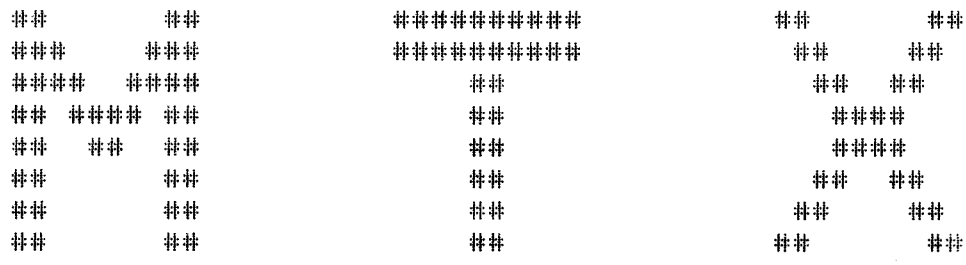


Info 4

MTX User-Club Deutschland

1. **Zweck:** Austausch von Tips & Tricks u.s.w.

2. **Programme** (nur **Selbstgeschriebenes**): Tausch von kurzen und einfachen Routinen. Besprechung von guten Programmen damit der Autor diese dann an Clubmitglieder verkaufen kann. Programme einfach an uns schicken, und wir liefern Verbesserungshinweise, Besprechung,...

3. **Mitglied** kann jeder werden! Keine Aufnahme oder Beitragsgebühr!

4. **Verpflichtungen** keine!

   Bitte: Einsendung von Tips & Tricks, Fragen, Antworten, kurzen Routinen, und was noch so zusammenkommt und andere interessieren könnte.

5. **Club-Info** wollen wir ca. monatlich verschicken. Das hängt von allen ab, da wir ja nicht rund um die Uhr am Computer sitzen können. Da brauchen wir die Hilfe aller Mitglieder!

6. **Kosten:** Wir berechnen ausschließlich Selbstkosten (Porto, Verpackung,..). Verständlicherweise verschicken wir **nichts**, wenn kein Geld da ist (s.u.)

   Da wir unseren Steckbrief nicht nur gegen Freiumschlag verschicken, ziehen wir denen, die ihn geschickt bekommen dafür DM -.70 vom Konto ab. (Einspruch ist selbverständlich jederzeit möglich.)

7. **Geld/Konto:** Für jedes Mitglied führt Herbert Herberg ein Konto, von dem die enstehenden Kosten jeweils abgehen. Der Kontostand wird regelmäßig mitgeteilt, und kann selbverständlich jederzeit erfragt werden!

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

8. **Kontaktadressen:** (alle derselbe Club!!)

Herbert Herberg
Sonnenau 2
2000 Hamburg 76
(040) 200 87 04

Thomas Pflaum
Leipziger Platz 1
8500 Nürnberg 20
(0911) 51 35 21

Moin, moin!

Achtung Martin Müller ist nicht mehr dabei!!! D.h. Ansprechpartner für den Club sind nur noch Thomas Pflaum und ich!!

Bislang habe ich in den Info's vermieden **Du** bzw. **Sie** zu verwenden, und dementsprechend einen recht eigenwilligen Stil der deutschen Sprache benutzt. Das finde ich aber eigentlich garnicht schön. Man liest das Info, das an die Mitglieder - also auch denjenigen der dies gerade liest - gerichtet ist,... und ist persönlich eigentlich garnicht angesprochen. Da die Mehrzahl der Mitglieder jünger und mehr für **Du** zu haben ist, bitte ich alle mir zu erlauben im Info die Anrede **Du** zu verwenden!! Danke!

Korrektur: Auf dem Deckblatt von Info 2 ist ein grausamer **Druckfehler**!!! Wir verschicken unser Erstinfo = Steckbrief (das ist nicht Info 1!) ja nicht nur gegen Freiumschlag. Deshalb kostet dieser Service alle, die es so per Post erhalten DM -.70 (siebzig Pfennig), da es sonst für uns zu teuer wird. In Info 2 steht statt DM -.70 leider eine DM 70.- ! Bitte nicht weiter beunruhigen!!

Folgende verständliche Kritik kommt bei mir ab und zu an: 'Du machst zu viel für die Floppy'. Nun dazu folgendes: Fast alles aus dem Info ist von Euch Mitgliedern - die nummerierten Seiten, die ich in NewWord eingebe sind durch Fragen von Euch verursacht! Und in Punkto Zuschriften sind die FDX'er wesentlich aktiver!

Wenn ich Euch verschiedene Info's e.t.c. in einem Umschlag zusammen verschicke (das tue ich auch wenn es was bringt), dann kostet es Euch natürlich nicht die Summe dessen, was die Sendungen einzeln kosten: Ihr spart Verpackung und vor allem Porto (2 mal DM 1.40 ist mehr als 1 mal DM 2.-). Genauso gibt es bei den Spielprogrammen von mir (s.u.) Mengenrabatt.

Auf vielfachen Wunsch habe ich nun in die Mitgliederliste die Telefonnummern soweit mir bekannt aufgenommen. Bitte überprüft auch diese mal in der Liste. Außerdem vermisse ich noch viele Fragebögen! Diese Fragebögen helfen Thomas und mir auch einzuschätzen, wo Interessenschwerpunkte liegen könnten (wievele haben FDX ist wohl die wichtigste Frage, da ja für nicht-FDX-er ein FDX-Info-Bericht nicht gerade bereichernd ist!)

In diesem Info liegen ein paar Seiten, die offensichtlich zu einigen aus Info 3 (z.B. F. Dersewski) gehören. Diese habe ich nicht in Info 3 gelassen, damit das Porto nicht von DM 1.40 auf DM 2.- für drei zusätzliche Seiten steigt.

W I C H T I G: Club-Treffen ist geplant: Nach Ostern, in Hamburg - mehr dazu s.u.

Telefonisch bin ich zu folgenden Zeiten erreichbar:
Dienstags 19.30 - 21.30 und Samstags 9.00 - 12.00
Bitte ruft mich nur zu diesen Zeiten an (es sei denn beide passen nicht). Zu beiden Zeiten ist Billig-Tarif für's Telefonieren! Und ich habe die Unterlagen zur Hand und Zeit!

Herbert Herberg

32 Seiten ROM- bzw. BASIC-Info (Herbert Herberg)
Wir haben leider kein kommentiertes ROM-Listing und auch keine Aussicht bald eines zu bekommen. Für jeden Hinweis sind wir dankbar!
Inhalt der 32 Seiten: Voll disassembliertes & kommentiertes Cassettenhandling, Jump-Tables für die BASIC-Befehle, Joystick & Keyboard-Tips (u.a. eine Routine, wie man für den Joystick R, d.h. Cursortasten, Mehrfachdruck erfragt) sowie VRAM-Aufteilung & Videocontrollerdaten.
Diese Unterlagen sind für das ROM-BASIC sowie das FDXB (Diskette) gültig! Nur: ROM Page 1 ab Hex 2000 liegt bei FDXB ab #4000.
Diese Seiten gibt es bei Herbert Herberg für DM 8.- (incl P+V)

Bisher erschienen und noch erhältlich: (Herbert Herberg)
Info 1 (18 Seiten, DM 4.70), Info 2 (45 Seiten, DM 8.15), Info 3 (46 Seiten, DM 8.60). Allerdings dauert die Lieferung ggf. etwas, da ich nur begrenzte Vorräte habe. Wer etwas davon haben möchte muß es mich wissen lassen! Nur das aktuelle Info gibt's automatisch!

Programme
Andreas Viebke: Flugsimulator DM 20.-
Andreas Viebke: Display & Enlarge DM 20.-

Herbert Herberg:
Die mit einem Stern gekennzeichneten Programme gibt es auch als Listing (DM -.30 je Seite). Ich tausche auch!
Preise netto, d.h. ohne Datenträger (DM 6.-) und P+V.
Die Programme mit einem Ausrufezeichen brauchen den 80-Schirm in der FDX **und** die 40-Zeichen-Grafik im MTX-Grundgerät.
Viele der u.g. Programme enthalten Maschinensprache (Assembler) und laufen nur auf dem MTX 500 und unter FDX-BASIC. Wer einen MTX 512 ohne FDX benutzen will, bei dem das Programm bei Hex 4000 (und nicht bei 8000) beginnt, muß vor dem laden des Programmes POKE 64122,0 (64122 = #FA7A) eingeben, um so den 512-er in einen 500-er zu wandeln.

4.- * Charactergenerator incl Zeichensatz
1.- * Character-Designer
4.- Labyrinth (durch ein Labyrinth hindurchfinden)
3.- Liner (plötzlich auftauchenden Linien ausweichen)
3.- Miner (Gold Berg finden)
1.- * Poker
2.- Shuttle (Bomben von einer Space-Shuttle fallen lassen)
3.- Brio (Geschicklichkeitsspiel: Kugel durch Labyrinth)
2.- * Breaktout (Mit Ping-Pong-Verfahren Steine zerstören)
2.- * Mamind (Mastermind = Kombinationsraten)
2.- * Jigsaw (Puzzle)
2.- * Missle (Städte vor Zerstörung schützen)
2.- * Anschiss (Mind over Electons)
1.- * Wasi (Nimble-Thimble, was immer das sei)
2.- Sandburg
6.- ! Railroad (Schienen legen mit bis zu 9 Spielern)
6.- ! Pferde (Pferdelauf mit mehreren Spielern)
4.- War-Plan (Flottenvernichtung)
2.- Lunar (Mondlandung)
2.- Willie (Ein gefräßiger Wurm)
5.- Vier-Gew (Vier gewinnt)
0.- Disketten-Konvertierung TA-PC -> MTX (d.h. gratis!)

Korrektur zu BASIC (Herbert Herberg)
der CRVS 5 - Befehl für die korrekte Ausgabe von Noddy-Seiten muß wie folgt aussehen: CRVS 5,0,1,0,39,24,40

Hard- und Software für MTX
Auch solche Dinge wie Silicon -Disks, EPROM-Software, viele Spielprogramme, Bücher, ... liefert:
Elektronik-Versand Karl-Heinz Harter, Salmstraße 13, 7550 Rastatt 15

Flachbandkabel (Herbert Herberg)
Pin 1 ist bei den meisten Computer-Flachbandkabeln (d.h. Kabeln die aus vielen nebeneinanderliegenden aneinandergeschweißten isolierten Drähten bestehen) deutlich gefärbt. Beim MTX ist Pin 1 vom Floppy-Anschlußkabel und Druckerkabel rot gefärbt!!
FDX: Pin 1 auf RS 232C-Karte: hinten, unten in FDX: rechts
Drucker: Pin 1: rechts
Falschherum angeschlossenes FDX-Kabel ist Mord der RS232C-Karte!!

Die erste Memotech-MTX-Diashow (Michael Köster)
Offener Brief an H. Herberg:
WOCHEN DER ARBEIT LIEGEN NOCH VOR MIR,UM DIE UNENDLICH SCHEINENDEN 300K EINER DISKETTE MIT HERRLICHER GRAFIK ZU FÜLLEN.VON DER SCHLICHTEN INK- GRAFIK,UM COMPUTERSCHRIFTSTÜCKE MIT IHREN HARDCOPYS ETWAS AUFZULOCKERN, BIS HIN ZU DEN FARBGRAFIKEN MÜSSEN SCHON AN DIE 40 BILDER ZUSAMMENKOMMEN. DAS KOPIEREN VON 300K MIT EINEM EINZELLAUFWERK BEREITET MIR SCHON JETZT ALPTRÄUME,DARUM HOFFE ICH AUF GENUG INTERESSENTEN UM MIR EIN ZUSATZLAUFWERK LEISTEN ZU KÖNNEN! OB MAN EINEN ESEL,DER CA.1.000.000 PUNKTE BERECHNET,ALS SHOW-MASTER(H.HERBERG) BEZEICHNEN KANN,WAGE ICH ZU BEZWEIFELN.WER JEDENFALLS 25.-DM(33.-MIT DISK U. VERP.)FÜR DIE ARBEIT VON EINEM HALBEN JAHR,FÜR ZU VIEL HÄLT UND AUF RAUBKOPIEN WARTET,GEHÖRT NICHT ZUM GUTEN PUBLIKUM. WER JEMALS DIE BRIGITTE BARDOT FARBGRAFIK ZU GESICHT BEKOMMT,WIRD ERKENNEN,DAß 49.152 PUNKTE*12.288 FARBMÖGLICHKEITEN BESSER WIRKEN ALS Z.B. DIE 64.000 PUNKTE *2000 FARBMÖGLICHKEITEN DES C-64.
BIS ZUM 06.02. FERTIGGESTELLT: B. BARDOT,GOOFY,HOCHZEITSFOTO,TELEPHON
IN VORBEREITUNG: EISVOGEL,MADCHEN
GEPLANT: PKW,BLUMEN,LANDSCHAFT,INTERRESS.OBJEKTE
WEITERE INFORMATIONEN SIND HIER WOHL NOCH VERFRÜHT!

TSCHÜSS,MICHAEL

Autostartunterdrückung beim FDX-BASIC: (Herbert Herberg)
Folgende Befehle hintereinander eingeben! Der / bedeutet <RETURN>
PANEL/D4AF5/C3/50/2/<BRK>L4AEE/
D.h. ab #4AF5 werden die Bytes #C3,#50,#02 eingetragen, mit anderen Worten: An der Hex-Adresse 4AF5 steht JP #0250.
Dann muß auf dem Schrim irgendwo CALL #0B40 und direkt darunter die Zeile JP #0250 stehen!!
Dann zurück ins BASIC und Laden. Aber viele Spielprogramme haben zerstörte BASIC-Pointer (d.h. Speicherinhalte, die sagen von wo bis wo das Programm steht)! Die zweiteiligen Programme von Continental Software bleiben auch gegenüber dieser Behandlung ungekrackt!!

Disketten von Fudschi, SS, DD für DM 3.60 (Götz Neumann)
Ich kann bei größeren Bestellmengen o.g. Disketten für je DM 3.60 besorgen (Vobis: SS, SD je DM 3.90, Anm. d. Red.). Dazu kommen natürlich noch Versandspesen! Wer Interesse hat schicke bitte Name, Anschrift und Telefonnummer, die Anzahl der gewünschten Disketten (und Unterschrift als Bestätigung) an mich (Götz Neumann, Am Steinicht 22, 8630 Coburg), und wenn genügend viele Bestellungen zusammen kommen werden die Disketten besorgt.

