeren Text, die Diskette ist voll, und ich will mittels DiJey mal kurz sehen, was ich auf eine andere Scheibe kopieren kann, um Platz zu schaffen. Und ich vergaß die Anmeldung (P2DOS Err R/O), oder das Format paßt nicht (P2DOS Err Bad Sector), oder was auch immer. Dann ernte ich statt einer DiJey-Fehlermeldung einen lapidaren Warm-Boot auf A: und mein schöner Text ist auch FUTSCH! Grand Malheur de Kack!
   Ab RAM 4.5 können Programme dem P2DOS mitteilen, daß es doch bitteschön diesen Mist zu unterlassen habe: eine Option die Ihr nicht mißbrauchen solltet - sie macht m.E. nur in Klickern wirklich Sinn.

   Außerdem lieferte DIR nach gewissen DiJey-Aktionen recht magische Resultate. Eine Änderung im ZCPR2 behebt eben dieses.

2. Wie es realisiert wurde:
   Es gibt zum einen neue Systemspuren sowie ein Programm P45FDX (P45SDX), welches NACH RAM 4.3 aufgerufen wird, um den Rest der Arbeit zu erledigen. Ja, P45?DX muß nach JEDEM start von RAM 4.3 laufen. Das alte P44?DX von KLICK.004 ist hiermit hinfällig.

3. Wie Du es erhältst:
   Entweder legst Du Dir KLICK.005 und/oder meine Diskette 'RAM 4.5' zulegst. Letztere hat eine Menü-Führung für die Anzeige einiger Informationen zu RAM 4.5 an und für sich sowie eine Menü-geführte Installation des Systems.

RAM 4.5 - Diskette (Herbert zur Nedden, 2000)

Mittlerweile ist der Wust von RAM 4.1 über 4.2, 4.3, 4.4 bis hin zu 4.5 recht undurchsichtig für viele geworden. Daher habe ich mich hingesetzt und eine Diskette zusammengestellt, die menügeführt Informationen zu RAM 4.5 bietet sowie durch die Installation des Systems führt. Alles was zu tun ist, ist die Diskette einzulegen und die Kiste einzuschalten. Vorher empfiehlt sich allerdings eine Backup-Kopie samt Systemspuren! Alles, was erforderlich ist, ein RAM 4.5-System zu starten ist mit drauf. Logischerweise auch RAM43.COM (daher muß Lizenz vorliegen) sowie einige wichtige Utilities drauf.

FORMSTAR-UPDATE 1K (Michel Keßler, 5600)

FORMSTAR existiert jetzt in der Version 1K und beherrscht nun auch die MSDOS-Formate. Ein Bug in FSK (Absturz nach Formatieren von 1C und 1D) ist beseitigt. Zusätzlich wird MFK.COM, eine Mini-Klix-Version, beigefügt, die nur noch die üblichen DS/DD-Formate (ohne MSDOS) beherrscht und auch nur im Automatik-Modus arbeitet, dafür aber um 8k kleiner ist (für schmale Klix-Heaps). Updates wie immer gegen Portoübernahme. MFK.COM ist im Preis von FormStar inbegriffen, einzeln kostet es 5,-- DM (o. P&V).

Programmbesprechung: (Michael Keßler, 5600)

DEBUGGER Z 8 E

Als notorischer Assemblierer größerer Programme braucht man halt stets einen Monitor/Debugger, um rauszufinden warums nicht klappt. Im Club scheint ja mittlerweile MONI favorisiert zu werden, sicherlich sprechen dafür auch gute Gründe. Nun habe ich MONI aber nicht (und ich brauche ihn auch nicht), da bei mir der

PUBLIC-DOMAIN-DEBUGGER Z 8 E

seine Pflicht erfüllt. Olaf mag mir verzeihen, wenn ich hier ein Konkurrenzprodukt bespreche, aber gerade einige Features machen eben diesen Debugger mit zum Besten was es gibt. Ich möchte die wichtigsten davon kurz aufzählen:

- Symbolischer Debugger:

  Wenn man ein Assembler-Programm erstellt, so gehören LABELS logischerweise zum Wichtigsten überhaupt. Man assembliert sein Programm und läßt sich dabei eine SYMBOL-TABELLE generieren (da sind alle Labels mit ihrer Adresse drin aufgeführt). Z8E kann eben diese Symbol-Tabelle zusätzlich zum COM-File laden und wenn man die LIST-Option wählt, kriegt man nicht nur den Assembler-Quellcode zu Gesicht, nein, auch alle Labels sind zu sehen. Das macht die Sache sehr viel übersichtlicher. Aber das ist nicht alles: Während des Debuggens kann man auch neue Labels setzen (und sogar Kommentare einfügen) und diese dann abspeichern. Dies ist wichtig, wenn man fremde Programme anschaut/ändert. Man kann bei allen Befehlen, die eine Adresse benötigen (LIST, ASSEMBLE, DUMP etc.) direkt den Namen des Labels angeben.

- Indirekte Adressierung:

  Z8E versteht indirekte Adressierung. Ich habe z.B. irgendwas im HL-Register und will diesen Wert als Breakpoint setzen. Dann kann ich einfach *B (HL) eingeben, und fertig ist das ganze. Diese indirekte Adressierung geht bei allen 16 Bit-Registern mit Ausnahme der Zweitregister (BC' etc.). Sie ist auch nicht auf Breakpoints beschränkt, sondern auch für EXAMINE MEMORY, GO usw.

- Speicherstellen ändern:

  Hier kann man dezimale, hexadezimale und ASCII-Werte eingeben, auch alles gemischt, und innerhalb EINER GANZEN ZEILE.

- Speicherstelle mit bestimmtem Inhalt suchen:

  Hier kann man ebenfalls nach ASCII-STRINGS, hexadezimalen und dezimalen Werten suchen, ebenfalls auch alles gemischt.

- Zugriff auf die Ports:

  Das ist für mich besonders wichtig, da ich viel mit dem Disc-Controller spiele. Hier stehen gleich zwei verschiedene Befehle zur Verfügung: Einerseits kann ich einfach irgendwelche Werte zu einem Port ausgeben (N-Befehl), andererseits wird der Port nach der Ausgabe auch gleich wieder eingelesen (Q-Befehl), dann kann man sofort Änderungen feststellen. Beispiel: OUT (44H) ist Control-Byte des Disccontrollers, IN (44H) ist das Status-Byte. Gebe ich was nach Port 44 aus, sehe ich sofort, was zurückkommt. Das ist echt super.

P r o g r a m m b e s p r e c h u n g: E A D / BACKUP

Das Allerbeste ist aber der bildschirmorientierte Trace-Modus. Hier ist der Bildschirm in drei Fenster aufgeteilt. Ganz oben werden die Registerinhalte und die Flags angezeigt. Darunter erscheinen ca. 20 Zeilen Assembler-Listing (ggf. mit Labels). Ein kleiner Pfeil bewegt sich dann auf dem Bildschirm von Befehl zu Befehl, man sieht also richtig IM LISTING, wie das Programm abgearbeitet wird. Zu allem Überfluß kann man sich als drittes Fenster noch einen Speicherdump ausgeben lassen, UND SIEHT SO DIREKT, WIE EINZELNE SPEICHERSTELLEN WÄHREND DES PROGRAMMABLAUFS VERÄNDERT WERDEN. Was will man mehr !!!

Ganz nebenbei kann Z8E natürlich alles, was ein Debugger sonst noch können muß, ein 92 Seiten starkes HANDBUCH (in Englisch) ist als DOC auf Diskette und auch die SOURCE ist nicht vergessen worden. Alles in allem ein empfehlenswertes Programmpaket, wenn auch einige wenige Bugs drin sind, so wird z.B. die Registerdarstellung (Fenster 1) im bildschirmorientierten Trace-Modus zerschossen, sobald auf den Zweitregistersatz zugegriffen wird. Hier braucht man aber bloß Return zu drücken und den Trace-befehl zu wiederholen, um alles wieder ins Lot zu bringen. Ein Bug, der in der Praxis kaum Bedeutung hat, zumal man, so Zeit genug vorhanden, diesen Fehler Dank der mitgelieferten Source selbst abstellen kann.

