MTX User-Club Deutschland

Info 36
15.02.1990

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	Christian Löhrmann	Hans Gras
Sonnenau 2	Grevenbleck 24	Statenhoek 49
2000 Hamburg 76	3005 Hemmingen 1	NL 1506 VM Zaandam
(040) 200 87 04	(0511) 41 78 77	(0031-75) 17 49 91

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

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

C l u b:
Clubtreffen Seite 1
d B a s e
Tastaturpuffer Seite 2
S o f t w a r e
Macken in DiJey Seite 2
L A N Seite 2
Wortmanager Seite 3
MENU 6.0 Seite 4
PROPRINT - PROFORMA Seite 5
FLUSH Seite 6
Neues auf PD Seite 7
Public-Domain CLUB.8xx/9xx Seite 8
R A M
RAM 6.0 Seite 9
CMDRUN.COM Seite 12
T u r b o - P a s c a l
Chain ohne Verlust des Drive/User Seite 20
Assembler-Listing von hinten drucken Seite 22
dynamische Arrays Seite 24
A s s e m b l e r
Programmier-Tips Seite 27
Stringvearbeitung per Macro Seite 30
N e w W o r d
Großes Bildschirmformat Seite 33
W o r d S t a r
Großes Bildschirmformat Seite 34
Was es ist und was es kann Seite 34
Befehlsübersicht Seite 35
H a r d w a r e
OMTI Seite 2
Klemmer Seite 37
Umbauer Seite 37
Maus Seite 37
Platinen in die FDX Seite 38
Zeichensätze für 80Z Seite 38
Megabytes Seite 42
c ' t
wo wollen die hin Seite 39
K o m i k
Phasen der RAM 6.0-Entwicklung Seite 44
Ein Arbeitstag Seite 45
Understanding the Technology Seite 46

Preis für dieses Info: DM 11,10

Clubtreffen (Herbert zur Nedden, 2000)
Als Termine kommen der 21. ~~und der 28~~. April in Frage. Es soll Samstag vormittags beginnen und bis Sonntag Mittag gehen - natürlich mit Übernachtung vor Ort.

RAM 6.x (Herbert zur Nedden, 2000)
Es läuft mit einer neuen Version von P2DOS und ZCPR 3.3. Allerdings fehlt noch das eine oder andere (Install, Teile der Dokum . Untersuchung diverser ZCPR 3.3-Tools). In Kürze wird es an die Tester gegeben, damit wir ein einigermaßen vernünftig getestetes System herausbringen.

Wordstar 4.0 (Herbert zur Nedden, 2000)
Ich habe dieses Teil für DM 250.- erstanden. Incl. gutem englischen Handbuch. Es kann mehr als NW, u.a. Proportionaldruck auf geeigneten Druckern und Rechnen.

Anzeigetexte samt Absender bitte schriftich an Herbert zur Nedden!

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

Herbert zur Nedden, Sonnenau 2, 2000 Hamburg 76, 040 - 2008704:
--> Interessiert Dich einer der von mir angebotenen Posten: Mach ein Angebot!
Meine hier genannten Preise sind nicht unbedingt unumstößlich!

- Lizensierter IBM PS2 Modell 50 Nachbau von Nomarell, 40MB-Festplatte 11 ms, RS 232, Centronics, Mausanschluß, Maus, 1MB RAM, VGA-S/W-Bildschirm, Seikosha 9-Nadel-Drucker SP120, große Tastatur (abgesetzter Cursor-Block), MSDOS 3.3, Neupreis über 12.000,- für VB 6500,-

- 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 arbeiteen.
- Ich vermittle jederzeit gebrauchte/neue Geräte und Teile der selben.
- MTX/FDX-System: DM 450.-
- Ich habe RAMs für 768kB-Erweiterung!
- Memotech Matrixdrucker DMX80: DM 350.-
- Panasonic-Drucker KXP 1091, d.h. eigentlich ein DMX80, jedoch mit Dip-Schalter für Papierlänge und NLQ-Schrift, IBM-tauglich: DM 450.-
- Uwe-Grass-ECB-Adapter als Platine mit zwei Steckplätzen: DM 55.-
- ECB-Buskarte, 10 Steckplätze, 5 bestückt mit kl. Netzteil: DM 50.-
- Elektor-ECB-Buskarte, 8 Steckplätze (alle bestückt): DM 50.-
- Disketten-Laufwerke: (gebraucht)
  - EPSON SMD 100: 3 1/2", einseitig, 40 Spur: DM 20.-
  - EPSON SD 560: 5 1/4", 80-Spur, HD-Lw., d.h. Formate 1A, 1B, aber nicht 03, 07, 09, 0A usw. DM 100.-
- NewWord Version 2.02 DEUTSCH, im Tausch gegen Original-NW-Diskette: DM 80.-
- NewWord Version 2.16, im Tausch gegen Original-NW-Diskette: DM 80.-
- Original Borland Turbo-Modula 2-Lizenz!, Handbuch, angepaßt auf MTX DM 200.-
- Original Microsoft-BASIC-Lizenz incl., M80/L80 Assembler/Linker, Handbuch, auf dem MTX lauffähig: DM 350.- (Microsoft liefert dieses Teil nicht mehr aus!)
- NonDoc-Editor von ComFood für CP/M incl. Source, zeilen- und bildschirmorientiert, Macro-Fähig, mit Handbuch im Ringordner: VB DM 30.-
- Z80ASM (der Z80-Assembler aus Club-PD): Listing und Handbuch auf Papier, ca. 1.4 cm DIN A4: VB DM 10.-
- Extended FDX-BASIC von Andreas Viebke mit div. Tools und Handbuch: DM 20.-
- 4 Stück 8-Zoll-Disketten: DM 20.-
- Einbau-Drehspulmeßgerät 0-50uA: DM 10,-
- Bücher
  - Mikroprozessor Interface-Techniken, A. Lesea/R. Zaks, 425 Seiten: DM 30.-
  - Vom Umgang mit CP/M, Bernd Pol, 376 Seiten: DM 30.-
  - CP/M Anwenderhandbuch, Thom Hogan, 303 Seiten: DM 25.-
  - Programmierung des Z80, Rodnay Zaks, 606 Seiten: DM 45.-
  - Software Schnellkurs SuperCalc, Wolfgang Maaß, 143 Seiten: DM 15.-
  - 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.-
- 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
  - Platinenstecker für Erweiterungen links am MTX-Grundgerät. Natürlich mit dem Gegenstück zu der Kerbe an Pin 5. je DM 4.-
  - Original-Memotech-Spielecassetten: Toado, Kilopede, Knuckles, Draughts, Reversi, Snappo, Blobbo, Utilities, Demo, StarCommand je DM 4.-;
  - 20 Leercassetten, mindestens C10 mit 2 Dosen für je 5 Cass: Zusammen DM 10.-
  - 10 Disketten FUJI HD 5 1/4", gebraucht & o.k. DM 35.-, Originalverpackt: DM 60.-
  - 10 Disketten 3M DD 5 1/4", gebraucht & o.k. DM 25.-

Liebe Leserin, lieber Leser,

die Öffnung der Grenzen macht auch vor dem MTX User-Club Deutschland nicht Halt. Wir haben ein neues Mitglied aus der DDR: Horst Schwäricke, Bergstr. 29, DDR-1509 Michendorf/Potsdam. Er hat DM-Einkommen kein und sucht Kontakt mit einem interessierten Clubmitglied zwecks Informationsaustausch. Er hat übrigens einen MTX 500. Wenn Du ihm etwas schicken magst, wird er sich sicherlich sehr freuen.

Dieses Info ist mal wieder etwas dicker, d.h. braucht DM 2.40 Porto. Daher habe ich gleich auch noch die eine oder andere Seite Komik hinten beigefügt - bei den letzten Infos mußte ich mit der Seitenzahl knausern, da ich wegen nur ein bis zwei Seiten nicht zusätzliche DM 0.60 Porto verbraten vollte.

In c't 2/90 verspielte Andreas Stiller in seinem Editorial viel Sympathie, wie Du aus dem enstprechenden Beitrag hinten in diesem Info entnehmen kannst. Es war schon erschreckend zu lesen.

Da meine Frau und ich vor haben, in Kürze mit dem Bau eines Hauses anzufangen (mir graust schon vor den Kredit-Zinsen!), möchte ich Euch um Nachsicht dafür bitten, daß ich vermutlich bis Ende des Jahres etwas weniger Zeit für den MTX User-Club Deutschland haben werde. Das Info wird es natürlich trotzdem geben, brauche aber etwas mehr Unterstützung in Form von Beiträgen, wenn das Info nicht dünner werden soll. Bitte schickt Eure Beiträge möglichst auf Diskette. Wenn der Beitrag ein Turbo-Pascal-Programm enthält, laßt bitte die eckigen und geschweiften Klammern drin. Ich drucke dann mit dem US-Zeichensatz. Ich finde, daß sich die (. und .) schlecht lesen lassen.

Auf dem CLUB-PD-Sektor hat sich einiges getan, was Ihr in diesem Info auch lesen könnt. Den KLICK-PD-Bereich haben Olaf und ich vernachlässigt, da wir uns um RAM 6.x gekümmert haben. Übrigens laufen alle KLICKer außer meinem Maustreiber und Michael Keßlers FORMSTAR auch unter RAM 6.x. Der Maustreiber ist ein 'Schweine'-Programm, was er auch sein muß, da er Zeichen in den Tastatur-Puffer stellt. FORMSTAR werden wir daher in geeigneter Version mit RAM 6.x ausliefern.

Euer Herbert

Clubtreffen

(Herbert zur Nedden, 2000)

Nachdem ich diverse Anmeldungen zum Clubtreffen erhalten habe, kann ich Euch folgende Termine anbieten: 21. ~~und 28.~~*) April. Beginn am Samstag Vormittag, so gegen 10:30 - 11:00 Uhr, Übernachtung vor Ort, also mit gemütlichem Beisammensein am Abend, Ende am Sonntag mittags. Wir versuchen zu klären, ob es mit der Übernachtung klar geht. Wenn dann alles konkret ist, erhältst Du, falls angemeldet eine Einladung mit der Aufforderung bis 2 Wochen vor dem Termin definitiv zu- oder abzusagen (um die Reservierung der Betten machen zu können). Damit Ihr evtl. Fahrgemeinschaften organisieren könnt, hier die Namen derer, die sich schon bei mir gemeldet haben, sortiert nach Postleitzahl:

B-9440 E. d'Hondt	CH-4303 Kaiseraugust	NL-2032 E. Kaschubinski
2000 H. zur Nedden	3300 H. Hansen	8520 R. Kirchhoff
2050 O. Krumnow	5138 R. de Wolff	8520 H. Oppmann
3000 D. Stobbe	5270 H. Traber	8520 G. Witzel
3070 C. Romanazzi	8039 U. Hönisch	8630 H. Göbel
3100 H. Ahrensfeld	8502 P. Lang	

*) Da 28. fällt weg wegen Platz im Fast. hof.

Übrigens: Bitte keine Raubkopien zum Clubtreffen mitbreingen!

D i e s e s u n d J e n e s

Tastaturpuffer unter dBase? (Dr. Holger Göbel, 8630; 09561/15131)

Olaf hat ja für TURBO-Pascal einmal die Variable CBREAK beschrieben. Wenn diese auf 'false' gesetzt wird, kann ein Programm nicht mehr mit ^C unterbrochen werden, und, was besonders angenehm ist, die Tastaturabfrage greift auf den RAM4-Tastaturpuffer zu (Ähnliches scheint übrigens die Compiler-Direktive (*$C-*) zu bewirken). Man kann also schon weiterschreiben, solange der Computer noch mit etwas Anderem beschäftigt ist.
Nun meine Frage: Gibt es so etwas auch für dBase?

Macken in DiJey (Herbert zur Nedden, 2000)

DiJey hat sicherlich die eine oder andere harmlose Macke, wie z.B. daß unten die beiden Zeilen beim Abbruch einer Funktion nicht immer vollständig wiederhergestellt werden. Lösche ich z.B. mit AL N alle Dateien und breche das mittendrin mit <BRK> ab, ist die Datei-Übersicht leer - die Dateien jedoch nicht alle weg. Wechsle ich das Laufwerk scheint die Anzeige des freien Platzes auf der Disk nicht zu stimmen; hier hilf das Kommando Z, um das Laufwerk neu anzumelden (wie das ^C unter CP/M). Aber zwei echte Macken sind mittlerweile auch bekannt: Bei Formaten mit einem Skew macht DiJey Mist, da er diesen einfach beim Lesen des Direktory ignoriert. Nach einem Rename auf einen anderen User solltest Du unbedingt das Laufwerk mit Z neu anmelden, sonst ...

RING und LAN (Herbert zur Nedden, 2000)

Zum Vernetzen von mehreren Memotechs ist das RING-ROM von Memotech nur für reine Grundgeräte interessant. Hans Gras hat ein LAN mit nur einem Memotech realisiert (nicht, daß das besonders sinnvoll ist, aber er hat nur einen). Dieses läuft unter MTX-Betrieb, FDX-BASIC mit 40 oder 80 Zeichen und unter CP/M wenn die FDXB 40-Version mit DISC QUIT verlassen wird. Dann läuft auf 80 Zeichen CP/M und das LAN in Interrupt auf dem VS 4. Was noch fehlt ist die Steuerung von der CP/M-Ebene aus und ein zweites ROM-Pack. (Wer hat das für Hans Gras?).

Unter RAM x.x könnte das LAN ja vielleicht in einem KLICK-Overlay unter Interrupt-Steuerung laufen.

OMTI (Herbert zur Nedden, 2000)

Unsere (Gerhard Witzel und ich ein wenig) ursprünglich entwickelte FDX-OMTI-Adapterkarte hat leider kleine Design-Fehler (von mir) dank falscher Informationen aus der c't: DACK3 (OMTI-Pin B15) muß an +5 Volt statt an Masse, AEN (OMTI-Pin A11) soll an Masse, der OMTI 5520A benötigt im Gegensatz zum 5510 +12 Volt, die +12 Volt gehören nicht an OMTI-Pin B12, sondern an OMTI-Pin B9.

S o f t w a r e: Wortmanager

Wortmanager 1.9 (Claudio Romanazzi, 3070)

Jetzt ist es geschafft. Man glaubt gar nicht, was es bedeutet, nur ein paar Bibliotheken auseinanderzunehmen, und dann, wie versprochen, eine große daraus zu machen. Da habe ich doch glatt ca. zwei Wochen daran gesessen. Dafür sind, sozusagen als Beute, etwa 5000 Wörter dazugekommen. Am Anfang dachte ich ja, man könnte die Bibliothek verdoppeln, doch es scheint, es gibt gar nicht so viele deutsche Wörter, auch wenn sie in ihren verschiedenen Formen der Beugung und Deklination gespeichert sind. Das will ich an einem Beispiel verdeutlichen. Vor Jahren übersetzte ich einmal das Doc zu der Programmiersprache REC für unsere Public Domain. Dieses Textfile ist 126 Kb kurz und ich dachte mir, laß doch den neuesten Wortmanager mal darauf los. Ergebnis: Ich konnte 'nur' 155 neue Wörter für Wortmanager gewinnen. Dafür entdeckte er bestimmt gut viermal soviele Rechtschreibefehler (pfui, wie unfein). Die Zeit zur Durchleuchtung betrug 56 Minuten, incl. Fehlerkorrektur und Verwaltung der neuen Wörter. Wortmanager zählte 14500 Wörter. Ich jedenfalls war's zufrieden und bis jetzt habe ich nur gehört, 'ohne komme ich gar nicht mehr aus' und ' trotz Korrekturlesens findet Wortmanager immer noch etwas' usw. Zum Schluß noch eine kleine Statistik. Beim Zusammenbau der neuen Bibliothek interessierte mich die Wortverteilung auf die einzelnen Buchstaben. Hier ist sie:

Nach Buchstaben			nach Häufigkeit		
A	=	6111	A	=	6111
B	=	4516	S	=	5393
C	=	192	B	=	4516
D	=	1985	G	=	4514
E	=	4265	E	=	4265
F	=	2306	V	=	3594
G	=	4514	K	=	3320
H	=	2473	H	=	2473
I	=	1094	F	=	2306
J	=	352	W	=	2248
K	=	3320	M	=	2105
L	=	1166	P	=	2089
M	=	2105	U	=	2074
N	=	1159	R	=	2050
O	=	504	Z	=	2024
P	=	2089	D	=	1985
Q	=	118	T	=	1548
R	=	2050	N	=	1159
S	=	5393	L	=	1166
T	=	1548	I	=	1094
U	=	2074	Ü	=	701
V	=	3594	O	=	504
W	=	2248	J	=	352
X	=	0	C	=	192
Y	=	1	Ä	=	178
Z	=	2024	Q	=	118
Ä	=	178	Ö	=	85
Ö	=	85	Y	=	1
Ü	=	701	X	=	0

Bibliotheken-Gehalt an Wörtern ist etwas mehr als 58000 (nach der Statistik habe ich weitere Wörter aufgenommen, deshalb die Differenz). Die mittlere Zugriffsgeschwindigkeit beträgt 2.3 Sekunden pro Wort, ist also etwas schneller geworden. Alle inzwischen noch aufgetretenen Fehler sind beseitigt, sowie Wünsche der Benutzer inzwischen verwirklicht. So gibt es bereits eine französische Version und eine belgische wird es geben. Die Dokumentation ist auf 36 Kb angewachsen, alle wollen halt alles wissen.

S o f t w a r e: MENU 6.0

Wie bereits erwähnt, wird dieses Programm nicht weiter gepflegt. Es wird in ein größeres und umfassenderes Programm, Wortmanager 2.x und höher, einfließen. Die Bibliothek wird umgebaut werden müssen, um Trennungsstellen für eine eingebaute Silbentrennung zu ermöglichen. Ich hatte zwar einen guten Algorithmus für die Silbentrennung, doch man kann sich krummlegen, es kommen keine 100% heraus. Deshalb habe ich diesen Weg gewählt. Außerdem habe ich noch Ideen, wie die Bibliothek verkürzt, d.h. noch weiter komprimiert werden kann. Dadurch wird alles noch etwas schneller. Außerdem plane ich das automatisierte Kreuzworträtsel. Geeignete Wörter müssen auch noch irgendwie gekennzeichnet werden.