Ausgabe eines Strings auf den 80-Zeichen-Schrim (Herbert Herberg)
Anbei ein Listing mit einem Programm, das einen Zeichenstring auf den 80-Zeichen-Bildschirm ausgibt. Dabei werden die Control-Codes wie die Hex-10 als Line-Feed, #0C als CLS nicht als solche interpretiert, sondern die entsprechenden Grafiksonderzeichen werden ausgegeben!

SuperCalc
Hat es jemand geschafft die aktuelle Zelle statt mit '<.....>' mit inverser Schrift kenntlich zu machen?

Einfacher EPROM'mer gesucht
d.h. eine Schaltung mit der man EPROM's programmieren kann.

BTX
Hat irgendjemand irgendwelche Hinweise oder Erfolge bzgl. BTX-MTX ?

BASIC (Herbert Herberg)
Speicherbelegung beim MTX-BASIC (ROM & FDX):
Im Speicher stehen erst das BASIC-Programm, dann die Noddy-Seiten und dann die Arrays (also Speicherstellen die mit DIM erzeugt werden).
Die genaue Lage der o.g. Daten ist natürlich auch irgendwo abgespeichert.
Aber ganz einfach ist es nicht, da ja im ROM-BASIC mehrere RAM-Pages, d.h. RAM-Seiten existieren (s. Handbuch 6. & 7. Seite von hinten), und das muß ja irgendwie dann auch notiert werden. Und im BASIC-Programm wird eine Programmzeile nie auf zwei verschiedene RAM-Seiten aufgeteilt, so daß ggf. am Ende einer RAM-Seite noch was frei bleibt. Fazit: Für jede RAM-Seite muß angegeben werden wie weit das BASIC-Programm noch auf dieser Seite geht. Bei Noddy und den Arrays wird geschnipselt, so daß man da nur Anfang und Ende (incl. Seite) braucht.
Mit <xxxx> meine ich den Inhalt von #xxxx, und mit <<xxxx>> bezeichne ich den Inhalt von #xxxx + 256 mal den Inhalt von #xxxx+1.

```
Höchste erlaubte Seite:          <FA7A>
BASIC Untergrenze:               <<FAAA>>
BASIC Obergrenze Seite 0         <<FAAC>>-1
                       1         <<FAAE>>-1
                       2         <<FAB0>>-1   u.s.w.
Noddy Untergrenze = BASIC Obergrenze der letzten BASIC-Seite plus 1
Noddy Obergrenze                 <<FAA4>>-1 auf Seite <FAA6>
Array Untergrenze =              <<FAA4>>   auf Seite <FAA6>
Array Obergrenze                 <<FACC>>-1 auf Seite <FACE>

Achtung <<FAA7>> auf Seite <FAA9> ist aktuelle BASIC-Seite!!!
```

CP/M Programme (Herbert Herberg)

ASM.COM	Assembler
BATCH.COM	Batch-Programm
BAUD.COM	Baud-Raten für RS 232C
BIOS.ASM	BOIS Assemblerlisting
CBIOS.ASM	BIOS Assemblerlisting
CBIOS.HEX	BIOS Hexfile (INTEL-Format)
CFIG8.COM	Televideo-Configurierer
COLDBOOT.COM	Kaltstart
CONFIG.COM	Disketten-Konfugurierprogramm
CONTACT.COM	Modem
CONTVI.COM	?
CPMGEN.SUB	CP/M Neusystemgenerierung BATCH-File
DDT.COM	Disassembler 8080 Code
DUMP.ASM	Hexdump Assemblerlisting
DUMP.COM	Hexdump
ED.COM	Editor
ENTER.COM	Eingaberoutine für BATCH
ERAQ.COM	Lösche Files
FDXB.COM	BASIC
FKEY.COM	Funktionstasten Belegung
FORMAT.COM	Formatieren (2 Laufw.)
H.COM	?
I.COM	?
INITIATE.COM	Test der Laufwerke & Baudraten setzen
LAST.LNK	Zu CONTACT
LOAD.COM	HEX-Format -> COM-Format
MOVCPM.COM	CP/M Änderung
MTX.COM	Umschalten auf ROM-BASIC
NCPM.COM	CP/M Änderung
OVERLAY8.COM	Televideo-Format-Overlay
PIP.COM	---> Herbert & Eva Gollnik
R.COM	?
RCHECK.COM	Diskettenprüfung
SIDISC.COM	Silicon-Disc-Programm
SISPOOL.COM	Drucker-Spooler
STARTUP.COM	Kommandos, die bei RESET sofort ausgeführt werden sollen einstellen
STAT.COM	Status abfragen und setzen
SUB.COM	BATCH
SUBMIT.COM	BATCH
SYSCOPY.COM	Systemspuren kopieran (2 Laufw.)
T.COM	?
VDEB.COM	80 Zeichen-Panel
WRTBIOS.COM	CP/M Änderung
WRTCPM.COM	CP/M Änderung
XSUB.COM	BATCH

Wir wollen gerne zu o.g. Programmen und somit zu CP/M an sich eine deutsche Beschreibung zusammenbekommen, und nach und nach im Info veröffentlichen! Bitte schickt uns deutsche Beschreibungen, die mögl. auch auf die Besonderheiten vom MTX hinweisen und vor allem deutlich machen, welche Routinen nicht mit nur einem Laufwerk funktionieren! Wir nehmen dafür gerne NewWord-Dokumente auf Diskette entgegen!

Der Parallele I/O Port (Herbert Herberg)
Das ist der leere Sockel neben der CPU im Grundgerät (d.h. der Tastatur) - übrigens ist nur ein leerer Sockel im Grundgerät, also ist die CPU das große IC neben dem leeren Sockel.
Nun mal etwas genauer der Sache auf den Grund gegangen:
Da gibt es zwei Speicher-IC's für je ein Byte, eines für den Ausgang und eines für den Eingang. Nennen wir diese Speicher mal <A> und <E>.
Dann vom BASIC aus gesehen:
OUT 7,I schreibt den Wert I in <A> (also raus in den Ausgang)
LET I=INP(7) liest den Inhalt von <E> (also vom Eingang)
Damit kann man noch nicht viel machen, aber ich kann von außen das was in <A> steht lesen (hurra - Daten vom MTX nach draußen) und etwas in <E> reinschreiben und das kann der MTX ja mit INP(7) lesen (also Daten rein).

MTX		I/O-Port		Aussenwelt
OUT 7,I	-->	<A>	-->	Pin's POTO,... OTSTB
I=INP(7)	<--	<E>	<--	Pin's PINO,... INSTB

Nun fehlt noch die Beschreibung wie die Außenwelt mit <A> und <E> kommuniziert:
Von <A> nach draußen: Wenn an OTSTB 0 Volt sind, so liegen die acht Bits des Bytes in <A> an den Pins POTO, POT1, ..., POT7. Ist OTSTB auf 5 Volt so sind die POTO,... hochohmig d.h. insbesondere ohne Infomation.
Von draußen nach <E>: Wenn an INSTB 0 VOLT sind, so werden die Werte, die an den Pins PINO, PIN1, ..., PIN7 liegen als Bits nach <E> geschrieben (Bit=0 bei 0 Volt, Bit=1 bei 5 Volt). Ist INSTB auf 5 Volt, so bleibt <E> unverändert, egal was an PINO,... ist.
Also in einfach:

OTSTB = 0	POTO,... = <A>		INSTB = 0	<E> = PINO,...
OTSTB = 1	POTO,... = ???		INSTB = 1	<E> unverändert

D.h. OTSTB bzw. INSTB = 0 Volt bewirkt einen Datenaustausch, ... = 5 Volt verhindert einen Austausch.

Will man mit einem Programm irgendwelche Zustände abfragen, so muß man INSTB an 0 Volt anschließen und die in Frage kommenden Signale an die Pins PINO,... legen. Mit LET I=INP(7) kann man die 8 Signale auf ein Mal einlesen. Der Pin PINi entspricht Bit i.
Um Relais anzusteuern (ACHTUNG nur TTL Relais oder Transistoren zwischenschalten wobei zwischen den Pin am Port und die Basis des Transistors ein 4,7 kOhm Widerstand sollte) muß OTSTB = 0 Volt sein. Mit OUT 7,I werden dann die entsprechenden Ausgägne POTO,... auf 0 bzw. 5 Volt gelegt. Auch hier: POTi entspricht Bit i. Soll z.B. POT2 und POT 5 = 5 Volt, und der Rest = 0 Volt sein: OUT 7,2^2+2^5.

Nachtrag zu Druckercodes (Herbert Herberg)
Im letzten Info (Nummer 3) habe ich EPSON mit DMX -Drucker verglichen, und dabei nicht gewußt, daß die vielgenannten EPSON-Kompatiblen schon untereinander nicht kompatibel sind!!!! Ei gucke da! Nun ja, ich habe den o.g. Vergleich mit DMX 80 & EPSON RX 80 gemacht!

TA-PC Diskettenformat (Herbert Herberg)
Anbei ist eine Diskettenstory von Bernd Preusing!!!
U.a. erwähnte er eine Vermutung, wie man es (auch mit einem Laufwerk) schaffen kann Disketten im TA-PC Format zu lesen und schreiben.
Dafür wird gebraucht: (System-Disc)
DDT.COM OVERLAY8.COM CFIG8.COM
Und falls man nur ein Laufwerk hat: COPY.COM
Ich will hier nur die Eingaben auf der Tastatur beschreiben, um den Vorschlag von Bernd zu verwirklichen:
<BRK> ist die BRK-Taste, / bedeutet RETURN, hinter dem Semikolon sind Kommentare. Die Eingaben sind **dunkel**.

```
A>DDT OVERLAY8.COM/          ; Rufe DDT auf um OVERLAY8.COM zu ändern

>S0DC4/10/./    ; In Speicherstelle 0DC4 schreibe eine 10 für den
                ; Befehl SUB 10 (statt SUB 12) -> Seite 6
                ; Der Punkt beendet das S-Kommando (S=Substitute)
>S0F19/40/00/04/0F/01/97/00/7F/00/C0/00/20/00/02/00/./
                ; Füge die Liste von Seite 7 oben ein
                ; Mit D0F19/ kann man die Eingabe überprüfen und ggf.
                ; mit S0F19/40/ u.s.w. korrigieren
```

Wer nur ein Laufwerk hat, also mit COPY arbeiten muß braucht noch:

```
>S0CBB/3E/00/00/./S0CC5/3E/00/00/./
```

Das waren die Änderungen. Nun Abspeichern:

```
<BRK>
A>SAVE 16 TAPCFMT.COM/    ; Nicht SAVE 18 .... wie bei Bernd
```

SAVE n datei.ext speichert n Sketoren auf Diskette unter dem Namen datei.ext und holt sich die Daten dafür aus dem RAM ab Speicherstelle Hex 100 folgende. Bemerke, daß CP/M ein Kommando das als Name.COM auf Diskette ist von Diskette nach Hex 100 folgende schreibt und dann ein GOTO Hex 100 ausführt. SAVE ist also das Gegenstück dazu.

So nun machen wir uns Laufwerk C (auch wenn wir nur ein Laufwerk haben und also C nicht existiert) zu einem TA-PC und B zu einem MTX:

```
A>TAPCFMT              ; TA-PC Format vorbereiten
A>CFIG8 B:03,C:08      ; Formate bestimmen. Der MTX meldet zwar, daß das
                       ; Laufwerk C TELEVIDEO ist, aber ist es nicht. Wir
                       ; haben den Text 'TELEVIDEO' nicht ersetzt.
```

Kopieren:
Mit 2 Laufwerken: als wenn nichts wäre... wie immer mit PIP

Mit 1 Laufwerk: Wenn die Datei TEST.COM kopiert werden soll
- MTX -> TA, dann ist 'source disk'=MTX-Diskette
  'destination disk'=TAPC-Diskette

```
COPY C:TEST.COM B:TEST.COM
```

- TA -> MTX, dann ist 'source disk'=TAPC-Diskette
  'destination disk'=MTX-Diskette

```
COPY B:TEST.COM C:TEST.COM
```

Die 'system disk' ist die MTX-Diskette mit COPY.COM
ACHTUNG: Mit einem Laufwerk alleine können wir nicht die TAPC-Diskette ins Laufwerk legen und mit CTRL-C oder BRK anmelden, also die DIRECTORY nicht lesen!!! (d.h. müssen wissen was drauf ist!)