Programmbesprechung: (Michael Keßler, 5600)

E A D = EDITOR / ASSEMBLER / DEBUGGER

E A D ist eine Programmierumgebung für Z80-Assembler (Macro-fähig). Editor, Assembler und Debugger sind in einem Programm vereinigt. Das Paket wendet sich in erster Linie an den Assembler-Anfänger, ist aber auch zum Erstellen kleinerer Sachen für den Fortgeschrittenen sehr interessant. Zuerst gelangt man zum Editor. Dort gibt man sein Programm ein (jeder wird den Editor wiedererkennen, er basiert auf Herberts Mtx-Edit). Nach Beendigung der Eingabe gelangt man per Tastendruck in den Assembler, das Programm wird übersetzt. Sollten Fehler aufgetreten sein, kann man wahlweise in den Editor zurück, oder aber in den Debugger gelangen. Wenn man in den Editor zurückgeht, so steht der Cursor genau auf dem Fehler (falls Tippfehler), GANZ GENAU WIE IN TURBO-PASCAL. Überhaupt ist die ganze Sache von der Gestaltung her ähnlich wie Turbo ausgelegt. Der Bedienkomfort ist auf die Spitze getrieben (eben genau das Richtige für Anfänger und solche die es erst werden wollen). Ca. 48K Source können eingegeben werden (halt wie beim Mtx-Edit), COM-Files können sofort erzeugt werden, ein Linken ist nicht möglich (keine RELs).

Das Programm wird an der Uni Wuppertal im Fachbereich Digitaltechnik (dort ist es auch entstanden) im Grundpraktikum als Lehrmittel für Assembler-Programmierung eingesetzt.

Betrifft BACKUP: (Holger Hansen, 3300)
(Kopierprogramm, das sich an Zeiteinträgen orientiert)

Abgesehen davon, daß es sich jetzt um eine ganze Programfamilie handelt, würde ich von den bisherigen Benutzern ganz gerne mal erfahren, ob es auf anderen Rechnerkonfigurationen überhaupt sinnvoll einsetzbar ist (Vor allem bei "nur" Diskettenbesitzern). Mit Ausnahme von Manfred Flume hat sich nämlich noch keiner mit Kritik irgendeiner Art und Weise gemeldet. Dies würde zwar bedeuten, daß die Programme perfekt sind, aber irgendwie kann ich es nicht glauben, weil ich diese Programme in erster Linie für mich schreibe. Ich würde mich auch über (konstruktive) Kritik zur dazugehörigen Dokumentation freuen, weil das mein großer Schwachpunkt ist. (Und nur wenn sich der geneigte Benutzer meldet, kann ich versuchen die Programme (Soft- und Paperware) zu verbessern!)

P r o g r a m m b e s p r e c h u n g: Grundsätzliches / Wort Manager

Programmbesprechungen (Dr. Holger Göbel, 8630; 09561/15131)

Schon oft wurde angeregt, Programme zu besprechen, seien es gekaufte, selbstgeschriebene oder über Public Domain allgemein zugänglich gemachte. Ich selbst finde das ausgesprochen gut und viel zu wenig praktiziert. So bin ich bei der Durchsicht **alter** Public-Domain-Disketten unseres Clubs auf wahre Schätze gestoßen, von denen ich bisher keine Ahnung hatte. Wenn ich nur weiter suche, werde ich sicher noch mehr davon finden (vgl. bitte dazu auch meinen Beitrag über die Pascal *.BIB, die Horst Kupka auf PD 13 zur Verfügung gestellt hat).

1. Der **WORTMANGAGER** von Claudio Romanazzi (Dr. Holger Göbel, 8630; 09561/15131)

Um so mehr war ich erfreut, als mich Claudio Romanazzi bat, seinen **WORTMANAGER** (ein Orthographie-Programm mit vielen nützlichen Funktionen), das er in diesem INFO vorstellt, zu testen und meine ehrliche Meinung dazu kundzutun.
Zunächst seien mir einige grundsätzliche Bemerkungen zu einem solchen Programm und zu seiner Programmierung erlaubt:

Jede Sprache ist geschichtlich gewachsen, hoch sensibel, von den Eigenarten der sie Sprechenden geprägt und muß deshalb stets zu Kompromissen bereit sein. Daraus folgt aber auch, daß sie in sich hochgradig inkonsequent ist. Es dürfte nur sehr wenige Regeln geben, die ohne Ausnahme gültig sind; schlimmer noch, viele Eigenarten scheinen fast regellos **einfach zu existieren.** Jeder Mathematiker würde ob eines solchen Konstruktes die Hände über dem Kopf zusammenschlagen, kein Ingenieur könnte damit etwas anfangen. Die Sprache ist aber nun eben gerade kein Konstrukt, und trotzdem geht jeder von uns mehr oder weniger selbstverständlich damit um. Dies ist eine im Einzelnen nicht zu verstehende Großartigkeit des menschlichen Geistes, der ein Computer der herkömmlichen Technologie nichts entgegenzusetzen hat; ob die sog. "Künstliche Intelligenz" mit ihren "assoziativen Speichern" oder "neuronalen Netzwerken" da einen entscheidenden Durchbruch bringen wird, wage nicht nur ich zu bezweifeln (vgl. auch z.B. c't 2/89).

Seitdem die Speicher der Computer größer, die Rechengeschwindigkeit schneller und die dumme Ahnungslosigkeit der "User" immer fataler werden, gibt es bei vielen Textprogramm-Paketen z.B. der MSDOS-Rechner ein Orthographie-Programm. Da die Computer des herkömmlichen Systems (Neumann) mit der menschlichen Sprache nicht mithalten können, müssen sie also zum Zweck der Orthographie-Überprüfung grundsätzlich in einer Bibliothek nachschauen, ob das entsprechende Wort dort gespeichert ist. Die Bibliothek selbst mußte natürlich erst von ein paar Emsigen erstellt werden.
Nun, diese Arbeitsweise des Computers wäre ja noch kein Nachteil, denn immerhin ist er ja unheimlich schnell, was soll's also? Fragt einmal Claudio, auf welche Schwierigkeiten er da gestoßen ist: Groß-/Kleinschreibung etwa (jedes Verb kann ja auch groß geschrieben werden, z.B. Goethe: "Wer sich zum Gesetz macht, das Tun am Denken, das Denken am Tun zu prüfen, der kann nicht irren; ...") oder die Behandlung von Abkürzungen. Ein anderes Beispiel: "Claudio ißt ein guter Programmierer" wird vom Programm sicher nicht als falsch erkannt, ein ganz schlauer Algorithmus würde vielleicht noch daraus machen "Claudio ißt einen guten Programmierer" (das passiert natürlich nicht beim Wortmanager!).

Wie Claudio in seiner Dokumentation zum Programm richtig erwähnt: Man muß wissen, was man vom Programm erwarten kann, nicht mehr und nicht weniger. Grammatikfehler wird es nicht finden und schlechten Schreibstil kann es schon gar nicht verbessern.

Programmbesprechung: Wort Manager

Aber: Jeder von uns kennt das: Man muß ein Dokument verfassen, man ist ja eigentlich auch sehr gut in der Rechtschreibung und man liest sich den Text bestimmt zweimal durch. Frühestens bei der Weitergabe des Dokuments merkt man aber, daß sich doch der berühmte Dreckfuhler-Teufel eingeschlichen hat. Ärgerlich, wenn es sich gerade um eine Bewerbung o.ä. handelt.
Und dafür ist so ein Orthographie-Programm gut. Ich selbst setze es erst in letzter Instanz ein; es erspart mir nicht, das Geschriebene selbst durchzulesen (ißt - ist!). Ich möchte es trotzdem nicht mehr missen, v.a., wenn noch einige Holprigkeiten ausgebügelt werden, die in der 1. Version halt noch bestehen (dazu später mehr).

Nun zum Programmieren eines solchen Werkes: **Ich** hätte es mir nicht zugetraut, Claudio hat ein Jahr dafür gebraucht. D.h. im Klartext: Es ist nicht nur ein hervorragend durchdachtes, im Bedienerkomfort seinesgleichen suchendes Programm, sondern auch noch ausgesprochen schnell. Dazu trägt bei, daß es rein in Assembler geschrieben ist und die Bibliothek nach einem intelligenten (von Claudio erdachten) Algorithmus aufgebaut ist. Was noch viel mehr wiegt ist der Pioniergeist, den Claudio hier an den Tag legt: "Es gibt für uns noch kein solches Programm - also schreib' ich es eben und stell' es den anderen zur Verfügung" oder "Newword kann zu wenig (z.B. keine Graphik) - ich programmiere etwas Besseres". Keine Spur da von frustrierten Usern, die den MTX, den C64 & Co. Zur Seite stellen, weil es jetzt IBM-PC'S gibt, und die sich später meinetwegen PS/2-Computer kaufen, weil die halt moderner sind und natürlich auch besser.

