MTX User-Club Deutschland

Info 42
12.05.1991

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.- 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:

Herbert zur Nedden
Alte Landstraße 21
2071 Siek
(04107) 99 00

Hans Gras
Statenhoek 49
NL 1506 VM Zaandam
(0031-75) 17 49 91

Telefon-Sprechzeiten
Herbert zur Nedden: Do 18 - 21 Uhr, Sa 9 - 14 Uhr
(Etwas klingeln lassen oder nochmal versuchen!)

I n h a l t s v e r z e i c h n i s

Preis für dieses Info: DM 8,80

Clubtreffen (Herbert zur Nedden, 2071)

Das Clubtreffen findet am Sa/So, den 29/30. Juni statt! Der Ort des Treffens ist das Hotel Kückenmühle, 3003 Ronneberg. Eine Übernachtung incl. Frühstück kostet DM 55.- pro Person im Doppelzimmer. Wenn Du kommen möchtest laß es mich bitte wissen. Solltest Du sogar übernachten wollen, dann solltest Du **so schnell es irgend geht** bei mir **definitiv** zusagen! Sonst kann es passieren, daß Du keinen Platz zum Schlafen bekommst!

Bitte (Hartmut Traber, 5270)

Die Flut von PD-, -Club- und Klick-Programmen ist für Einen kaum noch zu sichten. Bitte rafft Euch auf, nach Interessengebiet prägnante Programmbeschreibungen zu veröffentlichen!

Kontostand (Herbert zur Nedden, 2071)

Eine rote Markierung auf dem Umschlag bedeutet, daß er zu niedrig ist.

Anzeigetexte samt Absender bitte schriftlich an Herbert zur Nedden!

V E R K A U F (Preise sind i.a. ohne Porto & Verpackung)

Herbert zur Nedden, Alte Landstr. 21, 2071 Siek, 04107 - 9900:
--> Interessiert Dich einer der von mir angebotenen Posten: Mach ein Angebot!
Meine hier genannten Preise sind nicht unbedingt unumstößlich!
(Neue/geänderte Posten haben einen * statt des - vorne weg)

- Evtl. Grünmonitor: DM 100.-
- Rikadenki Plotter RY21, VB DM 1500,-
  Flachbettplotter, DIN A4, Aufnahme für Rotringstifte, incl. Handbuch und Schaltplan, 8085 CPU (Z80-aufwärtskompatibel), Centronics-Schnittstelle. Positioniergenauigkeit: 0,1 mm, Zeichengeschwindigkeit 200 mm/s, 19 Kommandos wie u.a. Kreise, Kreisbögen, Rechtecke, versicheiden gestrichelte Linien, Textausgabe in 4 Richtungen und verschiedenen Größen, absolute und relative Koordinaten, Markierungen auf Linien und Graphen. Kann in Textmodus gesetzt werden, um als Drucker zu arbeiten.
- Ich habe RAMs für 768kB-Erweiterung!

- dBASE Version 2.41: DM 100.-
- NewWord Version 2.02 DEUTSCH, im Tausch gegen Original-NW-Diskette: DM 40.-
- Original Microsoft-BASIC-Lizenz incl., M80/L80 Assembler/Linker, Handbuch, auf dem MTX lauffähig: DM 180.- (Microsoft liefert dieses Teil nicht mehr aus!)
- NonDoc-Editor für CP/M incl. Source, zeilen- und bildschirmorientiert, Macro-Fähig, mit Handbuch im Ringordner: VB DM 40.-
- Z80ASM (Club-PD Z80-Ass.): Listing und Handbuch, ca. 1.4 cm DIN A4: VB DM 5.-

- Infos 11-36, gebraucht VB DM 60.- frei Haus
- Bücher
  - Programmierung des Z80, Rodnay Zaks, 606 Seiten: DM 45.-
  - Mikroprozessor Interface-Techniken, A. Lesea/R. Zaks, 425 Seiten: DM 30.-
  - Operationsverstärker Anwendung, 164 Seiten, DM 10.-
  - ECA-Tabelle ttl-IC's , endet im Bereich der 74xx400-er: DM 20.-
  - ECA-Tabelle dat 1: Transistoren A..BUY: DM 5.-
  - ECA-Tabelle tht: Thyristoren, Triacs, ...: DM 5.-
  * Turbo Pascal, 3. Auflage von Herschel, Oldenbourg-Verlag: DM 15.-

- Einbau-Drehspulmeßgerät 0-50uA: DM 10,-
- Solange der Vorrat reicht:
  - MTX-Tasten je DM 1.-, Tastenkappen je DM -.50
  - EPROMS 2564 für je DM 15.-
  - Dynamische RAMs 4116 (VRAMs) 8 Stück: DM 25.-
  - Dynamische RAMs 3732 (32k x 1Bit) 8 Stück: DM 1.50
  - Statische RAM's 2k x 8 Bit (6116): je DM 2.-
  - TTL-IC's: 74LS175, 74LS368, 74LS173, 74LS158, 74LS258 je DM 0.50;
    74LS10, 74LS11, 74LS21 je DM 0.30
  - Original-Memotech-Spielecassetten: Toado, Kilopede, Knuckles, Draughts, Reversi, Snappo, Blobbo, Utilities, Demo, StarCommand je DM 4.-;
  - 10 Disketten FUJI HD 5 1/4", gebraucht & o.k. DM 35.-, Originalverpackt: DM 60.-

Liebe Leserin, lieber Leser,

Zusammen mit meiner Frau mabe ich unser neues Domizil nun endlich bezogen; der Innenausbau ist zwar immer noch nicht fertig, aber es ist schon toll ländlicher zu wohnen. Ich hoffe dennoch gelegentlich das eine oder andere Programm für unsere schwarze Kiste fertigstellen zu können! Wenn Du mich anrufen willst, laß es bitte etwas länger klingeln und versuche es ggf. etwas später nochmal; es der Weg zum Telefon ist im Haus länger als in unserer alten 48 m²-Wohnung.

Seit einiger Zeit tausche ich die Clubzeitschrift mit zwei anderen CP/M-orientierten Clubs aus: der Schneider/Armstrad User Group und dem SVI/MSX-Club Deutschland. Einige Infos zu den Clubs und evtl. weiteren folgt im nächsten Info; noch liegen zu viele meiner Unterlagen in unausgepackten Bananenkartons.

Ich würde mich freuen, wenn Du zu unserem Clubtreffen Ende Juni (genaueres s.o.) kommen könntest; mir liegen für das Treffen zwei Themenbereiche: neues um unseren Rechner und der Austausch mit anderen Clubs. Immerhin ist ZCPR 3.3 gewissermaßen unter CP/M zum Standard geworden, so daß viele Programme auf allen CP/M-Rechnern vernünftig laufen.

Euer

Michael Nickel

Aprilscherz ? (Herbert Oppmann, 8520)

Als ich das Info 41 in Händen hielt, und den Einsendeschluss für Anmeldungen zum Clubtreffen las (18.03.91), dachte ich zuerst, das sei ein Aprilscherz von HzN. Leider mußte ich dann feststellen, daß das ein Aprilscherz der Deutschen Bundespost war: abgestempelt in Hamburg am 08.03.91, angekommen in meinem Briefkasten am 04.04.91! Das macht eine Durchschnittsgeschwindigkeit von ungefähr 1 km/h. Man sollte wieder Staffelläufer einführen, wie es sie zur Römerzeit gab, das ging damals schneller. Und überleg mal, was das an Arbeitsplätzen schafft!

Silikon-Trottel ? (Hartmut Traber, 5270)

Im Info erscheint ja nicht mehr viel an Humor, mir scheint, die Diskussion darüber hat HzN etwas frustriert. Andererseits sorgen viele Club-Mitglieder dennoch, sicher aber unfreiwillig dafür, daß ich mich bei der Lektüre jeden Info's und fast jeder PD-, Club- und Klick-Diskette amüsieren kann! (Ich bin schon lange aus der Schule: Wer analysiert die Zeichensetzung in vorigem Satz?)

<u>Wie das?</u>
Für mich ist Silikon etwas, was sich manche Frauen hinter den Brustwarzen verstecken ließen! Silizium, englisch "silicon", neudeutsch: "Silikon" ist ein Element und verhilft <u>uns die Zeit zu vertreiben, als Si oder als "Silikon".</u> Die gedanklichen Assoziationen, die ich mit dem Begriff Silikon verbinde, sind erstens Aussprizen von Löchern und zweitens das Obige! Aber "Trottel"?
Bin ich unnormal oder ist das schwarzer Humor.....?
Ich schlage den neuen (richtigen?) Begriff: Silizium-Trottel vor! *Lacht nicht!*

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

Korrektur & Nachtrag

Olaf Krumnow: (Herbert zur Nedden, 2071)
Seine Anschrift lautet richtig August-Bebel-Str. 102c, 2050 Hamburg 80.

RAMLIB.LIB: (Herbert zur Nedden, 2071)
Ich bin anscheinend nicht in der Lage das richtige an die richtige Stelle zu kopieren. Auf KLICK.016 ist mein nächster Versuch; vielen Dank Jan Bredereke.

NewCat6: (Jan Bredereke, 2000)
Mein NewCat6 auf KLICK.012 sieht Dateien mit der Extension " " als nicht vorhanden an. Das ist mit der Version von KLICK.016 behoben.

Info 40-6: (Jan Bredereke, 2000)
Zu Info 40-6, über ECPs: XRUN, ALIAS.CMD und CHAINDU müsssen nicht unbedingt im Wurzelverzeichnis stehen. Meine wichtigsten Programme sind auf den SRAM-Disks I: und E:, die aber beide zur Sicherheit R/O sind. Damit ich Shell-Variablen benutzen kann, muß aber das Wurzelverzeichnis R/W sein, so daß ich dafür die RAM-Disk G15: benutze. Fast alle Programme lassen sich so patchen, daß sie ihre Dateien auch auf anderen Laufwerken finden. Z.B. zeigt ein Debugger am Anfang von ARUNZ/CMDRUN diverse Patch-Stellen im Klartext, u.a. "PATH", "ROOT", "SCANCUR" und "DU". Setzt man das Byte hinter den ersten drei Texten jeweils auf 0(=false) und die beiden Bytes hinter "DU" auf den gewünschten Wert (A:=0, I:=8), so sucht ARUNZ dort nach ALIAS.CMD. Wenn man am Ende von ALIAS.CMD vor XRUN eine Laufwerksangabe setzt, wird sie beachtet. Das gleiche gilt im Patchbereich von XRUN für eine Laufwerksangabe für CHAINDU. Die einzigen Dateien, die ich wirklich noch in das Wurzelverzeichnis kopieren muß, sind CMDRUN.COM und IF.COM.

Fragen & Antworten

A: (Jan Bredereke, 2000)
Zur Frage im Info 40-3 von Anton Reiser nach PostScript auf CP/M: Der fehlende Bildschirm-Grafik-Standard dürfte nicht das Problem sein; außer bei Display-Postscript ist das Ausgabemedium immer ein Drucker, und die sind ohnehin nicht genormt. Das Problem dürfte der Speicherplatz und damit verbunden auch die Rechenzeit sein: Postscript ist eine Seitenbeschreibungssprache, so daß erst alle Beschreibungen einer Seite (z.B. Rahmen um die ganze Seite) ausgewertet sein müssen, bevor die Seite gedruckt werden kann. Bei einem Rasterausgabegerät wie dem Nadel- oder Laserdrucker bedeutet dies, daß die Bitmap einer ganzen Seite gleichzeitig im Speicher gehalten werden muß. Je nach Auflösung ist man da schnell im Megabyte-Bereich. Und die Bearbeitung so großer Datenmengen kostet Zeit, so daß die doch relativ langsame Z80-CPU Probleme bekommen wird.