Assembler (Herbert Herberg)
1. Das BASIC versteht Z80-Assembler-Codes, aber die Programme DDT.COM und ASM.COM wollen 8080-Assembler-Code. Merkwürdig? Warum will ein Computer mit Z80 CPU für einen Assembler den Code einer anderen CPU? Nun das ist ganz einfach: Die 8080 ist ein Vorläufer der Z80, und die Z80 versteht die Maschinensprache der 8080 und noch einige weitere Befehle (wie z.B. #ED, #B0 = LDIR). Leider gehört aber zum Lieferumfang von CP/M (egal ob der Computer eine 8080 oder Z80 hat) noch DDT.COM und ASM.COM, und die wollen 8080-Code. Für den Benutzer heiß das: Der Hex-Maschinen-Code ist gleich (nur das Z80 mehr kann, und 8080 darum ggf. Code den die Z80 mag nicht versteht). Aber der Mnemonic-Code (d.h. der Code wie wir ihn vom BASIC her kennen mit Dingen wie LD (HL),A, DEC HL, u.s.w.) ist anders. Z.B.:
Hex: #3A,#00,#40 Z80: LD A,(#4000) 8080: LDA #4000
(Viele Assembler schreiben auch statt #4000: 4000H. Dann müssen Hex-Zahlen, die mit einem Buchstaben beginnen noch eine Null vorne weg haben, um sie von Labels zu unterscheiden).
Ich für meinen Teil mag den Code der Z80 lieber, da ich ihn für leichter lesbar halte.
Wer also mit DDT.COM wie in diesem Info beschrieben das OVERLAY8.COM verändern will und dann mit LODB7 den Teil, den Bernd Preusing in seiner Diskettenstory drin hat sehen will wird natürlich die Kommentare nicht finden, aber obendrein auch noch eine anscheinend völlig fremde Sprache. Das ist halt so! Abhilfe schaffen nur ein Z80 Debugger und Assembler (als Ersatz für DDT & ASM)

2. Warum so viel Assembler-Programme???
Nun ja, viele Dinge gehen nicht anders (wie z.B. die Interrupt gesteuerten Routinen als da wären u.a. die Uhr) und sind im BASIC auch unangenehm bzw. nervenaufreibend langsam!!! Wenn mir ein Programm in BASIC geschickt wird veröffentliche ich das natürlich auch, selbst wenn schon eine Assemblerfassung dagewesen ist.
Z.B. eine Hardcopy in Assembler kann man nach #E000 legen, und so bleibt sie im Speicher, wenn ein anderes Programm geladen wird. Man kann sie dann mit PRINT USR(14*16^3) starten oder über den Interrupt steuern.
Assembler bietet auch wesentlich mehr Möglichkeiten, die vom BASIC aus nicht funktionieren weil es sehr viel schneller ist. Ich habe ein Spielprogramm gebastelt, in dem in primitivster Spielstufe ein Zug durchaus 2 Minuten dauerte, nur das in der Spielstufe ein Zufallsgenerator fast genausogute Ergebnisse geliefert hätte. Dann habe ich das Programm in Assembler geschrieben, und den Rechenaufwand je Zug um den Faktor 100 bis 200 erhöht. Fazit ein Zug dauert jetzt schon ganze 2 Sekunden!! D.h. 100 - 200 mal so viele Rechen- und Vergleichsoperationen laufen in 1/60 der Zeit, also ca. 1000 mal so schnell!!! Und wer einmal CHESS (von Continental) in Spielstufe 4 gespielt hat bekommt rasch einen Rochus auf lange Antwortzeiten - die ja vermutlich nicht einmal notwendig sind.

Eine **große Bitte:** Bitte insbesondere Assembler-Programme gut kommentieren, sonst sind viele aufgeschmissen!!

Ein Paar kleine Hinweise zu Z80 Assembler:
Wenn etwas in Klammern steht, so ist das Objekt die Adresse des gemeinten: LD HL,#4000 lädt in HL den Wert #4000, LD HL,(#4000) lädt in HL den Wert der bei #4000 im Speicher steht!! CP (Compare=Vergleiche) betrifft immer das A Register und den genannten Kandidaten.
Vielleicht schaffen wir es mal eine Beschreibung der Assembler-Befehle zusammenzustellen. Wer in der Richtung etwas unternommen hat oder unternehmen will, bitte melden!

NewWord (Bernd Preusing & Herbert Herberg)
Die Druckerkommandos **^P^Q, ^P^W, ^P^E, ^P^R:**
Aufrufen vom Installationsprogramm kann sicherlich jeder, dann die Seite H = Special Patches (= Spezialflicken) anwählen, und dann davon Seite 3, wo unter G-K die o.g. Kommandos genannt sind.
Gebe ich G ein so sehe ich eine Reihe von Hexzahlenpaaren und darunter den Cursor, also soll ich eingeben. Aber wie?????? Na ja, jeder hat sicherlich mal ein Paar Ziffern eingetippt und mal geschaut, was dann passiert ... und war frustiert!
Hier die Lösung (wenn auch nicht gerade befriedigend)
FÜr jedes der 4 Druckerkommandos habe ich 6 Bytes (d.h. 6 Paare von Hex-Ziffern) Platz. Es werden zwar 16 Bytes angezeigt, aber das tut nichts zur Sache. Diese 6 Bytes haben folgende Bedeutung:
1. Byte Anzahl der folgenden Bytes, 2.-5. Byte die zu Übertragenden Zeichen. Will ich z.B. CHR$(27);"-";CHR$(0) rüberschicken, so muß ich unten in der Hex-Reihe (also nach Eingabe von G,H,I od. K) folgendes eingeben: (Nicht die Kommentare in Klammern)
03 (drei Bytes zum Drucker) **1B** (CHR$(27)) **2D** ("-") **00** (CHR$(0))
Klingt gut, nicht? Aber ...
Wenn die o.g. Prozedur korrekt vollstreckt, und das Installationsprogramm mit dem X-Kommando korrekt beendet wurde liefert der erste Test i.a. nicht die erwarteten Ergebnisse:
NewWord ist nicht bereit die Drucker-Control-Codes (die wir gerade mühsam eingebaut haben) mitten in einer Zeile an den Drucker zu schikken, sondern nur am Anfang oder Ende!!!!
Habe ich z.B. auf ^P^Q Unterstreichen an und auf ^P^W Unterstreichen aus, und die folgende Zeile eingegeben:
^P^QTest test test^P^W
So liefert der Drucker: Test test test (also ohne Unterstreichen!), da die Codes vom ^P^Q und ^P^W am gleichen Ende der Zeile verschickt werden!

Turbo Pascal (Herbert Herberg)
Ja, auch hierzu treffen hin und wieder Fragen ein. Nun diese will ich mal allgemein beantworten, da ich auch eine Weile im Handbuch gesucht habe, obwohl es eigentlich trivial ist:
Ausgabe auf den Drucker: write(lst,.....) bzw. writeln(lst,....),
wobei .... die auszugebenen Dinge sind.

Sprites (Anfrage)
Hat jemand mal etwas über Sprites, insbesondere Kollision mit Herausfinden, welche es waren gebastelt/herausgefunden ?

NewWord (Herbert Herberg)
Michael Möwe schickte mir einen allerliebsten Satz aus dem NEWORD DO IT YOURSELF: Now thet you have Newword, you can **misspell the names of thousands of people too**. Ist das nicht herrlich, NewWord gibt uns die Möglichkeit uns bei tausenden von Namen zu verschreiben!!! Mit NewWord soll man ja MERGE PRINTING können! Na was ist denn das? Nun ganz einfach übersetzt: MISCH-DRUCKEN, und bedeuten tut es: DRUCKEN EINES TEXTES UND DABEI ERSETZEN EINIGER TEILE MIT TEXT AUS EINER (o. mehreren) ANDEREN DATEIEN.

Wie das funktioniert will ich mal kurz skizzieren:
1. Datei, Typ Document (d.h. mit D in NewWord erstellen) mit dem eigentlichen Text darin. Nennen wir diese Datei mal TEXT. Und wir wollen diesen Brief an Otto, Peter und Humbert schicken, aber alle mit eigener Anrede und Dank für ein Geschenk:
2. Datei, Typ Nondocument (d.h. mit N in NewWord erstellen) mit den drei Namen und den Geschenken, die wir mal DATA nennen.
Ich brauche eine Dateien nicht als TEXT.**DOC** zu bezeichenen!! NewWord nimmt auch einfach TEXT.
Nun die Datei DATA ist einfach zu basteln:
1. Zeile: **Otto,den Teddy Bär**
2. Zeile: **Peter,"Hibiskus, Buch und Karte"**
3. Zeile: **Humbert,Kinobesuch & Torte**
(In Zeile 2 habe ich Gänsefüßchen gebraucht, damit nicht das Komma hinter Hibiskus als Trennzeichen gilt)
Und ein Beispiel für TEXT:

1. Zeile:	**.df data**	Daten aus der Datei data .df = data file = Daten-Datei
2. Zeile:	**.rv name,geschenk**	Lies die Werte für Name und Geschenk .rv = read variable = lies Variable (Variable=Veränderliche=Platzhalter)
3. Zeile:	**Lieber &name&!**	&name& wird durch den eingelesnenen Namen ersetzt, d.h. das was in name mir .rv eingelesen wurde. Im Text muß &name& stehen, damit NewWord weiß, das das eine Varaible und kein Text ist.
4. Zeile:	**Vielen Dank für &geschenk&.**	Hier wird &geschenk& ersetzt.
5. Zeile:	existiert nicht! Hier ist schon Ende	

Da also keine Leerzeilen oder Seitenvorschübe im Text sind, liefert der Ausdruck (Aufruf zum Drucken mit M und nicht P):
Lieber Otto!
Vielen Dank für den Teddy Bär.
Lieber Peter!
Vielen Dank für Hibiskus, Buch und Karte.
Lieber Humbert!
Vielen Dank für Kinobesuch & Torte.
Seitenvorschub liefert das Kommando **.pa**, das wie .df und .rv in der Datei stehen müsste.
ACHTUNG: Diese Kommandos mit einem Punkt als erstes Zeichen (sog. DOT-COMMANDS) müssen in Spalte 1 anfangen. Wer aus versehen im Text am Anfang einer neuen Zeile einen Punkt hat, der wird die Zeile auf dem Drucker lange suchen können, da alle Zeilen mit einem Punkt in Spalte 1 nicht gedruckt werden.

S C H A D E (Manfred Brodowski, Hölderlinstr. 10,4470 Meppen/Ems)
Manfred verläßt den MTX User-Club Deutschland! Grund: Fa. Profisoft hat und hat ihm seinen MTX 512 nicht repariert und zurückgeschickt, d.h. der Arme saß schon Monate ohne MTX! (Mein zweit-MTX 512 ist schon seit ??? bei Profisoft!). Er hat eine FDX von Vobis gekauft und will sie jetzt originalverpackt verkaufen!
Originalverpackte FDX Floppy-Station zu verkaufen!!!

Kopieren von ROM oder RAM BASIC-Bereichen in eine ASSEM-Zeile
damit der Sermon gelistet werden kann (Herbert Herberg)
Das erledigt mein Programm ROM-GET.BAS (Listing weiter hinten)

Wenn NEW <RET> ASSEM 1 <RET> <RET> eine Zeile der Form
4007 RET liefert, so ist BASBOT=#4000, sonst ist BASBOT=#8000

Folgende **Anpassungen** müssen gemacht werden:
1. Zeile 1, Adresse #8041 (bzw. #4041):
   BASBOT=#4000: OR #60 BASBOT=#8000: OR #A0
   (Dieser Befehl bewirkt, daß die Teile des ROM/RAM nicht irgendwo landen, sondern so, daß von der Adresse nur das höchste Byte falsch ist!)
2. Zeile 1, Adressen #80B4 & #80BB:
   FDX-System: OUT (7),A ROM-System: OUT (0),A
   (Damit wird die ROM-Page angewählt, was beim FDX-BASIC unzulässig ist, und so einfach unterbleibt.)
3. Zeile 7: LET X$="4011"
   BASBOT=#4000: so lassen BASBOT=#8000: LET X$="8011"
   (Die Zahl in den " ist die Adresse von PAGE in Zeile 1)
4. Zeile 30, erster Assembler-Befehl:
   Dies ist die wichtigste stelle und der große Trick des Programms, der nur mit Hilfe vom PANEL bewerkstelligt werden kann!!!
   Ein GOSUB 30 schneidet das Programm hinter Zeile 31 ab, d.h löscht alle Zeilen nach Zeile 31. Nun beim Programmieren gibt man einfach LD HL,#0000 ein, und merkt sich die Adresse des Befehls!!!! d.h. die vierstellige Hex-Zahl vorne in der Zeile (bei mir 86B3).
   Dann zu Ende programmieren!
   So nun geht's los: Ich werde mal annehmen, daß bei Euch der o.g. Befehl LD HL,#0000 bei #86B3 (wie im Listing steht). Dann ist offensichtlich #86B4 dieser Wert plus 1.
   Mit / meine ich die RET-Taste
   PANEL/DFAA7/<BRK> (Dann steht beim Cursor die Speicherzelle FAA7 mit dem Wert, wo das Programm aufhört.) Wir merken uns den Inhalt von FAA7 und von FAA8, z.B. in FAA7: 31, in FAA8: 85 (alles Hex). Dann:
   D86B4/31/85/<BRK> Also in 86B4 (das ist die Adresse vom LD HL,#0000 kommt der Inhalt von FAA7, und in die nächste Speicherstelle der Inhalt von FAA8. Dann muß ein **D**86B3 folgende Werte ab dem Cursor liefern: 21 31 85.
   Verlassen des PANEL mit <BRK> B Y

Wer das Ändern des LD HL,#0000 mit ASSEM 30 u.s.w. versuchen will, kann das gerne tun, wird aber dann zu seinem Erstaunen feststellen, daß sich dann auch der Inhalt von #FAA7, #FAA8 ändert, und damit sein LD HL,... wieder falsch ist.

Nun etwas Erklärung: Wenn das Programm einen ROM/RAM-Bereich kopiert, so wird Zeile 90 als REM-Zeile und Zeile 100 mit dem Assemblercode erzeugt, wobei die REM-Zeile den Zweck hat, zu erreichen, daß die HEX-Adressen sich von den echten (da wo das Zeug herkommt) nur in der höchsten Hex-Ziffer unterscheiden. D.h. z.B. bei BASBOT=#8000 wird #0000-#0100 kopiert nach #A000-#A100. Man kann also dann mit LIST 100,100 oder ASSEM 100 mal schauen. Wer LIST eingibt wird bei Zeile 90 eine Weile warten müssen!!!
Vorgehensweise: Abschneiden des ggf. hinter Zeile 31 kommenden Teil der ja durch ein RUN dort erzeugt worden sein kann (d.h. Löschen von Zeilen 90 und 100), und füllen des Bereiches mit Leerzeichen (damit der REM-Zeile keinen Schrott enthält). Berechnen aus VON die Zieladresse NACH, und die Länge LEN. Dann Berechnen CLEN = Länge der CODE-Zeile (100) und RLEN = Länge der REM-Zeile (90) und einschreiben der Längen, Zeilennummern, u.s.w. Dann wird das neue Ende des BASIC-Programms berechnet und in die Systemvariablen geschrieben und schließlich der eigentliche Kopiervorgang gestartet. Dann STOP.

Eingaben: Alle in Hex ohne #, Page von 0 bis 7.

Wer also ROM's kopieren will, muß mit dem ROM-BASIC arbeiten. Ich habe mir z.B. das BOOT-EPROM kopiert wie folgt:
ROM-GET starten in der MTX512-Version ohne FDX
Page 6, Von 0000, Bis 2000 (damit ist das ROM in Zeile 100)
Dann zweimal auf Kassette gespeichert mit Verify (das Programm ROM-GET incl Zeilen 90 und 100). Das sind übrigens über 16kByte!!!
(Wer faul wie ich ist: E.20 <RET> und dann mit Leerzeichen die 20 überschreiben - schon ist der SAVE-Befehl da!)
FDX-BASIC starten, von Kassette laden und auf Diskette schreiben.