Claudio

PS.
Beim Testen von 1.9 habe ich doch glatt noch 8 Fehler gefunden. Dabei habe ich mir solche Mühe gegeben und wollte es erst gar nicht durchlaufen lassen. Doch peinlich wäre es, findet die Fehler jemand anders!

MENU 6.0 (Claudio Romanazzi, 3070)

Peter Würfel hat mich angeschrieben und Vorschläge und Wünsche geäußert, die ein zu Ram 6.0 passendes Menu betreffen. Ihr alle habt bestimmt seine phantastische Oberfläche in einem der vergangen Infos gesehen. Da mein allseits bekanntes und wohl von niemand außer mir benutztes Menu 2.0 nur wenig geändert und verbessert werden müßte, werde ich Menu 6.0 schreiben (nach Erscheinen von RAM 6.0). Wer außer Peter noch Wünsche hat, Vorschläge etc., auch Kritik ist stets willkommen (man will ja doch nur das beste), der sollte mir in den nächsten Wochen schreiben. Nach Fertigstellung bekommt Peter eine Vorversion zum Testen. Er ist derjenige, der Menu am intensivsten nutzt. Danach wird es auf einer Klick-PD zu finden sein (weil doch einige Systemkenntnisse schon zum Bedienen von Menu 2.0 erforderlich sind).

Anm.d.HzN.: Zu ZCPR 3.3 gibt es ein MENU 3.x, welches sich als SHELL in das System einschleift. SHELL bedeutet u.a., daß es sich nicht mehr mit den Dingern wie ZEX und SUB beißen sollte. Claudio hat dieses Teil, so daß er auch die dort drin befindlichen Dinge übernehmen. Ich hoffe, Claudio nutzt dann MENU 3.x als Basis, da dieses ein ZCPR 3.x-Programm ist, welches dich die Informationen über diverse Systemtabellen (wie z.B. die Kommandozeile) aus dem System selbst holt, und nicht mehr installiert werden muß.
MENU 3.x kann u.a. mehere Menu-Dateien gemischt verarbeiten und kann sich im System Dateinamen hinterlegen, die dann in Variablen unter MENU verwendet werden können! So kann ich mir einmal im System merken, ich will mit der Datei XYZ arbeiten und Menu tut das. Später brauche ich die Datei ABC und Menu macht dann die Arbeit halt mit ABC. Leider kann MENU 3.x nicht all das, was Peter gerne haben möchte, so daß Claudio noch etwas tut kann!

WICHTIG: Claudio, bitte schreibe Menü so, daß es intern keine RAM 6.x-spezifischen Dinge benutzt - ich wüßte auch nicht welche! Wer in seinen Menüs hingegen mit RAM-Bildschirm-Steuerzeichen arbeitet macht es richtig! Vor allem diese in MENU fest hinterlegte Geschichte zum Booten von RAM finde ich ziemlich unpraktisch. Immerhin kommt es vor, daß ich mal eine andere Speicheraufteilung brauche, als die übliche - und Menü startet schon von alleine RAM.

S o f t w a r e: PROPRINT - PROFORMA

ProPrint - ProForma (Herbert zur Nedden, 2000)

Meinen Dank an Adreas Viebke, dessen neues Programm PROFORMA es mir ermöglichte, einen so witzigen Titel zu kreieren!

Erst einmal zu PROPRINT:

Andreas war nicht faul und hat sein eh schon tolles PROPRINT auch noch beigebracht, ein Inhaltsverzeichnis mit Seitennummern versehen zu können und es kann auch römisch paginieren! Zur Erinnerung: PROPRINT dient zum Drucken von Texten auf Diabolo-Kompatiblen Druckern (natürlich funktioniert es auch auf echten Teufeln (= Diablolos)!). Toll, meinst Du; das kann auch NewWord! Richtig, nur daß NewWord das nicht mit all den Features wie insbesondere Proportionalschrift, Rechnen ... kann. Auch echte lange Gedankenstriche sind kein Thema. O.K.? Soweit kann WordStar Version **4.0** mithalten; aber PROPRINT schafft das alles auch MEHRSPALTIG mit Proportionaldruck und Blocksatz! Weiterhin kann PROPPRINT rechtsbündig drucken (hallo Datum), echt zentrieren, tabellieren, festlegen wieviel zusätzlicher Platz die Buchstaben bei Fettdruck einrahmen soll, gesperrt drucken u.v.m.

Leider scheint Andreas den Befehlssatz von WordStar 4.0 nicht zu kennen, da er just in seiner letzten Version das Kommando .pr (= Proportional an/aus) auf .ps umbenannt, da weder NewWord noch WordStar dieses Kommando kennen. WordStar 4.0 jedoch kennt eben dieses Kommando - allerdings auch mit der gleichen Funktion.

Wenn PROPRINT nun auch mit anderen Druckern zurecktkäme wäre es nicht schlecht, oder ? Immerhin kennt der Diabolo nicht allzu viele Befehle.

Dazu meinte Andreas jedoch, die Schrittweite mit einer Sequenz Esc 1Eh n und Esc 1Fh n (n = Pixels+1 < 127) sowie den negativen Zeilenvorschub mit Esc 0Ah. Wenn nun ein anderer Drucker hierbei mithalten kann, wäre es grundsätzlich denkbar. Wenn sich aber gerade diese Mimiken ändern, sieht er schwarz, da diese Kommandos sehr stark genutzt werden (u.a. für Druckweg-Optimierung, Hoch/Tiefstellen, Schrittweite, separates (weil dann schnellers) Unterstreichen). Daher machte er mir Hoffnung für den Fall, daß (m)ein Drukker mithalten kann. O.K., falls statt Esc 1E n mehr als zwei Bytes vor dem n gesendet werden müssen, wäre das wohl machbar. Aber das wär's. Das größere Problem bei Matrix-Druckern dürfte der negative Zeilenvorschub sein! Schön wäre das schon - z.B. für schnelle Musterausdrucke auf dem Matrix-Drucker. Da allerdings mein EPSON LQ850 keinen negativen Zeilenvorschub kann .... Aber vielleicht gibt's auch andere Typenrad-Drucker? Ach da war doch noch etwas. Andreas hat den Spooler unter RAM beachtet, und vermeidet wenn es irgend geht, in einer Steuer-Sequenz 0Ch zu senden; außer für den Seitenvorschub an sich! Damit klappt die Einzelblatt-Verarbeitung besser! Also scheint es in der Tat ein Problem zu sein.

Nun zu PROFORMA:

Dieses Programm dient zum Formatieren eines Textes. Dabei ist der Clou dieses Programmes, daß es FUSSNOTEN verwaltet. Im Text werden lediglich die Fußnoten angesprochen und definiert - den Rest macht PROFORMA. Der einzige Nachteil, den ich bei meinem Test dieses Programmes entdecken konnte war die noch nicht vorhandene Dokumentation - die Andreas just am Erstellen ist. Es funktioniert ausgesprochen gut. Viel mehr kann ich noch nicht sagen, außer vielleicht, daß das Programm einiges an (mir noch unklaren) Optionen und Feinheiten bietet, und hervorragend mit PROPRINT harmoniert (watt'n Wunder!), jedoch auch mit NewWord, also insbesondere auch mit Deinem Drucker!

S o f t w a r e: FLUSH

Aber nur um Fußnoten zu Verwalten wäre es wohl zu viel verlangt, ein Programm zu schreiben. PROFORMA kann auch Text-Bereiche (z.B. Tabellen) zusammenhalten und verhindert sog. Witwen und Waisen - das sind alleinstehende Zeilen am Anfang bzw. Ende einer Seite. Andreas ist emsig die Doku am schreiben.

Jetzt noch etwas zu beiden:

Die Dokumentation von PROPRINT ist gut zu lesen und mit einigen Beispielen versehen, die das Einsteigen sowie das Ausnutzen der Fähigkeiten dieses Programmes erleichtern. Dabei hat Andreas Wert darauf gelegt, die Tips für 'Fortgeschrittene' in einen eigenen Teil des Handbuches auszulagern. Ich bin sicher, daß auch die Doku zu PROFORMA diese Qualtität aufweisen wird. Die Installation beider Programme übernehmen diese selbst - und zwar menügeführt!

FLUSH (Herbert zur Nedden, 2000)

Andreas Viebke hat ein Programm für den Public-Domain-Bereich geschreiben, welches eingentlich ein Kopierprogramm ist - jedoch auch einiges anders gleich mit erledigt.

Doch erst mal etwas zu einer weitern Idee in FLUSH: Disketten mit Namen zu versehen ist ja ein alter Hut: Eine Datei, deren Namen mit einem Minus-Zeichen (auch Bindestrich genannt) beginnt liefert den Disketten-Namen. Üblicherweise ist der Datei-Name eine Disk-Bezeichnung, die Extension hingegen eine laufende Nummer. Die CP/M-katalogprogramme basieren just hierauf. (Darum ist die Inhaltsverzeichnis-Datei der Public-Domain-Disketten mit so einem Namen versehen worden!) Irgendwo stieß ich auch auf die Idee, verschiedenen User-Bereichen ebenfalls Namen zu vergeben. Dies erfolgte mit Dateien, deren Namen mit dem #-Zeichen beginnen. Beide Varianten haben als Nachteil, daß dadurch oft so seltsame 0-kB Dateien in der Gegend herumschwirren können, daß die Namen recht kurz (nur 8+3 Zeichen) sind und das die Anzeige dieser Namen wenig fortschrittliche aussieht. Andreas hat richtig erkannt, daß die Bits der User-Angabe im Direktory folgende Bedeutung haben: 0-4 = User, 5 = Zeiteintrag und 7 = Gelöscht-Kennzeichen. Bleibt immerhin Bit 6 noch frei. Und er nutzt Bit 6, um Einträge mit User-Namen zu kennzeichnen. Diese Lösung hat den Vorteil, daß es keine Dateinamen sind, die CP/M 'sieht'. Daher kann er für den Namen immerhin 30 Zeichen (nämlich den Großteil des Dir-Eintrages auf der Diskette) vergeben - muß sich allerdings um die Anzeige dieser Namen selbst kümmern, da CP/M das nicht tut.

FLUSH kann:
- Namen von User-Bereichen einer Diskette auf dieser Speichern und anzeigen.
- Kopieren, und zwar optional ähnlich wie PIP, d.h. indem es erst eine temporäre Kopie auf der Zieldiskette anlegt, und diese erst nach Erfolg des Kopierens umbenennt und damit das alte Original löscht.
- Kopierte Dateien von der Original-Diskette löschen.
- Dateien von mehreren Laufwerken/Usern auf ein Ziel kopieren
- Kennt die CP/M- und die MsDos-Syntax für's Kopieren
- Dateien anzeigen und dabei die Tabulatoren abhängig vom Datei-Typ unterschiedlich expandieren
- Bestimmte Datei-Typen nicht anzeigen
- Bei der Anzeige vor- und rückwärts scrollen
- Bei der Anzeige von Dateien gleichmäßig (unabhängig von der Zeilenlänge) jedoch unterschiedlich schnell scrollen
- Unterstützt unterschiedlich viele Bildschirm-Formate
- Disketten Spurweise kopieren (wie COPYD)
- Format 09 auf Format FB (= 09 ohne Systemspuren jedoch mit 128 Dir-Einträgen) konvertieren!

Natürlich ist FLUSH installierbar und dokumentiert!

S o f t w a r e: Neues auf PD

Neues auf PD (Herbert zur Nedden, 2000)

Die im folgenden kurz erwähnten Programme findet Ihr auf einer der neuen CLUB-PDs. Auf welcher, könnt Ihr der Angebotsliste entnehmen!

CACG 1.0 (Claudio Romanazzi)

Der Kreuzworträtsel-Erschaffer in 'Primitiv'-Form als Baselkiste für unsere Anfänger, natürlich mit .COM-File für die nicht Assembler-mächtigen und .DOC mit reichlich Hinweisen. Die .MAC-Files sind überreichlich dokumentiert, doch ob der Menge vielleicht etwas verwirren.
Anm.d.HzN.: 'Primitiv' bedeutet, daß der PD-CACG keinen Anschluß an Claudios Wortmanager-Wortebibliothek hat.

Komplexes Pascal (Claudio Romanazzi)

Routinen, um mit komplexen Zahlen unter Turbo-Pascal zu rechnen.

Symbolrästel (Claudio Romanazzi)

Kleine Spielerei mit Symbolrätseln, Beispielen und Kernprogramm für eigene Schöpfungen.

Silbentrenner (Claudio Romanazzi)

Dieser Trenner ist besser als die seinerzeit auf PD herausgegebene Pascal-Version. Der Source (.MAC) ist reichlich kommentiert, .DOC ist auch dabei.

NWPDMXD (Wolfgang Dexheimer)

Das Programm NWPDMXD.MAC enthält einen erweiterten Druckertreiber für den DMX80. Der Treiber stammt im Original von Bernd Preusing (Erklärungen INFO 13/29, CLUB-PD.009). Durch die Erweiterung ist es möglich Rollenpapier beidseitig zu bedrukken. Ist die Nummer der ersten auszugebenden Seite (entweder erste Seitennummer des Textes z.B. über .PN eingestellt oder bei der Print Option hinter Anfangsseite angegeben) gerade, so werden nur die Seiten mit gerader Seitennummer ausgegeben, sonst die mit ungerader Seitennummer. Der Drukkertreiber (.MAC) kann wie schon mehrfach im INFO beschrieben übersetzt und in NWPRINT.OVR eingebaut werden. Nach Weihnachten werde ich einen kleinen Artikel fürs INFO schreiben, indem erklärt wird, wie das Ganze geht (Übernahme für andere Treiber möglich).

PLIST, PSTAT (Wolfgang Dexheimer)

PLIST enthält einige Verbesserungen zum unerschöpflichen Thema Pascal-Listings. Das original Programm ist von Dieter Schwarz nach einem CHIP Sonderdruck und befindet sich auf der Club-PD.003. Die Druckersteuerung geschieht über Konstanten, die zwischen Zeile 36 und 47 für den DMX80 eingestellt sind. PSTAT ist unverändert und liegt der Vollständigkeit halber bei.
- PLIST besitzt einen zusätzlichen Zeilenzähler, dadurch dadurch stimmt der Seitenumbruch auch, wenn der Quelltext längere Zeilen enthält.
- Kommentare werden in ITALIC ausgegeben.
- Die wichtigsten Optionen werden angezeigt und können sowohl direkt in der Kommandozeile eingegeben als auch auf Anfrage eingetippt werden.
- Das Datum (aus Adresse $48 bis $4A) und der File-Name (auch Include File-Name) werden im Seitenkopf ausgedruckt.
- Bei Strings wird auf deutschen Zeichensatz (D_BRD) umgeschaltet, um die Umlaute ausgeben zu können, anschließend wieder USA-Zeichensatz (D_USA) für die eckigen Klammern.

S o f t w a r e: Public-Domain CLUB.8xx/9xx

Dia-Programm (Peter Würfel)

Mein Dia-Programm ist nun soweit, daß ich es in PD gebe. Eigentlich hab ichs ja nur für mich gestrickt, aber vielleicht kanns jemand brauchen und außerdem hoffe ich, die eine oder andere Reaktion zu erhalten, denn an diesem Programm werde ich mit Sicherheit weiterbauen. Nur nachdem ich jetzt gute drei Monate mal wieder drangehängt habe (das ist in den letzten Jahren immer meine Herbstbeschäftigung gewesen) habe ich erstmal keine Lust mehr, ich muß jetzt mal das Programm zum Kompostieren ruhen lassen.
Das Programm läuft nicht ohne aufgerüstete 80-Zeichen-Karte (nicht Grafik, sondern die einfache Text-Aufrüstung für 96x28 Zeichen) und ist für einen DMX80 geschrieben. Nur auf Verdacht hin wollte ich keinen anderen Drucker programmieren, aber wenn ich weiß, daß jemand das Programm wirklich nutzen will, werde ich die entsprechende Anpassung gerne schreiben.
Die Programm-Sources und Listings zum Ausdrucken mit NewWord gibts bei mir auf Anfrage.

Public-Domain CLUB.8xx/9xx (Herbert zur Nedden, 2000)

Ich habe in letzter Zeit schlappe 25 MegaBytes CP/M-Software aus den U.S.A. erhalten und zusammen mit Olaf untersucht. U.a. sind wir auch so an den ZCPR 3.3 gekommen. Zu allem Überfluß sind die meisten dieser Programme komprimiert (SQUEEZE oder CRUNCH) und in Bibliotheken (Libraries) oder Archiven. Zu den ersten 20 MB erhielten wir eine Datei von ca. 300 kB mit einer Kurz-Doku (Inhalt was-ist-was), die immerhin über 60% der Dinge erläuterte. Wir haben angefangen, das Zeug zu sichten und auseinandezusortieren, um es dann auf PDs zu packen. Einiges wurde auch gleich gelöscht, z.B. weil drei unterschiedliche Versionen ein- und desselben Programmes dabei waren - da überlebte nur die neueste. Mit diesen Dingen werde ich die CLUB.8xx- und CLUB.9xx-PDs anfangen. Diese PDs kosten dann das selbe wie die SIG/M, CPM-U/G usw., d.h. die Hälfte der normalen CLUB-PDs. Ich wollte keine neue PD-Reihe anzufangen . Mal sehen, wie viel es wird - wenn zu reichlich, werde ich mir evtl. geeignete niedrigere Kosten für Euch ausdenken; sozusagen à la Mengenrabatt.
P.S.: CLUB.8xx/9xx fallen NICHT unter CLUB-PD-Abo!