F: (Wolfgang Dexheimer, 6719)
Nach Definition eines Fensters mit 'Esc Ä W AnfSpalte, AnfZeile, usw.' ist es anschließend mit 'Esc Ä Z usw.' nicht möglich, ein Zeichen in der Spalte AnfSpalte (die Zeile ist dabei beliebig außerhalb des Fensters) auszugeben, es wird ignoriert!
A: (Herbert zur Nedden, 2000)
Stimmt!

F: (Hans Gras, NL-1506)
Hat schon jemand ein Boot-EPROM gebastelt, welches von einer HardDisc bootet?
Anm.d.HzN: Boot-EPROM- und HardDisc-Treiber Sources gibt's bei mir.

F: (Hans Gras, NL-1506)
Wie können die Diskettenlaufwerksnachlaufzeiten verlängert werden?

S o f t w a r e: Einige Anmerkungen/Neuerungen

K 2 D o s: Mini-BDOS und großes System (Herbert zur Nedden, 2071)

Mittlerweile gibt's K2Dos Version 2.2, welche alle mir bekannten Fehler der Vorversionen nicht mehr hat. An dieser Stellen vielen Dank an Jan Bredereke, der einige Macken entdeckte. Das K2Dos-BDOS belegt 256 Bytes Platz (weniger ist nicht möglich, da es auf einer Seitengrenze beginnen muß) auf Bank 0; der Rest ist auf Bank ? verbannt ... steht halt im KLIX-Heap. Der CCP (unser ZCPR 3.3) steht als Datei ZCPR.CCP auf Diskette und kann so groß sein, wie er will.

In RAM 6.3 soll K2Dos dann integriert sein und obendrein noch weiter oben im Speicher stehen. Da der ZCPR nicht mehr auf 2kB in seiner Größe beschränkt ist, ist der RCP überflüssig; in den so frei werdenen Platz kommen dann andere Systembestandteile (Environment, Named Dirs, MCL usw.). Damit kann das BIOS nach EF00h und das BDOS nach EE00h. Höher geht nicht, da es zum einen eh schon oberhalb von F000h verdammt eng ist, u.a. wegen vielen Einsprungadressen und Variablen. Außerdem stehen bei F080h die Interrupt-Vektoren, die im BIOS stehen müssen, damit sie nicht überschrieben werden (schließlich dürfen Programme das BDOS überschreiben, wenn sie es nicht brauchen); und der Tastatur-Abfrage-Einsprung bei F016h läßt den BIOS-Jumptable ab F000h nicht zu.

LoadDir auf KLICK.014 (Jan Bredereke, 2000)

Die Alias-Betriebsart von LoadDir auf Klick.014 funtioniert immer noch nicht: Läßt man sich Dateien mit allen möglichen Extensions anzeigen und versucht dann eine Datei davon zu starten, so erzeugt LoadDir den Kommando-Alias "???". ARUNZ könnte ihn auch verarbeiten, aber ZCPR sagt leider schon vorher, daß dies kein wohlgeformter Dateiname sein kann, und ruft ARUNZ gar nicht erst auf.

Übrigens, wenn ich schon von Extensions rede, möchte ich noch eine Begriffsunterscheidung loswerden, weil es manchmal z.B. im Info etwas durcheinandergeht: Eine Datei-Extension ist der Teil des Dateinamens hinter dem Punkt, auch Dateityp genannt. Ein Datei-Extent ist ein 16 KByte großes Teilstück einer Datei. Die Aufteilung entsteht durch die Art der Verwaltung des Inhaltsverzeichnisses.

MTX-Edit (Jan Bredereke, 2000)

Auf Klick.014 äußert Claudio Kritik an einer Eigenschaft von Mtx-Edit, der ich mich anschließe: Mein Standard-Bildschirmformat ist 80*28, und wenn ich Mtx-Edit aufrufe, übernimmt es dieses Format. Nach ^O0 arbeitet es aber mit 80*25 weiter, ohne den Schirm umzustellen. Ergebnis: Bildschirmschrott in den untersten drei Zeilen!

Anm.d.HzN: Auf KLICK.016 ist der Source! Viel Spaß beim Ändern/Korrigieren/Erweitern; ich komme nicht dazu!

COPYD(B) (Jan Bredereke, 2000)

Und noch eine Idee: Kann man COPYD(B) mit FORM6 kombinieren? Formatieren und Kopieren in einem Rutsch würde wohl insbesondere beim Erstellen von PD-Disks viel Zeit sparen.

Anm.d.HzN: Im Prinzip: JA. Ich habe auch schon daran gedacht, aber es dann doch immer wieder verworfen, da es nicht ganz so einfach ist.

d B a s e: Fensterroutinen im KLIX

MS - Macke (Jan Bredereke, 2000)

In **MS.CHN** habe ich noch einen Fehler gefunden: In CPM-Dateinamen wird das 7.Bit nach dem Lesen nicht rückgesetzt (auch wenn es nicht angezeigt wird)! War z.B. das A-Bit gesetzt, so ist die Datei nach dem Kopieren auf eine MSDos-Scheibe dort erstens nur noch durch ein Wildcard zu erreichen (z.B. zum Löschen oder Zurücklesen) und zweitens ist sie dann verschieden von einer Datei desselben Namens ohne A-Bit.
Anm.d.HzN: Wird demnächst behoben.

BradFord-Zusatz (Jan Bredereke, 2000)

BradBold macht aus einem Bradford-Zeichensatz einen fetten Zeichensatz. Dieser ist besser erkennbar als das mehrfache Überdrucken durch "#dk x", und außerdem geht das Drucken auch schneller. BRADBOLD // gibt Hilfe.

DMX80 besser nutzen? (Jan Bredereke, 2000)

NwFilt61 ist ein Update zu NwFilt60 von mir. Der Filter für den DMX80 korrigiert jetzt auch einen Fehler beim Zeilenvorschub des DMX80. Dieser macht sich gerade unter Bradford bemerkbar: Nach einer Seite Drucken ist das Papier sonst nicht genau um eine Seite vorgeschoben.

Neue PD's (Herbert zur Nedden, 2071)

KLICK.015 mit ASCII 6.1 (Anzeige beider Zeichensatz-ROMs von der 80Z-Karte), XRUN 2.5, Edicta-Treiber 2.1, Spiel ISHIDO für Edicta-Karte (incl. Source, kann also an andere Karten angepaßt werden), ...
und
KLICK.016 mit K2Dos 2.2, MTX-Edit incl. Source, Fensterroutinen für dBase (siehe den Artikel von Wolfgang Dexheimer weiter unten), NwFilt61, ...
und
CLUB.057 mit Z80CMT (siehe unten), BradBold, Turbo-Pascal-Tips (siehe Holger Göbels Artikel weiter unten), Transfer (Peter Kretzschmars Apple-MTX-Kommunikation incl. Source) u.v.m.
sind fertig. Wenn Du ein PD-Abo hast, kommen die Scheiben mit dem Info.

dBase fensterlt (Wolfgang Dexheimer, 6719)

Seit einiger Zeit programmiere ich unter DRDOS mit dBase IV. Dabei gefiel mir das dort vorhandene Menüsystem sehr gut. Da unsere 'schwarze Kiste' dank Olaf Krumnow und Herbert zur Nedden in RAM 6.0 alles bereitstellt, was zu einem solchen System notwendig ist, wollte ich etwas ähnlich für dBase II realisieren. Zunächst erstellte ich mit Z80ASM einige .HEX Files die mittels LOAD xx, SET CALL TO 49152 und CALL xx zum gewünschten Ergebnis führten; aber das ständige nachladen ...

Deshalb versuchte ich mein Glück in meinem ersten größeren Programm für den KLIX (-> WinMen.MAC) und definierte mir eine Schnittstelle zu dBase (-> Aufruf.Z80).

d B a s e: Fensterroutinen im KLIX

Damit ist es nun möglich einer dBase Variablen den Namen des KLIX-Programmes (hier WinMen), einen Programm-Offset (das Programm WinMen enthält unter
Offset 0: Fenster öffnen
" 3: Fenster schliessen
" 6: gehe absolut und gib Text aus (siehe vorher beschr. Problem)
" 9: Zeilenmenü
" 12: Spaltenmenü
" 15: lösche Bildschirmbereich
" 18: Inkey-Funktion)
sowie die benötigten Werte (Zeilen, Spalten, Text etc.) zu übergeben und so das von dBase IV gewohnte ziemlich gut erhalten. Ich könnte nun die so entstandene Erweiterung auf vielen Seiten Wortreich erklären, das möchte ich mir und Dir ersparen, deshalb habe ich die Source-Codes von WinMen (WinMen.MAC), Aufruf (Aufruf.Z80), und die dazuhörenden Programme WinMen.KLX und Aufruf.HEX auf der Diskette (**Anm.d.HzN:** --> KLICK.016), sowie ein dBase II DemoProgramm AUFRUF.CMD das alle Möglichkeiten aufzeigt. Zunächst aber muß mit LOAD60 WinMen geladen werden, und in dBase muß ab Adresse 49152 AUFRUF bereitstehen.
(dBase-Befehle: LOAD Aufruf<RET>
SET CALL TO 49152<RET>
in AUFRUF.CMD enthalten).
Da das DemoProgramm nur eine einfallslose Hintereinanderreihung der Möglichkeiten ist, habe ich noch ein Programm (USHaupt.CMD), welches den sinnvolen Einsatz demonstriert, beigefügt; dazu werden folgende Programme im aktuellen Laufwerk (der Schnelligkeit wegen besser im RAM-Laufwerk) benötigt:

AUFRUF.HEX - Schnittstelle zu WinMen
dBase - dBase II
USHAUPT - Hauptprogramm (Aufruf: dBase USHAUPT), Auswertung von HMenü
USINIT - SET-Vorgaben, Aufruf laden, sowie die benötigten Variablen initialisieren
USPROC - die Logos und Menüdefinitionen
USUEBERW - die UMenü-Auswertung für die Überweisungen
UUAUSW - Menügesteuerte Auswahl eines Namens/BLZ/Kontos bei einer Überweisung
UUVERAEN - Eingaben für Überweisung machen, ggf. neuen Namen übernehmen
UUDRUCK - Überweisung ausdrucken (Formular 1 ohne Betragswiederholung und ohne Einrückung bei dewr Kontonummer, Formular 2 mit beidem; am besten mit Schmierpapier ausprobieren und ggf. an eigene Bedürfnisse anpassen!)
UUSPEICH - falls Empfänger/Auftraggeber nicht in Datei ggf. übernehmen und wenn gewünscht zukünftig als Vorschlag einsetzen
USSCHCK1 - Bar-Scheck1 eingeben und ausdrucken
USSCHCK2 - Euro-Scheck eingeben und ausdrucken
USSCHCK3 - Bar-Scheck2 eingeben und ausdrucken
ZAHLWORT - wandelt Betrag in Buchstaben um (Bsp: 110 wird zu einhundertzehn)

Wie vielleicht aus dieser Aufstellung hervorgeht ist es mit diesem Programm möglich 3 verschiedene Arten von Schecks sowie 2 Arten von Überweisungen auszufüllen und Namen / Konten / Bankverbindungen zu speichern und daraus auszuwählen. Bei Empf.-Überw. wird aus der Kontendatei (USKONTEN.DBF sowie den zug. Indexdateien USNAME.NDX, USBLZ.NDX und USKTONR.NDX) als Auftraggeber der bei UUSPEICH voreingestellte Wert eingetragen, bei Auft.-Überweisung entsprechend der Empfänger vorgeschlagen (beide können, müssen aber nicht identisch sein).
Dieses Programm ist Teil eines Buchführungssystems, welches ich sobald es fertig ist, auch im Club vorstellen werde. Falls man mit obigen Programmen Deiner Meinung nach was anfangen kann, so sind sie hiermit für die PD freigegeben.