Auch für Nicht-FDXer (Herbert Herberg)
Der CP/M-Tastaturtreiber findet sich auch im BASIC wieder!!!!
In der Speichestelle USERIO = #FD51 steht die Adresse des Tastaturtreibers (#3622 für ROM auf Page 1, #5622 für RAM).
Das, was ab #3622 (#5622) steht ist der Teil, der bei Bernd Preusing auf Seite 2 mit ASCAN bezeichnet ist. Natürlich sind die diversen Adressen anders, aber sonst ist es fast identisch! Vor einem RET-Befehl scheint das BASIC an einigen Stellen noch einen anderen CALL-Befehl (beim FDX: CALL #57C3) auszuführen.
Beachtet bitte, daß bei Bernds Listing nur Hex-Zahlen auftauchen, und das # überall fehlt!!
Ach so, wie schaue ich mir diese Dinge beim BASIC an? Nun ganz einfach: mit dem PANEL (s. BASIC-Handbuch) oder mit ROM-GET (s.o.)

New-Word auf ROM (Klaus Muerling)
Für DM 180.- verkauft Klaus Muerling, Mainstr. 34, 8702 Margretshöchheim sein NewWord auf EPROM (für MTX mit mindestens 64k RAM). Neupreis DM 298.-

FDXB / BASIC (Jürgen Adamczak)
Jürgen hat entdeckt, daß sein BASIC LEN(A$) akzeptiert, d.h. ohne Leerzeichen hinter LEN (also nicht LEN (A$)). Oder habe ich da was falsch verstanden?

BASIC - Speichern von Variablen mit dem Programm (Herbert Herberg)
Diese Beschreibung gilt für Kassette und Diskette!!
Häufig möchte man Variablen mit dem Programm abspeichern, und bei Arrays (d.h. Variablen die mit DIM angelegt wurden) klappt das auch ohne weiteres, aber die anderen? Da hilft nur ein Trick:
Will ich I mit abspeichern, dann ersetze ich I durch I(1) und packe oben ins Programm DIM I(1). Hat man I,J,K,L so ersetze I durch I(1), J durch I(2), ... und definiere I mit DIM I(4). Um aber nicht überall diese dummen I(1) u.s.w schreiben zu müssen geht auch folgendes:
Wenn I,J,K,L,X,Y mit abgespeichert werden sollen, so sieht das wie folgt aus:

Altes Programm:

```
 10 DIM A(10)
 20 REM Programmanfang
    hier kommt das Programm

1000 REM Abspeichern als Unterprogramm
1010 SAVE "Programm"
1020 RETURN
```

Neue Version:

```
 10 DIM ISAVE(6),A(10)
 20 REM Programmanfang
    hier kommt das Programm

1000 REM Abspeichern als Unterprogramm
1001 LET ISAVE(1)=I: LET ISAVE(2)=J: LET ISAVE(3)=K
1002 LET ISAVE(4)=L: LET ISAVE(5)=X: LET ISAVE(6)=Y
1010 SAVE "Programm"
1011 LET I=ISAVE(1): LET J=ISAVE(2): LET K=ISAVE(3)
1012 LET L=ISAVE(4): LET X=ISAVE(5): LET Y=ISAVE(6)
1020 RETURN
```

Dazu ein konkretes erbetenes Beispiel:
MTX Handbuch deutsch S. 57 oben bzw. englisch S. 63 mitte (Sortieren)
Folgende Zeilen müssen dazu eingefügt werden:
(Sogar mit VERIFY, d.h. Zeile 430,440,450 sind Luxus!)

```
  11 DIM ISAVE(1)
 310 PRINT: PRINT: PRINT "Soll geSAVEd werden ?"
 320 LET N$=INPUT$: IF N$="" THEN GOTO 320
 330 IF N$<>"J" AND N$<>"j" THEN STOP
 400 REM Abspeichern
 410 LET ISAVE(1)=I
 420 SAVE "SORT"
 430 PRINT "Bitte zurückspulen für VERIFY"
 440 INPUT "Fertig",N$
 450 VERIFY "SORT"
 460 LET I=ISAVE(1)
 470 GOTO 150
```

CP/M BDOS (Herbert Herberg)
Um unter CP/M auf den Rechner zuzugreifen verwendet man das BDOS = Basic Disc Operating System (=Grunddiskettenbetriebssystem).
Aufruf: CALL #0005 wobei im Register C steht, was man will:
Unten eine Beschreibung der BDOS-Funktionen, dabei werden die Parameter wie folgt beschrieben: Hinter dem E: stehen die, die vor dem CALL gebraucht werden, hinter R: die Rückgabeparameter. Mit den Großbuchstaben sind Register gemeint! Als Faustregel kann man sich merken, daß ein Byte, das vom Programm an das BDOS gehen im Register E stehen muß, und das BDOS Bytes im A-Register übergibt. Doppelbytes: zum BDOS in DE, vom BDOS in HL.

Für Dateizugriffe wird noch der FCB = File Control Block (= Dateikontrollblock) benötigt:

Byte	Beschreibung
0	Laufwerk. 0=aktuelles, A=1, B=2, ...
1-8	Dateiname in ASCII
9-11	Extension wobei Bit 7 eine Sonderfunktion hat: Byte 9: Bit 7=1 heißt read/only Byte 10: Bit 7=1 heißt sys-Datei
12	Extenstion-Nummer (einige Dateien brauchen mehr als einen Dir-Eintrag: sog. Extensions)
13,14	Interne Verwendung
15	Record-Zähler (0..128)
16-31	Interne Verwendung
32	Aktuelle Record-Nummer für seq. Zugriff (Muß am Anfang auf 0 gesetzt werden.)
33-35	Random-Record-Position, Low Byte:33, Mid Byte:34, High Byte:35

Für Rename (BDOS Funktion 23) muß der FCB etwas anders aussehen:
Byte 0-15 s.o., dann in Byte 16-31 genauso der 2. Name!

Also nun mal Programm: Der Aufruf einer BDOS-Funktion geschieht folgendermaßen:

```
   LD    C,Funktionsnummer
   CALL  #0005
```

Das ist alles, aber natürlich müssen die Parameter bearbeitet werden!

Funktion		Parameter
0	System-Neustart	-
1	Konsolen-Eingabe	R: Zeichen in A
2	Konsolen-Ausgabe	E: Zeichen in E
3	Reader-Eingabe	R: Zeichen in A
4	Punch-Ausgabe	E: Zeichen in E
5	Drucker-Ausgabe	E: Zeichen in E
6	Konsolen Ein- & Ausgabe	E: Zeichen in E für Ausgabe, E: #FF in E für Eingabe -> R: Zeichen in A
7	I/O-Byte holen	R: I/O-Byte in A
8	Setzen I/O-Byte	E: I/O-Byte in E
9	Ausgabe eines Strings	E: Adresse des Strings in DE, Ende des Strings markiert durch $
10	Zeile von Konsole	E: Adresse des Puffers, wobei das erste Byte die max. Anzahl der Zeichen ist.
11	Konsolen-Status	R: Null in A wenn nichts da
12	Versionsnummer	R: Versionsnummer in HL
13	**Rücksetzen des Disk**	Um R/O-Error zu vermeiden, ersetzt das CTRL-C in CP/M
14	Laufwerk anwählen	E: Nummer des Laufwerkes in E
15	Eröffnen einer Datei	E: FCB in DE, R: #FF in A falls nicht ex.
16	Schließen Datei	E: FCB in DE, R: #FF in A falls nicht ex.
17	Suche nächsten Eintrag	E: FCB in DE, R: #FF in A falls Ende
18	wie 17	
19	Löschen einer Datei	E: FCB in DE, R: #FF in A falls nicht ex.
20	Sequentiell lesen	E: FCB in DE, R: #FF in A falls E.O.F.
21	Sequentiell schreiben	E: FCB in DE, R: #00 in A falls o.k.
22	Datei erzeugen	E: FCB in DE, R: #FF in A: Directory voll
23	Datei umbenennen	E: FCB' in DE, R: #FF in A wenn nicht ex.
24	Login-Vektor holen	R: In HL: welche Laufwerke aktiv waren
25	Aktuelles Laufw. holen	R: Nr. des aktuellen Laufwerkes in A
26	DMA-Adresse setzen	E: In DE Adresse, Diskettenoperationen tauschen Daten zwischen dem Puffer mit der DMA-Adresse und der Diskette aus.
27	Bit-Map-Adresse	R: In HL Adresse der Bit-Map akt. Laufw.
28	R/O setzen	Setzt akt. Laufw. auf read-only
29	R/O Vektor holen	R: In HL: welche Laufw. read-only
30	Datei-Attribute setzen	E: FCB in DE
31	Adresse Laufw.-Param	R: HL Adresse DPB (s. Bernd Preusing Diskettenstory)
32	User-Code	E: Code in E zum Setzen, E: #FF in E, R: akt. Code in A
33	Random-Lesen	E: FCB in DE, R: Errorcode in A
34	Random-Schreiben	E: FCB in DE, R: Errorcode in A
35	Berechnen Datei-Größe	E: FCB in DE, Pos E.O.F in FCB
36	Random-Positionieren	E: FCB in DE

Für sequentielle und random Dateizugriffe steht die Adresse der Daten in der Datei (Zeiger auf Position in der Datei bzw. record-Pointer) im FCB im Byte 32 bzw. den Bytes 33-35. Die Daten werden zwischen der Diskette und der DMA-Adresse (d.h. dem Speicher ab DMA-Adresse) ausgetauscht.

FDXB und BDOS (Herbert Herberg)

H u r r a ! ! ! ! Ich habe es gefunden!

Statt CALL #0005 in CP/M ist der Aufruf im FDX-BASIC **CALL #6308.**
Aber nicht alle o.g. Funktionen sind verfügbar: Nur ab Funktion **12,** aber die restlichen dafür auch. Die ersten brauchen wir ja nicht, da die Bildschirm-Zugriffe,... schon im BASIC enthalten sind.
Nun zu dem berüchtigten Problem: Diskettenwechsel. Kein Problem mehr, man braucht nur folgendes Assembler-Programm:

```
   LD    C,13
   CALL  #6308
```

Und schon ist die Diskette angemeldet, die im Laufwerk liegt, und kann mit DISC SAVE bearbeitet werden, ohne einen read/only Fehler zu kriegen.
Mutig wie ich bin, habe ich an die Adresse des DISC QUIT Befehls diese Reset-Routine für Diskette geschrieben und hatte damit bis dato keine Probleme,... aber auf eigene Gefahr!!! Mit folgendem Programm kann man aus DISC QUIT ein Disketten-Reset:

```
      LD HL,RES    ; Reset-Routine
      LD DE,#770D  ; Adresse von DISC QUIT
      LD BC,5      ; 5 Bytes
      LDIR
      RET
RES:  LD C,13      ; Funktion 13 = Reset
      JP #6308     ; Kein CALL damit ein RET nicht gebraucht wird
```

Die Diskettenbeschreibung DPB (Disk-Parameter-Block), die von Bernd Preusing in seiner Diskettenstory gut beschrieben wurde (in diesem Info) steht im FDX-BASIC ab #6F63.
Ob bei den BDOS-Aufrufen im FDX-BASIC die Parameter genauso aussehen wie in CP/M weiß ich nicht! Mir ist schon mit der o.g. Funktion 13 gedient.
Für mutige: Hier die Beschreibung, wie man den Befehl DISC QUIT in FDXB.COM (d.h. dem FDXB auf Diskette) ersetzt, so daß gleich nach dem Aufruf des BASIC's der Befehl DISC QUIT ein Reset der Diskette bewirkt. Verlassen des BASIC's geht ja sowieso schneller mit RESET (die beiden Tasten neben der Leertaste):
Wie schon weiter oben: / = <RET>, Eingaben in **fett:**

```
A>DDT FDXB.COM/              ; DDT aufrufen
>D7810/                      ; Mal sehen ob wir richtig sind
   muß als erste Bytes liefern: 11 30 77 DD 21 A5 7F ...
>S7810/0E/0D/C3/08/63/./ ;Einfügen LD C,13    JP #6308
><BRK>                       ; Schluß DDT
A>SAVE 136 FDXBNEU.COM/      ; Neues FDXB abspeichern
A>FDXBNEU/                   ; Aufrufen
PANEL                        ; Assembler-Monitor
L770D                        ; Mal Prüfen
   Dann muß da stehen:    LD C,#0D     JP #6308
B Y                          ; Panel verlassen
Fertig!
```

ACHTUNG: Ich übernehme keine Garantie, daß die o.g. Änderungen nicht für anderes im FDXB tödlich sind, aber ich habe noch nie Probleme damit gehabt ... und halte es auch für sehr unwahrscheinlich!

DISC-Befehl-Jump-Table (Herbert Herberg)
(Diese Tabelle steht bei #7FCE)

Befehl	Adresse in HEX	Befehl	Adresse in HEX
DISC LOAD	7854	DISC SAVE	77D7
DISC PRINT	7B26	DISC INPUT	7AB6
DISC LINE INPUT	7AAC	DISC OPEN	78E0
DISC CLOSE	79C6	DISC KILL	79B0
DISC READ	7BD9	DISC WRITE	7BF9
DISC DIR	7747	DISC ERA	7793
DISC TYPE	79C1	DISC REC	7A52
DISC REN	79F6	DISC **RUN**	7A26
DISC QUIT	ZZ0D	DISC EOF	7D60

Frage: Was macht DISC RUN ??????