So - genug jetzt des Vorgeplänkels (es lag mir halt am Herzen), kommen wir zum Programm selber:

WORTMANAGER ist also ein Orthographieprogramm, das auf einen ASCII-Text losgelassen werden kann, auch wenn er z.B. Mit NEWWORD erstellt ist (mit kleinen Einschränkungen, s.u.). Es vergleicht das im Text auftauchende Wort mit den Wörtern, die in der Bibliothek abgelegt sind (mitgeliefert wird eine Bibliothek mit ca. 55.000 Wörtern!). Findet es ein entsprechendes, so ist alles in Ordnung. Im anderen Fall kann es ja sein, daß man sich vertippt hat. Dann kann man das Wort verbessern und WORTMANAGER schaut nach, ob das verbesserte Wort in der Bibliothek enthalten ist. Wenn ja, dann bessert er es gleich im Text aus (sehr nett!). Wenn nicht, dann kann man entscheiden, ob das Wort in die Bibliothek aufgenommen wird oder aber nicht (weil es z.B. zu speziell ist - wer nimmt schon an, daß ihm ein zweites Mal der König von Sparta namens "Agesilaos" begegnet!).
WORTMANAGER stellt vor der Übernahme eines großgeschriebenen Wortes noch die wichtige Frage, ob es **immer** groß geschrieben wird (aufpassen!: "Wir sitzen auf den Sitzen" ...).
Man kann also seine Bibliothek selbst erweitern. Darin liegt eine große Verantwortung, im Zweifelsfall immer im Duden nachgucken!
Die neu aufzunehmenden Wörter werden erstmal zwischengespeichert, zum einen in einem Servicepuffer (damit beim nächsten Auftauchen des Wortes nicht schon wieder danach gefragt wird) und in einer Datei NEU.SRT. Diese kann später editiert und evtl. korrigiert oder erweitert werden. Erst ein ganz spezieller Programmteil sorgt dafür, daß diese Datei NEU.SRT in die Bibliothek endgültig einsortiert wird. Das hat folgenden Grund: Eine Bibliothek von 55.000 Wörtern wäre riesig groß. Da bestimmte Wörter den gleichen Anfang haben, hat Claudio nicht immer jedes Wort ganz abgespeichert, sondern nur den Teil, der sich vom vorhergehenden unterscheidet; die Bibliothek ist also komprimiert. Das Einsortieren nun muß natürlich darauf Rücksicht nehmen und außerdem Platz für das alphabetisch einzusortierende Wort schaffen. Deshalb dauert das Einsortieren auch sehr lang: Claudio schreibt, bei 50.000 Wörtern etwa 20 Minuten (7 MHz Taktfrequenz). Da stellt er sein Licht unter den Scheffel. Auf meiner RAM-Disk braucht er bei 8 MHz 9 1/2 Minuten, auf meinem 3,5''-Laufwerk 12 Minuten. Wahrscheinich meint er die Rechner mit 4 MHz.

Programmbesprechung: Wort Manager

Da taucht nun auch gleich ein Problem auf: Der Speicherplatz. Die Bibliothek nimmt in der jetzigen Form etwa 185 kB in Anspruch. Wenn eine neusortierte Bibliothek entsteht, muß mindestens derselbe Platz nochmal auf der Diskette vorhanden sein. Laufwerke mit der Konfiguration 3 haben da keine Chance, mit der 9 wohl, aber wie lange noch? (die Bibliothek soll ja schließlich wachsen). Ich selbst werde schleunigst meine Speichererweiterungskarte auf 768 kB aufrüsten, denn auch die 512 kB reichen da nicht (schließlich läuft ja RAM4x auf der Speichererweiterung, und auf die KLICKER möchte ich deswegen nur sehr ungern verzichten).
Claudio sollte vielleicht vorsehen, daß die **neu** sortierte Bibliothek auf ein **anderes** Laufwerk gelegt werden kann; dann hätten auch die eine Chance, die nur zwei 5 1/4-Zoll-Laufwerke mit je 40 Spuren besitzen (wohl sehr viele).
Obwohl ja nun das **Einsortieren** auf einem echten Laufwerk ganz flott geht, würde ich beim **Testen** eines Textes trotzdem immer die Bibliothek in die RAM-Disk laden, denn das sequentielle Suchen auf der echten Diskette dauert schon furchtbar lang.
Weitere **Optionen** des Programms: Man kann auch Wörter wieder aussortieren oder man kann sie in der Bibliothek suchen, sinnvollerweise mit den Jokern ? und *.

Alles in allem also ein umfangreiches, wertvolles Programmpaket. Gleich noch etwas Wichtiges: Es läuft nur auf einem System mit dem Betriebssystem RAM > 3 (vielleicht auch =3), weil es wegen der Benutzerfreundlichkeit Bildschirmfenster aufbaut und aus Geschwindigkeitsgründen den MTX+++-Treiber verwendet.

Jetzt zu einigen Holprigkeiten, die noch ausgemerzt werden müßten:

1. Beim Verbessern eines Wortes findet man sich im NON-INSERT-Modus; ich selber würde den INSERT-Modus bevorzugen.
2. WORTMANAGER hat Schwierigkeiten mit dem Soft-Hyphen aus NEWWORD-Texten. Bei meiner WM-Version nimmt er immer zwei getrennte Wörter, die er dann natürlich nicht findet. Das passiert sowohl am Zeilenende als auch dann, wenn in der Zeile ein Wort seinen Soft-Hyphen von einer früheren Trennung noch trägt. Das kann ziemlich fatal werden, ein Beispiel: Im Text steht "lie=gen". WM erkennt "lie" nicht, ich übergehe das Wortfragment, d.h., es wird nicht in die Bibliothek aufgenommen. Jetzt müßte mich WM ja auch nach dem zweiten Teil des Wortes fragen, nach "gen". Da er aber "Gen" in seiner Bibliothek gespeichert hat, kommt zum Schluß "lie=Gen" heraus.
   Die Absätze sollten sowieso erst nach der Korrektur formatiert werden (mit ^B), denn es kann ja vorkommen, daß ein falschgeschriebenes Wort durch ein anderes ersetzt wird, und dann stimmt die ganze Formatiererei nicht mehr.
3. WM behandelt zwar Abkürzungen gesondert, bei mir hat er aber alle Abkürzungen in Großbuchstaben verwandelt: "z.B." wird zu "Z.B", "s.u." zu "S.U.". Und noch etwas: Ein nachfolgend klein geschriebenes Wort wird wegen des Punktes nach der Abkürzung einfach groß umgewandelt: "u.a. die ..." wird zu "U.A. Die ...".
4. Wenn einmal ein Leerzeichen vergessen wurde (z.B. "vondem" statt "von dem", so gibt es keine Möglichkeit, WM zu verklickern, daß man es jetzt richtig eingeben will, denn der Editor von WM kennt kein SPACE.
5. Hat man unter NEWWORD einen **Teil** eines Wortes durch Bold oder Unterstreichen hervorgehoben, so erkennt WM **zwei** Wörter, z.B.: Bei "^BZwischen^Bsumme" wird "summe" einfach groß geschrieben ("^BZwischen^BSumme").
6. Auch die Gliederungssymbole "a) .. b) .." wandelt WM einfach in die Großschreibweise um: "A) .. B) ..". Überhaupt scheint WM Schwierigkeiten mit Einzelbuchstaben zu haben, denn z.B. wird "er sah's" umgeändert in "er sah'S".
7. Wenn neue Wörter einsortiert werden, dann zählt immer ein Zähler auf dem Bildschirm mit. Ich kann mir vorstellen, daß die Umwandlung einer HEX-Zahl in einen dezimal darzustellenden Code und die Anzeige selbst doch einiges an Zeit vergeudet, die beim Einsortieren sowieso schon lang wird. Vielleicht sollte lediglich ein blinkendes Symbol o.ä. zu verstehen geben, daß WM gerade arbeitet.