Bis demnächst, Grüße aus dem Südwesten
und laßt die schwarze Kiste nicht sterben

A s s e m b l e r: Dokumentation im Source? / S o f t w a r e: .FOR

Z80Cmt - Doku im Source (Herbert zur Nedden, 2071)

Z80Cmt sucht alle Kommentare, die mit ;! beginnen aus einer Z80-Source-Datei samt deren INCLUDE-Dateien und schreibt das Resultat in eine neue Datei. Wenn ich alle Einsprünge (oder was auch immer sonst) mit derartigen Kommentaren im Source dokumentiere, kann ich mir mittels **Z80Cmt** diese Informationen aus dem Source extrahieren.

Entstanden ist diese Idee, da ich mich an ein größeres Z80-Programm zum Zwecke der Erweiterung desselben machen wollte und erst mal herausklamüsern mußte, was welche Routine eigentlich macht. Derartige Routine-Arbeiten sind mir jedoch ein Greuel, so daß ich entschied, daß das der Rechner zu machen hat.

Allerdings mußte ich natürlich erst mal die interessanten Kommentare entsprechend in ;!-Kommentare umsetzen. Der Einsprung in eine Routine sieht dann z.B. so im Source aus (das *kursive* hier nur zur Erläuterung):

```
AddHLA:   ;! Addiere A zu HL          <-- Zweck
          ;! I: HL = 1. Wert          <-- I: = Input-Register
          ;!    A  = 2. Wert
          ;! O: HL = Summe           <-- O: = Output-Register
          ;! V: Flags                 <-- V: = Veränderte Register
```

Fehlt eine der zu untersuchenden Dateien, bricht **Z80Cmt** ab. Müssen mehr als 256 Dateien analysiert werden, stürzt **Z80Cmt** ab - hierfür eine Abfrage einzubauen hatte ich keine Lust, denn dieser Wert erscheint mir eh unrealistisch hoch.

```
Die Extensions sind: Haupt-Source-Datei   .Z80
                     INCLUDE-Dateien      .LIB, falls nichts angegeben
                     Ausgabe-Datei        .CMT
```

Willst Du diese Einschränkungen ändern, tue es! Der Source ist anbei.

--> CLUB.057

What .FOR (Herbert zur Nedden, 2071)

Ich habe eine große Bitte an all diejenigen, die Software erstellen und im MTX User-Club Deutschland weitergeben, sei es nun gegen Bares oder als Public-Domain. Bei der Masse an Software aus den U.S.A. war es so, daß das, was zusammengehörte in einer Library steckte und dort auch eine Kurzinfo-Datei drin war. Diese Datei hat als Extension .FOR (von What for = wofür). Ich finde das mit der .FOR-Datei eine sehr praktische Idee, da ich dann immer weiß, wo ich nachschauen kann. Daher folgende Bitte: Wenn Du Software weitergibst, pack eine .FOR-Datei dazu, in der der Programmname, die Versionsnummer, ggf. Dein Copyright und ein 1-4-Zeiler mit einer Kurzinfo steht. Das kann z.B. folgendermaßen aussehen:

```
Z80Cmt  1.0   (C) Herbert zur Nedden
Holt ;!-Kommentare aus Z80-Sources und deren INCLUDEs
Zweck: Dokumentation im Source ermöglichen; mit Z80Cmt kann sie dort
rausgeholt werden.
```

S o f t w a r e: Einige Ideen

UISGE (Herbert Oppmann, 8520)

Ich hab mir das mal angeschaut und muß sagen: ich finde die Bewertungsfunktion gar nicht schlecht. Anstatt diese Funktion aufzublasen und damit das Spiel noch langsamer zu machen, würde ich als Verbesserung des Spiels vorschlagen, die Implementation so zu beschleunigen, daß man in höheren Spielstufen spielen kann, ohne zwischen den Zügen die Tante in Australien besuchen zu können. Ich habe mich mal hingesetzt und einige 'Handbremsen' ausgebaut, und kann jetzt in Stufe 3 so schnell spielen wie in Stufe 2.
Und in Stufe 3 habe ich erst ein einziges Mal gewonnen (ähem).

Von mir realisierte Ideen: (**Anm.d.HzN:** --> CLUB.057)
- die Funktion StellungOK wird nur von PruefeZug aufgerufen
  + StellungOK legt eine Kopie des Spielfeldes an, da es das Spielfeld verändert indem es die Steine markiert. PruefeZug hat aber bereits eine Kopie angelegt, und StellungOK darf diese Kopie müllen, muß also keine eigene Kopie anlegen
  + da der zuletzt ausgeführte Zug ja in PruefeZug bekannt ist, ist auch die Position eines Steines bekannt (die Zielposition des Zuges) und muss deshalb nicht in StellungOK jedesmal gesucht werden
- die Funktion Bewertung schaut nach, ob einer der Spieler zugunfähig ist. Dazu ruft sie jeweils ZugListe auf. Die Funktion ZugListe ist für diesen Zweck aber eine Nummer zu aufwendig, denn mich interessiert ja gar nicht die ganze Liste, sondern nur ob es noch einen zulässigen Zug gibt. Dazu habe ich eine Version von ZugListe namens ZugMoeglich erstellt, die nach dem ersten gefundenen zulässigen Zug aufhört und TRUE liefert, bei erfolgloser Suche FALSE

Weitergehende Ideen:
- es ist nicht notwendig, jedesmal zwecks Bewertung die Damen der Spieler zu zählen. Ganz am Anfang haben beide Spieler keine Damen, und die Anzahl der Damen verändert sich nur beim Ziehen (Springen) eines Steines über einen anderen hinweg. Zwei globale Variablen für die jeweilige Damenzahl, und die Funktion Ziehe etwas erweitert, und schon fällt die Erbsen- äh Damenzählerei weg.
- ZugListe muss jedesmal über das ganze Feld gehen und die Steine des Spielers suchen, der dran ist. Ein ARRAY von Stein-Positionen, am Anfang einmal initialisiert und dann von Ziehe jeweils aktualisiert, würde die Suche überflüssig machen. Praktischerweise sollten in diesem ARRAY keine x-/y-Koordinaten, sondern Zeiger in das Spielfeld stehen.
- zur Zeit legt PruefeZug jedesmal eine Kopie des Spielfeldes an. Da ein Zug eindeutig umkehrbar ist, könnte auch nach dem Test der Zug wieder rückgängig gemacht werden, was unter Umständen schneller geht als eine Kopie des Spielfeldes. Diese Idee kollidiert nun mit der bereits realisierten Idee, das Kopieren bei StellungOK einzusparen. Der Konflikt läßt sich aber mit der im vorigen Punkt angesprochenen ARRAY-Idee auflösen: StellungOK kann auf dem Originalfeld arbeiten und hinterher durch das ARRAY von Stein-Positionen gehen und jeweils die Markierung löschen.
- Die Funktion ZugListe probiert alle Züge aus, wirft das Ergebnis (die neue Spielstellung) wieder weg. Erst MiniMax führt die Züge 'richtig' aus. Das ist doppelte Arbeit. Wenn nun die Züge gleich durch rekursiven Aufruf ausgewertet würden, so wie sie anfallen, würde man sich die Doppelarbeit sparen. Das erfordert aber einen größeren Eingriff in die Programmstruktur, was vom Aufwand her einer Neu-Implementation gleichkommt.
- weitere Teile in Assembler, z.B. Ziehe (klar)

Anm.d.HzN: UISGE ist aus einer spontanen Idee heraus entstanden: Olaf und ich spielten UISGE (auf einem Brett mit echten Holzsteinen) und da kam einer von uns in Philosophieren: "Das mußte man doch auch mal programmieren können." In weniger als einer Stunde lief das Teil. Da das Herz eines jeden Strategiespielprogrammes die Bewertungsfunktion ist, kam uns die Idee mit dem Wettbewerb.

S o f t w a r e: Einige Ideen

LZH-Komprimierung (Hebert Oppmann, 8520)

Hat mich schon immer fasziniert. Nachdem ich durch die CLUB.90x-Disketten und andere Quellen C-Sources dazu hatte, und wußte, daß unter CP/M nur eine eingeschränkte Version davon implementiert wurde, hat mich interessiert, wo die Einschränkungen liegen. Ich habe die Datei UNLZH.REL (bzw. UNLZH.SLR) analysiert und reassembliert unter Zuhilfenahme der C-Source. Heraus ist jetzt UNL.MAC gekommen. Es ist recht, ja wie soll ich sagen, 'straight ahead' programmiert. Man kann die C-Source als Vorlage noch gut darin erkennen. Demzufolge ist es zwar besser als das, was ein C-Compiler erzeugen würde, aber bei weitem nicht optimal.
Anm.d.HzN: CLUB.057

Noch eine IDEE (Herbert Oppmann, 8520)

Neulich habe ich bei einem Kollegen, der einen AT besitzt, eine interessante Sache gesehen. Ein Utility, das sich resident in den Hauptspeicher klinkt und per HotKey aktiviert wird. Und zwar geht das so:
Ich bin gerade dabei, eine Kommandozeile einzugeben (entweder auf DOS-Ebene oder in einem Programm, das spielt keine Rolle). Ich tippe also 'WS HALLO' und jetzt weiß ich nicht mehr genau, wie das Teil hieß. Hieß es jetzt 'HALLO.TXT' oder 'HALLO.DOC' oder noch anders? Jetzt drücke ich auf den HotKey und voila! es steht 'WS HALLO.DOC' in der Eingabezeile. Das Utility schaut ab Cursorposition zurück bis zum letzten Separator (in diesem Falle das Leerzeichen), bildet die Maske 'HALLO*.*' und sucht auf dem aktuellen Direktory. Wenn es nix findet, dann piepst es mal, ansonsten stellt es den ersten gefundenen Dateinamen in die Zeile. Wenn mir der gefällt, dann drücke ich RETurn, und alles ist paletti. Gefällt er mir nicht, drücke ich einen zweiten HotKey, und es wird mir die nächste zu dieser Maske passende Datei präsentiert. Überzeugt mich das alles nicht so, drücke ich wieder auf den ersten HotKey und habe wieder meine ursprüngliche Eingabe dastehen. Nun kann ich z.B. die Maske verändern und es nochmal versuchen. Übrigens: es funktioniert auch sowas wie 'WS C:\WORK\'. Hier wird statt im aktuellen Direktory im angegebenen Direktory nachgeschaut!