Noddy auf den Drucker (Jürgen Adamczak)
Das in der letzten Info aufgeführte Programm zum Ausdrucken einer NODDY-Seite erscheint mit sehr umfangreich, da es nicht den im NODDY-Befehlssatz vorhandenen Befehl '*LIST' ausnutzt. Ich lasse eine NODDY-Seite folgendermaßen ausdrucken:
Zunächst richte ich eine zusätzliche NODDY-Seite, der ich den Namen PROG gebe, mit folgendem Inhalt ein:

```
PROG *L TEXT. *R
```

TEXT ist der Name der NODDY-Seite, die ausgedruckt werden soll.
Anschließend gebe ich diese BASIC-Befehle ein:

```
10 LPRINT CHR$(27);"Q";CHR$(39 oder 80)
20 PLOD "PROG"
30 LPRINT CHR$(27);CHR$(64)
```

Dazu (Herbert Herberg) folgendes:
BASIC-Zeile 10 legt den rechten Rand auf Spalte 39 (für 40-Zeichen BASIC) bzw. Spalte 80 (für 80-Zeichen NODDY) fest, und das kann nicht jeder Drucker!!!!!! Darum diese aufwendigen Programme!!!

CP/M *.DOC - Files auf der SYSTEM-DISC (Herbert Herberg)
Wer diese Files mit TYPE oder PIP anschaut wird eine Überraschung erleben/erlebt haben. Nun ja, diese Dateien sind ja schließlich auch NewWord Documents. Also müssen sie erst mal auf die NewWord-Diskette kopiert werden, und dann mit NewWord gedruckt werden. Aber in NewWord würde ich die Files erst mal als Document eröffnen (d.h. Bearbeiten), und mal sehen, ob nicht irgendwelche dusseligen Dot-Commands (d.h. Zeilen mit einem Punkt in Spalte 1) gleich oben im Text stehen, und diese ggf. entfernen!
Wie kriege ich diese Dateien auf eine Diskette mit NewWord? Nun:
1 Laufwerk: mit COPY.COM (siehe Info1/2, FDX-Handbuch)
2 Laufwerke: wie oben, oder mit PIP.COM.

C L U B T R E F F E N

Termin: **Samstag 27. und Sonntag 28. April 1985**, Ort: **Hamburg**

Wir wollen uns in Räumen der Uni Hamburg treffen, um Informationen auszutauschen, einander kennenzulernen und vor allem aufgetretene Probleme zu lösen ... und vieles anderes! Und das soll alles mit einem MTX/FDX vor Ort ablaufen (sonst sitzen wir ja nur und kopieren!). Wer etwas vorführen möchte: Disc/Cass mitbringen! Wir haben auch 40-Zeichen-Schirm angeschlossen! Einige Knabbereien und Getränke werde ich besorgen. Vorschläge hierzu sind willkommen!
Keine Raubkopien auf dem Treffen!!! (das gibt nur Ärger!!)

Da sich hoffentlich auch Mitglieder von außerhalb Hamburgs aufraffen, möchte ich alle Hamburger bitten mich wissen zu lassen, wieviele Gäste Ihr unter welchen Bedingungen (Schlafsack, ...) für die Nacht von Samstag auf Sonntag aufnehmen könnt. Das würde auch die Möglichkeit zu persönlichen Unterhaltungen (sogar mit MTX) geben!

Nun zu den Zeiten:
Samstag: Ab 14.00 Uhr an der Uni Hamburg, Von Melle Park 9 (im Gebäude ausgeschildert). Irgendwann abends Ende um u.a. Abendbrot zu essen und ggf. noch eine Kneipentour (oder was immer anliegt).
Sonntag: Ab 11.00 Uhr nochmals Uni HH (falls gewünscht) und irgendwann ist dann auch Schluß.

Und die Kosten:
Wir wollen ja sicherlich gerne beim Reden auch mal was trinken und essen. Ich habe deshalb vor für die Zeit an der Uni Getränke e.t.c. zu besorgen. Für Abendbrot zu sorgen halte ich für zu aufwendig. Ggf. könnt Ihr ja zu hause/bei Euren Gastgebern essen. Um es denjenigen, die von weiter her anreisen erschwinglicher zu machen sollen diejenigen, die dichter bei wohnen für die o.g. Genußmittel bezahlen.
In Zahlen: Für diejenigen, deren Postleitzahl mit einer 2 beginnt (in Deutschland) DM 15.- Falls noch was übrigbleibt wird das auf den Konten gutgeschrieben!

WICHTIG: Bitte anmelden bis spätestens 17. April 1985, und dabei die unten genannten Fragen alle beantworten (das ist ein **Muß**) sowie das Geld als Scheck befügen oder auf's Club-Konto überweisen!

1. **Name**
2. **Interessenschwerpunkte / Vorführpläne**
3. **Änderungsvorschläge**
4. **Übernachtung:** entweder a) oder b)
   a) Ich kann .. Mitglieder für eine Nacht (Schlafsack?) aufnehmen
   b) Ich möchte gerne in Hamburg übernachten können
   c) Gast- bzw. Gastgeber-Wunsch
5. **Besondere Wünsche**

Treffen
Wer auch ein Club-Treffen veranstalten will (vielleicht mal weiter im Süden) ... gerne!!!! Das Hauptproblem ist i.a. der Raum!!
Nur: Auf einem Treffen sollte keine!!!!! Rubkopien auftauchen.

Diskettenkonvertierung TA-PC -> MTX (Herbert Herberg)
Anbei ist ein Z80-Assemblerlisting eines Konvertierungsprogrammes, das die Daten auf einer TA-PC-Diskette so verschiebt, daß der MTX sie lesen kann.
ACHTUNG: Nur einmal auf eine Diskette loslassen!!! Das Programm verschiebt alles um 76 Sektoren nach vorne, und zweimal verschieben heißt um 152 Sektoren ... also Mist.

Nun zur Frage wie bekomme ich das Programm auf meinem MTX zum laufen:
1. Abtippen und mit einem Z80-Assembler bearbeiten (falls vorhanden)
2. Übersetzen in 8080-Assembler-Code, und dann eintippen und mit dem Programm ASM.COM von der System-Disc übersetzen;
3. Mit FDXB: Bloß nicht!;
4. Den HEX-Dump (s.u.) mit DDT.COM ab Hex 100 eingeben (S-Kommando), wenn fertig Ctrl-C, und SAVE 5 TATOMTX.COM;
5. Das Programm gibt es bei mir gratis auf Diskette (s. Liste Seite 2)

Hexdump des Programmes

```
0100  C3 35 02 C3 1B 4A C3 1E  4A C3 21 4A C3 24 4A C3
0110  27 4A C3 2A 4A C3 30 4A  00 00 00 00 00 00 00 00
0120  00 00 00 00 00 0C 0A 0A  0D 4B 6F 76 65 72 74 69
0130  65 72 75 6E 67 20 65 69  6E 65 72 20 54 41 2D 50
0140  43 20 44 69 73 6B 65 74  74 65 20 61 75 66 20 4D
0150  54 58 20 46 6F 72 6D 61  74 0A 0A 0A 0D 20 20 20
0160  20 20 20 20 28 43 29 20  31 39 38 35 20 20 48 65
0170  72 62 65 72 74 20 48 65  72 62 65 72 67 0A 0A 0A
0180  0D 44 61 73 20 50 72 6F  67 72 61 6D 6D 20 61 72
0190  62 65 69 74 65 74 20 6D  69 74 20 6E 75 72 20 65
01A0  69 6E 65 6D 20 4C 61 75  66 77 65 72 6B 3A 20 42
01B0  0A 0A 0A 0D 42 69 74 74  65 20 64 69 65 20 54 41
01C0  2D 50 43 20 44 69 73 6B  65 74 74 65 20 69 6E 20
01D0  4C 61 75 66 77 65 72 6B  20 42 20 65 69 6E 6C 65
01E0  67 65 6E 0A 0D 55 6E 64  20 3C 52 45 54 55 52 4E
01F0  3E 20 64 72 7D 63 6B 65  6E 0A 0A 0A 0D 24 0A 0A
0200  0D 4B 6F 6E 76 65 72 74  69 65 72 75 6E 67 20 66
0210  65 72 74 69 67 21 0A 0A  0A 0D 42 69 74 74 65 20
0220  3C 52 45 54 55 52 4E 3E  20 64 72 7D 63 6B 65 6E
0230  0A 0A 0A 0D 24 11 25 01  0E 09 CD 05 00 CD B8 02
0240  2A 01 00 11 FD B5 19 22  1B 01 21 03 01 23 4E 23
0250  46 78 E5 2A 1B 01 09 44  4D E1 70 2B 71 23 23 23
0260  B7 C2 4E 02 3E 04 32 1F  01 3E 18 32 20 01 3E 02
0270  32 21 01 3E 00 32 22 01  0E 01 CD 03 01 4E 23 46
0280  ED 43 1D 01 21 4E 03 22  23 01 06 00 04 C5 CD C2
0290  02 C1 B7 20 F7 21 4E 03  22 23 01 C5 CD 06 03 C1
02A0  05 20 F8 3A 1F 01 FE 62  20 DA 11 FE 01 0E 09 CD
02B0  05 00 CD B8 02 C3 00 00  0E 01 CD 05 00 FE 0D 20
02C0  F7 C9 3A 1F 01 4F 06 00  CD 06 01 3A 20 01 4F 06
02D0  00 ED 5B 1D 01 CD 15 01  4D 06 00 CD 09 01 ED 4B
02E0  23 01 CD 0C 01 CD 0F 01  3A 20 01 3C 32 20 01 FE
02F0  1A 3E FF C2 43 03 3E 00  32 20 01 3A 1F 01 3C 32
0300  1F 01 3E 00 18 3D 3A 21  01 4F 06 00 CD 06 01 3A
0310  22 01 4F 06 00 ED 5B 1D  01 CD 15 01 4D 06 00 CD
0320  09 01 ED 4B 23 01 CD 0C  01 CD 12 01 3A 22 01 3C
0330  32 22 01 FE 1A 20 0C 3E  00 32 22 01 3A 21 01 3C
0340  32 21 01 2A 23 01 11 80  00 19 22 23 01 C9
```

Hardware
Jemand rief mich an und erwähnte, daß sein selbstgebasteltes Druckerkabel mit 2 Metern, und das FDX-MTX-Kabel mit 1,5 Metern (er kann also mit der Tastatur spazieren gehen) tadellos funktionieren! Den Busstecker kann man im Elektronik-Fachhandel erhalten, das Kabel auch (ggf. zwei mal 30 Pins). Elektronic Bauteile Hans Lück, Lübecker Str. 134, 2000 Hamburg 76, Tel. 040 - 250 74 25 (das ist direkt bei der U-Bahn U1, Haltestelle Wartenau, Ausgang Richtung Innenstadt) hat die richtigen Teile. Er hat auch vieles andres preiswert, wie z.B. den DART für ca. DM 18.-. HW Elektronik, Eimsbüttler Chaussee 79, 2000 Hamburg 19, Tel.040-439 68 48 & 439 00 19 (Versand ab DM 30.- per NN) hat sehr preiswerte IC's: Z80A CPU 5.50, Z80A DART 15.75, und für Geschwindigkeitsfans: Z80A CPU 11.75, DART 23.40, CTC 13.-.

RS 232C-Karte Aufrüstung für unter DM 27.- ? (H.Wenzek & H. Herberg)
Die mitgelieferte RS 232C-Karte gibt es ja in drei Ausführungen:
1. Voll bestückt + Buchsen für Anschluß an der Rückwand des MTX
2. Teilw. " + " ....
3. Teilw. " ohne Buchsen
Bei den DM 27.- geht Hgen sicherlich von Typ 2 aus!
Es fehlen dann noch folgende IC's auf der Platine:
2x **MC 1489** für Sockel 12 & 14, 1x **MC 1488** für Sockel 13 und ein **Z80A-DART** für den großen Sockel. Beim Einstecken darauf achten, daß die IC's richtigherum sitzen. Beim IC ist Pin 1 an dem Ende mit einer kleinen Vertiefung im IC, auf der Platine steht eine kleine 1.
Aber leider reicht das so noch nicht: Das Signal DARTEN wird noch nicht erzeugt (welch ein Anschiß!). Nun ja, DART wird angesprochen über Port #0C-#0F, d.h. A_7-A_4 = 0, A_3-A_2 = 1 und IOREQ = 0. Zum Glück gibt's auf der Platine das Signal A432 ($\overline{A432}$ = 0 wenn A_4 = 0 und A_3-A_2 = 1). Folglich müssen die fünf Signale A_7, A_6, A_5, A432 und IOREQ = 0 sein. Das können wir mit einem 74**LS**32, einem 4-fach Oder-Gatter, erledigen. Die Pins des 74LS32 müssen wie folgt beschaltet werden:
1-11, 2-8, 3-4, 5-4PLA, 6-35DART, 7-10PLA, 9-7PLA, 10-8PLA, 12-9PLA, 13-11PLA, 14-20PLA. Das müsste so eigentlich funktionieren!

Dicke und dünne **Info's**
Dieses Info ist viel dicker geworden, als erwartet!! Ich habe mich anfangs mal gefragt: 'Wie schaffe ich es eigentlich ein Info mit mehr als 5-10 Seiten monatlich herauszugeben ???'. Nun ja, mittlerweile scheine ich in NewWord mehr zu produzieren (wobei Ihr alle auch mit Fragen helft!), ganz zu schweigen von den vielen hervorragenden Beiträgen, wo ich den Vorzug habe sie als erster zu lesen! Und oft bin ich einfach platt, was da alles so faszinierendes zusammenkommt!!! Euch allen ein ganz herzlichen Dank für die Hilfe!! Das klappt ganz toll!!! Die große Auflage und Seitenzahl hat übrigens den Preis je Seite gedrückt!!! Diese Info kostet zwar über DM 10.-, aber wenn ich die Seitenzahl reduziere ist auch das Angebot weg!!

Entschuldigt bitte, daß ich bei Schneiden von Bernd-Preusings Tastaturtreiber-Listing die Seitenüberschriften abgeschitten habe! Ich muß die Zettel auch DIN A4-Länge kürzen, und habe aus Versehen am falschen Ende geschnippelt!

1 oder 2 System-Discs
Auf der teilweise gelieferten 2. System-Disc sind nur drei Files, die auf der Haupt-System-Disc auch sind: FRM.COM, SYS.COM und COPY.COM, d.h. die SIngle-Disc-Routinen.

Fragebogen / Statistik
111 Namen stehen auf der Liste, 97 sind Mitglieder, **nur** 41 Fragebögen sind eingegangen: 28 FDX, 28 MTX 512, 13 MTX 500

W I C H T I G E N e u e r u n g!!
Kontostand steht jetzt hinter dem **#** über Eurer Anschrift!!! Auch hier tritt hin und wieder ein Minus-Zeichen auf! Bitte beachtet: -.15 heißt minus 15 Pfennig, plus 15 Pfennig: 0.15.