Programm-'Besprechung' Backitup (Dr. Holger Göbel, 8630)

Backitup von Holger Hansen ist schon toll, nicht ganz glücklich finde ich allerdings die Parameterübergabe: In CP/M ist es üblich (PIP, MFT51 und MDC z.B. machen es so), daß erst das Ziel und dann die Quelle angegeben werden, also:

```
PIP A:QUATSCH.COM = B:QUATSCH.COM.
```

Daran hat man sich gewöhnt, zumal diese Mimik es bei MFT51 erlaubt, mehrere Files anzugeben, also:

```
MFT51 A:=B: QUATSCH.COM *.PAS ANY.*
```

Und gerade diese Möglichkeit vermisse ich. Wenn ich also nicht nur alle *.PAS-, sondern auch alle *.COM-Dateien sichern will, so muß ich Backitup zweimal aufrufen. Das ist besonders deswegen etwas lästig, weil Backitup ziemlich lange braucht, bis es startet (vermutlich deswegen, weil es erst einmal die Directories nach den Datumseinträgen durchforstet). Vielleicht kann man dies doch noch ändern (obwohl ich natürlich weiß, daß MSDOS genau die Schreibweise wie Backitup verwendet - vielleicht ein Grund mehr, es anders zu machen).
Ein Zweites: Soviel ich gesehen habe, verwendet Backitup einen eigenen Bildschirmtreiber, der direkt in das VRam der 80-Zeichen-Karte schreibt und deswegen sehr schnell ist. Bei mir kommen da allerdings machmal die Attribute nicht mehr mit, so daß es schon passiert, daß ganz unmotiviert irgendwo etwas invers gedruckt oder unterstichen ist, auch unter 4 MHz. Vielleicht fehlen ein paar Warte-Takte für die 80-Zeichen-Karte?
Anm.d.HzN.: Holger, spendier mal ein paar Kondensatoren; siehe Info 34, S. 30

S o f t w a r e: RAM 6.0

RAM 6.x (Herbert zur Nedden, 2000)

Olaf und ich sind am WIRBELN!!! RAM 6.0 läuft nun schon seit dem 30.12.1989 als .COM - allerdings mit P2DOS und ZCPR2. ZCPR 3.3 einzubauen erwies sich doch als komplizierter als angenommen - die Doku ist nämlich nicht Public! Und dann erhielten wir auch noch NOVA-DOS, Z80DOS und ZRDOS; NOVA-DOS ist aus SuprBDOS entstanden, welches wiederum aus P2DOS entstand. Klarerweise wollten wir diese DOSse gelich für RAM 6.x nutzen - haben jedoch statt dessen einfach gute Ideen aus allen in P2DOS eingebaut und den unnützen Kram weggelassen. Näheres siehe RAM 6.x-Info.

Das BESTE unseres neuen P2DOS ist dabei ist:

Wurde eine Disk gewechselt, macht das BDOS unter RAM 6.x ganz alleine einen Disk-Reset, statt beim Schreiben den P2DOS Err Disk R/O zu melden!

Im Klartext: ^C ist relativ überflüssig geworden!

Unter RAM 6.x ist es möglich, ein **60k-CP/M** zu haben. Das konnte klappen, da ich zum einen alles (un)Mögliche auf Bank 1 ausgelagert habe und die Umschalterei auf Bank 1 für viele Routinen kompakter kodiert habe. Weiterhin landete der Puffer für physikalische Disketten/Festplatten-Sektoren auf Bank 1 und die CP/M-Tabellen der nicht-aktuellen Laufwerke auf dem Heap. Andererseits benötigt der ZCPR 3.3 einige Tabellen und Datenbereiche auf Bank 0 (2 kB sind da nix).

Die Speicheraufteilung eines 58k-CP/M unter RAM 6.x sieht so aus:

Adresse	was da so landet
0E100h	Beginn des BIOS, d.h. hier steht der BIOS-Jumptable Dahinter kommen Daten von CP/M für Diskettenzugriffe wie z.B. die Allocation-Vektoren etc.
0E2xxh	Free, d.h. Beginn des freien Speicherplatzes im BIOS
0EA80h	Toam, d.h. Ende des freien Speicherplatzes im BIOS Hier beginnt der Bereich mit den Named-Direktories (optional)
0EB80h	Hier beginnen ein Paar Puffer (ShellStack, Message-Buffer, Kommandozeile) des ZCPR 3.3
0ED00h	Hier beginnt der Environment-Block, d.h. der wichtigste Datenbereich des ZCPR 3.3
0EE00h	Hier beginnt das Flow-Command-Package des ZCPR 3.3
0F000h	Und hier beginnt das eigentliche BIOS, d.h. alles bis auf den Jumptable.
0F680h	Mitten im BIOS ist hier das I/O-Package (d.h. der Ersatz für das SYSIO.IO) versteckt.
0F900h	Weiter mit dem BIOS
0FA00h	Auch das Resident-Command-Package hat im BIOS Platz
0FF80h	Weiter mit dem BIOS
FFFFh	

S o f t w a r e: RAM 6.0

Viel mehr (BIOS auf Bank 1) ist nicht rauszuholen, da folgende Bedingungen schon beim Booten des MTX (also vor RAM 6.x) gelten müssen:

- diverse Memotech- und RAM-Einsprünge und -Variablen haben feste Adresse oberhalb von 0F000h, wo sie auch bleiben.
- Die ZCPR 3.3-Bereiche ShellStack, Message-Puffer, Kommandozeile und Environment-Descriptor müssen einen Platz haben, an dem sie bleiben.

Folglich haben wir diese ZCPR 3.3-Bereiche direkt vor 0EE00h gelegt, da ab dort Bernd Preusings RAM-Floppy-Treiber vom Boot-EPROM landen, das also der beste Platz ist.

Doch die schönste Theorie ...

Da gibt es so ein Boot-EPROM von Michael Keßler. Dieses unterstützt bekanntlich Disketten-Formate, für die der Sektor-Puffer 1 kB lang sein muß. Und der liegt dann direkt vor 0F000h (nämlich ab 0EC00h) - wo auch sonst? Und da sollen neue Dinge hin! Unterstützt das Boot-Eprom von Michael auch noch RAM-Floppies, so liegt der RAM-Disk-Treiber sogar davor von 0EB00h bis 0EB80h.
Was tun sprach Zeus! Ganz einfach: Bei Booten patcht das Boot-Bios, welches auf den Systemspuren steht Michaels Disketten-Treiber so, daß der der Sektor-Puffer woanders landet - nämlich weiter vorne! Darum kannst Du, wenn Du ein Boot-EPROM von Michael Dein Eigen nennst je nach Konstellation vermutlich nur ein 58k- oder ein 59k-CP/M laufen lassen. Schade eigentlich, aber leider nicht zu ändern!

RAM 6.x und Farbe

Wer an der 80-Zeichen-Karte einen Farbmonitor betreibt, wird sicherlich bei einigen Programmen an die ostfriesische Nationalflagge (Weißer Adler auf weißem Grund) denken, da schwarze Schrift auf schwarzem Grund selbst bei hochgedrehtem Kontrast nicht zu sehen ist. Die Ausgabe auf die 80-Zeichen-Karte verwendet die Bildschirm-Variable prtatt für das Attribut der Zeichen, die mit diversen Bildschirm-Sequenzen (^F, ^D, ^T, ...) verändert werden kann. Unter RAM 6.x hingegen verwendet der Bildschirmtreiber eine interne Variable für das Attribut, die sich inhaltlich von prtatt dahingehend unterscheidet, daß bei gleicher Vorder- und Hintergrundfarbe Bit 1 invertiert wird. Schwarze Schrift auf schwarzem Grund wird hierbei z.B. Grün. Ich habe Bit 1 genommen, da es beim Monochrom-Schirm keine Bedeutung hat. Somit könnt Ihr bei Farbdarstellung immerhin alles lesen!

Damit Ihr nun nicht glaubt, wir wären untätig, hier mal ein paar Zahlen:

Größe	was es ist
250kB	Dokumentation zu RAM 6.0, noch recht unvollständig!
600kB	Source + .COM von RAM 6.0 selbst
500kB	Source + .COM/.CHN der RAM 6.0-Dienstprogramme (wie FORM6, CFIG6, ...)
400kB	Source + .PR von P2DOS, ZCPR 3.3 und ZCBIOS3 sowie Flow- und Resident-Command-Package

Die letzten 400kB erforderten lediglich, daß bummelig 50kB eingehackt wurden, die anderen Teile kamen zu über 90% aus unseren Fingern.

In dieser Aufstellung sind die ZCPR 3.3-Dienstprogramme nocht nicht enthalten! Außerdem fehlt noch das Installationsprogramm von RAM 6.x und RAM 6.0 selbst fehlen auch noch Kleinigkeiten.

S o f t w a r e: RAM 6.0

Stand der Dinge und Neuerungen

1. RAM 6.0 **läuft.**

2. ZCPR 3.3 **läuft** mit RCP/FCP
   Hurra die Enten - endlich ist auch dieses Teil am laufen .....

3. P2DOS Version 1.3 **läuft** und hat zusätzlich folgende Features
   - Automatisches Laufwerks-Reset bei Schreibzugriffen auf eine gewechselte Diskette. Achtung: Hier kann der Cache etwas dazwischenfunken, daher die Doku zu RAM 6.0 aufmerksam lesen!
   - P2DOS-Errors können mit Esc übergangen werden ...
   - Neue Einsprünge Get-, Set- und UseStamp um die Zeiteinträge von Dateien lesen und setzen zu können. Damit können nun Programme ohne Tricks Zeiteinträge mitkopieren!
   - Die Funktionen Get- und Set-Time sind nicht mehr auf den Funktionsnummern 200 und 201, sondern auf den unter CP/M 3.x und daher von diversen ZCPR 3.x-Programmen und kommerzieller Software wie dem SLR-Assemblern geforderten 104 und 105. (Tip: SLR-Assembler: Adresse der Zeit-Routine = 5 = BDOS!)
   - Die BDOS-Funktion Set Time funktioniert nun auch, so daß DATE4 wieder in DATE zurück-konvertiert werden konnte; dieses nutzt nun nur noch die BDOS-Einsprünge und greift nicht mehr direkt auf die Hardware-Uhr zu.
   - Dateien, deren Name mit einem $ beginnt sind Public, falls Publics gewünscht werden. Dies ist insbesondere für SUBMIT wichtig, da es jetzt egal ist, auf welchem User $$$.SUB steht. Endlich kann ich so in einer .SUB-Datei den User umschalten.

4. Alpha-Lock hat seine unrpsüngliche Funktion wieder zurückbekommen, da u.a. schon einige Memotechler(innen) die beiden SHIFT-Tasten umverdrahtet haben.

Termin

Ich werde immer wieder mal gefragt, wann denn RAM 6.x nun fertig sei. Olaf und ich müssen Euch leider noch ein Bisserl vertrösten! Die ganze Sache ist doch erheblich aufwendiger und umfangreicher geworden, als wir dachten. Hier einige der Punkte, die uns Zeit koste(te)n:

- Wir können nicht zaubern und haben beide einen Job und Freunde (das ist auch gut so!)
- NOVA-DOS, ZRDOS und Z80DOS waren nicht eingeplant.
- ZCPR 3.3 war überhaupt nicht geplant und ist in sich doch um einiges Komplexer in der Installation als ZCPR 3. Obendrein haben wir keine Dokumentation dazu, da diese für US$ 70 gekauft werden müßte.
- Wir haben ZCPR 3.3 und zugehörige Programme in einem Wust von ca. 25 MB Software erhalten, die nicht so übersichtlich wie Public-Domain-Disketten mit Inhaltsverzeichnis-Dateien strukturiert war. Statt dessen bestanden diese 25 MB fast nur aus Libraries mit dort drin steckenden geCRUNCHten Dateien. (200 kB-Library --> 500 kB entkomprimiert!!)
- Wir kannten das Konzept und die vielen Ideen der ZCPR 3.x-Welt noch nicht!

Da es unser Ziel ist, Euch ein zuverlässiges System bereitzustellen, werden wir RAM 6.0 in Kürze drei Leuten zum Testen in die Hand drücken und sind guter Hoffnung RAM 6.0 allerspätestens zum Clubtreffen fertig zu haben.

R A M: CMDRUN.COM

CMDRUN - das was, warum und wie (Herbert zur Nedden, 2000)

Wenn ich unter RAM 4.x (und auch unter RAM 6.x) unter CP/M den Befehl XYZ eingebe sucht der ZCPR erst einmal, ob es einer der eingebauten Befehle wie DIR, ERA usw. ist. Wenn ja, wird dieser auch ausgeführt. Anderenfalls wird die Datei XYZ.COM entlang des Pfades gesucht. Ist diese Suche von Erfolg gekrönt, wird XYZ.COM auch gestartet. So weit so gut - das kann auch das gute alte CP/M.

Jetzt kommt etwas Feines von ZCPR 2 (und auch ZCPR 3.x) ins Spiel. Wird XYZ.COM nicht gefunden, sucht ZCPR das Programm CMDRUN.COM entlang des Pfades (ZCPR 2) oder nur auf dem letzten Laufwerk im Pfad, dem sog. Root (ZCPR 3.3) und falls vorhanden, wird dieses (also CMDRUN.COM) gestartet und mit der ursprünglich eingegebenen CP/M-Kommandozeile (die mit XYZ beginnt) versorgt. Wenn CMDRUN damit etwas anfangen kann - um so besser. So kann ich mittels CMDRUN das Starten von Programmen auf andere ausführbare Dinge erweitern.
Wird z.B. LRUNZ.COM (Library-Run für ZCPR) in CMDRUN.COM umbenannt, können so .COM-Programme direkt aus Libraries (.LBR) gestartet werden. Analog kann auch ein Pascal-Programm als CMDRUN.COM eingesetzt werden, welches .CHN-Dateien sucht und startet.

Mein CMDRUN.COM (Listing s.u., KLICK.009 und RAM 6.0) hingegen ist ein kleines Programm, welches in gewisser Hinsicht universell ist. Es kann nämlich Pascal-.CHN-Programme und .ZEX- und .SUB- und ...-Dateien suchen, und falls gefunden damit eine neue Kommandozeile aufbauen. Findet es z.B. ein .ZEX stellt es 'ZEX ' davor und packt dies in die Kommandozeile; und diese ist nun ausführbar - falls ZEX.COM gefunden wird! Bei Libraries (.LBR) wird z.B. gleich den Library-Manger (bei mir heißt er -.COM) aufgerufen.

Es ist vielleicht naheliegend, erst mal entlang des Pfades nach XYZ.CHN zu suchen (d.h. wenn Du XYZ eingeben hast, und der ZPCR damit nichts anfangen konnte, also CMDRUN startete). Falls gefunden wird XYZ.CHN gestartet. Ist dieses .CHN nicht zu finden, geht die Suche halt mit .ZEX weiter - wieder entlang des Pfades, usw. Das würde bedeuten, daß bei Eingabe eines ungültigen Kommandos immerhin die Laufwerke im Pfad je z.B. vier mal (.CHN, .ZEX, .SUB und .LBR) abgesucht würden. Ach ja, eigentlich könnte CMDRUN auch noch .KLX beachten ... und ...

Folglich mußte eine andere Idee her. Mein CMDRUN sucht nicht nach XYZ.CHN oder XYZ.ZEX usw., sondern schlicht und ergreifend nach XYZ.??? - d.h. ignoriert erst mal die Extension. Wurde CMDRUN fündig, wird in einer Tabelle nachgesehen, ob die gefundene Extension genehm (alos z.B. .CHN, .ZEX, .SUB oder .LBR) ist. Ist die Extension genehm, wird die neue Kommandozeile zusammengestellt, d.h. das Teil gestartet. Ist die gefundene Extension jedoch nicht in der Tabele in CMDRUN, wird auf dem gleichen Laufwerk weitergesucht, da noch ein weiteres XYZ.??? drauf sein könnte (z.B. weil ich erst das .PAS und dann erst das .CHN auf der Scheibe habe). Hat jedoch CMDRUN kein geeignetes XYZ.??? gefunden, nimmt es sich das nächste Laufwerk im Pfad vor.
Dieses Verfahren ist deutlich schneller als die o.g. erste Lösung und wird auch nicht langsamer, wenn noch .KLX und was-auch-immer dazu kommt. Jedes Laufwerk im Pfad wird nur einmal angesehen. Hast Du allerdings ein XYZ.CHN und ein XYZ.ZEX gewinnt der der beiden das Rennen, der zuerst erwischt wird.

Doch wenn schon denn schon: Wurde beim Aufruf ein Laufwerk angegeben (also so etwas wie C:XYZ), wird der Pfad nicht untersucht - dann wird nur das angegebene Laufwerk betrachtet. Wird kein Laufwerk angegeben wird aus Gründen der Bequemlichkeit erst einmal das aktuelle Laufwerk angesehen. CMDRUN ist sogar in Assembler geschrieben - das ist halt kleiner und schneller als Pascal. Soll ein .CHN gestartet werden, ruft CMDRUN mein CHAINDU (siehe Folgeartikel) zum Starten des .CHNs auf. Und da CMDRUN schon weiß, wo (Laufwerk/User) das gesuchte Teil ist, gibt es diese Information auch weiter.

R A M: CMDRUN.COM

Ich schrieb immer wieder von 'starten'. Wie geht das ? Nun ganz einfach. Zu jeder des Extensions (.CHN, .ZEX usw.) ist in CMDRUN hinterlegt, was vor das eingegebene XYZ gestellt werden muß, um es zu starten: .CHNs mit CHAINDU, .ZEX mit ZEX, .SUB mit / (ich habe SUB.COM in /.COM ummbenannt) und .LBR mit -. CMDRUN schnappt sich seine Kommandozeile und packt den entsprechenden Starter (CHAINDU usw.) davor, was mit residenten Kommandozeile unter ZCPR ein leichtes ist. Da jedoch in der residenten Kommandozeile noch weitere Kommandos stehen können - schließlich kann ich mehrere Kommandos getrennt durch Semikoli eingeben, müssen diese selbstredend auch drin bleiben. D.h. das neue erzeugte Kommando wird vorne in die Kommandozeile kopiert, und das was dort drin noch zum Abarbeiten anstand dahinter.

Einen Kompromiß an meine Faulheit habe ich jedoch gemacht: Ich bin (d.h. CMDRUN ist) so frei, einfach anzunehmen, daß die residente Kommandozeile mit ihren über 200 Bytes lang genug ist - schließlich kann ich unter CP/M nur 128 Zeichen eingeben und CMDRUN stellt max. neun Zeichen davor.