Ich habe ein wenig damit rumgespielt und muß sagen: enorm praktisch! Man gewöhnt sich schnell daran und vermißt es hinterher. Deswegen eine Frage an die 'Gurus': ist das machbar? Es muss bei uns ja nicht auf einem HotKey liegen, es genügt ja erst mal, wenn der normale Kommandozeilen-Editor so erweitert wird, daß er zwei zusätzliche Control-Tasten versteht, die dann die oben skizzierten Funktionen auslösen.

Anm.d.HzN: Gute Idee; mal sehen!

Wünsche an RAM 6.x (Herbert zur Nedden, 2071)

Hans Gras, NL-1506: Wenn die Bildschirmausgabe unter CP/M umgeleitet ist (z.B., wenn der Rechner 'remote' arbeitet, z.B. als Mailbox dient, sollte an Stelle des Blinkefensters die jeweilige Fehlermeldung über's BIOS ausgegeben werden.
Hartmut Traber, 5270: Superschneller Bildschirmtreiber wie FastScreen; Doppelstrich-Cursor.
Herbert Oppmann, 8520: o.g. HotKey-Idee.
Herbert zur Nedden, 2071: KLICK aufrufen können und gleich noch ein paar Tasten ausführen. Format Schneider Vortex erraten können. u.s.w. K2Dos einbauen.
Du: Hast Du noch Ideen?

T u r b o - P a s c a l: CP/M & MsDos

TURBO-Pascal unter CP/M und MsDos (Holger Göbel, 8630)

Es ist kein neues Thema, das ich aufgreife, Herbert hat in INFO 38/6 bereits Anregungen gegeben. Worum geht es?
Manche unserer Programme sollen manchmal auch unter MsDos laufen (weil wir halt so gute Programmierer sind). Mit TURBO-Pascal ist das möglich, da die Version 3.0 beide Betriebssysteme unterstützt. Unter CP/M verfaßte Quelltexte können also, nachdem sie auf eine MsDos-Scheibe kopiert sind, von einem MsDos-Rechner kompiliert werden.
Leider jedoch nur mit gewissen Einschränkungen. Meine Erfahrungen diesbezüglich habe ich mit einem Schneider PC 1640 gesammelt und möchte sie Euch mitteilen.

1. Sonderzeichen

Fangen wir mit dem letzten Schritt an: Der Quelltext muß etwas manipuliert werden, da die Umlaute und die eckigen Klammern unterschiedlich kodiert werden. Herberts Programm CONVERT (CLUB.052) erledigt das. Allerdings ist die Handhabung dieses Programms etwas unkomfortabel, wenn gleich mehrere Dateien konvertiert werden sollen. Ich habe dieses Programm deswegen (mit Hilfe des Source von FILZ) so erweitert, daß jetzt in der Kommandozeile Wildcards akzeptiert werden. Der Aufruf **CONVERT *.PAS /CD** funktioniert jetzt also. (Ohne vom Thema abschweifen zu wollen: In FILZ steckte ein Fehler, der Dateinamen mit 8 Zeichen Länge betraf. Diesen habe ich behoben und die korrigierte Form Herbert gegeben).

2. Funktionstasten

Die Funktionstasten liefern unter MsDos keine vernünftigen CTRL-Codes. TURBO wandelt sie in sog. 'erweiterte Scans' um, die mit einem ESC beginnen. Wenn man also so wichtige Tasten wie die Pfeil- oder PgUp- und PgDn-Tasten abfragen will, muß man die Prozedur **Read (kbd,ch)** durch **ch := TastIn** ersetzen:

```
const                    (*so sind die Fkt-Tasten bei uns i.a. belegt*)
  CsrUp = ^E;
  CsrDn = ^X;
  CsrRi = ^D;
  CsrLe = ^S;
  Home  = ^Z;
  PgUp  = ^R;
  PgDn  = ^C;
  Ins   = ^V;
  EOL   = ^Y;
  RET   = ^M;
  Del   = ^G;
  BS    = ^H;
  ESC   = ^A;


(**** fuer CP/M: ****)

FUNCTION TastIn: Char;
Var ch,c: Char;
BEGIN
  Read (kbd,ch);
  If ch = ^Q then if keypressed then Begin
      (*HOME = ^Q^R unter NewWord, nach ^Z umwandeln*)
   Read (kbd,c);
   If c = ^R then ch := Home;
  End;
  TastIn := ch;
END;
```

T u r b o - P a s c a l: CP/M & MsDos

```
(**** fuer MsDos: ****)

FUNCTION TastIn: Char;
Var ch: Char;
BEGIN
  Read (kbd,ch);
  if (ch = ESC) and keypressed then Begin
   Read (kbd,ch);
   Case ch of
     'H': ch := CsrUp;
     'P': ch := CsrDn;
     'M': ch := CsrRi;
     'K': ch := CsrLe;
     'G': ch := Home;
     'I': ch := PgUp;
     'Q': ch := PgDn;
     'R': ch := Ins;
     'S': ch := Del;
     'u': ch := EOL; (*CTRL-End*)
    End; (*Case*)
   End; (*if ch = ESC*)
  TastIn := ch;
END;
```

3. Bildschirmattribute

Der Monochrom-Bildschirm der MsDos-Rechner bietet Normal- und Fettdruck sowie Unterstrich, Blinken und inversen Hintergrund. Helleuchtenden Hintergrund (wie bei uns) kann er nicht. Der Bildschirmtreiber überprüft, ob Schrift- und Hintergrundfarbe kompatibel sind (schwarze Schrift auf schwarzem Hintergrund geht nicht). Ähnlich wie bei uns ist der Unterstrich auf dem Monochrom-Schirm durch Ausgabe einer Farbe zu erreichen. Vernünftig funktioniert dies allerdings nur im Graphikmodus. Der Bildschirm in MsDos bietet übrigens standardmäßig 25 Zeilen und 80 Spalten.
Zur Verwaltung von Schrift und Hintergrund sowie zum An-/Abschalten des Cursors dienen folgende Prozeduren (das Hinmalen von Pfeilen habe ich von Herbert):

```
type
  Farbe = (norm,bright,uline,revers,invint);
var
  AktSchrift, AktHGrund: Frabe;


(**** fuer CP/M: ****)

PROCEDURE Schrift (c: Farbe);
Begin
  Write (Chr(6)+Chr(0));
  Case c of
    uline:       If Aktschrift = norm then Write(^F#1) Else Write(^F#5);
    norm:        Write(^F#2);
    bright:      Write(^F#4);
  End;
  If c in [norm,bright] then Aktschrift := c;
End;

PROCEDURE HGrund (c: Farbe);
Begin
  Write (Chr(6)+Chr(0));
  Case c of
    revers:      Write(^F#16);
    invint:      Write(^F#48);
   End;
End;
```

T u r b o - P a s c a l: CP/M & MsDos

```
PROCEDURE Blink_an;
Begin
  Write (^N);
End;

PROCEDURE Blink_aus;
Begin
  Write (^O);
End;

PROCEDURE Cursor_an;
Begin
  Write (^^);
End;

PROCEDURE Cursor_aus;
Begin
  Write (^_);
End;

PROCEDURE Pfeile;
Begin
  Write (#27,'GHIJK',#27,'S');
End;

PROCEDURE CrtInit;
BEGIN
  write(^Ä'S'^L^Ä'ÄT2'); (*Newword-Tabelle*)
  AktSchrift := norm; AktHGrund := norm;
END;

PROCEDURE CrtExit;
BEGIN
  write (^Ä'ÄT1');  (*CPM-Tabelle*)
  Cursor_an;
END;


(**** fuer MsDos: ****)

type
  regpack = record
              ax,ah,bx,bh,cx,dx,bp,si,di,ds,es,flags: integer;
            end;
var
  recpack: regpack;                (*record for MsDos call*)

PROCEDURE Schrift (c: Farbe);
Begin
  case c of
   norm: Begin
          AktSchrift := LightGray;
          Textcolor (Aktschrift);
          AktHGrund := Black;
          TextBackground (AktHGrund);
         End;
   bright: Begin
            Aktschrift := White;
            Textcolor (Aktschrift);
           End;
   uline: Begin
           If AktSchrift = LightGray then Aktschrift := Blue
           Else AktSchrift := LightBlue;
           Textcolor (Aktschrift);
          End;
  End;
End;
```

T u r b o - P a s c a l: CP/M & MsDos

```
PROCEDURE HGrund (c:Farbe);
BEGIN
  case c of
   norm: Begin
          Aktschrift := LightGray;
          Textcolor (Aktschrift);
          AktHGrund := Black;
          TextBackground (AktHGrund);
         End;
   revers,invint: Begin
          Aktschrift := Black;
          Textcolor (Aktschrift);
          AktHGrund := LightGray;
          TextBackground (AktHGrund);
         End;
   End;
End;

PROCEDURE Blink_an;
BEGIN
  If Aktschrift < 16 then Aktschrift := Aktschrift + 16;
  Textcolor (Aktschrift);
END;

PROCEDURE Blink_aus;
BEGIN
  If Aktschrift > 15 then Aktschrift := Aktschrift - 16;
  Textcolor (Aktschrift);
END;

PROCEDURE Cursor_an;
Begin
  With recpack do Begin
    AX := $0100;
    CX := $0B0C; (*CH: 1. Scanlinie, CL: letzte Scanlinie*)
    Intr($10,recpack);
  end;
End;

PROCEDURE Cursor_aus;
Begin
  With recpack do Begin
    AX := $0100;
    CX:= $1000;
    Intr($10,recpack);
  end;
End;

PROCEDURE Pfeile;
Begin
  Write(chr(24),chr(25),chr(26),chr(27));
End;

PROCEDURE CrtInit;
BEGIN
  GraphMode;
END;

PROCEDURE CrtExit;
BEGIN
  TextMode;
  Cursor_an;
END;
```

T u r b o - P a s c a l: CP/M & MsDos

4. Fensterroutinen

TURBO-Pascal für MsDos untertützt den Fenstergebrauch mit Hilfe der Prozedur **Window (x1,y1,y1,y2)**, den Rahmen um das Fenster muß man sich aber selber zeichnen. Etwas vorsichtig muß man daher mit den Koordinaten (GOTOXY) umgehen:

```
(**** fuer CP/M: ****)

PROCEDURE SetWindow (x,y,b,h);
Begin
  Write (#27,'Ö',Chr(x+32),Chr(y+32),Chr(b+32),Chr(h+32));
End;

PROCEDURE Zeichne_Rahmen (x,y,b,h: Integer);
BEGIN
  Write (ESC,'O',chr(x+32),chr(y+32),chr(b+32),chr(h+32));
END;


(**** fuer MsDos: ****)

PROCEDURE SetWindow (x,y,b,h);
Begin
  Window (x,y,x+b,y+h);
End;

PROCEDURE Zeichne_Rahmen (x,y,b,h: Integer);
Const
 Lio = #218;
 Liu = #192;
 Reo = #191;
 Reu = #217;
 Wgr = #196;
 Skr = #179;
Var i: Integer;
BEGIN
  x := Succ(x); y := Succ(y); (*zum Ausgleich 0/0 --> 1/1*)
  Gotoxy (x,y); Write (Lio);
  For i := 1 to b do Write (Wgr);
  Write (Reo);
  For i := 1 to h do Begin
   GotoXY (x,y+i); Write (Skr);
   GotoXY (x+Succ(b),y+i); Write (Skr);
  End;
  GotoXY (x,y+Succ(h)); Write (Liu);
  For i := 1 to b do Write (Wgr);
  Write (Reu);
END;
```