Dieses Info kostet: DM 11.05

```
;*****************************************
;*                                       *
;* Disketten-Konvertierung TA-PC AUF MTX *
;*                                       *
;*           (C)  Herbert Herberg        *
;*                  Sonnenau  2          *
;*                2000 Hamburg 76        *
;*                                       *
;*****************************************

          ORG       100H
          JP        START

MAXSEC    EQU       1AH         ;Max. Sektor +1
MINSEC    EQU       0           ;Min. Sektor
MAXTRK    EQU       98          ;Max. Track +1
RDTRKS    EQU       4           ;Anfangs-Track TA-PC
RDSECS    EQU       24          ;Anfangs-Sektor TA-PC
WRTRKS    EQU       2           ;Anfangs-Track MTX
WRSECS    EQU       MINSEC      ;Anfangs-Sektor MTX

BIOS:                           ;BIOS Einsprünge
SELDSK:   JP        4A1BH       ;Laufwerk anwählen
SETTRK:   JP        4A1EH       ;Track anwählen
SETSEC:   JP        4A21H       ;Sektor anwählen
SETDMA:   JP        4A24H       ;DMA anwählen
READ:     JP        4A27H       ;Sektor lesen
WRITE:    JP        4A2AH       ;Sektor schreiben
SECTRA:   JP        4A30H       ;Sektor-Übersetzung
          DEFB      0,0,0       ;Ende der Tabelle

OFFSET:   DEFS      2           ;Offset des BIOS
XLT:      DEFS      2           ;Übersetzungstabelle
TRKR:     DEFS      1           ;Track Read
SECR:     DEFS      1           ;Sector Read
TRKW:     DEFS      1           ;Track Write
SECW:     DEFS      1           ;Sector Write
PUFF:     DEFS      2           ;Puffer-Adresse

LOGON:    DEFB      0CH,0AH,0AH,0DH
          DEFB      'Kovertierung einer TA-PC Diskette auf MTX Format'
          DEFB      0AH,0AH,0AH,0DH
          DEFB      '         (C) 1985  Herbert Herberg'
          DEFB      0AH,0AH,0AH,0DH
          DEFB      'Das Programm arbeitet mit nur einem Laufwerk: B'
          DEFB      0AH,0AH,0AH,0DH
          DEFB      'Bitte die TA-PC Diskette in Laufwerk B einlegen'
          DEFB      0AH,0DH
          DEFB      'Und <RETURN> drücken'
          DEFB      0AH,0AH,0AH,0DH,'$'
LOGOU:    DEFB      0AH,0AH,0DH
          DEFB      'Konvertierung fertig!'
          DEFB      0AH,0AH,0AH,0DH
          DEFB      'Bitte <RETURN> drücken'
          DEFB      0AH,0AH,0AH,0DH,'$'


START:    LD        DE,LOGON
          LD        C,9
          CALL      5                 ;Textausgabe
          CALL      WAIT              ;Warten auf RETURN

;Setzen der BIOS-Einsprünge
          LD        HL,(0001H)        ;Warm-Boot-Adresse
          LD        DE,0-4A03H        ;Wurde mit 4A03H angenommem
          ADD       HL,DE
          LD        (OFFSET),HL       ;Korrekturdifferenz
          LD        HL,BIOS           ;BIOS-Einsprungtabelle
          INC       HL
JUMPS:    LD        C,(HL)
          INC       HL
          LD        B,(HL)            ;Hole Adresse
          LD        A,B               ;Für Ende-Test
          PUSH      HL
          LD        HL,(OFFSET)
          ADD       HL,BC             ;Plus Offset
          LD        B,H
          LD        C,L
```

```
            POP     HL
            LD      (HL),B
            DEC     HL
            LD      (HL),C          ;Korrekte Adresse einschreiben
            INC     HL              ;Nächster
            INC     HL
            INC     HL
            OR      A               ;Ende der Tabelle: Null
            JP      NZ,JUMPS

;Initialisieren der Pointer
            LD      A,RDTRKS        ;Anfangs-Track für Read
            LD      (TRKR),A
            LD      A,RDSECS        ;Anfangs-Sektor für Read
            LD      (SECR),A
            LD      A,WRTRKS        ;Anfangs-Track für Write
            LD      (TRKW),A
            LD      A,WRSECS        ;Anfangs-Sektor für Write
            LD      (SECW),A

            LD      C,1             ;Laufer k B
            CALL    SELDSK          ;DPH-Adresse in HL
            LD      C,(HL)
            INC     HL
            LD      B,(HL)          ;XLT-Adresse in BC
            LD      (XLT),BC        ;XLT = Übersetzungstabelle
                                    ;Wäre für MTX nicht notwendig, da Null

;Konvertieren
MOVE:       LD      HL,PUFFS        ;Puffer-Anfang
            LD      (PUFF),HL
            LD      B,0             ;Sektor-Zähler
LESEN:      INC     B
            PUSH    BC
            CALL    READS           ;Lies Sektor und erhöhe Sektor
            POP     BC
            OR      A
            JR      NZ,LESEN        ;Lies bis Spurende
            LD      HL,PUFFS        ;Anfang Puffer
            LD      (PUFF),HL
SCHRE:      PUSH    BC
            CALL    WRITES          ;Schreib Sektor und erhöhe Sektor
            POP     BC
            DEC     B
            JR      NZ,SCHRE        ;Bis alles gelesene weg
            LD      A,(TRKR)        ;Fertig  ?
            CP      MAXTRK
            JR      NZ,MOVE         ;Nein

            LD      DE,LOGOU
            LD      C,9
            CALL    5               ;Textausgabe
            CALL    WAIT
            JP      0               ;Warm-Boot

;Warten auf RETURN
WAIT:       LD      C,1             ;Konsolen-Eingabe
            CALL    5
            CP      0DH             ;RETURN ?
            JR      NZ,WAIT         ;Nein
            RET

;Sektor lesen und Pointer erhöhen
READS:      LD      A,(TRKR)
            LD      C,A
            LD      B,0
            CALL    SETTRK          ;Track anwählen
            LD      A,(SECR)
            LD      C,A
            LD      B,0
            LD      DE,(XLT)
            CALL    SECTRA          ;Sektor-Übersetzung
            LD      C,L
            LD      B,0             ;Phys. Sektor in BC
            CALL    SETSEC          ;Sektor anwählen
            LD      BC,(PUFF)
            CALL    SETDMA          ;DMA Anwählen
```

```
          CALL      READ                ;Sektor lesen
          LD        A,(SECR)
          INC       A                   ;Nächster Sektor
          LD        (SECR),A
          CP        MAXSEC              ;Nächste Track ?
          LD        A,0FFH              ;Nein, NonZero: nicht EOT
          JP        NZ,ENDE
          LD        A,MINSEC
          LD        (SECR),A
          LD        A,(TRKR)
          INC       A
          LD        (TRKR),A            ;1. Sektor nächster Track
          LD        A,0                 ;Zero: EOT
          JR        ENDE

;Sektor schreiben und Pointer erhöhen
WRITES:   LD        A,(TRKW)
          LD        C,A
          LD        B,0
          CALL      SETTRK
          LD        A,(SECW)
          LD        C,A
          LD        B,0
          LD        DE,(XLT)
          CALL      SECTRA
          LD        C,L
          LD        B,0
          CALL      SETSEC
          LD        BC,(PUFF)
          CALL      SETDMA
          CALL      WRITE
          LD        A,(SECW)
          INC       A
          LD        (SECW),A
          CP        MAXSEC
          JR        NZ,ENDE
          LD        A,MINSEC
          LD        (SECW),A
          LD        A,(TRKW)
          INC       A
          LD        (TRKW),A

;Puffer-Pointer erhöhen
ENDE:     LD        HL,(PUFF)           ;Pufferpointer weiter
          LD        DE,80H
          ADD       HL,DE
          LD        (PUFF),HL
          RET

PUFFS     EQU       $                   ;Puffer-Anfang

          END
```

Bitte Beschreibung im Info 4 beachten!

ACHTUNG: Je TA-PC-Diskette nur e i n Mal aufrufen!!!
(Sonst ist alles futsch!)

Es sind nach der Konvertierung die Dateien der Diskette, deren Directory-Einträge in den ersten 2kByte der Directory stehen lesbar! Auf die zweite Hälfte der Directory kann der MTX nicht zugreifen!!!

Das obige Listing ist für einen Z80-Macro-Assembler, d.h. ist so weder für FDXB noch für DDT brauchbar! Siehe dazu Info4.

Kurze Beschreibung einiger Befehle:
ORG * legt Anfangsadresse (hier Hex 100) fest.
EQU * weist einem Label einen Wert zu:
WERT EQU 3 / LD A,WERT liefert: LD A,3
DEFB DB im BASIC
DEFS DS im BASIC

*: Akzeptiert BASIC (FDXB) nicht. Auch mag es weder zwei Labels für einen Befehl (BIOS/SELDSK) noch Kommentare, die in Zeilen ohne Befehl stehen!

25

Die Diskettenstory (Bernd Preusing)

Da ja anscheinend immer noch große Verwirrung bezüglich dieses Themas herrscht, möchte ich hier nun einmal genau beschreiben, wie der MTX bzw. die FDX-Floppy ihre Daten auf die kleinen Scheiben bringt.
Hierbei beschränke ich mich ausschließlich auf die 5-Zoll-Formate, da erstens bei den 8-Zoll-Formaten nicht diese Unklarheiten bestehen und zweitens garantiert niemand ein 8-Zoll-Laufwerk angeschlossen hat (übrigens ist meine Controller-Karte im 8-Zoll-Bereich gar nicht bestückt, DIP-Schalter sind nicht drauf, sondern Lötbrücken, und die Busanschlüsse sind hinten auch nicht herausgeführt VOBIS,VOBIS).

Fangen wir am Besten ganz unten auf der physikalischen Ebene an, und arbeiten uns dann langsam zum CP/M vor:

Herz der Controller-Karte ist der FD1791. Er beherrscht u.a. für single density das IBM 3740 Format und für double density das IBM SYSTEM 34 Format, die auch beide vom MTX benutzt werden. Bezüglich näherer Einzelheiten verweise ich auf das Datenblatt.
Anschließbar sind alle Shugart-kompatiblen Laufwerke von single sided (SS), single density (SD), 40 Spuren bis double sided (DS), double density (DD), 80 Spuren (entspricht Typ 00 bis 07).
Die meisten werden wohl ein DS,DD,40Trk-Laufwerk (Typ 03) besitzen und als zweites entweder gar nichts oder das gleiche oder ein DS,DD,80Trk (Typ 07).
Bei allen CONFIGs 00 bis 07 gibt es 16 Sektoren pro Spur; bei single density 128 Bytes/Sektor, bei DD 256 Bytes pro Sektor. Das ergibt bei Typ 03 vier KByte pro Spur mal 2 Seiten mal 40 Spuren = 320 KByte Gesamtkapazität, bei Typ 07 das Doppelte: 640 KByte.
Und hier hatte die Firma MEMOTECH ihre erste dunkle Stunde! Man kann nämlich leicht 18 Sektoren a 256 Byte auf eine Spur bringen, das ergäbe bei Typ 03 schon 360 KB (siehe TELEVIDEO-Format!) und bei Typ 07 720 KByte; wenn man die Sektor-Größe auf 512 oder 1024 Bytes erhöht, gehen auch 10 bzw. 5 Sektoren auf eine Spur, wodurch sich die Disk-Kapazitäten auf 400 (Typ 03) und 800 KByte (Typ 07) erhöhen. Das sind 25% mehr!

Die Sektoren sind nummeriert von 1 bis 16 (im Gegensatz zu den Spurnummern, die bei 0 beginnen). Jeder Sektor besteht aus einem ID- und einem Daten-Teil. Im ID-Teil sind Spur-, Seiten- und Sektor-Nummer abgespeichert. Wenn der Controller einen bestimmten Sektor lesen oder schreiben soll, liest er solange die ID-Teile, bis er den Sektor mit der entsprechenden Nummer gefunden hat. Daraus folgt, daß die Sektoren nicht in ihrer numerischen Reihenfolge auf der Spur liegen müssen. Genau das macht der MTX: er legt die Sektoren in der Reihenfolge

1,12,7,2,13,8,3,14,9,4,15,10,5,16,11,6

mit dem FORMAT-Programm auf der Spur an. Wie man sieht, liegen zwischen zwei numerisch aufeinanderfolgenden immer zwei andere Sektoren. Auf diese Weise bleibt dem Computer beim Lesen von mehreren zusammengehörigen Sektoren immer noch genug Zeit, die Daten im RAM zu bewegen (was besonders unter CP/M nötig ist). Wären die Sektoren in der richtigen Reihenfolge auf der Spur, müßte der Controller immer eine volle Umdrehung der Scheibe abwarten, bis er den nächsten Sektor lesen könnte. Dies ist also eine Maßnahme, die die Geschwindigkeit des Systems beträchtlich erhöht. Das ist nicht zu verwechseln mit dem 'sector scope' oder 'skew factor' des CP/M! Obiges ist nichts, worüber man beim Schreiben oder Lesen von Daten nachdenken müßte; der einzige Effekt ist der der Geschwindigkeitserhöhung! Beim 'sektor scope' hingegen sind die

Sektoren meist in der richtigen Reihenfolge auf der Spur und CP/M rechnet anhand einer Tabelle die logischen Sektornummern in physikalische um. Der Effekt ist der gleiche, allerdings ist es z.B. unter einem anderen Betriebssystem und ohne die Umrechnungstabelle unmöglich, die Daten korrekt einzulesen.

Kommen wir nun zur zweiten schwarzen Stunde von MEMOTECH: zum logischen Disketten-Format unter CP/M. Hier hat MEMOTECH im wahrsten Sinne des Wortes etwas Einmaliges geschaffen!