Unter ZCPR 2 und RAM 4.x sind die Adressen des Pfades und der residenten Kommandozeile fest. Mit ZCPR 3.3 und RAM 6.0 ändert sich dies drastisch zum besseren. Hier die Mimik für den ZCPR 3.3:

Unter RAM 6.x mit dem ZCPR 3.3 steht im Speicher der sogenannte Environment-Descriptor, und dort drin stehen diverse Adressen wie unter anderem die des Pfades und der residenten Kommandozeile. Bleibt nur die Frage, wo dieser Environtmen-Descriptor steht. Hier hilft uns der ZCPR 3.3 freiwillig und gerne! Beginnt mein Programm in der Form:

```
        jp      Start           ; geht zum Beginn
        db      'Z3ENV',1       ; diese Kennung bittet den ZCPR 3.3, uns
Z3Env:  dw      0               ; hier die Adresse des Env-Descr. zu schreiben
                                ; wenn das Programm gestartet wird.
```

Wenn ich nun weiß, wo was in Environment-Descriptor steht (und das weiß ich), finde ich alle Datenbereiche des Systems, die ich brauche. Und nun kommt der Clou: Läuft der ZCPR 3.3 nicht, dann bleibt der Wert bei Z3Env: Null. Dies nutze ich aus, um zu erkennen, ob mein CMDRUN unter RAM 4.x mit ZCPR 2 oder unter RAM 6.x mit ZCPR 3.3 läuft.

```
; CMDRUN

; Dieses Programm sucht entlang des Pfades nach der genannten Datei -
; die Extension allerdings ignorierend.

; Vor Aufruf der jeweiligen Funktion das akt. Lw. wieder eingeschaltet!

; Je nach der ersten gefundenen Datei, deren Extension in der
; Liste der gültigen Extensions ist, wird das zugehörige Start-
; Programm (z.B. ZEX) vorangestellt, und die Chose gestartet.

Starter         MACRO   Ext,Prog                ; MACRO zum generieren
                local   lex,lpr                 ; der Tabelle mit den
lex:            db      '&Ext'                  ; Extension und den zuge-
                if      ($-lex) ne 3            ; hörigen Startern.
                ds      3-($-lex),' '           ; <- EXT dreistellig!
                endif
lpr:            db      '&Prog'
                ds      9-($-lpr),0             ; <- Kommando 8-Stellig
                ENDM                            ;    plus Ende-Null
```

R A M: CMDRUN.COM

```
PutA            MACRO                           ; MACRO, weil ich zu faul
                ld      (de),a                  ; war diese vier Zeilen
                call    Glotze                  ; zig-mal einzuhacken.
                inc     de
                inc     c
                ENDM

CmdLin          equ     80h                     ; CP/M-Kommandozeile
FCB             equ     5ch                     ; FileControlBlock
Drive           equ     04h                     ; akt. Laufwerk
SearchF         equ     11h                     ; BDOS: erste Suche
SearchN         equ     12h                     ; BDOS: Folge-Suche
SetDma          equ     1ah                     ; BDOS: setze DMA
GetUsr          equ     20h                     ; BDOS: hole User
SetUsr          equ     20h                     ; BDOS: setze User
GetDrv          equ     19h                     ; BDOS: hole Laufwerk
SetDrv          equ     0eh                     ; BDOS: setze Laufwerk
VduOut          equ     0ffd3h                  ; MTX:  Ausgabe auf Glotze

                jp      start

                db      'Z3ENV',1               ; Kennung für ZCPR 3.3
Z3Env:          dw      0                       ; dies füllt der ZCPR 3.3

                ; Dies ist die Tabelle mit den Extensions und den zugehörigen
                ; Aufrufen. Wenn Du willst, kannst Du sie gerne erweitern;
                ; z.B. um etwas wie   Starter <KLX>,<LOAD50>

Tabelle:        Starter <CHN>,<CHAINDU>         ; .CHN -> CHAINDU
                Starter <SUB>,</>               ; .SUB -> /
                Starter <ZEX>                   ; .ZEX -> ZEX
                Starter <LBR>,<->               ; .LBR -> -

; Hier stehen die unter ZCPR 2 (d.h. RAM 4.x) gültigen Adressen der
; benötigten Datenbereiche.

                ; Jeder Eintrag des Pfades beinhaltet zwei Bytes:
                ; das Laufwerk (A: = 1) und den User. Ein '$' bedeutet
                ; nimm das jeweils aktuelle. Das Ende des Pfades kennzeich-
                ; net ein Eintrag, bei dem das Laufwerk = 0 ist.
ZCPRPath:       dw      0e96eh                  ; Pfad
                ; Die residente Kommandozeile hat folgenden aufbau:
                ; Zeiger:  dw  ?   Adresse des nächsten Zeichens, welches
                ;                  abzuarbeiten ist.
                ; MaxLen:  db  ?   max. Länge des Zeileninhaltes
                ; AktLen:  db  ?   aktuelle Länge des Zeileninhaltes,
                ;                  d.h. Anzahl der gültigen Zeichen
                ; Zeile:   ds  ?   Und hier die Zeile selbst, wobei hinter
                ;                  dem letzten Zeichen ein 00h steht.
                ; Der Einfachheit halber habe ich hier gleich zwei dieser
                ; Adressen.
ZCPRCmdLin:     dw      0e983h                  ; ZCPR-Kommandozeile
ZCPRCmdPtr:     dw      0e980h                  ; Pointer dazu

Papo:           dw      0                       ; Path-Pointer
CurrUser:       db      0                       ; Akt. User
CurrDrive:      db      0                       ; akt. Lw.
PathUser:       db      0                       ; User aus Path
PathDrive:      db      0                       ; Lw. aus Path
```

R A M: CMDRUN.COM

```
Start:      ld      a,0c9h
            ld      (100h),a            ; Sperre für GO

            ld      h,(Z3Env)           ; Adresse Env-Descr.
            ld      a,h
            or      1                   ; läuft ZCPR 3.3
            jr      z,NoZCPR3           ; nein, also nehmen wir
                                        ; an, daß ZCPR2 läuft
            ld      de,9
            add     hl,de               ; hier steht Path-Adresse
            ld      e,(hl)
            inc     hl
            ld      d,(hl)              ; DE = Adresse vom Pfad
            ld      (ZCPRPath),de       ; merken
            ld      de,14
            add     hl,de               ; hier steht CmdLine-Adr.
            ld      e,(hl)
            inc     hl
            ld      d,(hl)              ; DE = Adresse CmdLine
                                        ; d.h. Adresse Zeiger
            ld      (ZCPRCmdPtr),de     ; merken
            inc     de
            inc     de
            inc     de                  ; DE = Adr. CmdLine +3
                                        ; d.h. Adresse akt. Länge
            ld      (ZCPRCmdLin),de     ; merken

NoZCPR3:    ld      hl,(ZCPRCmdPtr)     ; Hole Zeiger auf den noch
            ld      e,(hl)              ; abzuarbeitenden Rest der
            inc     hl                  ; Kommandozeile
            ld      d,(hl)              ; DE = Zeiger auf Rest
            ld      hl,OldLine          ; Dort hin Rest retten
            ex      de,hl
GetOld:     ld      a,(hl)              ; Zeichen für Ende-Abfrage
            ldi                         ; LD (DE),(HL)
            or      a                   ; Ende (Zeichen = 00h) ?
            jr      nz,GetOld           ; nein

            ld      e,0ffh
            ld      c,GetUsr
            call    5                   ; hole akt. User
            ld      (CurrUser),a
            ld      c,GetDrv
            call    5                   ; hole akt. Lw.
            ld      (CurrDrive),a
            ld      hl,(ZCPRPath)
            ld      (Papo),hl           ; merke Path-Beginn
            ld      hl,FCB+9            ; Extension im FCB
            ld      (hl),'?'            ; Extension = ???
            inc     hl
            ld      (hl),'?'
            inc     hl
            ld      (hl),'?'
            call    SearchIt            ; Suchen (s.u.)
            ret     nz                  ; Erfolg: Willie go
            ld      a,(FCB)             ; war Lw. angegeben ?
            or      a
            jp      nz,NotFound         ; ja, dann Pfad ignorieren
```

R A M: CMDRUN.COM

```
PathLoop:       ld      hl,(Papo)
                ld      a,(hl)                  ; Lw. aus Pfad
                or      a
                jp      z,NotFound              ; Pfad-Ende: pech
                cp      '$'                     ; aktuelles Lw. ?
                jr      nz,NotAktDrive          ; nein
                ld      a,(CurrDrive)
NotAktDrive:    ld      (FCB),a                 ; Lw. in FCB
                ld      (PathDrive),a           ; und merken
                inc     hl
                ld      a,(hl)                  ; User aus Pfad
                inc     hl
                ld      (Papo),hl               ; Pfad-Zeiger merken
                cp      '$'                     ; aktueller User ?
                jr      nz,NotAktUser           ; nein
                ld      a,(CurrUser)
NotAktUser:     ld      e,a
                ld      (PathUser),a            ; User merken
                ld      c,SetUsr
                call    5                       ; und setzen
                call    SearchIt                ; Suchen (s.u.)
                push    af                      ; Erfolgsmeldung merken
                ld      a,(CurrUser)
                ld      e,a
                ld      c,SetUsr                ; wieder akt. User
                call    5
                ld      a,(CurrDrive)
                ld      e,a
                ld      c,SetDrv
                call    5                       ; und akt. Lw.
                pop     af
                jr      z,PathLoop              ; nicht gefunden: weiter
                ret                             ; gefunden: Starten

; SearchIt sucht die Dateien auf dem eingestellten Laufwerk. Dabei muß der
; FCB schon initialisiert sein, was ein leichtes ist, da der ZCPR das schon
; gemacht hat und CMDRUN Laufwerk und die Extension ??? eingetragen hat.

; Wird eine passende Datei gefunden, wird die Extension in der Tabelle gesucht.
; Bei Mißerfolg wird weitergesucht; bei Erfolg die Kommandozeile aufgebaut.

SearchIt:       ld      de,DMA
                ld      c,SetDma
                call    5                       ; setze DMA
SearchFile:     ld      de,FCB
                ld      c,SearchF               ; erste Suche
SearchFLoop:    call    5                       ; Suche
                ; BDOS liefert entweder Offh = nix passendes da oder
                ; die Position des DIR-Eintrages im Sektor (0 bis 3).
                cp      Offh
                ret     z                       ; nix gefunden: ade
```

R A M: CMDRUN.COM

```
                ; Zum finden der Extension:
                ; DMA-Adresse + (Akku * 32) + 9
                add     a,a                     ; *2
                add     a,a                     ; *4
                add     a,a                     ; *8
                add     a,a                     ; *16
                add     a,a                     ; *32
                ld      hl,DMA+9                ; Extension
                ld      d,0
                ld      e,a
                add     hl,de                   ; gefundene Extension
                ld      de,Tabelle
                ; HL zeigt auf Extension, DE auf die Tabelle
SearchExt:      ld      a,(de)
                or      a                       ; Tabellen-Ende ?
                jp      z,SearchFNext           ; Pech
                ld      b,3                     ; Länge der Extension
SearchELoop:    ld      a,(de)
                cp      (hl)
                jp      nz,SearchENext          ; Extension passt nicht
                inc     hl
                inc     de
                djnz    SearchELoop
                ; wenn ich lier lande, paßt die Extension
                ; also Kommandozeile aufbauen und zur Beruhigung des Benutzers
                ; auch mit dem A> (oder B> ...) davor ausgeben.
                ld      a,0bh
                call    Glotze                  ; Cursor hoch
                ld      a,13
                call    Glotze                  ; und an Zeilenanfang
                ld      a,(Drive)
                and     0fh
                add     a,'A'
                call    Glotze                  ; Lw.
                ld      a,'>'
                call    Glotze                  ; und >  (ergibt A>)
                ex      de,hl
                ld      de,(ZCPRCmdLin)         ; hier steht akt. Länge
                inc     de                      ; und da soll Kommando hin
                ld      c,0                     ; Anzahl Zeichen
PutCmdLoop:     ld      a,(hl)                  ; Kommando-Zeichen aus
                or      a                       ; Tabelle (0 = Ende)
                jr      z,PutCmdOK              ; Kommando fertig
                PutA                            ; In ZCPR-Kommandozeile
                                                ; und auf Glotze sowie
                                                ; Pointer & Zähler: +1
                inc     hl                      ; nächsten Kommandozeichen
                jr      PutCmdLoop

PutCmdOK:       ld      a,' '
                PutA                            ; Leerzeichen dazu
                ld      a,(PathDrive)
                or      a                       ; Drive/User von Pfad ?
                jr      z,PutCmdName            ; nö
                add     a,'A'-1
                PutA                            ; ja, dann Lw. eintragen
```

R A M: CMDRUN.COM

```
                ld      a,(PathUser)
                ld      hl,0ff0ah                       ; h=-1, l=10
CmdUserLoop:    inc     h
                sub     a,l
                jr      nc,CmdUserLoop
                add     a,l                             ; a=a mod 10, h = a/10
                ld      l,a                             ; also a=Einer, h=Zehner
                ld      a,h                             ; nun  l=Einer, a=Zehner
                add     a,'0'                           ; Zeichen draus machen
                PutA                                    ; und in Kommandozeile
                ld      a,l
                add     a,'0'
                PutA                                    ; Einer entsprechend
                ld      a,':'
                PutA                                    ; und der :
                ; nun noch CP/M-Kommandozeile dahinter setzen
PutCmdname:     ld      hl,CmdLin                       ; CP/M-Kommandozeile
                ld      a,(hl)
                or      a                               ; länge = 0
                jr      z,PutLinOK                      ; dann fertig
                ld      b,a                             ; Länge merken
                inc     hl                              ; Längenbyte übergehen
                ld      a,(hl)
                cp      ' '                             ; Erstes Zeichen = ' '
                jr      nz,PutLinLoop                   ; nein
                inc     hl                              ; Leerzeichen weglassen
                dec     b                               ; Länge -1
PutLinLoop:     ld      a,(hl)
                PutA                                    ; CP/M-Kommandozeile rüber
                inc     hl
                djnz    PutLinLoop
PutLinOK:       ld      a,';'                           ; Trennzeichen in CmdLine,
                ld      (de),a                          ; damit Rest dahinter kann
                inc     de
                inc     c
                ld      hl,OldLine
PutOld:         ld      a,(hl)
                push    bc                              ; ldi verändert c!
                ldi
                pop     bc
                inc     c
                or      a
                jr      nz,PutOld

                ld      a,5
                call    Glotze                          ; EOL auf Glotze
                ld      hl,(ZCPRCmdLin)
                ld      (hl),c                          ; Länge neue Zeile
                inc     hl                              ; da beginnt der Text
                                                        ; in der CmdLine
                ex      de,hl                           ; nach DE
                ld      hl,(ZCPRCmdPtr)                 ; CmdLine-Zeiger
                ld      (hl),e
                inc     hl
                ld      (hl),d                          ; Zeiger eintragen
                xor     a
                dec     a
                ret                                     ; Erfolg melden
```

R A M: CMDRUN.COM

```
                ; Hier landen wir, wenn die Extension nicht genehm ist!
                ; Zeiger für nächsten Extension-Vergleich setzen
SearchENext:    inc     hl
                inc     de
                djnz    SearchENext
                ; HL wieder zurück auf Anfang der Extension um weiter in der
                ; Tabelle suchen zu können
                dec     hl
                dec     hl
                dec     hl
                ; DE auf nächste Extension in der Tabelle, also um 9 weiter
                inc     de
                inc     de
                inc     de
                inc     de
                inc     de
                inc     de
                inc     de
                inc     de
                inc     de
                jp      SearchExt

                ; Und hier landen wir, wenn es heißt
                ; 'Suche die nächste Datei auf der Diskette'
SearchFNext:    ld      c,SearchN
                jp      SearchFLoop

                ; Zeichen auf Bildschirm ausgeben, dabei jedoch BC
                ; unverändert lassen
Glotze:         push    bc
                ld      c,a
                call    VduOut
                pop     bc
                ret

                ; Diese Routine schließlich und endlich gibt die unheilige
                ; Meldung 'Watt'n dat ?' aus, wenn garnichts passendes ge-
                ; funden wurde. Die CP/M-Meldung 'XYZ?' war mir zu öde.
NotFound:       ld      hl,Wattn
                xor     a
PrtStr:         ld      c,(hl)
                cp      c
                ret     z
                call    VduOut
                inc     hl
                jr      PrtStr
Wattn:          db      'Watt''n dat ?',13,10,0

                ; Hier ist der DMA-Puffer, d.h. der Bereich, wo das BDOS
                ; den Direktory-Sektor hin kopiert.
DMA:
OldLine         equ     DMA+80h
                end
```

Anm.d.HzN.: Ich habe das Listing hier angedruckt, da es in meinen Augen recht lehrreich ist (insbesondere in Sachen Assembler und BDOS) und aufzeigt, wie ZCPR 3.3-Programme aussehen sollten.

T u r b o - P a s c a l: Chain ohne Verlust des Drive/User

Chain ohne Verlust des Drive/User (Herbert zur Nedden, 2000)

Bekanntlich kosten .COM-Dateien von Turbo-Pascal reichlich Platz auf der Diskette bzw. schlimmer noch auf der RAM-Floppy. Daher ist es eine weit verbreitete Idee, unter RAM 4.x die Pascal-Programme als .CHN zu übersetzen und durch einen geeignetes Programm (welches das .CHN mit Chain(..) aufruft) zu starten.

Dummerweise muß ich aber unter Pascal zumindest den User aktivieren, wo das .CHN ist, damit es dann auch geladen und gestartet werden kann. Und dann läuft das dumme Teil zwar an, nur ist evtl. nicht der User aktiv, wo ich mich eigentlich befinde. Es muß doch eine Möglichkeit geben, dachte ich mir, wie ich den ursprünglichen User (und das Laufwerk) wieder einschalten konnte - natürlich, ohne das .CHN zu ändern.

Folglich habe ich mal mittels MONI untersucht, was eigentlich beim Chain-Befehl von Turbo-Pascal abläuft. Da fand ich u.a. folgendes am Ende dieser Aktion:

```
LD  DE,(0101h)      ; Lade Programm-Startadresse von 0101h
LD  (DMA),DE
....                ; Lade gewünschtes .CHN an die DMA-Adresse
JP  100h            ; Simuliere NeuStart
```

Also habe ich den Sprung bei 100h verbogen, und hier eine kleine Routine eingeschleift, die Drive und User wieder auf den ursprünglichen Wert zurücksetzt. Das was etwas dümmlich ist, ist daß Turbo-Pascal die DAM-Adresse von 101h holt, also mußte ich auch diesen Befehl verbiegen.