6. Datum

Es ist ja manchmal praktisch, die Zeit automatisch abzufragen, wenn man schon eine Uhr hat. Ab dem XT, meist schon PC, besitzen die MsDos-Rechner Uhren-Chips. Auch unsere MTX-Großrechenmaschinen dürften meist ein solches Utensil ihr eigen nennen. Bei der Abfrage des Datums habe ich mich entschieden, hier mal die (sehr umständliche) CP/M+ -Manier zu verwenden, da man sie eben auch unter CP/M+ auch verwenden könnte:

```
(**** fuer CP/M: ***)

PROCEDURE Datumholen (Var Year,Month,Day: Integer);
Var Datumpuffer: Array[1..8] of Byte;
    i,Tage,Jahr,Vierer,Rest,Einzeljahre,Summe,Monat: Integer;
    Tage_im_Monat: Array[1..12] of Integer =
      (31,28,31,30,31,30,31,31,30,31,30,31);
```

```
BEGIN
  Bdos (105,Addr(Datumpuffer)); (*Tage seit 1.1.1978*)
  Tage := Datumpuffer[1] + 256*Datumpuffer[2];
  Tage := Tage-1097; (*Anfang 1981*)
  Vierer := Tage div 1461; (*Wieviele Vierer-Bloecke?: 3*365 + 366*)
  Rest := Tage Mod 1461;   (*Wieviele Jahre ausserdem?*)
  Einzeljahre := Rest div 365;
  If Einzeljahre = 4 then Begin  (*31.12. eines Schaltjahrs*)
   Einzeljahre := Pred(Einzeljahre);
   Rest := 365;
  End
  Else Rest := Rest mod 365;     (*Wieviele Tage in diesem Jahr?*)
  If (Einzeljahre = 3) AND (Rest > 60) then Rest := Pred(Rest); (*29.2.*)
  Jahr := 1980 + Vierer*4 + Succ(Einzeljahre);
  Monat := 0;
  Summe := 0;
  Repeat
   Monat := Succ(Monat);
   Summe := Summe + Tage_im_Monat[Monat]
  Until Summe > Rest;
  Year := Jahr;
  Month := Monat;
  Day := Tage_im_Monat[Monat] - (Summe - Succ(Rest));
END;


(**** fuer MsDos: ****)

PROCEDURE Datumholen (Var Year,Month,Day: Integer);

(* recpack ist definiert wie unter 3. beschrieben *)

Begin
  with recpack do begin
   ax := $2a shl 8;
  end;
  MsDos (recpack);                          (* call function *)
  with recpack do begin
   Year := cx;
   Day := dx mod 256;
   Month := dx shr 8;
  end;
End;
```

7. Drucker

In der MsDos-Welt gibt es nicht nur den EPSON-Standard, leider sind da auch IBM-Drucker weit verbreitet, die die Kodierung der Umlaute handhaben wie MsDos. Da wir MTX-ler kaum IBM-Drucker verwenden dürften, betrachten wir nur den MsDos-Fall: Wenn Zeichen an einen EPSON-Drucker gegeben werden, müssen sie erst einen Filter durchlaufen. Zum Glück bietet TURBO-Pascal die Möglichkeit, den LstOutPtr zu verbiegen:

```
(**** fuer MsDos: ****);

Type
  DrTyp = (IBM,EPSON);
var
  ownlstptr,orglstptr: integer;
```

T u r b o - P a s c a l: CP/M & MsDos

```
PROCEDURE DruckFilter (ch: Char);
  (*Die Bezeichnungen habe ich von Herberts CONVERT*)

const (* CP/M oder EPSON*)
      c_kl_ae = #$7b; c_gr_ae = #$5b;
      c_kl_oe = #$7c; c_gr_oe = #$5c;
      c_kl_ue = #$7d; c_gr_ue = #$5d;
      c_eszet = #$7e;
const (* DoMessDos oder IBM*)
      d_kl_ae = #$84; d_gr_ae = #$8e;
      d_kl_oe = #$94; d_gr_oe = #$99;
      d_kl_ue = #$81; d_gr_ue = #$9a;
      d_eszet = #$e1;
      d_gkl_a = #$7b; d_gkl_z = #$7d;
      d_ekl_a = #$5b; d_ekl_z = #$5d;

BEGIN
  If DrTyp = EPSON then
  case ch of
    d_kl_ae,d_gkl_a : ch:=c_kl_ae;
    d_gr_ae,d_ekl_a : ch:=c_gr_ae;
    d_kl_oe         : ch:=c_kl_oe;
    d_gr_oe         : ch:=c_gr_oe;
    d_kl_ue,d_gkl_z : ch:=c_kl_ue;
    d_gr_ue,d_ekl_z : ch:=c_gr_ue;
    d_eszet         : ch:=c_eszet;
  end;

  LstOutPtr := orglstptr;
  write (lst,ch);
  lstoutptr := ownlstptr;
END;

(* Vorher unbedingt aufrufen: *)
PROCEDURE DrInit;
BEGIN
  orglstptr := LstOutPtr;
  ownlstptr := Ofs(DruckFilter);
  Lstoutptr := ownlstptr;
  DrTyp := IBM; (*oder EPSON*)
END;
```

8. Datenformate

Will man außer Programmen auch Daten zwischen den beiden Systemen hin- und herschaufeln, so gilt es hier zu beachten, daß TURBO-Pascal unter CP/M grundsätzlich am Anfang eines Files die Zahl und Länge der Sätze abspeichert. Unter MsDos ist das nicht notwendig, da hier bereits im Directory die genaue Länge des Files zu finden ist. Aus diesem Grund ist es notwendig, Daten erst in ein ASCII-Format umzuwandeln, welches dann vom jeweilig anderen System rückverwandelt wird. Zum Glück unterstützt TURBO-Pascal diese Umwandlung.
Ein Manko gibt es noch: Kompatibel sind alle Datenformate bis auf die Zeigertypen: diese sind unter CP/M 16 Bit groß, unter MsDos 32 Bit. Da müßte also unter CP/M immer ein Dummy-Zeiger als Platzhalter mitgeführt werden.
Beispielhaft möchte ich eine solche Umwandlung zeigen. Zu bedenken ist dabei, daß man nicht immer weiß (bei einem Record z.B.), wie groß das Datenfeld ist. Deshalb markiere ich Anfang und Ende durch ein Dummy-Byte:

T u r b o - P a s c a l: CP/M & MsDos

```
const
  Adresszahl: 100;

type
  Adresse = Record
             Name: STRING[20];
             Vorname: STRING[15];
             Alter: Integer;
             (* .... usw.usf.*)
            End;
  AdressFile: File of Adresse;

var
  a1: Byte; (*Dummy*)
  Adr: Array [1..Adresszahl] of Adresse;
  a2: Byte; (*Dummy*)
  AdrText: Array [1..MaxInt] of Byte Absolute Adr;
  AdrDatei: AdressFile;
  AdrTextDatei: Text;

(**** fuer CPM: ****)

FUNCTION Laenge (Var Beginn,Ende): Integer;
BEGIN
  Laenge := Abs (Addr(Ende) - Addr(Begin)) - 1;
END;

(**** fuer MSDOS: ****)

FUNCTION Laenge (Var Beginn,Ende): Integer;
BEGIN
  Laenge := Abs (Ofs(Ende) - Ofs(Beginn)) - 1;
END;

(**** fuer beide: ****)

PROCEDURE Dat_Txt;
Var i,j,l: Integer;
BEGIN
  Assign (AdrDatei,'ADRESSE'+'.DAT');
  Assign (AdrTextDatei,'ADRESSE'+'.TXT');
  Reset (AdrDatei);
  Rewrite (AdrTextDatei);
  l := Laenge (a1,a2);
  For j := 1 to Adresszahl do Begin
   Read (AdrDatei,Adr);
   For i := 1 to l do Write (AdrTxtDatei,AdrText[i]:4);
   Write (PTextDatei,0:4);
  End;
(* es wird ein Byte mehr abgespeichert, weil es zwischen
   MsDos und CP/M Unterschiede mit dem Dateiende gibt;
   gelesen wird die tatsaechliche Laenge *)
  Close (AdrDatei);
  Close (AdrTextDatei);
END;

PROCEDURE Txt_Dat;
Var i,j,l: Integer;
BEGIN
  Assign (AdrDatei,'ADRESSE'+'.DAT');  Assign (AdrTextDatei,'ADRESSE'+'.TXT');
  Rewrite (AdrDatei);                  Reset (AdrTextDatei);
  l := Laenge (a1,a2);
  For j := 1 to Adresszahl do Begin
   For i := 1 to l do Read (AdrTextDatei,AdrText[i]);
   Write (AdrDatei,Adr);
  End;
  Close (AdrDatei);                    Close (AdrTextDatei);
END;
```

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

9. Warum überhaupt MsDos?

Diese Frage habe ich mir während der Tüftelei ob dieses Transfers oft genug gestellt. Ich will andererseits nicht schon wieder über jenes 'System' lästern, jeder von uns weiß zur Genüge, was er an unserem Rechner hat. Da halt andere nicht unbedingt von diesem Sachverhalt zu überzeugen sind, muß man manchmal ein paar Ebenen hinabsteigen, um die Unverbesserlichen zu beglücken. Mögen obige Routinen zu jenem Behufe nützlich sein.