Programmbesprechung: Wort Manager

8. Einige Wörter in der Bibliothek sind - trotz der Durchsicht von Claudio - noch falsch. Mir ist z.B. aufgefallen, daß "durcheinander" immer groß geschrieben wird (da hat einer beim Eingeben nicht aufgepaßt!). Da WM kommentarlos in einem solchen Fall einfach "Durcheinander hinsetzt, wird dieser Fehler in jedem "korrigierten" Text auftauchen, auch wenn ich es primär richtig geschrieben habe - ohne daß ich es merke! Vielleicht sollte, bevor die Bibliothek nicht ganz "sauber" ist (das wird sie wohl nie sein), grundsätzlich von WM nachgefragt werden, ob ein Wort gegen ein anderes eingetauscht wird (es sei denn bei der Option, wo WM **alle** Buchstaben in Großbuchstaben umwandelt).
9. Es existiert ja ein Servicepuffer (s.o.). Dieser ist jedoch immer nur für eine Sitzung aktiv. Bei einer neuen Sitzung müßte NEU.SRT, sofern vorhanden, in diesen Servicepuffer eingelesen werden (falls genügend Platz ist), denn WM schaut nicht in NEU.SRT nach, ob man das Wort schon einmal übernehmen wollte, aber halt noch nicht in die Bibliothek einsortiert hat (was man wegen des Zeitaufwandes sowieso nicht so oft tut).

A. So komisch es klingt - die Bibliothek ist doch noch ziemlich klein. Da sind allerdings alle aufgerufen, sie im Laufe der Zeit zu vergrößern (die Bibliotheken sollen durch Claudio eingesammelt, durchgesehen und zusammengefaßt werden).
B. Etwas Penibles zum Schluß (für ein Orthographieprogramm aber doch von Bedeutung): WM meldet sich mit: "nnnnnn Worte an Bord". Es müßte hier "Wörter" statt "Worte" heißen, denn die beiden Pluralformen haben unterschiedliche Bedeutung.

Das sind also doch einige Kritikpunkte, die eine "Version 1" über sich ergehen lassen muß. Ich habe sie vorab Claudio geschickt und bin sicher, daß er sie zwischenzeitlich, soweit möglich, berücksichtigt hat.
So bleibt mir am Ende nur noch zu Wünschen, daß Claudio sein nächstes Projekt genauso bravourös löst, nämlich ein verbessertes Textprogramm auf die Beine zu stellen, das die Silbentrennung automatisch durchführt und Graphik einbinden kann.

Anm.d.HzN.:

1. Ich war so frei, die Numerierung von Holgers 'Kritik'-Punkten auf Hex-Numerierung (1-B statt 1-11) umzunumerieren, da ich zu spät merkte, daß es über 9 hinausgeht, und keine Lust verspürte, der Schönheit wegen die gesamte Reformatierung auf 80 Zeichen je Zeile mit eingerücktem Text (^O^G) zu wiederholen.
2. Thema 'Service-Puffer' (Holger's 9.): Die Idee habe ich Claudio mit folgendem Hintergrund näher gebracht - und zu meiner großen Freude hat er sie auch sehr prompt realisiert: Insbesondere in Briefen habe ich i.a. in Form von Eigennamen und ähnlichem seltene Worte, die ich dann einfach für gültig erklären kann, damit ich nicht in der Anschrift, der Anrede und sonstwo auf den Namen Claudio mit 'ist ok' reagieren muß. Er soll dieses Wort erst einmal akzeptieren. Anschließend würde ich diesen aus NEU.SRT löschen, wenn ich das Wort nicht auf Dauer in meine Bibliothek haben möchte (ich glaube 'Claudio' war ein schlechtes Beispiel).
   In Kurz: Ich dachte diesen Service-Puffer für temporäte individuelle Zwischenspeicherung von Worten, die durchaus aus NEU.SRT auch verschwinden sollten. Liegt mir in der Tat etwas an dem Inhalt von NEU.SRT, nehme ich es halt in die Bibliothek auf.

P r o g r a m m b e s p r e c h u n g: Turbo-Pascal-Tools

2. WINDOW., HILFE., SELEKT. und MASKE.BIB von Horst Kupka

(Dr. Holger Göbel, 8630; 09561/15131)

Diese Programmierhilfen für TURBO-Pascal habe ich, wie schon gesagt, auf PD 13 jetzt erst gefunden. Sie sind jedoch so gut, daß ich mich wundere, warum selbst "Vielprogrammierer" noch nicht auf sie aufmerksam geworden sind (wahrscheinlich, weil sie noch nicht besprochen wurden).
Für wen sind diese Routinen, was leisten sie und wann sollte man sie einsetzen?

Gedacht sind sie natürlich für Pascal-Programmierer. Die Vorteile der Programmiersprache Pascal brauche ich, glaube ich, nicht zu erwähnen (zu meiner Schande muß ich gestehen, daß ich sie selber nur unzulänglich beherrsche, aber ich will fleißig üben). Jedenfalls ist es bei Pascal so, daß man Programmbibliotheken anlegen kann, die dann in beliebige Programme eingebunden werden können und da ihren Dienst tun. Man muß ihnen nur immer einige Parameter übergeben, und schon arbeiten sie flexibel und zuverlässig, da sie ja möglichst allgemein gehalten und ausgetestet sind.
Bemerkenswert ist, daß die Routinen von Horst kein RAMxx voraussetzen (obwohl sie da natürlich auch funktionieren), sie laufen nach geringer Anpassung auf jedem CP/M- (vielleicht auch MSDOS-) -Rechner.
Die Module dieser Programmbibliothek dienen v.a. dazu, Programme benutzerfreundlicher zu machen. Da die Programmflut immer größer wird und auch immer mehr Benutzer zu erreichen sucht, ist Bedienungskomfort ein nicht zu unterschätzendes Element des Programmierens.

WINDOW.BIB ist also ein Modul, das Fenster verwalten kann. Fenstertechnik ist in modernen Programmen sehr beliebt geworden (KLICK benutzt ja auch ein solches). Wenn der Benutzer irgendeine Eingabe macht, z.B. Hilfe sucht, so öffnet sich ein Fenster, und die geforderte Hilfe steht darinnen.
Eigentlich ist es ja kein Fenster ("Window"), denn durch ein solches müßte man ja durchschauen können (vielleicht ist das im angloamerikanischen "Kultur"kreis anders). Unser Fenster jedenfalls ist mehr wie ein Blatt Papier, das sich über den gerade aktiven Bildschirm legt, und das bei Bedarf wieder weggenommen werden kann. Schlimmer noch: man kann auch ein neues Papier auf dieses drauflegen, vielleicht ganz bedeckend, vielleicht nur halb usw. usw. Der Vorteil ist nicht nur, daß alles ganz nett aussieht, man hat auch noch das Gefühl, daß das eigentliche Hauptprogramm immer noch da ist und man nur ein paar Hilfsblätter drübergelegt hat.
Der Vorteil, daß WINDOW.BIB auf allen CP/M-Rechnern laufen kann, kehrt sich für den zum Nachteil, der seinem MTX für immer die Treue geschworen hat und die WINDOW42.INC-Routinen von Olaf Krumnow kennt. Diese sind ganz auf den MTX zugeschnitten, großteils in Assembler geschrieben und deshalb konkurrenzlos schnell.

HILFE.BIB ist ein wirklich starkes Programm, das die Fenstertechnik gleich ausnutzt. Wie gesagt, Benutzerfreundichkeit ist alles, und jeder Benutzer braucht auch mal Hilfe. Deswegen wird in einem Auswahlmenü eines Programms häufig auch "Hilfe" angeboten. Mit HILFE.BIB geht das nun so vonstatten: Der Programmierer hat vorher eine Hilfsdatei "HILFE.TXT" nach gewissen Regeln erstellt. HILFE liest diese ein, denn darin werden ja hoffentlich ein paar Ratschläge untergebracht sein.
In diesem Hilfetext können ja jetzt z.B. auch ein paar "Unterhilfen" angeboten werden. Auch das verwaltet HILFE.BIB: Die Unterhilfen können angewählt werden, es legt sich dann ein entsprechendes "Unterfenster" drüber usw.
Ein Beispiel: Ein Programm (das ist sogar auch auf PD 13 von H.K.!) verwaltet Adressen. Im ersten Menü stehen so all die Funktionen, die das Programm kann, z.B. neue Adressen eingeben und natürlich auch die Hilfefunktion. Die wähl ich mal aus. Das Hilfefenster gibt ein paar Informationen und ich darf angeben, wo ich speziell Hilfe brauche, etwa beim Adressen-Neueingeben oder -Suchen oder -Löschen usw. Zum Schluß bin ich schlauer als vorher und kann mich mit "Quit" wieder ins Hauptprogramm zurückhangeln.