CP/M V2.x kennt nur Sektoren a 128 Byte, d.h. jeder physikalische Sektor enthält 2 logische Sektoren (im Folgenden mit LS abgekürzt)(ich rede jetzt nur noch von den Typen 03 und 07!).
Das Betriebssystem will wissen, wieviele 128-Byte-Sektoren auf einer Spur sind; dies wären dann naiv betrachtet 32 LS/Spur, aber weit gefehlt: CP/M sieht 26 !
Des weiteren muß man CP/M eine sogenannte 'Blockgröße' in KByte angeben. Das Betriebssystem sieht die Diskette nur als eine bestimmte Anzahl von Blöcken (Nummerierung beginnt bei 0), die es der Directory oder den einzelnen Files zuweist. Selbst wenn ein File nur 1 Byte lang ist, belegt es bereits einen ganzen Block auf der Diskette!
Das ist der Preis dafür, daß einzelne Files beliebig gelöscht werden können und der Platz dann wieder frei ist. CP/M verwaltet für jedes angesprochene Laufwerk eine sogenannte BAM (block allocation map), die ihm anzeigt, welche Blöcke belegt oder noch frei sind. Welche Blöcke ein File belegt, ist in der Directory eingetragen. Die Blockgröße ist bei beiden Typen 2 KByte. Die Anzahl dieser Blöcke auf der Disk -1 (also die höchste Blocknummer) muß man auch angeben.
Dann will CP/M noch wissen, wieviele Systemspuren auf der Diskette z.B. für CP/M selbst oder anderes reserviert sind; dies können 0 bis 65535 sein. Auf diese Weise wird z.B. eine Harddisk in mehrere logische Disks unterteilt. Diese Spuren zählen nicht, die Angabe wird nur benutzt, um den Beginn von Block 0 festzulegen. Für alle Typen 00 bis 07 sind 2 Systemspuren reserviert, d.h. Block 0 bzw. die Directory beginnt bei Spur 2, Sektor 1 (logisch!).
Zusammengefaßt: 26 LS/Spur, 2 Systemspuren, Blockgr. 2KByte

Jetzt wollen wir einmal Typ 03 durchrechnen (Typ 7 in Klammern/:

Zuerst müssen wir die logische Reihenfolge der physikalischen Sektoren kennen; sie laufen von Spur 0, Seite 0, Sektor 1 bis 16 über Spur 0, Seite 1, Sektor 1 bis 16 über alle Spuren bis Spur 39 (79), Seite 1, Sektor 16 durch. (Das ist nicht klar, das Bondwell-Format läuft z.B. erst Seite 0 ganz durch und geht dann weiter bei Seite 1, Spur 0.)
Nun stellen wir diese physikalischen Sektoren in einer langen Schlange aufgereiht vor und hacken jeden Sektor in zwei Teile, dann haben wir eine lange Reihe logischer Sektoren, insgesamt 2*2*40*16 = 2560 (5120). Jetzt machen wir alle 26 Sektoren einen Schnitt und erhalten die logischen Spuren 0 bis 97 (0 bis 195) und eine Restspur, die nur die Sektoren 1 bis 12 (1 bis 24) enthält.
Jetzt rechnen wir die Anzahl der Blöcke aus; 1 Block = 2 KByte = 16 logische Sektoren. Von den 2560 (5120) LS ziehen wir die beiden Systemspuren (also 52 LS) ab, dann bleiben 2508 (5068). Teilen wir diese in 2K-Blöcke auf, indem wir durch 16 teilen, erhalten wir 156, Rest 12 (316, Rest 12), d.h. CP/M selbst sieht nur die Blöcke 0 bis 155 (0 bis 316), also Spur 2, Sektor 1 bis Spur 97, Sektor 26 (Sp 2, LS 1 bis SP 195, LS 12), die restlichen

1,5 KByte werden niemals angerührt. Das sind übrigens die physikalischen Sektoren Spur 39 (79), Seite 1, Sektor 11 bis 16. Diese Sektoren kann man z.B. für den Datenschutz, zur Ablage eines Diskettennamens oder des Datums verwenden.

Wie man sieht, ist es also recht kompliziert, aus den Angaben für den gewünschten logischen Sektor und die log. Spur die physikalischen Angaben Spur, Seite und Sektor zu errechnen. Dabei leuchtet mir überhaupt nicht ein, was der Sinn dieses komplizierten Verfahrens sein soll.
Es ist schon richtig, daß durch die Angabe von zwei Systemspuren à 26 Sektoren genau der benötigte Platz für das CP/M-System bereitgestellt wird (6,5 KByte), aber dafür bleiben am Ende zumindest für die beiden besprochenen Formate jeweils 1,5K ungenutzt, was in einem 'normalen' Format ganz genauso wäre. Auch das Argument, auf diese Weise die vielen verschiedenen Formate einfacher Verwalten zu können zieht nicht, denn normalerweise berechnet CP/M alles selbst, soweit es sich um 128-Byte-Sektoren handelt, und die Umrechnung auf doppelt große Sektoren ist eine einfache Division der Sektornummer durch 2.
Zum Zweiten, was nützen mir die vielen verschiedenen Formate, wenn kein einziges ein 'Standardformat' ist! Jeder CP/M-Rechner ist nur soviel wert, wie ich dafür auch Software bestellen kann; nicht einmal Turbo-Pascal, das angeblich auch in den exotischten Formaten erhältlich ist, gibt es für uns!
Der einzige Grund, das 26LS-Format beizubehalten, ergibt sich aus der Faulheit der Software-Leute bei MEMOTECH: Alle Systemprogramme, die Blocks lesen oder schreiben (d.h. mehrere aufeinanderfolgende log. Sektoren), wie z.B. SYSCOPY oder auch die Blockread-Routine im Monitor, gehen unerschütterlich davon aus, daß 26 Sektoren auf jeder Spur sind. Dabei ist es sehr einfach, anhand des ersten Bytes des DPB (s.u.) die Anzahl der Sektoren pro Spur zu ermitteln, wie das z.B. VDEB sehr gut vormacht!
Wenn man ein fremdes Format (z.B. TELEVIDEO) mittels COLDBOOT zum A-Laufwerk 'erheben' (also nicht nur davon kopieren), dann muß vorher durch SYSCOPY das System auf die Diskette gebracht werden. Dabei ist zu bedenken, daß das Format mindestens 26 LS pro Spur und mind. 2 Systemspuren besitzt. In diesem Fall werden durch SYSCOPY immer nur die ersten 26 LS einer Spur beschrieben, was aber nichts ausmacht, denn beim Booten werden auch nur diese gelesen.

Doch kommen wir nach diesem kleine Exkurs zu den Tabellen im BIOS, die CP/M als Beschreibung der Formate benötigt:

Für jedes angschlossene Laufwerk gibt es einen sog. Disc Parameter Header, der 16 Byte lang ist und CP/M die Beschreibung des Formates und Speicherplätze für bestimmte Operationen zugänglich macht.

DPH

XLT	0000	0000	0000	DIRBUF	DPB	CSV	ALV

Jedes Element darin ist ein 16-Bit-Wort. Die Bedeutung ist im Einzelnen:

XLT — Adresse der Umrechnungstabelle für sector scope, falls benutzt, sonst 0000. Laufwerke mit gleichem skew factor benutzen dieselben Tabellen

0000 — Dieser Bereiche benutzt das BDOS für Berechnungen etc., der Anfangswert ist egal.

DIRBUF Adresse eines 128-Byte-Bereiches für Directory-Operationen des BDOS. Alle DPHs benutzen denselben Bereich.

DPB Adresse des Disk Parameter Blocks für dieses Laufwerk. Laufwerke mit gleichen Eigenschaften benutzen dieselbe Tabelle.

CSV Adresse eines Speicherbereichs zum Feststellen eines Diskettenwechsels. Verschieden für jedes Laufwerk.

ALV Adresse des 'allocation vectors' zum Feststellen der Diskettenbelegung (Blöcke!). Für jedes Laufwerk versch.

Der Aufbau des Disk Parameter Block (DPB), der von einem oder mehreren DPHs benutzt werden kann, ist etwas komplizierter:

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
16b	8b	8b	8b	16b	16b	8b	8b	16b	16b

Hierbei bedeutet 8b ein Byte und 16b ein Word (2 Bytes in der Reihenfolge Low-Byte, High-Byte), also insgesamt 15 Bytes lang.

SPT Anzahl der Sektoren pro Spur (128 Byte !!)

BSH ist der sog. block shift factor, bestimmt durch die Blockgröße

EXM ist die Extent-Maske, bestimmt durch die Blockgröße und die Anzahl der Blöcke auf der Disk

DSM ist die höchste Blocknummer (Blockzahl-1), bestimmt also die Kapazität der Diskette

DRM bestimmt die Anzahl der Directory-Einträge (-1), AL0 und AL1 legen reservierte Dir-Blocks fest

CKS ist die Größe des Directory check vectors

OFF bestimmt die Anzahl der reservierten Spuren

Die Werte von BSH und BLM bestimmen implizit die Blockgröße, die im DPB nicht direkt angegeben wird. Hat man sich für eine bestimmte Blockgröße entschieden, ergeben sich BSH und BLM zu:

BLS	BSH	BLM
1024	3	7
2048	4	15
4096	5	31
8192	6	63
16384	7	127

Der Wert von EXM hängt von BLS ab und davon, ob DSM <= oder >255 ist, mit anderen Worten, ob für die Blocknummern in der Directory 1 oder 2 Byte reserviert werden müssen:

BLS	DSM<256 EXM	DSM>255 EXM
1024	0	unmöglich
2048	1	0
4096	3	1
8192	7	3
16384	15	7

Der Wert von DSM ist die größte Blocknummer des Laufwerks, das Produkt aus BLS und (DSM+1) ergibt die Gesamtkapazität in Bytes und muß natürlich kleiner oder gleich der physikalischen Kapazität sein, die Systemspuren nicht mitgerechnet.
Der DRM-Eintrag ist die Anzahl der Dir-Einträge -1; daraus ergeben sich eindeutig die Werte für AL0 und AL1, die wie folgt aufgebaut sind:

! AL0	! AL1
!00!01!02!03!04!05!06!07!08!09!10!11!12!13!14!15!	

Dabei ist Pos. 0 das Bit 7 von AL0 und läuft bis Pos. 15 als Bit 0 von AL1 durch. Die für die Directory reservierten Blöcke werden als 1-Bits dargestellt und von links aufgefüllt. In je 1 K gehen 32 Dir-Einträge a 32 Byte. Ist die Blockgröße also 2k, dann ergeben sich z.B. für 64 Einträge die Werte #80,#00 und für 128 Einträge #C0,#00.

Der CKS-Wert bestimmt sich so: ist das Medium auswechselbar (also Floppy,), dann ist CKS=(DRM+1)/4, ist es fest (Hard-Disk oder RAM-Disk), dann ist CKS=0.

Beim DPH fehlen noch die Erklärungen zu den beiden Adressen CSV und ALV. Die Größe der Bereiche bestimmt sich aus dem DPB:
die Größe des für CSV reservierten Bereiches ist CKS Bytes, die Anzahl der reservierten Bytes für ALV ist (DSM/8)+1. Für jedes Laufwerk müssen eigene Bereiche festgelegt sein!

Sehen wir uns jetzt einmal einige konkrete DPBs an:

	SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
a) CFIG03:	1A,00,	04,	0F,	01,	9B,00,	3F,00,	80,	00,	10,00,	02,00
b) CFIG07:	1A,00,	04,	0F,	00,	3A,01,	7F,00,	C0,	00,	20,00,	02,00
c) CFIG08:	48,00,	04,	0F,	01,	AA,00,	3F,00,	80,	00,	10,00,	02,00
d) CFIG50:	1A,00,	03,	07,	00,	68,00,	1F,00,	80,	00,	00,00,	02,00

a) und b) haben 26 Sektoren pro Spur (leider!), c) hat #0048=72 Sektoren pro Spur, das ergibt sich aus 18 Sektoren a 256 Bytes=36 a 128 Bytes mal 2 Seiten = 72 (eine Spur läuft also über 2 Seiten). Alle haben eine Blockgröße von 2 KByte (BSH und BLM) und 2 Systemspuren. a) hat als größte Blocknummer #009B = 155, also 156 Blöcke a 2K, was ich weiter oben auch schon ausgerechnet hatte, c) hingegen hat laut dieser Tabelle #013A+1 = 315 Blöcke, Platz auf der Disk haben jedoch 316, hier werden uns also glatt 2K unterschlagen! Dies kann man aber durch einfaches Ändern des entsprechenden Bytes von #3A auf #3B korrigieren.
d) ist ein von mir 'erfundenes' Format, mit dem ich durch meine 128K-Karte und das VRAM auf dem MTX 500 eine 105K-Ramdisk habe (2*48K RAM + 16K VRAM, Blockgröße 1K, 32 DIR-Eintr.).

Die restlichen Werte müßten jetzt eigentlich klar sein.

Zum Schluß noch ein paar Tips:

OVERLAY8 ist das Tor zur CP/M-Welt !!

Mit Hilfe dieses Programms können wir alle Formate verarbeiten, die 256 Bytes pro Sektor in double density haben.
Es überschreibt bei seinem Aufruf den Disk-Handler im Bereich #FA00..#FFFE und führt dann einen Kaltstart durch. Darin enthalten sind der DPB für das Format 08 und eine kleine Routine, die die logischen Sektoren in physikalische umrechnet. Diese Beiden braucht man nur nach seinen Wünschen zu modifizieren und man hat ein neues Format zur Verfügung!
Die Routine liegt von #0DB7..#0DCF, der DPB bei #0F19..#0F27, beide werden nachher um genau #F000 nach oben verschoben.