Bevor es zum Listing kommt, hier noch ein Paar Informationen:

```
BDOS-Funktion   14   = Selektiere Laufwerk
BDOS-Funktion   25   = Ermittle aktuelles Laufwerk
BDOS-Funktion   32   = Ermittle bzw. Setze User
```

Mein Programm macht folgendes:

- Verbiege den Sprung bei 100h auf meine Routine
- Ersetze den LD DE,(0101h) durch ein LD DE,Adresse
- Schiebe meine Routine nach 105h
- Hole akt. Laufwerk/User und packe diese Information in meine Routine
- Suche XYZ.CHN
- Stelle Rest der Kommandozeile in die selbe
- Chain(XYZ.CHN)

```
PROGRAM ChainDu;            (* NUR FUER TURBO-PASCAL 3.0 *)

PROCEDURE SetDriveUser;     { meine Routine, die nach 105h kommt }
begin
   inline($0E/$0E/          { 105  ld    c,14                    }
          $1E/$00/          { 107  ld    e,0   <- wird gepatcht  }
          $CD/$05/$00/      { 109  call  5                       }
          $0E/$20/          { 10c  ld    c,32                    }
          $1E/$00/          { 10e  ld    e,0   <- wird gepatcht  }
          $CD/$05/$00/      { 110  call  5                       }
          $3e/$01/          { 113  ld    a,01h                   }
          $32/$2b/$1c/      { 115  ld    (1c2bh),a               }
          $21/$E2/$20/      { 118  ld    hl,20e2h                }
          $22/$01/$01/      { 11b  ld    (0101h),hl              }
          $E9);             { 11e  jp    (hl)                    }
end;
```

T u r b o - P a s c a l: Chain ohne Verlust des Drive/User

```
var d:          file;
    i,p,u,j:    integer;
    CPMCmdLine: string[$80] absolute $0080;
    NewLin:     string[$80];

begin
   { JP an Adresse 100h auf meine Routine verbiegen }
   mem[$0101] := $05; mem[$0102] := $01;

   { LD DE,(101h) fuer DMA-Startadresse statt von 101h von $119 }
   mem[$1c2b] := $19;

   { meine Routine nach 105h schieben }
   for i := 0 to 25 do
       mem[$105+i] := mem[Addr(SetDriveUser)+i];

   { akt. Laufwerk und User holen und in meine Routine eintragen }
   { die stellen, die hierbei veraendert werden sind oben        }
   { mit <-- wird gepatcht gekennzeichnet                        }
   i := BDOS(25); mem[$108] := i;
   i := BDOS(32,$ff); mem[$10f] := i;

   if ParamCount = 0 then   { ist ueberhaupt was zu tun }
      halt;

   NewLin := ParamStr(1);      { Name der Datei, die gesucht werden soll }
   { Falls ':' im Dateinamen, ist die erste Stell das Laufwerk           }
   { Dahinter kann evtl. der User stehen - also wird da mal nachgesehen  }
   p := pos(':',NewLin);
   u := 0;
   if p > 1 then
      while (copy(NewLin,2,1) in ['0'..'9']) do
            begin
            u := 10*u + ord(copy(NewLin,2,1))-ord('0');
            delete(NewLin,2,1);
            BDOS(32,u);
            end;

   { .CHN an den Dateinamen anhaengen }
   if pos('.',NewLin) = 0 then
      NewLin := NewLin+'.CHN';

   assign(d,NewLin);    { .CHN-Datei schnappen }

   NewLin:='';   { neue Kommandozeile aufbauen }
   if paramcount>1 then
      for i:=2 to paramcount do
          NewLin:=NewLin+' '+paramstr(i);
   CPMCmdLine:=NewLin;

   chain(d);    { und Chain-en }
   end.
```

WICHTIG: ChainDu erwartet in der Aufrufzeile den Namen des .CHN mit DU: davor, es sucht nicht entlang des Pfades und prüft nicht, ob die .CHN-Datei da ist. Ich habe ChainDu auch so geschrieben, daß es mit meinem o.g. CMDRUN zusammenarbeitet, und CMDRUN liefert die geforderten Informationen!

T u r b o - P a s c a l: Assembler-Listing von hinten drucken

Assembler-Listing von hinten drucken (Herbert zur Nedden, 2000)

Was soll dieser Mist denn nun schon wieder, werden sicherlich einige Fragen! Das ist leicht erläutert! Ich habe einen Drucker mit Einzelblatt-Einzug, der unangenehmerweise die Seiten in der falschen Reihenfolge ablegt, d.h. Seite 2 kommt auf Seite 1, usw. bis die letzte Seite ganz oben auf liegt. Im Rahmen der Programmierung von RAM 6.0 waren hin und wieder Listings von Teilen von RAM 6.0 fällig, die über 60 Seiten lang sind. Da hatte ich keinen Bock auf Sortieren!

Was was naheliegender, als ein Prográmmchen einzuhacken, welches die Seitenvorschübe in der .PRN-Datei sucht und sich merkt und dann beim letzen Seitenvorschub mit dem Drucken beginnt. Anschließend vom Seitenvorschub davor bis zum nächsten usw. - also schlicht und ergreifend folgendes tut:

- Suche alle Seitenvorschübe und merke die Stelle in der .PRN-Datei
  Das Dateiende kennzeichnet ein ^Z im letzten Sektor. Alles was in diesem Sektor dem ^Z folgt wird ignoriert!
- Grase diese Seitenvorschübe von hinten nach vorne ab und drucke bis jeweils zum nächsten.

```
PROGRAM PrtPrn;

var f:      file;                              { Die .PRN-Datei              }
    fn:     string[20];                        { Der Dateiname               }
    Zeiger: array[1..255] of record            { Merker fuer Seitenvorschuebe}
                             sek: integer;     { - Sektor                    }
                             adr: integer;     { - Adresse im Sektor         }
                             end;
    Seiten:  integer;                          { Anzahl Seiten               }
    Sektor:  integer;
    Buffer:  array[0..127] of char;            { Sektor-Puffer               }
    i,j,k,s: integer;

begin
   if ParamCount = 0 then                      { ohne Dateiname geht nix     }
      begin
      writeln('Dateiname fehlt');
      halt;
      end;

   fn := ParamStr(1);                          { Dateiname                   }
   if pos('.',fn) = 0 then
      fn := fn+'.PRN';                         { Default-Extension: .PRN     }
   assign(f,fn);
   reset(f);                                   { Datei oeffnen               }

   Zeiger[1].sek := 0;                         { Erste Position: Dateianfang }
   Zeiger[1].adr := 0;                         { (auch ohne Seitenvorschub)  }
   Seiten := 1;
```

T u r b o - P a s c a l: Assembler-Listing von hinten drucken

```
   for Sektor := 0 to filesize(f)-1 do          { Datei durchsuchen          }
       begin
       BlockRead(f,Buffer,1);                   { Sektor lesen               }
       for i := 0 to 127 do                     { und abgrasen               }
           case Buffer[i] of
           ^Z: begin                            { ^Z = Dateiende             }
               Seiten := succ(Seiten);          { merken und                 }
               Zeiger[Seiten].sek := Sektor;    { Rest des Sektors auf Leer- }
               Zeiger[Seiten].adr := i;         { zeichen damit keine Seiten- }
               for s := i to 127 do             { vorschuebe mehr drin sind  }
                   Buffer[s] := ' ';            { (faule Programmierung)     }
               end;
           ^L: begin                            { ^L = Seitensorschub        }
               Seiten := succ(Seiten);          { merken!                    }
               Zeiger[Seiten].sek := Sektor;
               Zeiger[Seiten].adr := i;
               end;
           end;
       end;
   close(f);                                    { Datei wieder dicht machen  }

   if Seiten < 2 then                           { lohnt es sich zu drucken ? }
      begin
      writeln('Da ist ja nix');                 { nein                       }
      halt;
      end;

   reset(f);                                    { zurueck zum Dateianfang    }
   writeln(Seiten-2,' Seiten werden nun gedruckt');
   writeln;                                     { Drucker-Initialisierung!!  }
   write(lst,^M^['@'^['g'^['A'^F^['1'^H);       { *** SIEHE HINWEIS UNTEN *** }
   write('Seite:    ');

   for s := Seiten-1 downto 1 do                { von hinten nach vorne      }
       begin
       write(^H^H^H,s:3);
       k := Zeiger[s].sek;                      { Sektor und                 }
       i := Zeiger[s].adr;                      { Adresse im Sektor          }
       seek(f,k);                               { Sektor suchen              }
       BlockRead(f,Buffer,1);                   { und einlesen               }
       while (k <> Zeiger[s+1].sek) do          { Falls naechsts ^L nicht in }
          begin                                 { diesem Sektor die Sektoren }
          for j := i to 127 do                  { bis zum naechsten ^L aus-  }
              write(lst,Buffer[j]);             { drucken                    }
          i := 0;
          k := succ(k);                         { aber auch mitzaehlen und   }
          BlockRead(f,Buffer,1);                { einlesen                   }
          end;
       if Zeiger[s+1].adr >= i then             { nun den Teil in dem Sektor }
          for j := i to Zeiger[s+1].adr do      { drucken, in dem das naechste}
              write(lst,Buffer[j]);             { ^L steht bis zu eben diesem }
       end;
end.                                            { FINITO                     }
```

HINWEIS: Mitten im Programm steht mein Drucker-Initialisierungs-String. Dieser stellt den EPSON LQ 850 auf Mikrodruck mit linkem Rand von 8 Zeichen ein. Vermutlich mußt Du diesen writeln(lst,...) auf Deinen Drucker anpassen.

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

TURBO-PASCAL (Dr. Holger Göbel, 8630; 09561/15131)

Beim Programmieren mit TURBO-Pascal (ich arbeite an einem Statistik-Programm) habe ich verschiedene Erfahrungen entweder zusammengetragen oder auch selbst gemacht. Davon möchte ich Euch ein paar mitteilen:

Dynamische Arrays

TURBO-Pascal verlangt in der Variablendeklaration die Festlegung der Größe eines Arrays. Das kann auch hinderlich sein. In meinem Statistikprogramm z.B. soll das Array, das die Daten aufnimmt, so groß wie möglich sein, damit nicht ständig auf Diskette ausgeladen werden muß. Außerdem soll das Array auch variabel sein, d.h., der Benutzer soll die Dimension während des Programmlaufs bestimmen dürfen.
Horst Kupka hat zur Lösung dieses Problems einen CHIP-Artikel ausgegraben. Dessen Idee ist folgende:
TURBO-Pascal kann den Heap variabel verwalten (der Heap ist der freie Platz zwischen Programmende und den Stacks). Also bietet es sich an, ein Array in diesen Heap zu setzen. Der Programmierer ist dann allerdings selbst verantwortlich für die Verwaltung des Arrays. Das sieht so aus:
Zunächst muß festgelegt werden, welcher Art die Elemente des Arrays sind (z.B. Byte, Integer oder Real).
Dann wird ein Schein-Array angelegt (Array[1..1]) und ein Zeiger definiert, der auf dieses Schein-Array zeigt.
Und schließlich wird mit Hilfe der Prozedur GetMem der Zeiger auf den Heap-Anfang gesetzt. So kann auf jedes Element (z.B. eine Real-Zahl) im Heap zugegriffen werden. Wenn das Array z.B. den Namen 'DatenArray' trägt, so wird das siebte Element mit Hilfe des Zeigers so angesprochen: **DatenArray ^[7].**
Im Heap stehen jetzt also hintereinander die Elemente des Arrays. Der Programmierer muß sich die Position eines bestimmten x-dimensionalen Arrays mittels einer Prozedur errechnen.
Damit es nun nicht zu kompliziert wird, wollen wir nur **zwei**dimensionale Arrays betrachten, vergleichbar also einer Tabelle mit S Spalten und Z Zeilen. Ich habe es mir angewöhnt, die Elemente des Arrays spaltenweise anzuordnen (erst Spalte 1, dann Spalte 2 ... bis Spalte S).
Zur Verwaltung eines dynamischen Arrays sind also folgende Vereinbarungen notwendig:

```
TYPE
  Typ            = (Byte_Typ, Integer_Typ, Real_Typ);(*Element-Typ des Arrays*)
  Byte_Array     = ARRAY [1..1] OF Byte;             (*Schein-Array*)
  Integer_Array  = ARRAY [1..1] OF Integer;          (*dto*)
  Real_Array     = ARRAY [1..1] OF Real;
  Byte_Pointer   = ^Byte_Array;                      (*Zeiger auf Byte_Array*)
  Integer_Pointer = ^Integer_Array;
  Real_Pointer   = ^Real_Array;

CONST
  TypSize: ARRAY [0..2] OF Integer = (1,2,6);
                (*Platzbedarf fuer Byte, Integer oder Real*)

VAR
  DimZeile, DimSpalte: Integer;  (*Dimension des Arrays*)
  DatenArray: Real_Pointer;      (*so kann z.B. ein Real-Array heissen*)
```

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

```
PROCEDURE NewArray (VAR Pointer; Element_Typ: Typ; Groesse: Integer);
  (* Array anlegen; Uebergabeparameter: Array-Name;
                                        Typ (z.B.: Real_Typ);
                                        Groesse des Arrays *)
VAR AnyPointer: ^Integer Absolute Pointer;
BEGIN
  GetMem (AnyPointer,TypSize[Integer(Element_Typ)]*Groesse);
END;

PROCEDURE DispArray (VAR Pointer; Element_Typ: Typ; Groesse: Integer);
  (* Array freigeben; Uebergabeparameter wie oben, die Groesse muss
     exakt mit der Groesse aus NewArray uebereinstimmen *)
VAR AnyPointer: ^Integer Absolute Pointer;
BEGIN
  FreeMem (AnyPointer,TypSize[Integer(Element_Typ)]*Groesse);
END;


FUNCTION DatenPos (Spalte, Zeile: Integer): Integer;
       (*Position berechnen*)
BEGIN
  DatenPos := Pred(Spalte)*DimZeile + Zeile;
END;

(**** Beispiel: Wir definieren ein Array namens DatenArray mit den
       Dimensionen DimSpalte und DimZeile, speichern etwas ab und holen
       es wieder hervor  ****)

BEGIN
  DimSpalte := 20;       (* Dimension des Arrays *)
  DimZeile := 10;
  NewArray (DatenArray,Real_Typ,DimSpalte*DimZeile); (*Array anlegen*)

  DatenArray ^[DatenPos(5,3)] := 5;       (* ein Element ablegen *)
  Writeln (DatenArray ^[DatenPos(5,3)]); (* und wieder hervorholen *)
  Repeat until keypressed;                (* zum Anschauen *)
               (* zum Schluss das Array wieder wegnehmen *)
  DispArray (DatenArray,Real_Typ,DimSpalte*DimZeile);
END.
```

Vorher muß natürlich der Programierer sicherstellen, daß der Heap das Array überhaupt aufnehmen kann, was er mit der Funktion MemAvail überprüft.
Mit dem bis jetzt Dargelegten können also Arrays angelegt und entfernt werden, sie sind dynamisch geworden.
Meine nächstes Anliegen allerdings, daß der Heap nicht mehr die Größe eines Arrays begrenzen soll, muß anders gelöst werden.
Die Idee ist zwangsläufig die, daß ein Array auf Diskette angelegt wird und das Array im Speicher quasi nur ein kleiner Ausschnitt des Gesamtarrays ist.
Deshalb muß die Funktion DatenPos für die Diskettenverwaltung sorgen. Die Programmtechnik darzulegen, würde hier zu weit führen. Wenn Interesse besteht, kann ich meine Routinen auf PD zur Verfügung stellen.
Übrigens ist es auch möglich, mehrere Arrays im Heap anzulegen, die alle mit Diskettenverwaltung bearbeitet werden können.

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

Nun zu zwei Einschränkungen:

1. Wenn der Heap zu stark beansprucht wird, laufen rekursive Prozeduren nicht mehr so ohne weiteres, da sie selbst mit ihrem Rekursions-Stack den Heap dynamisch verkleinern. Hier kann man sich allerdings verschiedene Lösungsmöglichkeiten einfallen lassen.
2. Die WINDOW-Routinen von Olaf benutzen ebenfalls den Heap. Da ich inzwischen Programme sehr gerne mit der Window-Technik gestalte, habe ich die Window-Routinen umgeschrieben. Jetzt lagern sie den Inhalt des 80-Zeichen-Schirms in das VS4-RAM aus, hier stehen ja 16 kByte zur Verfügung. Zusätzlich habe ich noch gleich Routinen geschrieben, die ein horizontales Scrollen erlauben (PD).

Noch ein kleiner, 'unsauberer' Trick:
In meinem Statistik-Programm sind meine Daten-Arrays als Real-Arrays angelegt. Sie erlauben einen wahlfreien Zugriff aus Diskette. Ich wollte allerdings auch einige kleine Strings in dieser Real-Datei unterbringen. Glücklicherweise brauchten die Strings nicht länger als 6 Zeichen zu sein, so daß ich den String einfach auf eine Real-Zahl lege, die ja 6 Bytes benötigt. Probleme gab es nur deshalb, weil ein String in seinem ersten Byte immer seine Länge abgelegt hat. Die Lösung sieht so aus:

```
TYPE
  Str6 = STRING[6];
  FilVar: FILE OF Real;
VAR
  StZahl: Real;                (* Die Reihenfolge von StZahl, *)
  Dummy:  Byte;                (* Dummy,                      *)
  RealSt: Str6 ABSOLUTE Dummy; (* und RealSt ist so notwendig *)
  Datei:  FilVar;

BEGIN
  Dummy := 6; (* ist notwendig *);
  Assign (Datei,'TEST');
  Rewrite (Datei);
  RealSt := 'Hallo ';
  Write (Datei,StZahl);
  Close (Datei);
END.
```

Die ersten drei Variablen müssen in der obengenannten Reihenfolge abgelegt sein. Dadurch kommt die Länge des Strings auf Dummy zu liegen und RealSt[1] liegt exakt auf dem Beginn der StZahl, also auf Addr(StZahl). Es ist nun möglich, einen String mit der Länge 6 nach RealSt zu kopieren und ihn dann als StZahl in einem Real-File abzuspeichern.

A s s e m b l e r: Programmier-Tips

Programmier-Tips für den Assembler-Programmierer (Herbert zur Nedden, 2000)

'Vorgeplänkel'

Es ist doch immer wieder verblüffend, wieviele Zeilen Assembler-Source beim Programmieren mit Hilfe einiger (oft sogar einfacher und kleiner) Macros eingespart werden können. Obendrein erhöhen diese Macros die Lesbarkeit des Sources, falls der Programmierer etwas Phantasie bei der Vergabe der Namen der Macros einfließen läßt.
Für die Konvertierung von Binär nach Hex oder Dezimal kann man sicherlich die geeignete Routine aus der SYSLIB nehmen. Wie man das aber selbst programmieren kann ist deswegen nicht minder interssant - vor allem, weil dann das Linken mit der Syslib entfällt.