P r o g r a m m b e s p r e c h u n g: Claudio's Antwort

SELEKT.BIB ist nun das Modul, das eigentlich zwangsläufig dazugehört, es wird in HILFE.BIB natürlich genutzt. Es ist für die gedacht, die gerne mit Cursor-Tasten, Joysticks oder - vornehmer - mit Mäusen umgehen:
Ein Menü bietet ja immer einige Auswahlmöglichkeiten an. Bei SELEKT stehen diese untereinander, eine, die aktive, ist hell unterlegt. Man kann sich nun irgendeine aussuchen (na was denn sonst), aber auf angenehme Weise: Zum einen ist es erlaubt, den Anfangsbuchstaben des Menüpunktes einzugeben (die müssen dann natürlich dann auch unterschiedliche Anfangsbuchstaben haben); zum anderen kann man auch mit den Cursortasten das aktive Feld nach oben oder unten verschieben und mit RET an"klicken".

MASKE.BIB erlaubt es dem Benutzer, seine Daten in eine Maske einzugeben (jeder dBase-Benutzer kennt das); z.B. ist bei dem Adressenprogramm ein bestimmtes Eingabefeld für den Vor- oder Nachnamen, die Straße usw. vorgesehen. Der Cursor springt nach Abschluß der einen Eingabe automatisch zur nächsten. Das klingt selbstvertändlich, ist aber gar nicht so einfach zu programmieren.

Anm.d.HzN.: Für die Adreßverwaltung auf CLUB.036 habe ich auf dieses MASKE.BIB deutlich erweitert und u.a. um verschiedene Handhabung des Cursors, wenn das Ende des Eingabefeldes erreicht wurde und diverse Spezial-Feldtypen (Datum, Zahl mit 2 Nachkommastellen, Bitfeld, u.s.w.). Daher empfehle ich, gleich MASKE.INC von CLUB.036 zu verwenden - es ist sozusagen ein MASKE.BIB-Upgrade.

Resümee: Die Module erleichtern benutzerfreundliches Programmieren sehr. Ich werde sie in Zukunft gerne nutzen. Eine wirklich schöne Sache wäre es, wenn Olaf und Horst sich zusammentäten, und diese ganze Bibliothek mit den schnelleren Fensterroutinen von Olaf zum Laufen brächten. Da sieht man schon gar nicht mehr, wie schnell das Fenster auf- und wieder abgebaut wird. Ich hätte diese Aufgabe gerne selber übernommen (obwohl ich, wie gesagt, noch fleißig üben muß), aber Olaf hat seine Routinen sehr schlecht (eigentlich gar nicht) kommentiert und Horst die seinen lückenhaft. Da tut sich jeder schwer, der nicht ganz fit ist. Denn auch hier gilt, daß Benutzerfreundlichkeit alles ist, auch wenn der Benutzer in diesem Fall ein Programmierer ist.

Antwort auf Holgers Programmbesprechung zu WortManger (Claudio Romanazzi, 3070)

Nachdem Holger meinen Wortmanager gründlich zerrissen hat (danke Hoger!), möchte ich zu den einzelnen Punkten Stellung nehmen:

zu 1) Ansichtssache! Wenn man 50.000 Wörter geprüft hat, tränen einem die Augen und man macht doch hin und wieder Fehler, weil eben auch mal mehrere Fehler in einem Wort stecken können. Deswegen hat sich bei mir bewährt, den Overwrite-Modus zu verwenden (ist auch einfacher zu programmieren). Das zu korrigierende Wort wird der Aufmerksamkeit des Benutzers bis zum bitteren Ende vorgeführt.

zu 2) Die Schwierigkeiten mit den Softhyphen sind erledigt. Da war doch tatsächlich noch eine Fehler drin. Ich hatte übersehen, daß Newword ein Softhyphen (1fh), das nicht verwendet wird, als 1eh parkt. Die Abfrage auf diesen Wert hatte ich nicht abgefragt. Wie gesagt, ist das Problem jetzt erledigt.

P r o g r a m m b e s p r e c h u n g: Claudio's Antwort

zu 3) Wortmanager behandelt einzelne Buchstaben als Nicht-Wort. Sie werden deswegen nicht auf Syntax geprüft und können deswegen auch nicht als Abkürzungen gespeichert werden. Ich habe jetzt folgenden Kompromiß geschlossen: Wird Großschreibung automatisiert und ist das letzte Wort eines Satzes eine Abkürzung (auch einbuchstabig), so wird die Großschreibung am Anfang eines Satzes nicht abgefragt. Das bedeutet, man muß selbst darauf achten, am Satzanfang großzuscheiben, falls man sich angewöhnen sollte (durch Wortmanager ja möglich), Satzanfänge einfach klein zu schreiben. Eine andere Möglichkeit sehe ich im Moment nicht. Daß die Einzelbuchstaben großgeschrieben wurden, war ein Flagfehler. Der ist ebenfalls beseitigt.

zu 4) Jetzt ist Wortmanager 'Space' bekannt.

zu 5) Noch ungelöst. Ich mache mir aber Gedanken. Spätestens mit den gesammelten Bibliotheken gibt es ein Update, das dieses Problem löst. Geplant sind auch freie Eingaben von Laufwerken und Usern und eine Zählfunktion, die (na was wohl) die Anzahl der Wörter eines Textes zählt (wichtig für Schriftsteller?!).

zu 6) Auch das ist erledigt. Hier trieb derselbe Flagfehler, wie in 4) sein Unwesen.

zu 7) Hier hat Holger die Maschinensprache etwas unterschätzt. Bei 50.000 Wörtern benötigt die gesamte Zahlenausgabe ca. 5 Sekunden! Mir gefällt es so. Man sieht, der Kerl schafft was und es tut sich was. Außerdem weiß man mit dem Wert der Zahl, wo die Einsortiererei sich ungefähr befindet. Bei einer Sortierzeit von zwanzig Minuten will ich das ab und zu wissen.

zu 8) Die Bibliothek ist in dieser Hinsicht jetzt korrigiert.

zu 9) Neu.Srt wird jetzt vor dem Testen eines Textes in den Servicepuffer eingeladen. Ist Neu.Srt zu groß für den freien Speicherplatz (ca. 40 Kb), wird mit einer Fehlermeldung abgebrochen. Aber Achtung: Je größer Neu.Srt ist, desto länger braucht der Check eines unbekannten Wortes, weil ja erst mal alle (!) Wörter gelesen werden müssen. Bei 40Kb können das mehr als 1000 Wörter sein.

zu A) No comment! Leute, Ihr seid gefragt!

zu B) Wie ihr im vorliegenden Text sehen könnt, habe ich mich gebessert. Selbst die Statuszeile der Bibliothek im Wortmanager ist korrigiert.

Zu allem möchte ich noch anmerken, daß Holger sich richtig Arbeit gemacht hat. Das hat zur Folge, das der Wortmanager besser geworden ist. Mit dem Problem konfrontiert, einen Artikel für das Info zu schreiben, lassen sich doch meistens die Worte nicht finden, es sei denn, man hat was zu sagen. Das ist hier bei Holgers Programmkritik in hervorragender Weise gelungen. Das gilt natürlich auch für die danach folgenden Kritiken an den Pascal-Programmen. Danke Holger!

L e s e r b r i e f: Peter Lang, 8502

Peter Lang, Banderbacher Str.24, 8502 Zirndorf

Gemischtwaren !

Von Lastern, Computerzeugs und dem, was verdammt gut ist !

Der Club ist aller Laster Anfang!