Datum und Feiertage (Holger Göbel, 8630)

Wie man das Datum vom Uhren-Chip erhält, habe ich oben dargelegt. Oft ist es aber auch noch wichtig zu wissen, mit welchem Wochentag dieser Tag getauft ist, und ob an diesem Tag gerade ein Feiertag ist.
Den Algorithmus zur Bestimmung des Wochentages habe ich aus einem dBase-Buch, er funktioniert von 1900 bis 2099. Mit den Feiertagen hingegen ist es so eine Sache. Zunächst gibt es da diese verschiedenen Regelungen in den Bundesländern. Das läßt sich programmtechnisch durch etwas Fleiß aber lösen. Etwas mehr Schmalz verbraucht die Berechnung der osternabhängigen Feiertage. Ostern liegt ja auf dem ersten Sonntag nach dem ersten Frühlingsvollmond; und von Ostern hängen eine Menge Feiertage ab (Pfingsten, Christi Himmelfahrt usw.). Es bleibt also nichts anderes übrig, als sich ein bißchen mit den Mondzyklen zu beschäftigen, von denen es eine ganze Reihe gibt (s. Meyers Enzyklopädie ...).
Nun zu den Routinen:
Die Berechnung der Monats- und Wochentage spricht für sich. Die Feiertage lege ich in einem zweispaltigen Array FT (1. Spalte für den Monat, zweite für den Tag) ab.

```
Const
  FTZahl = 20;  (*Zahl der Feiertage, muesste reichen*)
Type
  Vars = Record
           BL:  STRING[2];  (*Bundesland: Ba, BW, NW, Sa, SH usw.,
                              Au fuer Augsburg*)
           konf: STRING[2]; (*Konfession: ka - ev *)
         End;
Var
  FT: Array [1..FTZahl,1..2] of Integer;
  Variablen: Vars;

FUNCTION Monatstage (Jahr,Monat: Integer): Integer;
VAR Days: Byte;
BEGIN
  If Monat = 2 then Begin
    If (Jahr MOD 4 <> 0) OR
      ((Jahr MOD 4 = 0) AND (Jahr MOD 100 = 0) AND (Jahr MOD 400 <> 0))
    then Days := 28
    Else Days := 29;
  End
  Else If Monat in [1,3,5,7,8,10,12] then Days := 31
  Else Days := 30;
  Monatstage := Days;
END;
```

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

```
FUNCTION Wochentag (Jahr,Monat,Tag: Integer): Integer;
 (* Sonntag = 1 *)
VAR wo: Integer;
BEGIN
  Jahr := Jahr Mod 1900;
  If Monat > 2 then Monat := Monat - 2
  Else Begin
   Monat := Monat + 10;
   Jahr := Pred(Jahr);
  End;
  wo := (Pred(13*Monat)) div 5 + Tag + Jahr + Jahr div 4 - 34;
  wo := Succ(wo) - 7*(wo div 7);
  Wochentag := wo;
END;


PROCEDURE Feiertage (Jahr: Integer);
CONST Mondzyklus = 29.53059;

Var i: Integer;
    Tage,Ph1900,Mondphase: Real;
    Vollmondmonat,Vollmondtag,Ostertag,T,Diff: Integer;

 PROCEDURE Datumeintragen (OffTage,FTNr: Integer);
   (*fuer osternabhaengige Feiertage*)
 CONST Monatsarray: Array [3..6] of Integer =
                          (0,31,61,91);    (*Tage seit dem 1.3.*)
 Var i: Integer;
 BEGIN
   i := 7;
   Repeat
    i := Pred(i);
   Until OffTage > Monatsarray[i];
   FT [FTNr,1] := i;
   FT [FTNr,2] := Offtage - Monatsarray[i];
 END;

BEGIN
  For i := 1 to FTZahl Do Begin
   FT[i,1] := 0; FT[i,2] := 0;
  End;

  FT[1,1] := 1; FT[1,2] := 1; (*Neujahr*)
  With Variablen Do Begin
   If (Bl = 'Ba') OR (Bl = 'Au') OR (Bl = 'BW') then Begin
    FT[2,1] := 1; FT[2,2] := 6; (*DreiK¦nig*)
   End;
  End;

         (*Osternabhaengige Feiertage:*)

  Ph1900 := 29.2 - Mondzyklus/2; (*29.2 war die Phase im Neumond-Zyklus am
                          1.1.1900, hier interessiert aber der Vollmond!*)
  Jahr := Jahr Mod 1900;
  Tage := (Jahr+0.0)*365 + Jahr div 4 + 79; (*21.3.*)
  Mondphase := Frac((Ph1900 + Tage)/Mondzyklus) * Mondzyklus;
  Diff := Trunc(Mondzyklus - Mondphase);
  If Diff > 10 then Begin
   Vollmondtag := Diff - 10;
   Vollmondmonat := 4;
  End
  Else Begin
   Vollmondtag := Diff + 21;
   Vollmondmonat := 3;
  End;
  T := Wochentag (Jahr,Vollmondmonat,Vollmondtag);
  T := (8 - T) MOD 7;
  Ostertag := Vollmondtag + T;
```

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

```
  Datumeintragen (Ostertag-2,3);     (*Karfreitag*)
  Datumeintragen (Ostertag,4);
  Datumeintragen (Succ(Ostertag),5); (*Ostermontag*)
  Datumeintragen (Ostertag+39,6);    (*Christi Himmelfahrt*)
  Datumeintragen (Ostertag+49,7);    (*Pfingstsonntag*)
  Datumeintragen (Ostertag+50,8);    (*Pfingstmontag*)
  With Variablen Do Begin
   IF (Bl = 'BW') OR (Bl = 'Ba') OR (Bl = 'Au') OR (Bl = 'He')
   OR (Bl = 'NW') OR (Bl = 'RP') OR (Bl = 'Sa') then
   Datumeintragen (Ostertag+60,9);    (*Fronleichnam*)
  End;

  FT[10,1] := 5; FT[10,2] := 1;   (*1. Mai*)
  With Variablen Do Begin
   If ((BL = 'Ba') AND (Konf = 'ka')) OR (BL = 'Sa') OR (Bl = 'Au')
   then Begin
    FT[11,1] := 8; FT[11,2] := 15; (*Mariae Aufnahme*)
   End;
  End;
  FT[12,1] := 10; FT[12,2] := 3;    (*Tag d. dt. Einheit*)
  With Variablen Do Begin
   IF (Bl = 'BW') OR (Bl = 'Ba') OR (Bl = 'Au') OR (Bl = 'NW')
   OR (Bl = 'RP') OR (Bl = 'Sa') then Begin
    FT[13,1] := 11; FT[13,2] := 1; (*Allerheiligen*)
   End;
  End;

    (*Buss- und Bettag: *)
  T := Wochentag (Jahr,12,25); (*1. Weihnachtsfeiertag*)
  If T = 1 then T := 16 Else T := 24 - T; (*4. Advent - 32 Tage*)
  FT[14,1] := 11; FT[14,2] := T;

  FT[15,1] := 12; FT[15,2] := 24; (*Heiliger Abend*)
  FT[16,1] := 12; FT[16,2] := 25; (*1. Weihnachtsfeiertag*)
  FT[17,1] := 12; FT[17,2] := 26; (*2. Weihnachtsfeiertag*)
  FT[18,1] := 12; FT[18,2] := 31; (*Silvester*)

  If (Bl = 'Au') then Begin (*Friedensfest in Augsburg*)
   FT[19,1] := 8;
   FT[19,2] := 8;
  End;
END;


FUNCTION Feiertag (Jahr,Monat,Tag: Integer): Boolean;
Var Feier: Boolean;
    T,i: Integer;
BEGIN
  Feier := false;
  For i := 1 to FTZahl Do Begin
   If Monat = FT[i,1] then If Tag = FT[i,2] then Feier := true;
  End;
  T := Wochentag (Jahr,Monat,Tag);
  Feier := Feier OR (T = 1);
  Feiertag := Feier;
END;


BEGIN  (*Demo-Bloedsinn*)
  Feiertage (1991);
  If Feiertag (1991,12,24) then Write ('Heute ist Feiertag');
END.
```

Noch nicht beruecksichtigen konnte ich die Feiertagsregelung in den neuen Bundesländern. Das kann man ja nachtragen.

H a r d w a r e: SiDisc in der SDX / 80Zeichen-Karte

SDX-SIDISK (Helmut Grothe, 8000)

Ich habe einen auf der Hauptplatine auf 512 k aufgerüsteten MTX mit einer 3.5 '' SDX mit zusätzlich eingebauter 512 k SIDISK, die ich leider bisher nicht nutzen konnte.

Beim Installieren von RAM 6.1 fiel mir nun auf, daß INST61 (abgesehen von Bank 0) 512 k Speicherplatz meldet, d.h. daß der Rechner insgesamt 576 k haben muß! Das bestätigt die bereits früher gehegte Vermutung, daß die SIDISK einfach eine Speichererweiterung ist, die parallel (aber offensichtlich um 64 k verschoben) zu den 512 k auf meiner Hauptplatine läuft.

Um das zu überprüfen, habe ich zunächst die Leitung CAS2(invertiert) zu den oberen 256 k auf der Hauptplatine unterbrochen (s. Info 40 S.21). Der MTX lief damit genauso wie vorher. Beim Unterbrechen von CAS1(invertiert) hingegen ging gar nichts mehr. Daraus schließe ich, daß die ersten 64 k (Bank 0) auf der Hauptplatine des MTX liegen müssen, während die SIDISK in der SDX den darauffolgenden Speicher bis 576 k belegt. Damit ist also der Bereich von 65 k bis 512 k doppelt belegt. Dieser parallelliegende Speicherbereich (immerhin 448 k) sollte nun natürlich ebenfalls genutzt werden.

Im Prinzip läßt sich sicherlich das PROM für die 512 k Erweiterung auf der Hauptplatine (s. Info 40 S.20-22) so programmieren, daß von den 512 k im MTX die ersten 64 k auf Bank 0 liegen und dann 208 k ab 577 k bis 784 k folgen. Die restlichen 240 k können wohl nicht so ohne weiteres für eine Aufrüstung auf 1024 k verwendet werden (oder hat da vielleicht irgendjemand eine Idee ?).

Da ich im Moment keine solche Idee habe, bin ich zunächst einmal anders vorgegangen. Um den restlichen direkt adressierbaren Speicherbereich bis zu 784 k zu erschließen, habe ich einfach das Signal P3 an Pin 15 des PROMs invertiert (s. Info 40 S. 21). Dazu wird die Verbindung zwischen Pin 15 am PROM und Pin 18 am darunterliegenden PAL unterbrochen. Pin 18 (PAL) wird mit Pin 12 am Huckepack über IC 5B liegenden 74LS00 verbunden, Pin 11 direkt daneben wird mit Pin 15 am PROM verbunden. Schließlich noch Pin 13 vom 74LS00 mit Pin 14 (+ 5 V) desselben ICs verbinden.

Die Reihenfolge der einzelnen Bereiche im Speicher wird dadurch natürlich etwas durcheinandergewürfelt, aber das merkt man ja von außen nicht. Wichtig ist, daß beim Wiedereinschalten des MTX nun wirklich 784 k verfügbar sind!

80Zeichen-Karte: Ideen (Jan Bredereke, 2000)

Zur 80-Zeichen-Karte habe ich zwei Ideen, die ich weitergeben möchte, weil ich nicht so schnell dazu kommen werde, sie umzusetzen: In Info 38-41 wurde vorgeschlagen, das EPROM mit der Kästchen-Grafik gegen ein EPROM mit weiteren Sonderzeichen auszutauschen, z.B. griechischen Buchstaben und weiteren Rahmenzeichen. Das fand ich etwas unschön, weil ich die Kästchengrafik nicht gern verlieren würde. In Info 35-38 wurde ein EPROM mit vier von Hand umschaltbaren Zeichensätzen beschrieben. Und nun meine erste Idee: Man könnte doch die bei Monochrom-Monitoren unbenutzen Bits im Attribut-Byte benutzen, um das EPROM umzuschalten. Dies läßt sich auch sehr leicht in alle vorhandene Software installieren, z.B. NewWord (wie in 38-41 beschrieben). (Diese Lösung ist für Farbmonitorbesitzer natürlich völlig uninteressant.) Und meine zweite Idee: Ich brauche für meine Diplomarbeit ständig neue, ungewöhnliche Sonderzeichen. Was wäre daher mit einem Zeichensatz-RAM anstelle eines EPROMS? Man könnte jeweils die gewünschten Sonderzeichen hineinladen und immer weitere definieren.