Definition von Control-Codes

Häufig sind Zeichenfolgen, die an den Bildschirm geschickt werden sollen mit Control-Codes gespickt. Sehen wir uns erst mal ein Beispiel an. Der folgende Text soll ausgegeben werden:

Bitte **Quell-Diskette** einlegen und <RET> drücken.

Bitte beachte, daß 'Quell-Diskette' hell dargestellt werden soll. Zur Erinnerung: auf dem Bildschirm kann hell mittels ^T und normal mit ^R aktiviert werden (ja, es geht auch über ^F und Esc P jeweils gefolgt von einem weiteren Zeichen). Bleibt die Frage, wie deses ^R und ^T im Source kodiert wird. Pascal-Programmierer wissen das: einfach mit ^R und ^T; aber die Assemlber-Fans (wozu ich auch gehöre) neigen i.a. zu einer der folgenden Varianten:

```
            db      'Bitte ',14h,'Quell-Diskette',12h
            db      ' einlegen und <RET> drücken.$'
```

oder

```
   Hell     equ     'T' and 1fh
   Norm     equ     'R' and 1fh

            db      'Bitte ',Hell,'Quell-Diskette',Norm
            db      ' einlegen und <RET> drücken.$'
```

(wobei das $ am Ende für die BDOS-Stringausgabe als Ende-Kennung dient).
Folgendes MACRO erzeugt für diverse Control-Codes Konstanten, wie z.B. Cntl_T mit dem Code von ^T:

```
            IRPC    X,<ABCDEFGHIJKLMNOPQRSTUVWXYZ>
   Cntl_&X  equ     '&X' and 1fh
            ENDM
```

Damit kann der o.g. Text (und viele viele andere entsprechend) kodiert werden:

```
            db      'Bitte ',Cntl_T,'Quell-Diskette',Cntl_R
            db      ' einlegen und <RET> drücken.$'
```

Diese Variante gefällt mir eigentlich recht gut, da ich erkenne, was das für Codes sind (wer weiß schon, was 14h ist) und keine Labels wie Hell, Norm usw. verbrate und mir merken muß.

A s s e m b l e r: Programmier-Tips

ld hl,(ix+Konstante)

Diesen Befehl braucht man doch gelegentlich, z.B. um aus einem Datenbereich wie dem BDOS-DPH eine Adresse (also einen 16-Bit-Wert) zu holen. Folgendes MACRO macht die Chose übersichtlich (Aufruf: LDIX Konstante):

```
; dieses Macro 'emuliert' den nicht vorhandenen befehl  LD HL,(IX+Nr)
LDIX    MACRO   Nr
        ld      l,(ix+Nr)
        ld      h,(ix+Nr+1)
        ENDM
```

BDOS/BIOS-Aufruf:

Mit einem MACRO sieht der BDOS- oder BIOS-Aufruf wesentlich lesbarer aus.

```
Bdos    MACRO   Func
        ld      c,Funk
        call    5
        ENDM

Bios    MACRO   Par
        ld      a,Par
        call    CallBIOS
        ENDM
```

Allerdings muß noch für's BIOS die Routine CallBios dazu:

```
; mehr zum Warum und Wieso dieser Routine findest Du in Info 34, S. 21 ff.

CallBIOS: push  de              ; Merke DE-Register
        dec     a               ; nun ist 0 = WarmBoot
        ld      d,a             ; nach D mit der Funktion
        sla     a               ; A = 2*A
        add     a,d             ; A = A+D = 3*A
        ld      d,0
        ld      e,a             ; nun DE = 3*(FunktionsNummer-1)
        ld      hl,(1)          ; Adresse WarmBoot = Funktion 1
        add     hl,de           ; nun HL = Adresse der Funktion
        pop     de              ; DE zurück
        jp      (hl)            ; und aufi geht's
```

Definiert man noch z.B. folgende Konstanten, sieht der BDOS-Aufruf Disk-Reset bzw. der BIOS-Aufruf zum Wählen eines Laufwerkes wie folgt aus:

```
Reset   equ     0dh             ; BDOS Disk-Reset
SelDrv  equ     09h             ; BIOS Drive-Select

        BDOS    Reset
        BIOS    SelDrv
```

A s s e m b l e r: Programmier-Tips

Konvertierung Byte in HEX-Darstellung:

```
; Hexzahl aus A in Speicher ab HL im Display-Format ablegen
HexToMem: push    af              ; merke A
          srl     a
          srl     a
          srl     a
          srl     a               ; Bit 3-0 = Bit 7-4, Bit 7-4 = 0
          call    HeToMeNib       ; Nibble (= halbes Byte) konvertieren
          pop     af              ; hole A zurück (für Bits 3-0)
          and     0fh             ; Bit 3-0 = Bit 3-0, Bit 7-4 = 0

HeToMeNib:add     a,90h           ; und diese Routine zum Konvertieren
          daa                     ; von 0 -> '0', 1 -> '1', ... 9 -> '9'
          adc     a,40h           ; sowie von 10 -> 'A',  ... 15 -> 'F'
          daa                     ; ist einfach pfiffig und aus einem
          ld      (hl),a          ; Assembler-Buch abgeschrieben worden!
          inc     hl              ; (Sie funktioniert trotzdem!)
          ret
```

Konvertierung 16-Bit-Wert in Dezmial:

Die folgende Routine wandelt die Zahl in HL in eine Dezimal-Zahl um, und gibt sie über die Routine ConOut (das kann z.B. die Bildschirm-Ausgabe sein) mit führenden Leerstellen aus:

```
NumOut:   ; HL dezimal mit führenden Leerzeichen ausgeben
          ld      b,' '           ; erst einmal führende Leerzeichen
          ld      de,10000        ; Größenordnung = 10000
          call    ZifferOut       ; Zehn-Tausender ausgeben
          ld      de,1000         ; Größenordnung = 1000
          call    ZifferOut       ; Tausender ausgeben
          ld      de,100          ; Größenordnung = 100
          call    ZifferOut       ; Hunderter ausgeben
          ld      de,10           ; Größenordnung = 10
          call    ZifferOut       ; Zehner ausgeben
          ld      a,1
          jp      DigitOut        ; Einer ausgeben

ZifferOut:ld      a,0ffh          ; A = -1
ZiffOutL: or      a               ; lösche Carry-Flag
          sbc     hl,de           ; Zahl minus Größenordnung
          inc     a               ; Zähler +1
          jp      nc,ZiffOutL     ; Überlauf ? Nein, dann -> ZffOutL
          add     hl,de           ; Überlauf aufgetreten, wieder weg damit
          or      a               ; A = wie oft Größenordnung in Zahl steckt
          jp      z,NibbleOut     ;     also die Ziffer. Ist diese <> 0?
DigitOut: ld      b,'0'           ; ja, dann nicht mehr führende Leerzeichen
NibbleOut:add     a,b             ; Ziffer + '0' bzw. Ziffer + ' '
          ld      c,a
          jp      ConOut          ; Ausgeben

ConOut:   ; Zeichen in C ausgeben
          ; mußt Du selbst schreiben, z.B. via BIOS oder BDOS
          ; mit dem o.g. Macro
```

A s s e m b l e r: Stringvearbeitung per Macro

Bei der Assemblierung eine Zahl in Hex ausgeben lassen:

```
DISPHEX   MACRO   msg,lab
          .printx /msg lab/        ; Gib msg und lab aus
          ENDM

DISP      MACRO   lab
          .radix  16               ; Zahlenbasis = 16 (also Hex)
          DISPH   lab,%lab         ; Rufe DISPHEX mit lab und Wert von lab
          .radix  10               ; Zahlenbasis = 10 (also Dezimal)
          ENDM
```

Mit dem Aufruf

```
DISP      KarlOtto
```

wird auf dem Bildschirm bei der Assemblierung der Text 'KarlOtto' sowie dessen Wert in Hex ausgegeben. Das ist recht praktisch, um gleich einge Adresse bei der Assemblierung zu erfahren - z.B. für Prüfungen.

Warum so, und nicht anders: Wird ein Macro aufgerufen, und einem Parameter das Zeichen '%' vorangestellt, wird nicht der Parameter, sondern dessen Wert an das Macro üebrgeben. Warum meine Makro-Schachtelung funktioniert? Bitte Raten! Lösung kommt, falls Intersse im nächsten Info!

Stringvearbeitung per Macro (Herbert zur Nedden, 2000)

Aus den USA erhielt ich die Lösung zu einer Herausforderung (d.h. Knobelaufgabe) an Macro-Freaks mitsamt dem Text dieser Herausforderung. Olaf und ich überlegten uns, ob wir erst mal nur die Knobelaufgabe ins Info schreiben sollten damit Ihr mal etwas zu tütfeln habt. In Anbetracht der Lösung haben wir das sein gelassen.

Die Herausforderung (The 'Challenge') (von Ben Grey)

> Ein Macro soll mit einem Parameter, der an erster Stelle evtl. das Zeichen '#' vorangestellt hat, aufgerufen werden. Falls das '#' da ist soll der Rest dieses Parameters auf eine Art und Weise verarbeitet (z.B. in einen DW-Befehl kodiert) werden. Fehlt das '#' soll eine andere Verarbeitung (z.B. ein DB) erfolgen.
>
> Dies ist ein Spezialfall des allgemeinen Problems ein Macro zu basteln, welches den übergebenen String in zwei Teile zerlegt.

Anmerkung (von Herbert zur Nedden)

> Das Problem liegt nicht darin, das erste Zeichen zu erwischen, sondern dieses ggf. vom String anzuknipsen!

Die Lösung (von Bob Freed)

> Unglücklicherweise fehlen den meisten Z80-Assembler (insbesonder dem M80 und dem SLR-Assembler) Stringverarbeitungs-Mimiken, die zur Zeit der Assemblierung greifen. Der SLR-Assember für MsDos bietet hier mit Variationen des EQU-Befehls Möglichkeiten, mit denen die Lösung der 'Challenge' einfach ist. Da diese Feinheiten fehlen, glaube ich, daß ich eine etwas komplexe (und irgendwie unelegante) Lösung der 'Challenge' habe. (Ich kann das Knobeln nicht lassen.)

A s s e m b l e r: Stringvearbeitung per Macro

```
; Diese Lösung ist für den SLR Assembler Z80ASM+ geeeignet. Sie wurde
; nicht mit SLRs Z80ASM oder Microsofts M80 getestet, sollte allerdings
; auch mit diesen funktionieren.
;
; Der Aufgabensteller hat diese Koblelei wegen der intellektuellen
; Kuriosität verbreitet. Er wird sicherlich nicht den Gebrauch so obskurer
; Macro-Nutzung empfehlen. Vor allem kann er einer guten Programmierer-
; Kobelaufgabe nicht wiederstehen.

;
; Vorgeplänkel
;
; Das Macro MSTR erzeugt ein String-Argument für einen Assembler-Pseudo-
; Opcode.
; Das Macro MCHR ergänzt diesen String um das nächste Zeichen, falls der
; Parameter OP = MSTR ist. Anderenfalls erzeugt es den endgültigen Assembler-
; Pseudo-Opcode (der als Parameter OP angegeben wird).
; Der Haupt-Trick hierbei ist, daß MCHR erst durch MSTR generiert.
; Dadurch definiert sich MCHR selbst um, falls es mit dem Parameter
; OP = MSTR aufgerufen wird.

MSTR    MACRO   STR
MCHR    MACRO   OP,CHR
        OP      STR&&CHR        ;; siehe Kommentar unten
        ENDM
        ENDM

; Beachte das Doppel-Ampersand ('&&'). Das erste wird bei der Ausführung
; (d.h. Expansion) von MSTR (d.h. bei der Definition von MCHR) umgesetzt;
; das zweite wird bei der Ausführung von MCHR umgesetzt.

; Die Lösung
;
; Das Macro DWB erzeugt abhängig vom ersten Zeichen des übergebenen
; String STR entweder ein DW oder ein DB. Ist das erste Zeichen das
; '#' wird dieses '#' entfernt und ein DW mit dem Rest des Strings
; als Parameter erzeugt. Anderenfalls (kein '#') wird ein DB erzeugt.

; Zuvor eine Anm.d.HzN.: Der Macro-Aufruf

;       IRPC    CHR,<STR>
;       .....
;       ENDM

; bewirkt, folgendes: (STR ist ein String)

;       for i := 1 to length(STR)
;           begin
;           CHR := i-tes Zeichen von STR
;           .....
;           end;
;
; Der Assembler verwendet die spitzen Klammern <...> als Begrenzer für
; Strings in Macros.
```

A s s e m b l e r: Stringvearbeitung per Macro

```
DWB  MACRO  STR                    ; Beginn der Definition von DWB

FIRST  DEFL  NOT 0                 ; FIRST = false
                                   ; FIRST ist der Merker, ob wir das erste
                                   ; Zeichen des Strings untersuchen

    IRPC  CHR,<STR>                ; String STR Zeichenweise abarbeiten

        IF  FIRST                  ; erstes Zeichen ?
            IFDIF  <CHR>,<#>       ; ist das erstes Zeichen <> '#' ?
                DB  STR            ; DB-Opcode erzeugen
                EXITM              ; Macro IRPC abbrechen, da fertig
            ELSE                   ; wenn erstes Zeichen = '#' ?
FIRST           DEFL 0             ; FIRST = false
                MSTR               ; MSTR aufrufen, also MCHR initialisieren
            ENDIF                  ; Ende IFDIF <CHR>,<#>
        ELSE                       ; nicht erstes Zeichen
            MCHR  MSTR,<CHR>       ; MCHR aufrufen
        ENDIF                      ; Ende IF FIRST

    ENDM                           ; Ende IRPC

    IF NOT FIRST                   ; Wenn FIRST = false (also '#' da)
        MCHR  DW                   ; DW-Opcode mittels MCHR erzeugen
    ENDIF                          ; Ende IF NOT FIRST

ENDM                               ; Ende DWB
```

Anm.d.HzN.: Ich hab's getestet - es funktioniert. Um dieses Teil allerdings zu verstehen sollte es in der Tat einfach mal laufen gelassen werden. Das geht mit Folgenden Pprogramm-Zeilen: (MACRO aber davor definieren!!!)

```
DWB       123                      ; Erzeuge DB 123
DWB       #456                     ; Erzeuge DW 456
```

N e w W o r d: Großes Bildschirmformat

NwSchirm und Newword 2.17 (Jan Bredereke, 2000)

Herberts Erweiterung von NWSCHIRM auf Newword Version 2.17 kann mit dieser Newword-Version nicht immer richtig zusammenarbeiten, da die Größe der internen Tabellen über den Bildschirm nicht beachtet wird. Sie laufen folglich bei großen Schirmformaten über, wenn man mal mit einer Newword-Version arbeitet, bei der der zu NwSchirm gehörige Newword-Patch nicht ausgeführt wurde. Z.B. gibt es eine Tabelle, die die Zeilenlängen enthält, eine mit Flags, ob die Zeilen leer sind, und zwei mit der Kopfzeile.

Da sowohl NW216 diese Tabellen besitzt, muß wohl auch NW217 sie haben.

Ergebnis eines Tabellenüberlaufs müßte u.a. sein, daß bei großen Formaten beim Blättern der Schirm vor dem Schreiben einer neuen Zeile nicht immer richtig gelöscht wird, sondern ein Rest der vorigen Seite gelegentlich stehenbleibt. Dies sieht man z.B. bei den Leerzeilen in Assembler-Listings, die ab und zu plötzlich nicht mehr leer scheinen.

Im Quelltext von NWSCHIRM steht nach dem Label "VarTab217:", welche Tabellen bzw. welche Zeiger auf solche Tabellen benötigt werden. Wie findet man nun die Lage dieser Zeiger heraus?

1. Starten von NW und warten, bis sich alles beruhigt hat. Warmboot über Klick und "SAVE 60 N1.COM".

2. "GET 100 NW.COM", patchen der Bildschirmbreite WID um 1 größer mit Klick-Monitor (Adresse steht auch in NWSCHIRM.MAC nach "VarTab217:"). Danach "GO" und ebenfalls Warmboot und "SAVE 60 NWIDP1.COM".

3. Dito mit HITE+1.

4. Vergleichen von N1.COM und NHITEP1.COM, z.B. mit DiJey. Interessant sind die Stellen, wo der Wert um 1 bis 3 gesunken ist. Wenn dort Zeiger in den Bereich von ungefähr OCBOOh stehen, sind wird schon fast am Ziel. Denn jetzt ist NW nochmal zu starten und die Gebiete, in die die Zeiger zeigen, sind zu inspizieren. Dann erkennt man schon die Tabellen. Was für Tabellen genau gesucht werden, steht in NWSCHIRM.MAC.
   Bei veränderter Höhe findet man insbesondere die Tabelle mit den Zeileninhaltsflags, die Tabelle der Zeilenlängen und die Tabelle mit dem Inhalt der rechten Spalte.
   Die Adresse von hitem1 (hite minus 1) wird man auch finden, aber zumindest für Newword 2.17 ist sie ja schon bekannt.

5. Dito mit N1.COM UND NWIDP1.COM. Interessant sind Stellen, wo der Wert um 1 oder 2 gesunken ist.
   Bei veränderter Breite findet man insbesondere die beiden Kopfzeilen. Den Zeiger auf die erste Adresse hinter der zweiten Kopfzeile findet man so nicht, da er sich nicht ändert. Aber da man jetzt weiß, worauf er zeigt, kann man ihn mit MONI in NWIDP1.COM suchen.
   Auch die Adresse von widm1 wird man finden, aber sie ist ja wie hitem1 schon bekannt.

6. Wenn endlich alle Werte gefunden sind, können sie die Nullen in der Tabelle hinter dem Label "VarTab217:" ersetzen. Siehe die Kommentare dort. Damit die Werte auch beachtet werden, ist noch folgendes nötig: Die Routine "Test217:" ist (bis zu ihrem Ende "ret") zu löschen, ebenso die beiden Zeilen

```
call Test217
jp   z,Tab217
```

   Anschließend sollte noch das Label "Tab217:" gelöscht werden.

W o r d S t a r: Großes Bildschirmformat und was es ist/kann

WordStar 4.0 mit großem Bildschirm-Format (Herbert zur Nedden, 2000)

WordStar 4.0 ist ja eine Fortentwicklung von NewWord. Also habe ich Jan gebeten, mein WS 4.0 mal entsprechend zu untersuchen, was von Erfolg gekrönt war.