Anfang 1984 mußte ich notgedrungen meinen TI99 und meinen SHARP700 in die Ecke verbannen, Ende der Fahnenstange. Wegen großer zu bewältigender Datenmengen (Zeitschriftenvertrieb, Abonnementen-Verwaltung) waren die Kisten wohl nur noch als Spielzeug anzusehen. Also was tun? Vobis-Groß-Anzeigen für Memotech-Rechner! So ein Ding mal genauer unter die Lupe genommen. Naja! Andere artverwande Silicon-Trottel verglichen, und entschieden! Ne komplette Memotech Anlage war (damals) immer noch erheblich billiger als diverse Konkurrenz-produkte. Nach einigen Wochen Einarbeitung in Newword, (FDX-) Basic und CPM wurde doch schon bald klar, daß auch hier bald nicht mehr viel geht!
Hardware? Software? Utilities? Service? Informationen? Pustekuchen! Aber man las (liest) ja schließlich die CT, Abteilung Userclubs! Und da stand eines verregneten Tages zu lesen: MTX-Userclub, Herbert **Herzberg** (!!!!) ...
Na immerhin. Kontakt hergestellt, Kohle überwiesen, Mitglied geworden, Infos bekommen. Und in den Infos gab es außer wertvollsten Informationen auch noch Hinweise darauf, wo und bei wem es viele viele gute und günstige Sachen gibt!

Dies ist jetzt so ziemlich fünf Jahre her! Aus dem anfänglichen Gedanken, die schwarze Kiste wieder zu verscheuern wurde nichts! Im Gegenteil!
Aus meinem Blechtrottel wurde ein PC (rein optisch). Die Klappertastatur ist ersetzt durch ein Cherry Restpostenterminal (54 x 30cm) mit ergonomischen Styling (Wow!). Logisch, daß da viel von dem was gut und teuer ist eingebaut wurde. Und im letzten Jahr gabs noch einen gleichwertigen Bruder. Den hat eine Nürnberger Firma (für die ich zeitweise arbeite) geordert, weil sie von den Leistungen eben dieses Gerätes so optimal begeistert war! Hier arbeitet jetzt ein Memotech in der Vertriebsabteilung im Datenverbund mit PC's anderer Ab-teilungen, über softwaregesteuerten Umschalter mit sündhaft teueren Printern und in näherer Zukunft sogar nochmal mit ein oder zwei Geschwistern. Warum? Weil außer den Eingeweihten absolut keiner den Unterschied zu einem wirklichen PC feststellen konnte. Wie denn auch? (Eigenständige Turboprogramme ohne Ausstieg ins CPM !).
Und ich arbeite jetzt of zig Stunden die Woche an meinem "Memotech-PC". Hätte nie gedacht, daß es mal soweit kommt. Aber ehrlich! Ohne dem ganzen Club-Gerümpel (das ist liebevoll gemeint !!!) gäbe es bei mir todsicher nix mehr Memotech!
Mein Lebensablauf (Arbeit u.Privat) hat durch oben genanntes total andere Dimensionen erhalten. Faszination? Neugier? Technik-Geilheit? Was auch immer, es ist zu einem Laster geworden! Zwar nicht unbedingt der Gesundheit abträglich aber doch manchmal bedenklich.
Und wer oder was ist mit schuld, hä ..... ?

Was bootet da durch Nacht und Wind ...

es ist das Eprom vom Keßler, und zwar geschwind! Na gut, Klassiker zu vergewohltätigen ist also auch nicht meine Stärke. Apropos Stärke! Für all diejenigen, die nicht von irgendwelchen Silicon- oder ähnlichen Discs booten, war dieses neue Boot-Eprom (von Michael Kessler) wohl der absolute Traum. Da hat man vier 80-Spur Laufwerke, kann die (danke!) auch noch kinderleicht im HD-Modus betreiben (= eintausendvierhundertvierzig Kilobytes) und mußte von solch dämlichen 03er Disketten booten (dreihunderzwanzig Kilobytes).

L e s e r b r i e f: Peter Lang, 8502

Peter Lang, Banderbacher Str.24, 8502 Zirndorf

Das tat unsagbar weh! Doch mit sowas nervigen ist ja jetzt -Gott sei's gelobt getrommelt und gepfiffen- endgültig Schluß! Erstens ist jetzt nur noch eine Diskette zum booten nötig, weil da alles drauf paßt was das Herz begehrt, vorrausgesetzt man benutzt eine Diskette im "1A" Format. Hier auch mal größten Dank an Bernd Preusing für seine "BP"-Disc-Formate. Zweitens gehts verdammt flott (liegt an Bernd's flotten Formaten). Und drittens ist das Boot-Eprom auch noch komfortabel, aber das merkt man erst, wenn mal keine oder eine falsche Disc drin ist. Kurz und gut: Schnell, klein, schwarz! (Hä?) Nennt sich Boot-Eprom, ist eine Wohltat und wurde hervorragend dokumentiert. Das Preis-Leistungs- Verhältnis ist dem Club-Preisniveau angepaßt günstig.
Ein starkes Stück also!

Jedem sein eigener DiJey

Die Fülle der Klick-Programme und deren Qualität scheint sich unaufhaltsam zu steigern. Eines der spektkulärsten Produkte scheint mir (außer RAMxx selbst) "DIJEY" von Olaf Krumnow zu sein. Welch geistige Arbeit und welcher Arbeitsaufwand in diesem Projekt stecken, kann wahrscheinlich nur einer nachvollziehen, der sich selbst schon erfolglos an ähnlichen Sachen probiert hat. Hier auf Einzelheiten dieses mächtigen Werkzeugs einzugehen wäre müßig. Es ist einfach ein absolutes MUSS, mit DiJey zu arbeiten. Die Programmbesprechungen im Info bezüglich "DiJey" versprechen garantiert nicht zuviel. Auch an der Dokumentation von "DiJey" ist nichts auszusetzen. Lediglich der zum vernünftigen Arbeiten nötige Speicherplatzverlust auf der Ramdisc ist etwas schmerzlich (liegt aber in der Natur der Sache). Bei der täglichen Arbeit mit "DiJey" kommen so manche Wunschvorstellungen ans Tageslicht. Da wahrscheinlich jeder der den "DiJey" hat, auch wirklich täglich und immer wieder damit arbeitet, bin ich der Meinung daß "DiJey" jetzt nicht in den Dornröschenschlaf versinken sollte. Jeder der das Ding nutzt sollte sich Gedanken über eventuelle Verbesserungen machen. Und die auch kundtun. Ich fang hier mal an damit.
Mich nervt zuweilen, daß DiJey (fast) nicht installierbar ist. Eine (feste) Installation für Vertikal- oder Horizontal-Directory, Sortiermodus, mit/ohne Größenanzeige und/oder User sowie Logged-Drive wäre sehr sinnvoll (siehe XDIR, GENINS etc.).
Aber grundsätzlich möchte ich "DiJey" nicht mehr missen! Jeder soll's haben!

Der Star im Formatwirrwar

Michael Keßler's (der schon wieder!) FORMSTAR nennt sich wohl nicht zu Unrecht so. Dieses fast schon zu komfortable Formatierprogramm kann ich eigentlich äußerst kurz kritisieren: Spitzenklasse. Mit den über Hundert verschiedenen Formaten, die dieses FORMSTAR bearbeiten kann (ob geCONFIGt oder nicht!), der wohl nicht zu verbessernden Bedienerfreundlichkeit und der Tatsache, daß das Ganze auch noch im Klick laufen kann, wird wohl jeder Anwender Schwierigkeiten haben, da was rumzumeckern. Mein Wunsch: Disketten-Checker (RCHECK) und XSYS (das ist das dazugehörige Syscopy) gleich mit einbauen. Wäre äußerst praktisch und zeitsparend!

L e s e r b r i e f: Peter Lang, 8502

Peter Lang, Banderbacher Str.24, 8502 Zirndorf

Gemischtwaren !

Abendessen oder Zigarettenpause?

Gesundheitsfanatiker würden sich für das Erste entscheiden. Aber in Anbedracht der Existenz von COPYD (Herbert z.Nedden) wirds da bei mir schon schwierig. Folgender Fall: Nach acht Stunden Turbositzung hat man allerhand geändert, gelöscht und neu geschreibselt. Nach Programmierergesetz soll man ja mindestens zwei Sicherungskopien der Arbeitsdiskette machen (für den Fall daß Murphy wieder mal gnadenlos zuschlägt). Da mir so Sachen wie NSWEEP irgendwie nicht so recht zusagten, wurde halt mittels PIP *.* alles "hinübergePIPt". Wer mit "1A" Disc's arbeitet weiß bescheid: Zeit für's Abendessen! Lediglich zwischen zwei Portionen Spagheroni mal einen Stock höher, Diskette gewechselt und .. nochmal. Weiteressen. Doch damit ist's Schluß! Ganze Disketten werden nur noch mit COPYD kopiert. Spurweise. Und das dann auch noch so schnell, daß die Zeit gerade noch für 'ne Lülle reicht.