H a r d w a r e: RLL-Festplatte am MTX

RLL-Festplatte am MTX

(Ulrich Taschke, 7500)

Vor einiger Zeit bekam ich von einem Kollegen einen OMTI-Controller geschenkt. Er meinte, er hätte an seinem IBM damit Schwierigkeiten, außerdem wollte er sich eine andere Festplatte kaufen. Da ich in den Infos von Festplatten am MTX gelesen habe, habe ich ihn gefragt, und er verkaufte mir dann auch noch seine Platte dazu.
Das erste, was mich etwas irritierte, war, daß der Controller weder 5510 noch 5520 hieß. Und diese Typen wurden bisher ja ausschließlich am MTX eingesetzt. Was mir Hoffnung gab, war, daß auf dem Datenblatt der 5527 und der 5520 gemeinsam abgehandelt wurden, also konnte da der Unterschied nicht allzu groß sein. Eine Nachfrage bei Holger Hansen ergab auch, daß sich der 5527 bei den 3 Befehlen, mit denen er im RAM 6.x angesprochen wird identisch verhält, wie der 5520. Also ließ ich mir von Gerhard Witzel die OMTI-Adapterkarte machen und ging ran ans Werk.

Zur Hardware: Ich habe es mir einfach gemacht. Von meinen 2 Diskettenlaufwerken habe ich eins aus der FDX rausgeschmissen und dafür die Platte eingebaut. Meine Hoffnung, daß das das Netzteil mitmacht, hat sich bisher nicht zerschlagen.

Und was heißt RLL? Das ist so ziemlich die einzige Frage, die ich direkt beantworten kann, es heißt RUN LENGTH LIMITED, und bezeichnet ein spezielles Aufzeichnungsverfahren, mit dem die Daten auf die Platte geschrieben werden. In den paar Artikeln, die ich über Festplatten durchgeschmökert habe, war von RLL jeweils nie die Rede, lediglich von MFM, das ja das Aufzeichnungsverfahren der 5510 und 5520 ist. Ich kann also keinen großen Vorteil von RLL gegenüber MFM nennen, außer, daß die Plattenkapazität etwa 1.5 mal so groß wird, weil man mehr Sektoren pro Spur hat (beim 5527 fest auf 26*512 Bytes gegenüber 17*512 Bytes beim 5520). Meine SEAGATE ST238R hätte also mit MFM "nur" 20 MByte, mit RLL hat sie 32 MB. Wichtig ist auch, daß man spezielle für RLL geeignete Festplatten braucht, daher auch das R in der Plattenbezeichnung. Ich stelle mir das so ähnlich vor wie bei den Disketten mit Single- und Double Density. Aber viel wichtiger als das war mir, daß die Platte überhaupt läuft.

Und da gab es so einiges. Ersmal war das Wirrwarr um die 12V-Spannungsversorgung nicht sehr ermutigend. Aber es war kein Problem: So, wie Gerhard die Adapterplatine verdrahtet hat, ist alles in Ordnung. Die IBM-XT-Slots sind ja auch festgelegt, und so kann man relativ einfach nachvollziehen, was wohin gehört.
Sehr einfach hat mir die Festplatteninstallation auch das Programm HDSERV von Holger Hansen gemacht. Denn damit kann man sehr einfach und schnell prüfen, ob die Festplatte vom System angesprochen werden kann, oder nicht. Und bei mir konnte sie nicht! Etwas Suche brachte dann den Fehler: Es lag an der Verdrahtung auf der OMTI-Adapterkarte. Laut Beschreibung im c't-Artikel "Harddisk-Controller einmal anders" (4.87) und dem Plan für unser OMTI-Interface müssen die beiden Adreßbits 5 und 6 vertauscht werden. Sonst liegt der Controller nicht auf I/O-Adresse $48, sondern $28. Und damit läßt er sich nicht so einfach ins RAM 6.x integrieren. Man braucht dafür nur 2 Lötbrücken auf der Adapterkarte über Kreuz zu legen, und schon geht's.

Auch für die Anbindung der Festplatte ins RAM 6.x System hat Holger Hansen eine wichtige Hilfe geleistet. In seinem Artikel in Info 38 bekommt man ausführlich alle Schritte vorgekaut, die man durchführen muß.

H a r d w a r e: RLL-Festplatte am MTX

Trotzdem war bei mir immer nur die äußerste Partition ansprechbar. Wollte ich eine andere Partition ansprechen, endete das immer mit einem Rechnerabsturz (nachdem die Platte einige Male hin und her geschnurrt hat). Scheinbar verträgt meine Platte die langsame Stepzeit von 3ms nicht. Und die ist in RAM 6.x so voreingestellt. Zum Glück gab's da auch wieder einen Beitrag im Info, in dem steht, wie man eine andere Steprate ins RAM patcht. (Info 38, von HzN. Die Adresse für RAM 6.1 ist übrigens 9161H). Erst damit ließen sich bei mir die Laufwerke K: bis N: ansprechen.

Nächste Hürde: Ich habe für Laufwerk J: eine Systemspur vorgesehen. Theoretisch müßte das reichen, denn eine Spur hat bei mir 4*26*512 Bytes = 52K. Und auf der #03-Diskette werden gerade 2 Systemspuren à 26*256 = 6.5K, also 13K fürs System reserviert. Und trotzdem hieß es jedesmal, wenn ich SYSCOPY J: eingegeben habe "Speicherplatz reicht nicht aus" o.ä.! Der Grund (Ich hoffe, ich habe Herbert richtig verstanden) : Irgendeine Funktion im Original-Memotech-Boot-ROM geht einfach davon aus, daß ein Laufwerk pro Spur immer nur 26 logische 128-Byte-Sektoren hat. Sind die voll, dann macht die Routine einen Step. Und diese Funktion gibts immer noch in RAM. Das System braucht also nur länger als 3.25K zu sein, schon braucht man 2 Systemspuren, egal wie groß die sind. Es blieb mir also nichts anderes übrig, als nochmal 52K fürs System zu spendieren.

Und nun war endlich alles so weit: Ich konnte hergehen, und den besten Skew-Faktor ermitteln. Das Dumme ist nur, daß der bei mir =0 ist! Und so schnell ist mein 4MHz-Z80A auch wieder nicht. Hier sind meine 2 Methoden, mit denen ich das ausprobiert habe, vielleicht findet einer von Euch ja einen grundsätzlichen Denkfehler darin:

1.: Inhalt einer beliebigen Diskette von Partition J: auf K: kopieren und Zeit stoppen.
2.: Ein Programm, das eine Datei erzeugt, und in einer Schleife (per BlockWrite) irgendwelche Daten rausschreibt. Die Datei ist etwa 3 MB lang. Die Laufzeit des Programmes habe ich dann einfach gemessen.

Die letzte Erklärung, die Holger hatte, ist, daß der Controller immer eine komplette Spur der Festplatte in einem eigenen Cache hält, und somit ein Skew wirkunkslos wird. Ich tendiere auch zu dieser Erklärung.

So, das waren meine Erfahrungen mit der Installation meiner Festplatte. Jetzt tut sie wie selbstverständlich ihren Dienst, und ich bin froh, daß ich jetzt keine Disketten mehr schaufeln muß.

H a r d w a r e: Harddisk ja oder doch? und RAM 6.x

Anm.d.HzN: Ich habe den folgenden Artikel aus zwei Briefen von Hartmut zusammengestellt (und dabei einen Teil herausgenommen und an eine andere Stelle ins Info gepackt). Daher kann es sein, daß der eine oder andere Gedankensprung drin ist.

Ein paar Gedanken zur Harddisk und RAM 6.x (Hartmut Traber, 5270)

Es macht richtig Spaß, mal einfach ohne Rücksicht auf das Speichermedium Platz in Anspruch zu nehmen. Die Formate 1A-aufwärts waren ja schon toll, aber nun habe ich das Vielfache! (Mit der Schnelligkeit ungefähr der RAM-Floppy!).

Der Output kann sich erheblich verbessern, da das Suchen nach Disketten nunmehr wegfällt, (der Vergangenheit angehört), man hat die Möglichkeit, alle interessierenden Programme, Daten usw. ständig im Griff zu haben.
Ich kann nur den Fans empfehlen, den Rechner aufzurüsten. Die Kosten insgesamt sind doch erheblich und reichen an den Anschaffungpreis eines neuen, modernen(?) Systems heran, aber der Aha-Effekt belohnt den Fan.

Da hat doch einer mal 'nen tollen Trick gefunden?
Sorry, ich kann nicht alles, was da so kommt, sofort verarbeiten und nachhalten. Life hieß das, und darin war was verborgen: etwa eine schnelle Bildschirmausgabe? Natürlich von Olaf, zu finden auf Klick006 ! Wie wäre es, diese Routinen in RAM6X als Treiber einzubauen ? Auf den Televideo-Treiber könnte man mittlerweile sicher verzichten.

Anm.d.HzN: Der dort verwendete Bildschirmtreiber, dessen sich auch mein DirEdit bedient ist sehr rudimentär! Er kann nur Zeichen und Attribute ausgeben! Sogar die Cursorpositionierung muß das Programm selbst machen, d.h. aus (x,y) die Adresse des Zeichens im Video-RAM ermitteln! Daher nützt dieser Treiber vermutlich nur solchen Programmen, die das alles selbst in der Hand haben. Und dann sollten diese den FastScreen-Treiber in sich selbst beinhalten, da er erst dann richtig schnell ist, da dann das Programm den rechten Druchgriff hat.

Ich sitze zwar jetzt vor dem Rechner und tippe ein, da merkt man natürlich nichts, aber unter WS ^B gedrückt, dauert es. Oder ausgestiegen und ein Directory aufgerufen, ist das Floppy- oder HD-Laufwerk schon stille, und dann kommt die Ausgabe. Mit dem Cache ist das noch viel spürbarer. Da ist eine Bremse drin!
Haltet es mir nicht vor, ich teste im Club gerne die Hardware, aber Software und insbesondere diese komplizierte vom RAM61 ist mir zu hoch. Aber wer es kann, sollte sich des obigen Software-Problems mal annehmen.
Die Performance unseres Systems ist schon hoch, könnte dadurch aber noch gesteigert werden.

Jetzt noch ein Kommentar zum Leserbrief von Claudio, Info 41 - 14, zweiter Absatz, unter Berücksichtigung der Tatsache, daß die Harddisk fast so schnell ist wie die SRAM-Floppy, ich habe den Eindruck, daß sie sogar schneller ist (bei 6 MHz):
Ich finde es gut, daß man sich mal grundsätzliche Gedanken macht und diese dann auch zur Diskussion stellt. Über die Festspeichernutzung habe ich mir vor Jahren auch schon einmal Gedanken gemacht, nämlich nach Einbau der ersten SRAM-Floppy. Wenn ich nicht irre, habe ich das auch mitgeteilt.