NWSCHIRM auf KLICK.009 kann daher auch das Bildschirm-Format von WS 4.0 umschalten, so WS 4.0 gepatcht ist. Dieser Patch ist auch dabei.

WordStar: Was ist es - was kann es (Herbert zur Nedden, 2000)

Ich habe eine LIZENZ für WordStar Version 4.0 erstanden. Das Teil hat mich ganze DM 250.- gekostet. Hast Du Interesse, dann schick mir einen Scheck über DM 250.- und in ca. 8 Wochen hast Du WS 4.0.

WS 4.0 kam in Form von 6 Disketten - alle Teile schön aufgeteilt und so gut wiederzufinden; insgesamt bummelig 800 kB. Dazu ein sehr gutes, allerdings englisches Handbuch.

Warum macht WS 4.0 eigentlich Sinn? Hier einige Feinheiten dieses Programmes, die ich so in loser Folge zusammengestellt habe:

- 148k Druckertreiber mit einer sehr umfangreichen Auswahl (wie vor einiger Zeit von Hans Gras im Info schon gelistet).
  Die Druckertreiber unterstützen die Drucker deutlich besser als NewWord, was insbesondere für Hoch/Tiefstellen und Proportionalschrift gilt.
- WS 4.0 kann rechnen! Mal eben eine Summe von Zahlen als Spaltenblock markieren, zwei Kommandos eingeben - und schon steht das Resultat im Text.
- Deutlich mehr Punkt-Kommandos (siehe Übersicht unten)
- Gebe ich z.B. als Dokumentname A:TEST.DOC B: ein, wird A:TEST.DOC auf dem Laufwerk A: bearbeitet, jedoch das Resultat auf B: gespeichert.
- Die Messages und Hilfe-Texte stehen in getrennten Dateien; die Hilfe-Texte müssen nicht da sein! Das spart Platz.
- Ich kann mir diverse Eingaben in WS speichern und Abrufen (= Shorthands). Klar, wir haben bombige F-Tasten, aber nichtsdestotrotz ist das ganz praktisch.
- Ich kann beim Mischdruck (Merge Print) auf die Zeilen- und Seitennummer zugreifen.
- Ich kann die Ausgabe an einen Drucker in eine Datei leiten.
- ^QQ wiederholt das folgene Zeichen bis ich in stoppe.
- ^QW und ^QZ scrollen bis zum Anhalten durch den Text.
- WS kennt Marker, die ich mit ^K0 bis ^K9 setze und mit ^Q0 bis ^Q9 dort hin positionieren kann.
- Suche vor/rückwärts nach einem Zeichen. Mit ^QG<RET> bin ich subito am Absatzende.
- WS 4.0 kann Inhalts- und Stichwortverzeichnisse erstellen!!!
- Rechtschreib-Prüfung und zugehörige Worte-Bibliothek für englisch ist dabei.
- User-Area Listing ist auf Diskette dabei.
- Für die Installation gibt es WSCHANGE. Damit kann ich so zeimlich ALLES patchen und installieren, wonach mir ist. Es ist gut menügeführt und kümmert sich auch ansständig um die Dinger, die bei NewWord als 'Special Patches' in Hex eingegeben werden mußten.
- Ich habe für WS 4.0 ein Instalations-Patch-Starterset zusammengestellt - allerdings nur für die RAM 4.x-oder-besser Umgebung.

W o r d s t a r: Befehlsübersicht

WordStar Befehlsübersicht (Herbert zur Nedden, 2000)

Da einem doch immer wieder das eine oder andere bei der Aufzählung der Features eines Programmes durch die Lappen gehen kann, hier einfach eine Übersicht über Befehle und Kommandos von WS 4.0. Ich gehe allerdings davon aus, daß Du NewWord kennst und liste hier nur die neuen Befehle auf:

Opening Menu

?	zeige Speicherplatz an
I	Stichwortverzeichnis eines Textes erzeugen (Index)
T	Inhaltsverzeichnis eines Textes erzeugen (Table of Contents)
S	Rechtschreibprüfung (englisch)

Sorthands

Esc	Shorthands = WS 4.0-interne Funktionstasten, die Du selbst definieren kannst; wirken auch bei Editieren eines Textes.
Esc ?	Aufruf Shorthand-Editor
Esc =	Ergebnis der letzten Rechnung in den Text
Esc $	Ergebnis der letzten Rechnung formatiert in den Text
Esc #	Formel der letzen ^QM-Rechnung in den Text

Help

^J?	Help zum Bildschirmaufbau
^J.	Help zu Punktkommandos

Cursor-Bewegung

^QB, ^QK	gehe zu Block-Anfang/Ende (ACHTUNG: NewWords ^QB --> ^QU)
^QI	gehe zu Seite (Document) bzw. Zeile (Non-Document)
^QP	gehe zu letzter Cursor-Position
^QV	gehe zu letztem Suchen/Ersetzen oder Block
^QG, ^QH	gehe zu Zeichen vorwärts bzw. rückwärts
^QW, ^QZ	kontinuierliches Scrollen hoch bzw. runter
^QF, ^QA	zusätzliche Option: Anzahl wie oft es passieren soll
^K0-^K9	Setze Marke 0 bis 9
^Q0-^Q9	Gehe zu Marke 0 bis 9

Block/Datei

^KN	Spaltenblöcke
^KI	Spaltenblöcke mit Erhalt der Spaltenpositionen
^KM	Zahlen in Block mit Vorzeichen aufsummieren
^K'	Block in Kleinbuchstaben
^K"	Block in Großbuchstaben
^KL	Laufwerk wechseln (damit dann das Dir angezeigt wird!)
^KF	Direktory anzeigen

Rechnen

^KM	Zahlen in Block mit Vorzeichen aufsummieren
^QM	Formel mit + - * / ( ) ausrechnen
.MA	Formel mit + - * / ( ) ausrechnen und in Variable für Mischdruck (Merge Print)

Randausgleich

.PM	Linker Rand für erste Zeile eines Absatzes (SUPER!!!) (In .RR-Zeile ein 'P' für Paragraph-Margin)
.AW	Randausgleich an/aus
.PS	Proportionalschrift an/aus
^QU	unter NewWord ist dies ^QB, d.h. ^B bis Textende
^P§	Feste Spalte kenntlich machen - für Tabellen bei Proportionaldruck

W o r d s t a r: Befehlsübersicht

Druckersteuerung

.BN	Wechsel Papiereinzugsschacht
.LQ	Letter-Quality-Druck an/aus
.SR	Pixel um die Hoch/Tiefgestellte Schrift über/unter die Grundlinie gestellt werden soll. Bei 0: Minischrift des Druckers.
.XL	Sequenz für Seitenvorschub neu definieren (Normal: ^L)
^PF	Phantom-Space = definierbares Zeichen (für Sonderzeichen)
^PG	Phantom-Rubout = definierbares Zeichen (für Sonderzeichen)

Kopf/Fußzeilen (Header/Footer)

^PK	bestimmte Leerzeichen nur auf (un)geraden Seiten drucken. (So ist z.B. Seitennummer rechts - links - rechts - ... mögl.)

Schlagwort- und Inhaltsverzeichnis (Index und Table of Contents)

^PK	Anfang/Ende eines Schlagwortverzeichnis-Eintrags
.IX	Schlagwortverzeichnis-Eintrag
.TX	Inhaltsverzeichnis-Eintrag

Mischdruck (Merge Print)

.SV	Variablenzuweisung
.MA	dto. mit Rechnen
&var&	wird durch Variableninhalt von var ersetzt
&var/O&	Zeile enfällt, falls var leer, sonst wie &var&
&var/F&	Variable var beim Einfügen in den Text entsprechend dem zuvor definierten Format F aufbereiten. Format-Steuerzeichen: Links/Rechts/Mittig/Ziffer/führende Null/... Die Formate werden mit .SV zuvor individuell definiert
&#&	Nummer der aktuellen Seite
&_&	Nummer der aktuellen Zeile
.GO	Gehe zu Ende oder Anfang des Textes

Sonst

^OB	Weiche Leerzeichen anzeigen (Soft-Spaces)
^Q?	Zähle Buchstaben bis hierher
^QQ	Wiederhole folgendes Zeichen bis zum Abwinken

H a r d w a r e: Klemmer - Umbauer - Maus

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

Dieser Tip ist nicht von mir, sondern von Hans-Henning Herder. Da er ihn aber nicht preisgibt, verrate ich ihn (den Trick und Henning):
Meine Tasten haben fürchterlich geklemmt, sie zu ölen habe ich mich nicht getraut, weil das Öl die Kontakte zu Unkontakten machen könnte. Flüssiges Autowachs (z.B. car plate von Johnson) hat nun die angenehme Eigenschaft, fest zu werden, und so die Kontakte auf die Dauer nicht zu verschmutzen. Man heble also die Tastaturkappen ab, nehme eine Spritze mit einer feinen Kanüle und gebe ganz vorsichtig einen winzigen Tropfen von diesem flüssigen Wachs auf die Mechanik. Dann wird die Taste ein paarmal durchgedrückt - und funktioniert hervorragend (bei mir seit nun 2 Jahren). Wenn aus Versehen etwas zu viel Wachs in die Taste gelangen sollte, so daß der Kontakt doch benetzt wird, so bleibt nichts anderes übrig, als diese Taste auszulöten und den Kontakt z.B. mit Reinigungsbenzin zu säubern (ist mir nur bei einer Taste passiert).

An alle Umbauer (Dr. Holger Göbel, 8630; 09561/15131)

Viele von uns basteln gerne am Computer. Erfahrungsgemäß treten am Ende einer solchen Aktion immer wieder völlig unverständliche Effekte auf, die keine Systematik haben.
So kann ich nur empfehlen, wirklich alle Verbraucher sternförmig mit einer zentralen Masse zu verbinden, möglichst auch die 5V-Versorgung. Selbst das Durchschleifen der Masse von 2 Laufwerken (bei insgesamt 4 Lw) hat bei mir schon Probleme gebracht.

Ein weiteres Phänomen hat mich kürzlich geärgert: Am Flachbandkabel für die Laufwerke habe ich einen Quetschstecker versetzen müssen: Also habe ich in aufgemacht und an anderer Stelle wieder aufgequetscht. Leider traten immer wieder Diskettenfehler auf, die erst behoben waren, als ich ein ganz neues Flachbandkabel eingebaut habe: offensichtlich war die Verbindung nicht mehr einwandfrei gewesen.

Zu unsrer 8-MHz-Platine: Sie lief bei mir immer einwandfrei, seit kurzem aber hatte ich öfters Diskettenzugriffsprobleme. Abhilfe hat die Vergrößerung des Kondensators geschaffen, der am 74LS123 die Haltezeit für kurzfristiges Umschalten auf 4 MHz bestimmt. Er ist jetzt 20 uF groß und außerdem kein Tantal- sondern ein einfacher Elektrolytkondensator (Horst Kupka sagte mir, daß der viel robuster ist).

Maus (Herbert zur Nedden, 2000)

Horst Kupka hatte einige Probleme beim Anschluß seiner Maus. Im Laufe seiner Ermittlungen, die von Erfolg gekrönt waren, machte er folgende Entdeckungen:

* Send vom MTX zur Maus muß auf -12 Volt liegen, sonst tut sich nix.

* Wird DTR kurz unterbrochen (d.h. -12 V -> +12 V -> -12 V), so sendet die Maus ein großes M als Kennung.

* An der RS 232-B läuft die Maus auch als Zweitasten-Maus

* An der RS 232-A läuft sie nur als Dreitasten-Maus.

H a r d w a r e: Platinen in die FDX - Zeichensätze für 80Z

Platinen in die FDX

(Herbert zur Nedden, 2000)

Unlängst spendierte ich dem Rechner meines Bruders eine 512kB-c't-SRAM-Floppy in der FDX, d.h. mit einem FDX-ECB-Adapter. Im Prinzip funktionierte die Platine auch, aber nicht richtig. Bad Sector war eine recht häufige Fehlermeldung. Die Platine selbst jedoch war O.K. - schließlich lief sie in meiner Kiste.

Nachdem ich direkt auf einige der IC's (LS 245 und noch einer) der SRAM-Floppy einen 100 nF-Kondensator gelötet hatte, lief die Chose.

Die Schaltpläne der FDX gaben mir einigen Aufschluß über den Grund der Probleme: Der Floppy-Controller und dir 80-Zeichen-Karte belasten den Datenbus mit je fünf IC's! D.h. die Treiber auf der FDX-Interface-Karte müssen 10 Eingänge treiben, was schon recht happig ist. Sollen Daten von der SRAM-Floppy zum MTX, muß der Datenbus-Treiber der SRAM-Floppy (72LS245) seinerseits 11 ICs treiben (5x Floppycontroller, 5x 80Zeichen, 1x FDX-Interfacekarte). Die IC's auf der Interface-Karte alle ihre eigenen Block-Kondensatoren und schaffen daher ihre Arbeit. Auf der c't-SRAM-Floppy waren die Kondensatoren etwas weniger großzügig verteilt, so daß die IC's dort mit der Last nicht klar kamen.

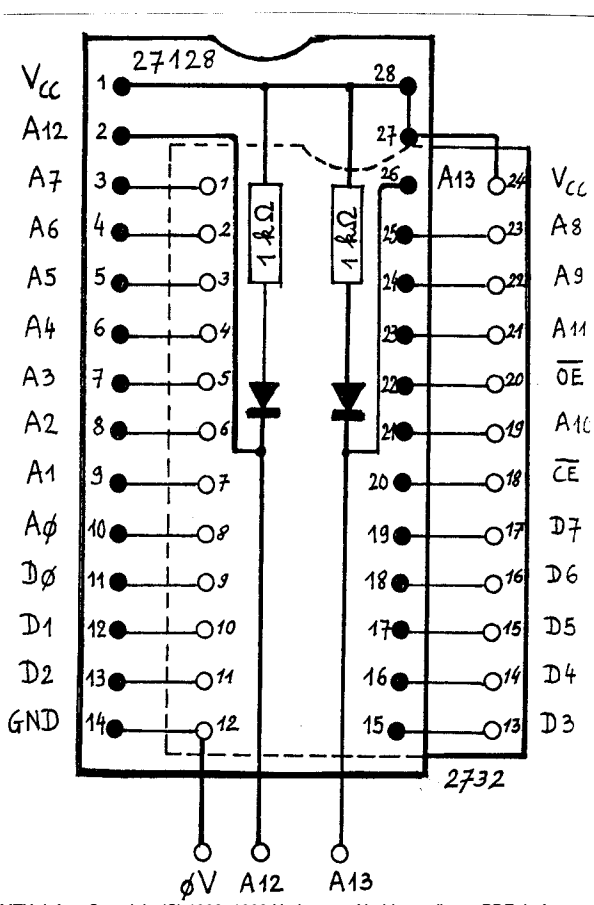
Evtl. könnt Ihr Eure Probleme mit anderen Platinen in der FDX (z.B. Harddisc-Controller) ebenfalls dadurch beheben, daß Ihr auf die IC's, die Signale an den Bus senden (können) 100 nF-Kondensatoren lötet.

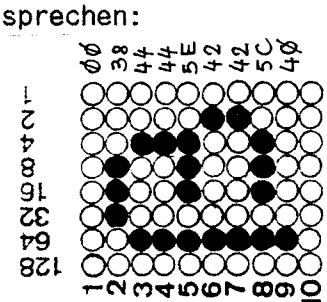
Zeichensätze für die 80-Zeichen-Karte

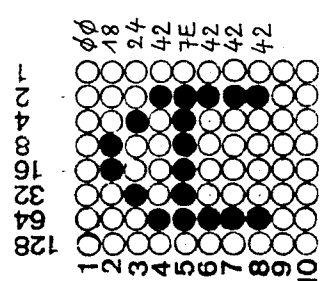
(Herbert zur Nedden, 2000)

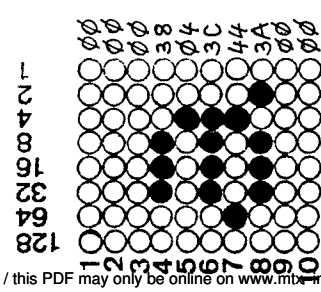
Erik d'Hondt bietet ein EPROM mit vier Zeichensätzen für die 80-Zeichen-Karte an. Der jeweils gewünschte Zeichensatz kann entweder per Schalter oder falls Du eine auf HiRes-Grafik aufgerüstete Karte hast per Software umgeschaltet werden.

Hier ein Paar Zeichnungen dazu, die für sich selbst sprechen:









c ' t: wo wollen die hin

Leitartikel in c't 2/90 (Dr. Holger Göbel, 8630; 09561/15131)

In der c't 2/90 hat Andreas Stiller einen Leitartikel geschrieben, der ein Überbordwerfen bisheriger Systeme zugunsten eines neuen, rationellen Standards fordert. U.a. schreibt er: 'Schlimmer noch als bei den Programmiersprachen scheint sich bei den Betriebssystemen der Hang zum Ewig-gestrigen zu manifestieren. Heute noch frönen einige Altvordere dem CP/M, welches etliche seiner schlechten Eigenschaften an MSDOS weitervererbt hat.' Ohne den Wert einer Weiterentwicklung auch der Computertechnologie - besonders für Wissenschaft und Wirtschaft - abstreiten zu wollen, bin ich der Meinung, den Leitartikel so nicht stehenlassen zu können, und habe einen Leserbrief geschrieben. Vielleicht interessiert er Euch:

An den
Verlag Heinz Heise GmbH & Co. KG
Postfach 61 04 07
Helstorfer Straße 7
3000 Hannover 60

Betr.: Leitartikel zu Heft 2/90, 'Alte Zöpfe ab!'