Mein Notebook gehört mir

Dieses NOTE oder wie es auch immer heißen mag (von Bernd Preusing) hat ja schon einige Monate am Buckel. Ich erwähne es hier, weil es eines meiner vielbenutzten Programme ist. Der NOTE-Hauptteil liegt im Klick und ist ein verdammt schnelles (Mini-) Textprogramm. Ruft man NOTE im CPM nochmal auf, erhält man ein Bedienermenü zum Laden, Sichern, Löschen und installieren. Der Nachteil: Damit der Wechsel zwischen Bedienteil und dem Teil im Klick in einem halbwegs vernünftigem Tempo abläuft, sollte NOTE auf der Ramdisc sein - und das frißt Speicherplatz. Nachdem ja mittlerweile Diskettenzugriffe aus Klick kein Problem mehr darstellen, müßte es doch möglich sein, dies zu implementieren. Warum ? Ganz einfach: Kurz mal 'nen Brief o.ä. schreiben und ausdrucken macht der moderne MTX-User nicht mehr mit Newword. Er nimmt NOTE schreibt seinen Text im Klick (es fällt einem ja immer wieder was ein), saved das Ganze ab und ruft BRADFORD zum drucken auf! Soweit es sich nicht um riesige Mengen Texte handelt ist das m.E. die eleganteste und schnellste Lösung. Also! Wär's machbar?

Spinn ich?

Nur ganz kurz: Den Patch zur Umlautgroßdarstellungsumschaltung (Auweia!) für RAM4.x aus dem letzten Info eingegeben, gestartet, ha - gleich 'ne Taste gedrückt - paßt. Nach einiger Tipparbeit Text durchgesehen: öha! Ganz schön doof! Zu dumm zum tippen! Also verbessert. <ß> gedrückt, <?> erhalten! Und umgekehrt. Man kann sich zwar daran gewöhnen, aber sein müßte es nicht!

Guter Printer für 145 Mark ?

Wer seinen gut erhaltenen DMX-Drucker verkauft, kann locker 350,- dafür bekommen. So, jetzt 145,- aus der Schatulle, dazuaddiert und schon hat man einen "Panasonic KX-P1081". Fast (= 98%) DMX-kompatibel, um einiges schneller (120 z.), mit NLQ-Umschaltung, mehr Befehlsvorrat (manches davon überflüssig) und ein hellbeiges Äußeres. Sehr nützlich ist ein Abstandhalter zwischen Papier und Farbband um zuverlässig die Schmiererreien verhindert wenn das Druckgut mal zu dick ist. Kurzurteil: Der KX-P1081 ist die 495,- Märker wert.

L e s e r b r i e f: Peter Lang, 8502

Peter Lang, Banderbacher Str.24, 8502 Zirndorf

Gemischtwaren !

Begeisterung !

Obwohl Besitzer eines NLQ-fähigen Druckers bin ich doch glatt auf die "Werbekampagne" für BRADFORD (Aaaron Contorer) hereingefallen. Doch dieses NLQ-Druckprogramm erfüllt nicht die gesetzten Erwartungen! Es übertrifft sie bei weitem. Die im Info abgedruckte Reference war nicht mehr aktuell (neue Version), das mitgelieferte englische Manual enthielt nur etwa 30% der nötigen Informationen und ich stand am Schlauch. Doch welch ein Jubel! Es war ein ins holländische übersetztes Manual anbei. Nach vielen nächtlichen Stunden hatte ich dann doch etwa die Hälfte der Möglichkeiten von BRADFORD herausklabüsert. Vollständigkeit gelang mir nicht, da Holländisch stellenweise schwerer ist als eine englische Anleitung. Da ich aber voll von BRADFORD und seinen Möglichkeiten begeistert bin, würde ich (bei Interesse) das englische Handbuch eindeutschen. Wer würde mir dazu das Originalhandbuch zur Verfügung stellen? Bezüglich der Qualität von BRADFORD soll folgende Begebenheit mein Urteil ersetzen: Bei der Sichtung von BRADFORD-Ausdrucken (Präsentation) stellte ein IBM-PC-User der in diesem Beitrag schon erwähnten Firma fest: 'Ihr müßt ja ganz schön Geld bewilligt kriegen. So ein Programm kostet doch mindesten 450 Mark!'. Falsch! Gibt's bei Herbert als Public-Domain und ist spottbillig!

Was ich noch sagen wollte ...

Es ist gar nicht so einfach über die Werke anderer zu schreiben. Erstens weil schon mal die Auswahl schwierig ist. Zweitens könnten diejenigen, über deren Arbeiten man nichts schreibt glauben, daß ihre Sachen weniger wertvoll seien. Und drittens können solche Kritiken wohl nie voll objektiv sein.
Zu erstens: Die Auswahl für diesen Beitrag ist willkürlich. Mit eine Rolle gespielt hat die Häufigkeit der Nutzung und wann die Dinger erschienen sind.
Zu zweitens: Hier sollte sich ja wohl kaum einer Sorgen machen müßen! Ich habe über den Club noch nie ein schlechtes Programm erhalten. Wenn also nix darüber geschrieben wird dann liegts entweder an Schreibfaulheit oder aber das Ding ist so gut daß es nichts zu schreiben gibt. Siehe Olafs Artikel aus einem der letzten Infos (Bin ich der ideale Programmierer?). Herr Krumnow, was will man denn an ihren Dingern recht viel kritisieren? Das Zeugs ist gut! Punktum! Alles Andere wäre m.E. Haarspalterei! Was müßte sich da Bernd Preusing Gewissensfragen stellen! Der hat uns ja mit RAMx.x die ganze Chose erst möglich gemacht. Recht viele Lobeshymnen über diese excellente Software habe ich nicht gelesen. Sicher freut sich jeder der 'Macher' über Lob. Aber bei so vielen Könnern wäre dann ja wohl das halbe Info eine Kritik. Seit euch allesamt mal sicher: Wenn einer mit euren Sachen richtig unzufrieden ist, dann wird garantiert gemault und kritisiert. - Danke für die geneigten Ohren (Augen)!

TEAC hat ein Rad ab !

Kurzbeschreibung: Vor einem Jahr zwei TEAC GFR eingebaut. Alles wie Info und TEAC-Manual. Im Januar '89 und Februar '89 jeweils noch ein GFR zugelegt. Wegen Zeitmangel wurden die Drives noch nicht anstelle der Epson bzw. FV eingebaut. Doch jetzt wars soweit. Aber ...: Die beiden neueren GFR waren zum Großteil unterschiedlich zu den Älteren! Der gesamte Inhalt sieht wie das Innenleben eines 3,5" aus. Die Jumper-Felder und deren Bezeichnungen entsprechen nicht mehr dem Manual bzw. Info. Gespart wurde auch: Das Februar-GFR hat keine Headload-Spule mehr! Ist schon Sch.... was die mit einem machen.

B r a d f o r d: Kommandos

BRADFORD
Kommandos

=== RUNTIME-OPTIONEN ===

Diese beginnen mit "/" und werden Bradford beim Start eingegeben!