H a r d w a r e: Harddisk ja oder doch? und RAM 6.x

Wie läuft es unter CP/M und ohne Speichererweiterung? Wir booten von einem x-beliebigen Laufwerk, dieses ist dann als A: erklärt und das System holt sich alles, was es braucht von diesem Laufwerk. Unter RAMx ist das anders: RAMx richtet die eigenständige RAM-Disk A: ein, die dann unterschiedlich genutzt wird, nämlich als Heap (was immer das auch ist, steht ja im Handbuch) und als Floppy A:. Boote ich von der SRAM-Floppy, dann muß ich die Klicker und das System (dies automatisch) nach A: in den Heap holen und habe sie dann doppelt in der Kiste. Man kann ja einwenden, die Klicker gehören nicht in die schnelle SRAM-Floppy, sie werden nur beim Booten geholt, und das geht ja auch von einem echten Laufwerk. Das dauert mir aber zu lange und deswegen habe ich den kostbaren schnellen Speicherplatz in I: bisher für Klixe geopfert, anstatt dort andere nützliche Programme unterzubringen.

Nachdem jetzt meine Hard-Disk läuft, kommt natürlich der Wunsch auf, von dieser zu booten. Alle Platzprobleme wären damit behoben, ein einheitliches Laufwerk J: allerdings Voraussetzung; und dazu: die HD ist so schnell wie die SRAM-Floppy! Aber auch da: die Daten sind gedoppelt.

Mir leuchtet folgendes nicht ganz ein:
A: ist DRAM- und z. B. I: ist SRAM-Floppy und damit das eine z. T. echter Kernspeicher, aber auch Floppy A:, und das andere, nun ja, Floppy I:.
Beide Floppies sind in etwa gleich schnell und damit für mich gleichwertig (daß die Behandlung des Heaps und der Floppy A: vom Betriebssystem völlig unterschiedlich erfolgt, ist natürlich klar).
Ich meine wie Claudio, daß sich dann das Betriebssystem an die Hardware anzupassen hat, mit Geburtshilfe von Herbert und Olaf!
Wir sollten tatsächlich den Kern-Speicher, d.h. die gesamte Speichererweiterungs-Platine als solchen auch reservieren. RAM6x muß dann eben auf ein Non-removable-Laufwerk zugreifen, oder auf die HD, so man hat,und muß deswegen installierbar gemacht werden! Ich könnte mir vorstellen, da die Bank-Umschaltung klappt, daß dann auch Programme so geschrieben werden, daß sie 768 kB auf einen Schlag kopieren können, daß auch unsere Standard-Software von Freaks so gepacht wird, daß die z.B. 60-K-Grenze für sie nicht mehr besteht! Das wäre doch was: mehr Hauptspeicher als popelige PC und XT!
Ich stimme also Claudio bei seinen Überlegungen voll zu, wobei meine Überlegungen sicher nicht maßgebend sein sollten, da ich viel zu wenig von unserem Rechner und dem Betriebs-System verstehe und als Nutzer der gesammelten Erkenntnisse im Club höchstens Praxis-Tips einbringen kann.
Ich bitte dies konstruktiv aufzufassen! Ein weiteres Lob über RAMx erübrigt sich, da dieses System selbstverständlich Jeden sofort überzeugt, der auch nur mal kurz 'reingeschaut hat!
Innerhalb dieses so vortrefflichen Systems mal einen Rückschritt nach "good old MTX-CP/M" zu tun kann wiederum Fortschritt sein. (Nicht alle dSM waren möglicherweise so furchtbar dunkel?).
Wie wäre es mit einem Ideen-Wettbewerb?
Hartmut

H a r d w a r e: Speicher

Kern-Speicher vs. SRAM-Floppy (Herbert zur Nedden, 2071)

Ich möchte versuchen, etwas Klarheit in die Unterschiede zwischen dem KLIX-Heap und einer SRAM-Floppy bringen, um so einige der immer wieder mal auftauchenden Ideen und Diskussionen zur Fragestellung, wo KLIX-Overlays stehen sollten und ob die interne RAM-Floppy A: so, wie sie jetzt ist Sinn macht.

Erst mal die technische Seite

Diesen Teil bitte nicht einfach nach dem Motto "das versteh' ich nicht" oder "das interessiert mich nicht" u.s.w. überspringen. Ohne diese Informationen sind meine weiter hinten kommenden Antworten zwar verständlich, nur das Warum nicht klar.

Die Z80-CPU kann auf zweierlei Arten Informationen von außen holen. Zum einen durch Speicher-Zugriffe, bei denen die 16 Adreßpins die Adresse liefern und das Signal MREQ (= Memory-Request = Speicher-Anforderung) aktiv ist. Weiterhin kann die Z80-CPU über sog. Port-Zugriffe, bei denen die unteren acht Adreßpins die Adresse liefern und das Signal IOREQ (InputOutput-Request = Ein/Ausgabe-Anforderung) aktiv ist.

Weiterhin kennt die Z80 zweierlei Arten von Informationen: Daten und Befehle. Befehle sind die Dinger, die die Z80 ausführt, Daten ist das, was die Z80 bearbeitet, z.B. von einer zu einer anderen Stelle transportiert.

Die Z80 kann max. 64kB über Speicher-Zugriffe erreichen, da mit einer 16-Bit-Adresse mehr nicht möglich ist; dieser Speicher heißt Hauptspeicher oder auch Kernspeicher. Beim MTX kann jedoch ein Teil (nämlich die unteren 48kB) dieses Speichers umgeschaltet (gebankt) werden, so daß unsere Z80 auf 768kB per Speicherzugriff erreichen kann. Soll ein Programm allerdings mehr als 64kB verwenden, so muß es sich selbst (bei uns anstandshalber via RAM 6.x-Einsprung) um die Bankumschaltung kümmern.

Die Z80 zwar nur 256 verschiedene Port-Adressen ansprechen, aber i.a. genügt das allemal. Über Port-Zugriffe kommuniziert die Z80 i.a. mit 'Geräten', die in der Regel mehr oder weniger eigene Intelligenz und Hardware haben, z.B. serielle Schnittstellen oder Floppy-Controller. Viele CPUs bieten garkeine besonderen Port-Zugriffe; bei denen müssen dafür bestimmte Speicheradressen geopfert werden.

> Und nun kommt der KNACKPUNKT: Die Z80-CPU kann sich Befehle nur per Speicherzugriff holen (mal von Interrupt-Acknowledge abgesehen).

Soll die Z80 ein Programm ausführen, muß dieses daher im Hauptspeicher stehen, d.h. dort, wo die CPU per Speicherzugriffsbefehl dran kommt.

Die SRAM-Floppy hingegen wird (übrigens wie auch Disketten und Festplatten) über Port-Befehle angesprochen und Daten transferriert. Die Eigenintelligenz der SRAM-Floppy ist übrigens die, daß sie einen internen Zähler hat, so daß die Z80 nach Übermittlung der Spur/Sektor-Angabe den gewünschten Sektor lesen bzw. schreiben kann, indem sie 128 Port-Zugriffe macht; d.h. sie muß nur angeben, welchen 128-Byte-Block sie will und das Adressieren der 128 Bytes auf der SRAM-Floppy macht diese selbt. Da der Zugriff auf die SRAM-Floppy nur über Port-Befehl erfolgt, kann **kein Programm** direkt auf der SRAM-Floppy (oder Disketten

H a r d w a r e: Speicher

Unter RAM 6.x wird der Hauptspeicher (also die üblichen 768kB) für das aktuelle CP/M-Programm (die sog. TPA), das Betriebssystem (ZCPR, P2DOS und RAM 6.x), die interne RAM-Floppy A: und den KLIX-Heap verwendet. Das einzige, was nicht im Hauptspeicher stehen muß ist die RAM-Floppy A:.

KLIX-Overlays sind nämlich Programme (ich weiß, daß Du das auch weißt), die einmal in den Hauptspeicher geladen und dann bei Bedarf auch genau dort, wo sie hingeladen wurden laufen. Das einige Konsequenzen:
1. Da das Programm erst mal Platz im Hauptspeicher benötigt, muß dieser für das Programm besorgt und reserviert werden (RAM-Einsprung GetBlk).
2. Erst wenn bekannt ist, wo dieser Speicherblock liegt (d.h. an welcher Adresse), ist bekannt, wo das Programm laufen soll. Da Programme üblicherweise nicht an jeder beliebigen Adresse laufen müssen die Adressen im Programm erst mal angepaßt werden.
3. Nun steht das Programm irgendwo im Hautpspeicher (genauer auf irgend einer Bank außer Bank 0 und 1, die ja beide für CP/M-Programme und RAM 6.x gebraucht werden, im sog. Heap) und kann aus dem KLICK heraus aufgerufen werden.

Der Witz der KLIX-Overlays ist ja gerade, daß die Programme einmal geladen werden, und dann jederzeit aufgerufen werden können ohne das laufende CP/M-Programm deswegen abbrechen zu müssen oder zu stören.

Folglich müssen die KLIX-Overlays, wenn sie laufen sollen, entweder dann in den Hauptspeicher geladen werden (was Zeit kostet) oder schon dort stehen (was die Regel ist).

Zu folgenden Wünschen möchte ich daher hier vorerst endgültig Stellung nehmen:

F: Warum KLIX-Overlays auf einer SRAM-Floppy und im Heap?
A: Weil sie auf der SRAM-Floppy nicht laufen können! Sie müssen in den Hauptspeicher! Wann und von wo Du sie lädst ist egal.

F: Warum können KLIX-Overlays nicht erst bei Bedarf geladen werden?
A: Olaf's LOADER ist im Source auf einer KLICK-PD! Wer mag, kann ihn sich gerne als KLIX-Overlay schreiben, welches dann ein KLIX-Overlay auf Wunsch lädt. Diese Version des LOADERS kann sich ja auf alle KLICK-FTasten legen um beim Drücken einer noch unbelegten aktiviert zu werden um was-auch-immer zu laden.

F: Warum können sich die KLIX-Overlays nicht gemeinsame Datenbereiche teilen?
A: Einige KLIX-Overlays brauchen Ihren Datenbereich dauerhaft (z.B. MTX-Edit), andere zumindest auf der eigenen Bank. Bei diesen ist's schon schlecht mit dem Teilen. Braucht ein KLIX nur zeitweilig Speicher, sollte es sich diesen dann per GetBlk holen und später per FreBlk freigeben. Einen Automatismus dafür wäre kaum sinnvoll - evtl. sogar garnicht - machbar.

F: Kann A: aus dem Kern-Speicher verschwinden?
A: Theoretisch JA, aber das werde ich nicht tun! RAM 6.x ist schon komplex genug! Jetzt noch derartige "Schmankerl" einzubauen habe ich nicht vor - das würde RAM 6.x deutlich verkomplizieren - und das ist es eh schon. Außerdem gehen einige der zu RAM 6.x gehörenden Programme davor aus, daß A: die interne RAM-Floppy ist - die müßten dann auch geändert werden.
Du kannst die interne RAM-Floppy ja auf 2kB Nutzgröße redizieren - wenn Du diese 2 kB nicht über hast, kann ich Dir nicht helfen. Sinnvolles Minimum für A: ist jedoch 16kB, wenn Du mit SUB arbeitest.
Mein Arbeitslaufwerk ist eine 2MB-DRAM-Floppy G:, A: ist 48kB groß und beim KLICK-Warmboot wird automatisch G: angewählt - dazu gibt's die ja Möglichkeit, bei der Installation von RAM 6.x eine F-Taste zu installieren, die beim KLIX-Warboot auszuführen ist.