Sehr geehrte Damen,
sehr geehrte Herrren,

noch vor 2 Jahren hätte mich Herrn Stillers Leitartikel maßlos irritiert, inzwischen ist er fast bezeichnend für Ihre Zeitschrift geworden. Die c't hat sich sukzessive zu einem Organ für Computer-Hochtechnologie entwickelt. Natürlich sind Artikel dieser Richtung hochinteressant für den engagierten Leser. Haben Sie aber auch bedacht, welchen Druck Sie auf den Computeranwender ausüben, wenn Sie darüber vergessen, sich um seine Ausrüstung zu kümmern, die er sich vor 2 oder 5 Jahren gekauft hat? Gerade der Computeranwender scheut sich doch, als 'veraltet' angesehen zu werden. Reden Sie nicht einer Gesellschaftsmanie das Wort, die, von der Industrie natürlich angeheizt, das Bessere als den Todfeind des Guten einschätzt? Vielleicht ist es doch erlaubt, Parallelen zum Dilemma zwischen Steigerung des Bruttosozialprodukts und Ökologie zu ziehen.
Selbstvertändlich darf technische Forschung nicht auf einem bestimmten Niveau so ohne weiteres eingefroren werden. Ist es aber wirklich notwendig, daß der Computeranwender zu Hause einen 16-, 32- oder 64-Bit-Rechner besitzt? Dieser Text ist mit dem, wie Sie sagen, 'grauhaarigen' Wordstar geschrieben auf einem 'ewiggestrigen' CP/M-System, von einem 'Altvorderen', der Computerei aus Leidenschaft betreibt (die wohl wichtigste Motivation übrigens, die auch Ihre Zeitschrift am Leben erhält, nicht etwa irgendeine Notwendigkeit). Welche Möglichkeiten in so einem 'alten Zopf' stecken, ahnen Sie wahrscheinlich nicht. Die Tüftelei aber den Clubs zu überlassen ist schon deswegen problematisch, weil dies die Leute ausgrenzt, die den Mut haben, nicht jeden Rummel mitzumachen, und die Clubs auch deswegen immer mehr ausgedünnt werden.

In diesem Sinne fand ich den Leitartikel recht vorlaut, zudem etwas traurig und vielleicht auch beschämend für den Autor, der bisher durch engagierte, auch politische, Beiträge meinen Respekt erworben hatte.

Mit freundlichen Grüßen

c ' t: wo wollen die hin

Folgenden Leserbrief schickte ich gleichzeitig ebenfalls an c't

(Herbert zur Nedden, 2000)

Leserbrief

Ganz schön arrogant ? - von einem "Altvorderen"

Ich habe c't und mc abonniert, da diese beiden Zeitschriften nach meiner Meinung (die ich mit einigen anderen teile) nicht auf einen Rechnertyp fixiert sind und sowohl Neuigkeiten, Grundlagen, Ideen als auch Praxistips und Projekte herausbringen. Dabei waren die Informationen oft auch für solche Anwender, die nicht das neueste und schnellste haben durchaus interessant.

Dies wurde für mich insbesondere auch in einigen Anschreiben an die Leser(innen) auf Seite 3 von c't deutlich, wo hinterfragt wurde wozu dieser Technologie- und Geschwindigkeitsrausch gut sein sollte. Unerfreulicherweise degradieren diverse Computerzeitschriften alles was 'kleiner' als MsDos-AT ist (also MsDos-PC/XT, CP/M, Commodore C46 usw.) zu einem 'Nichts' und erklären MsDos-AT's zum Home-Computer. Ich frage mich, wo diese Selbstherrlichkeit her kommt. Mit seinem Anschreiben in c't 2/90 hat jedoch Andreas Stiller in meinen Augen den Vogel abgeschossen. Glaubt die c't-Redaktion denn etwa, daß die Leser alle zuhause 'ne VAX oder mindestens eine 20-MHz-80386 stehen haben? Ist c't für private Anwender oder soll c't jetzt eine Zeitschrift für professionelle werden?? Wenn ja, bitte ich die c't-Redaktion dies auch mal deutlich zuzugeben, damit die, die nicht mehr angesprochen werden die Konsequenzen ziehen können!

Ich kann durchaus nachvollziehen, wie toll es sein muß, so einen Super-Turbo mal laufen zu sehen! Nur dürften die meisten privat genutzten Computer eher die Bereiche Textverarbeitung und Haushalts- und Auto-Buchhaltung als Anwendungen fahren. Dazu kommen all die emsigen Bastler. Diese User brauchen Tips und Hinweise zum Progammieren und Löten! Klar - vieles ist sicherlich schon dagewesen! Aber nicht alles! Selbst Projekte für IBM-Kompatible haben mir für CP/Mund den ECB-Bus gute Anregungen geliefert.

Ich leite einen Computer-Club, der sich um CP/M rankt. Im Laufe der Zeit habe ich (und damit stehe ich nicht alleine - auch andere CP/M-orientierte Clubs geben mir Recht) das Gefühl bekommen, daß CP/M-User recht aktiv sind, während Ms-Dos-User vorwiegend Anwender sind, die fertige Programme laufen lassen. Wenn ich mir nur mal so überlege, daß z.B. im Laufe von 1989 ZCPR 3.4 als Ersatz für den CP/M-CCP (Command Console Processor) und NovaDos (aus P2DOS entstanden) als CP/M-BDOS-Ersatz als PUBLIC DOMAIN herausgekommen sind; oder z.B. MicroPro WORD-STAR Professional Release 4.0 für CP/M herausgebracht hat, welches in der CP/M-Version auch Inhalts- und Schlagwortverzeichnisse unterstützt und rechnen kann - dann frage ich mich warum Computer-Zeitschriften dieses **irgnorieren**! Gerade von c't hätte ich erwartet, daß in Anbetracht der guten Kontakte in die USA derartige Informationen den Lesern nicht entgehen dürften!

Wir haben mal unser 8-MHz-Z80-CP/M-System mit einem 8-MHz-IBM-Kompatiblen mittels einem Turbo-Pascal-Benchmark verglichen. Es war ein Kopf an Kopf Rennen! Bei Interger war der IBM schneller; bei Bildschirmausgaben unser CP/M. Es ist doch schon bemerkenswert, daß die 16-Bit-CPU die 8-Bit-CPU nur bei Integer schlägt, oder?

c ' t: wo wollen die hin

Über alle CPU's läßt sich c't aus - nur nicht über die Z280. Warum denn bloß; es gibt von verschiedenen Anbietern fertige Z280-Karten, die in ihrer Leistung eine 80268 (wenn nicht gar mehr) in die Tasche stecken können. Vergleiche ich die verschiedenen CPUs, kann ich nur noch mit dem Kopf schütteln. Die 80186 und 80286 könne im protected Mode die 8086 nicht nachbilden; die 68000 und 68020 sind bei Traps inkompatibel. Die Z280 hat diese Probleme nicht und bietet das, was erst neuere andere CPUs bieten: Memory-Mamagement-Unit, 4 DMA-Kanäle, 3 Counter-Timer, 1 ser. Kanal und Cache on Chip, ist von Haus aus Multiprozessor-System tauglich und kann seriell gebootet werden. Sollte die Peripherie langsamer sein, spendiert die Z280 je nach Zugriffsart Waits oder geht mit 1/2 bzw. 1/4 der CPU-Frequenz auf den Bus. Für den ECB-Bus gibt es dermaßen viele gute Karten; alle Welt redet von EPROM/GAL/PAL-Programmiergeräten - für den ECB-Bus gibt es soetwas schon lange!

Weiterhin muß ich mich doch gegen die Behauptung von Andreas Stiller aus c't 2/90, Seite 3 wehren, daß die Macken von CP/M in MsDos übernommen wurden! CP/M ist erheblich besser konzipiert als MsDos - nur in Sachen Bildschirmtreiber und Diskettenformate nicht so standardisiert! Wenn ich an die 640k-Grenze, die von der festen Adresse des Bildschirmspeichers kommt, oder die Probleme mit mehr als 32 MB auf einem Laufwerk denke, kann ich nur Mitleid mit MsDos empfinden! CP/M kennt feste Adressen nur von 0-100h. Je nach Kapazität des Laufwerks werden die Blöcke mit 8- oder 16-Bit-Blocknummern verwaltet; große Laufwerks-Kapazitäten sind damit kein Problem! Bei der Ausnutzung der Disketten scheinen sich viele (auch neue) Systeme auch schwer zu tun. MsDos gibt sich mit 1.44 MB auf 3.5"-Disketten zufrieden! Unter CP/M habe ich da 1.76MB. Auch Probleme mit Memory-Mapped-I/O, womit zusammenhängende Speicherbereiche zerstückelt werden sind bei einer Z80 und Z280 fremd. Dinge wie Cache, Druckerspooler, residente Programme etc. gehen unter CP/M auch.

Man sollte sich wirklich hüten, die Anwender, die etwas anderes einsetzen als man selbst für gut hält - sei es nun, weil es ihnen Spaß macht oder schlicht und ergreifend aus finanziellen Gründen oder aus Bequemlichkeit (weil man es kennt und hat und es genügt) - nicht einfach für dumm erklären! MicroPro z.B. bedankt sich im Handbuch für die CP/M-Version von WORDSTAR Professional Release 4.0 bei den CP/M-Freunden für ihre Existenz, denn ohne CP/M hätte es evtl. WORDSTAR garnicht gegeben. Fährt Herr Stiller immer das neueste und modernste Auto mit umfangreicher Bord-Elektronik und nennt den, der einen guten VW-Käfer fährt einen Altvorderen??? Immerhin sind beides Autos und erfüllen für den Eigner ihren Zweck, oder ?

Leserbrief dazu aus c't 3/90 (von Edmund Ramm, kein MTX-ler)

Nicht von Morgen

Auch Sie scheinen leider dem Irrtum verfallen zu sein, daß alle Dinge, die nicht von morgen sind, veraltet oder schlecht sein müssen. Warum sollte ich kostbare CPU-Zeit dafür verschwenden, micht von einer Graphikoberfläche zu einem nagetierschiebenden Ikonenklicker degradieren zu lassen, wenn es die Eingabe einiger kurzer Befehle auch tut und ich dabei sogar die Hände auf der Tastatur belassen kann? In Punkto Geschwindigkeit nehme ich es unter Verwendung von ED.COM mit jedem Rodent-Pusher und z.B. WORD auf! Bevor ich von CP/M auf ein anderes Betriebssystem wechsele, muß schon etwas wesentlich Besseres vorbeikommen (ist evtl. in Sicht in Form von Unix, natürlich ohne Graphikoberfläche, ab nach /dev/nul damit!), denn seit ich CP/M-80 auf einem Z280 fahre, kann ich über ATs und dergleichen nur noch grinsen.

H a r d w a r e: Megabytes

Ein bis drei MegaBytes als RAM-Floppy (Herbert zur Nedden, 2000)

In Anbetracht der zum einen (noch) sinkenden Preise für Megabit-Chips, d.h. für die schnuckeligen RAMs von denen acht zusammen ein MegaByte ergeben, und dem zum anderen immer höher werdenden Platzbedarf für KLICK-Overlays stellte sich die Frage, was man aus diesen Informationen machen kann.

Zwei Lösungen drängen sich auf:

1. Durchbrechen der 784 kB-Grenze für Speicher auf der Haupt- bzw. Speichererweiterungsplatine; eine Idee, die von RAM 6.x unterstützt wird, und zwar immerhin bis 1552 kB.
2. Eine vernünftige RAM-Floppy für den ECB-Bus.

Für Lösung 1 spricht, daß dieser Speicher recht relativ bequem zwischen der internen RAM-Floppy, dem Platz für KLICK-Overlays und dem Spooler aufgeteilt werden kann. Lösung 2 kann als Vorteil verbuchen, daß eine RAM-Floppy auf dem ECB-Bus schreibgeschützt werden kann, Reset-Fest ist und auch ggf. mal in einen anderen Rechner gesteckt werden kann.

Ich habe erst einmal Lösung 2 verfolgt, da sie leichter zu realisieren und zu testen ist. Das Zugriffs-Timing ist unkritischer und die Karte stört den Hauptspeicher und damit die CPU nicht. Immerhin mußte ich auch erst einmal Erfahrungen mit den MegaBit-Chips sammeln.

Um es gleich vorweg zu nehmen: Ich habe es geschafft, eine 1 MB-RAM-Floppy ans Laufen zu bekommen, und auf der Platine ist noch Platz für zwei weitere MB.

Kommen wir zur Hardware und damit auch zum Thema Refresh. Meine Idee war es, die von c't angebotene 1 MB-DRAM-Floppy-Platine, die (mein Gott wie praktisch) schon von RAM 4.x unterstützt wird, als Grundlage zu nehmen. Immerhin muß für eine RAM-Floppy folgendes zusammengestrickt werden:

1. Interface zum Bus. Das sind eine Portdekodierung und Bustreiber.
2. Zwischenspeicher für Sektor- und Spur-Nummer.
3. Zähler. Es werden beim Lesen/Schreiben immer ganze Sektoren zu 128 Bytes übertragen, also müssen je Übertragung 128 Speicherstellen angesprochen werden. Und dafür soll auf die Platine der Zähler, der automatisch nach der Übertragung eines Bytes um eins weiterzählt, so daß ein Sektor mit 128 IN- bzw. OUT-Befehlen übertragen werden kann.
4. Da war doch noch etwas ... ach ja, die RAMs
5. Refresh für die RAMs. Dynamische RAMs haben nämlich die dumme Angewohnheit, ihre Daten nach einer recht kurzen Zeit zu vergessen, wenn man sie nicht auffrischt - sozusagen ein Kurzzeitgedächnis.

Ich hatte wahrlich keine Lust, diesen Gramusel vollständig neu aufzubauen zumal es die o.g. Platine von c't gibt, die mir diese Arbeit zum guten Teil abnimmt. Da die c't-Platine max. 1 MB mit 256-kiloBit-Chips unterstützt, werden alle acht Bit der Sketor-Nummer und Bit 0-2 der Spur-Nummer als Adressen an die RAMs angelegt, während Bit 3-4 der Spur zur Selektion der Bank dienen. Da MegaBit-Chips zwei Adreßbits mehr benötigen, nehmen wir halt Bit 0-4 der Spur als Adresse und Bit 5-6 der Spur zur Selektion der Bank.

Unpraktischerweise habe die MegaBit-Chips eine deutlich andere Pin-Belegung als 256k-Bitter haben, landeten die MegaBitter kurzerhand auf einer kleinen Huckepack-Platine, die 'fliegend' über die c't-Platine gelötet wurde. Nachdem ich das alles zusammengelötet hatte, funktionierte die Karte sogar auf Anhieb - verlor jedoch konsequent ihre Daten.

H a r d w a r e: Megabytes

Und damit kommen wir zum leidigen Thema Refresh. Der Refresh kann auf mehrere Arten erzeugt werden. Zur Erinnerung: Der Refresh dient dem Auffrischen des Kurzzeitgedächnisses der RAMs. Folgende zwei Arten des Refresh interessiert mich:

1. Der RAS-Only-Refresh. Hierbei muß an die Adreßpins der RAMs eine Adresse angelegt und RAS aktiviert werden. Beim nächsten Refresh muß dann die um eins erhöhre Adresse angelegt werden. Dieser Refresh verlangt also etwas Elektronik zum Zählen.
2. Der CAS-Before-RAS-Refresh. Dieser ist recht praktisch, da einfach vor einem RAS der CAS aktiviert werden muß. Die RAMs wissen dann, daß sie ihre Erinnerung auffrischen sollen und erzeugen sich ihre Refresh-Adresse intern selbst.

Da die c't-Platine auch billige RAMs unterstützt, die den CAS-Before-RAS-Refresh nicht können, ist ein Zähler für den RAS-Only-Refresh auf der Platine drauf, den ich auch gleich nutzen wollte. Dummerweise brauchen die MegaBit-Chips eine 9-Bit-Refresh-Adresse und die 256k-Bitter nur 8-Bit. Daher ist der Refresh-Zähler auf der c't-Platine zu kurz.

Na gut, dann halt den CAS-Before-RAS-Refresh: Den Refresh auf der Platine abklemmen und die CAS-Signale der RAMs zusätzlich beim Refresh aktivieren. Klingt gut, die Karte blieb aber vergeßlich!

Klar, wenn ich nicht richtig lesen kann ... Der heißt doch CAS-Before-RAS-Refresh, Betonung auf RAS. Erfreulicherweise erzeugt die Karte den geeigneten RAS schon für ihren RAS-Only-Refresh. Nachdem ich diesen aktiviert hatte, lief die Karte.

Nun zum Löten: (ICxx/nn = Pin nn von IC xx)

1. Neue Nutzung der Bits der Spur-Nummer:
   Trenne: IC8/13 und IC8/14 hochbiegen, IC7/13 und IC7/14 von Masse abtrennen
   Lege: IC4/5 - IC7/13, IC4/2 - IC7/14, IC4/19 - IC8/14 und IC4/16 -IC8/13
2. Refresh:
   Das teure IC 11 (74LS539) entfällt ersatzlos!
   Ein 74LS08 muß her; dieses IC enthält vier UND-Gatter mit je zwei Eingängen und einem Ausgang. In die vier CAS-Leitungen wird je ein UND-Gatter eingeschleift und die vier dann noch freien UND-Eingänge mit RFSH verbunden.
   Dazu werden die Pins IC8/9, IC8/10, IC8/11 und IC8/12 alle hochgebogen und an je einen Eingang unterschiedlicher UND-Gatter angeschlossen. Die Ausgänge dieser UND-Gatter werden an die jeweiligen Anschlüsse unter den hochgebogenen Pins des IC8 gelötet. Die nun noch freien Eingänge der vier UND-Gatter kommen an RFSH. Ach so, Masse und +5 Volt für den LS08 nicht vergessen!
3. Jumper: J4: 1-2, J5: 1-2, J6: RFSH, J7: Schalter für Schreibschutz

Übrigens habe ich für die Huckepack-Platine, die die acht MegaBit-RAMs aufnimmt Layouted, d.h. mit den Edding-Leiterplatten-Symbolen geklebt und anschließend ätzen lassen. Allerdings ist es die Frage, ob es nicht schneller zu Löten geht, wenn man einfach eine Lochrasterplatine nimmt und diesen Draht, dessen Isolierung beim Löten schmilzt verwendet. Immerhin müssen auf der Platine satte 8*18 (RAMs) + 8*2 (Block-Kondensatoren) und 18 (Anschlüsse) Löcher gebohrt werden, und zwar recht präzise, damit die IC-Sockel auch passen.

Sorry, aber diese Seite musste ich entfernen, weil ich für eine Veröffentlichung außer in der ehemaligen, gedruckten Form keine Freigabe habe.

Sorry, aber diese Seite musste ich entfernen, weil ich für eine Veröffentlichung außer in der ehemaligen, gedruckten Form keine Freigabe habe.

Sorry, aber diese Seite musste ich entfernen, weil ich für eine Veröffentlichung außer in der ehemaligen, gedruckten Form keine Freigabe habe.