/Axxx	Anfang, erste zu druckende Seite ist "xxx"
/Cxxx	Copies, Schriftstueck "xxx" mal drucken
/D	Draft, schneller Probedruck (halbe Dotanzahl)
/P	Pause zwischen Seiten
/Zxxx	Ende, letzte zu druckende Seite ist "xxx"

=== KOMMANDO'S ===

Beim Start oder im Text (auf Spalte 1 !) beginnen diese mit "."
Im Text auf der Zeile verteilt beginnen diese mit "\"

..	Kommentar, diese Zeile wird nicht gedruckt
AIxx	Spaltenabstand "xx" Dots (0 .. 99), Normal = 3
BMxx	Bottom Margin, unterer Rand, "xx" freie Zeilen unten
CNxxx	Copy Number, Seitenzahl auf "xxx" setzen, "xxx" = "+" addiert
CPxxx	Conditional Page, sieht nach ob noch "xxx" Zeilen vorhanden
DCCy	Define Commando-Character, normal: "\", auf Char "y" setzen
DCLy	Define Leading-Character, normal: ".", auf Char "y" setzen
DKx	Darkness (1 .. 4), Druckstaerke, Zeile "x" mal drucken
DM	Display Message, Nachricht auf Bildschirm
FNFONT	Font Name, Zeichensatzwahl (max. 5 je Zeile)
FFFONT	Font For, Zeichensatz fuer Kopf- u. Fusszeile
FO...	Footer, Text u. Kommandos "..." fuer Fusszeile
FLxx	Footer Line, Zeilennummer "xx" der Fusszeile
HE...	Header, Text u. Kommandos "..." fuer Kopfzeile
HLxx	Header Line, Zeilennummer "xx" der Kopfzeile
HMx	Horiz. Magnify, horizontale Verbreiterung "x" (1 .. 9, norm = 2)
LMxx	Left Margin, linker Rand "xx" Spalten (0 .. 79)
LXxxx	Left exact, li. Rand genau in "xxx"/240 inch (0 .. Papierbreite)

B r a d f o r d: Kommandos

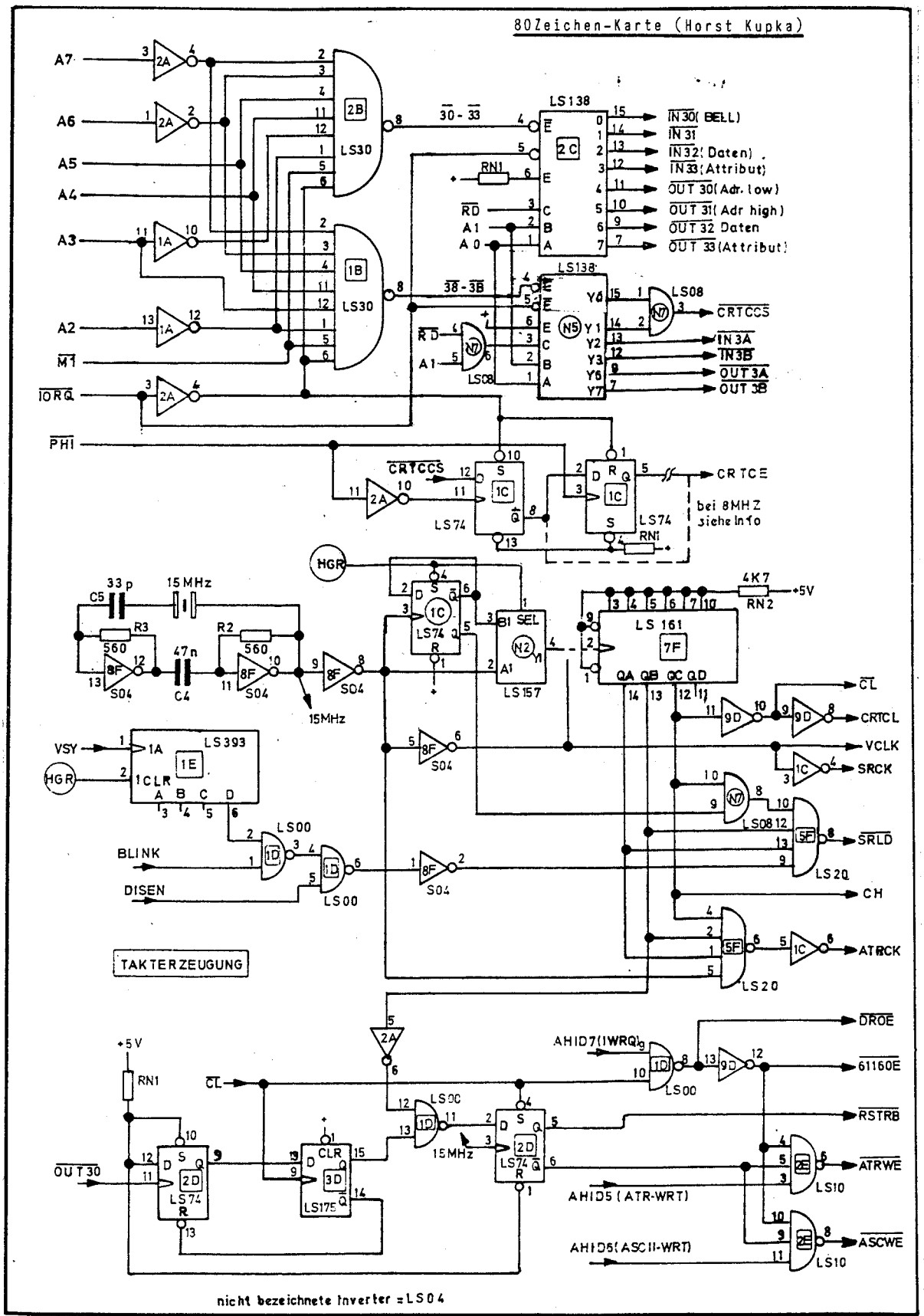
MCyr	Macro Character, Macro "y" erhaelt Wert "r" (Char)
MDy ...	Macro Definition, Macro mit Bezeichnung "y" (A .. Z) auf String "..." definieren
MNyxx	Macro Numberswitch, fuehrt Macro "y" aus, wenn Zahl "xx"
MOx	Monospace, Proportionalschrift "x" = 1: AUS, "x" = 0: AN
NL	New Line, Neue Zeile, wirkt wie <RETURN>
OV	Overprint, drucke naechste Zeile ueber diese Zeile
PA	Page Break, Seitenumbruch, neue Seite beginnen
PLxx	Page Length, Seitenlaenge in Zeilen "xx" (1 .. 254, norm.: 72)
PNxx	Set Page Number, Setze aktuelle Seitennummer auf "xx"
POxx	Page Offset, Linker Randsteller auf "xx" (0 .. 79)
PR	Print, drucke Kommando-Character, z.B.: PR \, PR .
RJ	Reset Justify, Zentrierten oder Blocksatz AUSschalten
RMxx	Right Margin, rechter Rand "xx" (0 .. 79)
RXxxx	Right exact, re. Rand genau in "xxx"/240 inch (0 .. Papierbreite)
SNxx	Section Number, auf "xx"
SPy	Spacing, Zeilenabstand, "y"=Q: 3/4, "y"=H: 1 1/2, "y"=2: doppelt
STyx	Set Text, "y" = L: Linksbuendig, "y" = R: Rechtsbuendig "y" = C: zentriert, "y" = J: Blocksatz "y" = H: Halb-Blocksatz, "y" = A: Blocksatz, zentr. "x" = 1: nur eine Zeile
SUB	Subscript, tiefgestellt drucken
SUP	Superscript, hochgestellt drucken
SU0	(SU-null!) wieder normal drucken (SUB u. SUP aus)
TAx	Tab, Tabulatorabstand = "x"
TMxx	Top Margin, oberer Rand, "xx" freie Zeilen oben
ULx	Underlining, Unterstreichen, x=0: aus, x=1: einfach, x=2: doppelt
WK	Wait Keypress, wartet auf Tastendruck
XMy	Execute Macro, fuehre Macro "y" aus
XX	Inline-Kommando, diese Zeile nicht drucken

Die Kommandos koennen in Gross- oder Kleinbuchstaben eingegeben werden. Runtime-Optionen und "."-Kommandos koennen bei Bradford-Start gemischt werden. z.B: BRADFORD Textfile /C2 /Z15 .ML10 .FNpizza
Druckt die Datei "Textfile" mit 2 Kopien, bis Seite 15, mit 10/10 inch rechten Rand und nimmt Zeichensatz "pizza".

H a r d w a r e: 80Zeichen-Karte

Hochauflösende Grafik mit der 80Zeichen-Karte (Herbert zur Nedden, 2000)

Auf der PD CLUB.040 ist die Umbaubeschreibung für Horst Kupka's 80Z-HiRes-Grafik. Hier zumindest in kleiner Form die Schaltpläne, damit Ihr über Ostern schon mal basteln könnt. Im nächsten Info kommt das alles ausführlicher/besser. Hier paßt es nicht mehr rein, da das Info zu dick würde und eh schon in Druck ist (daher auch als Seitennummer einfach 'Anhang'). So jetzt schnell zu Druckerei!



H a r d w a r e: 80Zeichen-Karte