Hier nun erstmal die Original-Routine für CONFIG08:

```
**************************************************************************
 Eingaben: HL: log. Spur    (0..xx)     (H immer 0 !)
           E: log. Sektor  (1..SPT)
 Ausgang:  HL: phys. Spur
           E: phys. Sektor
           D: Seite (0..1)
 Die phys. Spur muß in #FEC7 abgelegt werden, der Ausgang erfolgt
durch JP #FD71.
**************************************************************************

FDB7        LD   A,L              ;Spurnummer bleibt gleich
FDB8        LD   (#FEC7),A        ;abspeichern
FDBB        DEC  E                ;Sektornummern (0..SPT-1)
FDBC        XOR  A                ;clear carry
FDBD        LD   A,E              ;log. Sektor (128 Byte)
FDBE        RRA                   ;durch 2 = phys. Sektor (256 Byte)
FDBF        INC  A                ;wieder Nummern 1..
FDC0        LD   D,0              ;Seite 0 annehmen
FDC2        LD   E,A              ;E = phys. Sektornummer
FDC3        SUB  #12              ;-18 probieren
FDC5        JP   C,#FD71          ;<18, also fertig
FDC8        JP   Z,#FD71          ;=18, auch
FDCB        LD   E,A              ;war >18, ...
FDCC        INC  D                ;..also Seite 1
FDCD        JP   #FD71            ;und fertig
```

Nun habe ich z.B. Disketten im **BONDWELL-Format**, die folgendermaßen aufgebaut sind:
18 Sektoren pro Spur a 256 Byte DD, nummeriert 0..17, 40 Spuren, 2 Seiten. Log. Reihenfolge: erst Seite 0,Spur 0..39, dann Seite 1, Spur 0..39. 2 Systemspuren, 128 Dir-Einträge, Blockgröße 2K. Daraus ergibt sich der DPB zu: (=350K CP/M-Kapazität)

24,00, 04, 0F, 01, AE,00, 7F,00, C0,00, 20,00, 02,00

und die Routine lautet:

```
FDB7        DEC  E                ;log. Sektornummern 0..35
FDB8        SRL  E                ;/2 = phys. Sektornummern 0..17
FDBA        LD   D,0              ;Seite 0 annehmen
FDBC        LD   A,L              ;log. Spur
FDBD        SUB  #28              ;minus 40
FDBF        JP   C,#FDC4          ;<40, also ok
FDC2        LD   L,A              ;phys. Spur nach L
FDC3        INC  D                ;Seite 1
FDC4 L1:    LD   A,L
FDC5        LD   (#FEC7),A        ;phys. Spur merken
FDC8        JP   #FD71            ;und fertig
```

Nach den Informationen, die ich aus dem letzten Info über das **TA-pc-Format** habe, müßte folgendes funktionieren:

DPB: 40,00, 04, 0F, 01, 97,00, 7F,00, C0,00, 20,00, 02,00

Die Umrechnungs-Routine braucht gegenüber dem Original bei #FDC3 statt SUB #12 nur in SUB #10 geändert zu werden.

Diese Änderungen macht man am Besten mit 'DDT OVERLAY8.COM' oder 'VDEB OVERLAY8.COM', bringt die neue Routine und den DPB ein (alles, was nachher bei #Fxyz liegt ist im OVERLAY8 bei #0xyz !), drückt dann CTRL-C und gibt anschließend das Kommando

SAVE 18 OVERTAPC.COM o.ä. ein.

Will man dann ein entsprechendes Format bearbeiten, ruft man z.B. OVERTAPC auf und danach CFIG8 B:03,C:08 o.ä. Danach kann man dann lesen, schreiben und kopieren nach Herzenslust.

Wer zu den unerschütterlichen Disk-Jockeys gehört und immer noch nur ein Laufwerk besitzt, braucht ein Programm, das zwischen verschiedenen Laufwerken kopieren kann und trotzdem zwischendurch zum Diskettenwechsel auffordert. Glücklicherweise ist COPY so nett und tut das: COPY A:x.y C:x.y kopiert auf einem Laufwerk das File x.y vom Format des Laufwerks C zum Format des Laufwerks A, wenn man in OVERLAY8 zusätzlich folgende Änderung vornimmt:

bei #0CBB und #0CC5 statt LD A,(#FFE8) (#3A,#E8,#FF)
nun LD A,0 NOP (#3E,#00,#00)

Dadurch wird immer das physikalische Laufwerk 0 angewählt, aber trotzdem das Format des log. Laufwerks A, B, C, etc. benutzt!

Nachtrag: Aufbau der Directory

Jeder Directory-Eintrag besteht aus 32 Byte und ist wie folgt aufgebaut:

```
US F1 F2 F3 F4 F5 F6 F7 F8 T1 T2 T3 EX S1 S2 RC
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
```

US	enthält die USER-Nummer, unter der der File abgespeichert ist (0 bis 31). Ist US=#E5, dann ist der Eintrag frei, also unbenutzt.
F1..F8	Filename in ASCII, Großbuchstaben, Bit7=0
T1..T3	Filetyp in ASCII, Großbuchstaben, Bit7=0. Falls Bit7 von T1=1, dann R/O-File, falls Bit7 von T2=1, dann SYS-File (kein List bei DIR).
EX	ist die Anzahl der EXTENSIONS (s.u.)
S1,S2	egal
RC	bestimmt mit die Länge des Files
D0..DF	enthalten die Blocknummern, die der File belegt. Falls die Anzahl der Blöcke auf der Diskette <256 ist, bedeutet jedes Byte einen Block, sonst jeweils 2 Byte.

Die Länge eines Files ermittelt man, indem man den Eintrag sucht, bei dem RC kleiner als 128 ist (große Files haben mehrere DIR-Einträge!) und dann 128*EX+RC berechnet. Das Ergebnis ist die Länge in 128-Byte-Sektoren.

Ich hoffe, hiermit wenigstens etwas Klarheit in das 'Disketten-Wirrwarr' gebracht zu haben.

Diese Patches sind von DIGITAL RESEARCH.

Aufruf jeweils mit DDT xx.COM, korrigieren mittels Sxxxx oder Axxxx, danach SAVE xx yy.COM.

1. Fehler beim SET-Befehl in ASM.COM, der manchmal fälschlicherweise einen 'phasing error' erzeugt:

```
alt:   1DAD: CD 52 13          neu:    1DAD: CD 8D 1B
       1B8D: 00 00 00 ....             1B8D: CD 52 13 B7 CA B5 1D C9
```

SAVE 32 ASM.COM

2. Fehler in SUBMIT.COM, der dafür sorgt, daß Jobs nur auf Drive A laufen können, weil der $$$.SUB-File auf das momentan eingeloggte Laufwerk kommt. Jetzt kommt der $$$.SUB-File immer auf Drive A.

```
alt:   05BB: 00                neu:    05BB: 01
```

SAVE 5 SUBMIT.COM

3. Drei Fehler in PIP.COM: Das erste Problem tritt auf, wenn die Start- und Stop-Strings beim Kopieren von ASCII-Files gleich lang sind:

```
alt:   1168: 3A 62 1F 32 F7 1D 21 62 1F 36 00 3A F9 1D 3C 32 F8 1D
neu:   1168: 21 62 1F 7E 32 F7 1D 36 00 21 F9 1D 7E 36 00 3C 2B 77
```

Das zweite Problem tritt beim Kopieren von Objekt-Files auf:

```
alt:   0713: 3A 5E 1F 21 04 1E   neu:   0713: 3A 04 1E 21 5E 1F
       1099: 3A 04 1E                   1099: 3A 5E 1F
       1640: 3A 04 1E                   1640: 3A 5E 1F
```

Wenn PIP zum logischen PRN-Kanal kopieren soll, nimmt es immer den physikalischen LST-Kanal. Die dritte Korrektur behebt das:

```
alt:   0C69: 36 80             neu:    0C69: 00 00
```

SAVE 29 PIP.COM

4. Die folgende Erweiterung von INITIATE.COM ist von mir und sorgt dafür, daß die Umlaute bei ALPHA LOCK mit umgeschaltet werden, d.h., daß sie zu den Buchstaben zählen:

```
alt:   012B: C3 00 00          neu:    012B: C3 80 02
                                       0280: AF 21 65 F2 77 23 77 23 77
                                       0288: C3 00 00
```

SAVE 2 INITIATE.COM

FDX-TASTATURTREIBER

DISASSEMBLIERT UND KOMMENTIERT BERND FREUSING 1/85

```
                        ORG     0F000H

F000            MOUT:   DS      1       ;MERKER FÜR LETZTES OUT 5
F001            MCHAR:  DS      1       ;MERKT GETIPPTE TASTE (ASCII)
F002            NEWCH?: DS      1       ;MERKER LETZTER OFFSET, NEUE
                                        ;TASTE WIRD NUR ANGENOMMEN,
                                        ;WENN (F002)=0
F003            BITS:   DS      1       ;EINZELFLAGS

;       BIT5: 0=SCROLL, 1=PAGE (FÜR CODE DER PAGE-TASTE)
;       BIT7: 1=ALPHA LOCK EIN
;       BIT2: 0=NORMAL-BELEGUNG, 1=ZEHNERBLOCK GIBT ZIFFERN
;             (MUSS EXTERN GESETZT WERDEN!)

F004            CNTR:   DS      2       ;ZAHLER FÜR AUTO-REPEAT

;       #8800: NICHTS GEDRüCKT
;       #0800: NEUE TASTE GEDRÜCKT (ZAHLT RUNTER BIS 255, DANN
;                                   BEGINNT REPEAT)
;       #00xx: WAHREND REPEAT ALS TIMER

F006                    DS      1       ;NICHT BENUTZT

                ;SHIFT-TASTEN ABFRAGEN, Z=SHIFT

F007  3EBF      SHIFT?: LD      A,BF    ;NUR BIT6=0
F009  3200F0            LD      (F000),A        ;MERKEN (WOFÜR?)
F00C  D305              OUT     (05),A
F00E  DB05              IN      A,(05)
F010  CB77              BIT     6,A     ;SHIFT1 GEDRÜCKT?
F012  C8                RET     Z
F013  CB47              BIT     0,A     ;SHIFT2?
F015  C9                RET

                ;TASTATUR ABFRAGEN, BIS WAS GEDRÜCKT

F016  ED7354F0  KBIN:   LD      (F054),SP       ;EIGENEN STACK SETZEN
F01A  3176F0            LD      SP,F076
F01D  C5                PUSH    BC
F01E  0603      KBIN10: LD      B,03    ;ENTPRELLUNG, MUSS 3*GLEICH SEIN
F020  CD76F0            CALL    F076
F023  28F9              JR      Z,F01E  ;WARTEN, BIS TASTE
F025  3A02F0            LD      A,(F002)
F028  4F                LD      C,A     ;ALTE MERKEN
F029  CD76F0            CALL    F076    ;NOCHMAL NACHSEHEN
F02C  3A02F0            LD      A,(F002)
F02F  B9                CP      C       ;TASTE GLEICH GEBLIEBEN?
F030  20EC              JR      NZ,F01E ;NEIN, ALLES NOCHMAL
F032  10F5              DJNZ    F029    ;INSGES. 6 GLEICHE SCANS
F034  3A01F0            LD      A,(F001)        ;TASTENCODE
F037  C1                POP     BC
F038  ED7B54F0          LD      SP,(F054)       ;ALTEN STACK WIEDER
F03C  C9                RET

                ;STATUS ABFRAGEN, A=FF: WAS DA,    SONST 0

F03D  ED7354F0  KBRDY:  LD      (F054),SP
F041  3176F0            LD      SP,F076
```

```
F044  CD76F0            CALL    F076
F047  3E00              LD      A,00
F049  2804              JR      Z,F04F  ;NICHTS DA, FERTIG
F04B  3202F0            LD      (F002),A        ;AUF 0 SETZEN, WEIL
                                                ;SONST BEIM NACHSTEN
                                                ;KBIN-AUF-RUF UNGULTIG
F04E  3D                DEC     A       ;FF=TASTE GEDRUCKT
F04F  ED7B54F0  KBRD10: LD      SP,(F054)
F053  C9                RET

F054            SPOLD:  DS      2       ;MERKER FÜR ALTEN STACKP
                STACK:  DS      32      ;16-LEVEL STACK

                ;BESTIMME TASTENCODE
                ;Z: NICHTS NEUES, NZ:CODE IN (F001)

F076  E5        SCAN:   PUSH    HL
F077  C5                PUSH    BC
F078  D5                PUSH    DE
F079  2104F0            LD      HL,F004 ;REPEAT TIMER
F07C  56                LD      D,(HL)
F07D  23                INC     HL
F07E  5E                LD      E,(HL)
F07F  3E88              LD      A,88
F081  BA                CP      D       ;LAUFT REPEAT?
F082  280F              JR      Z,F093  ;NEIN, NORMALSCAN
F084  7A                LD      A,D
F085  B3                OR      E       ;TIMER = 0 ?
F086  2808              JR      Z,F090  ;JA, TASTE WIEDER GÜLTIG
F088  1B                DEC     DE      ;NEIN, TIMER-1
F089  73                LD      (HL),E
F08A  2B                DEC     HL
F08B  72                LD      (HL),D
F08C  7A                LD      A,D
F08D  B3                OR      E       ;TIMER JETZT 0 ?
F08E  2003              JR      NZ,F093
F090  3202F0    SCAN10: LD      (F002),A        ;TASTE GÜLTIG
F093  CD9EF0    SCAN20: CALL    F09E    ;ASCII-CODE HOLEN
F096  2002              JR      NZ,F09A ;WAS DA, OK
F098  3E00              LD      A,00    ;NICHTS DA
F09A  D1        SCAN30: POP     DE
F09B  C1                POP     BC
F09C  E1                POP     HL
F09D  C9                RET

                ;ASCII-SCAN MIT CTRL-AUSWERTUNG
                ;NZ:TASTE GEDRÜCKT

F09E  3EFB      ASCAN:  LD      A,FB    ;BIT2=0
F0A0  3200F0            LD      (F000),A        ;MERKEN
F0A3  D305              OUT     (05),A
F0A5  DB05              IN      A,(05)
F0A7  CB47              BIT     0,A     ;CTRL-TASTE GEDRÜCKT?
F0A9  2804              JR      Z,F0AF  ;JA
F0AB  CDB9F0            CALL    F0B9    ;NEIN, NORMALE AUSW.
F0AE  C9                RET

F0AF  CDB9F0    ASCA10: CALL    F0B9    ;HOLE ASCII-CODE
F0B2  C8                RET     Z       ;NICHTS DA, OK
F0B3  E61F              AND     1F      ;IN CTRL-CODE UMWANDELN
F0B5  3201F0            LD      (F001),A        ;UND MERKEN
F0B8  C9                RET
```

```
                        ;SCAN MIT AUSW. VON SHIFT, ALPHA LOCK ETC

F0B9  CD07F0    SSCAN:  CALL    F007    ;SHIFT GEDRUCKT?
F0BC  2831              JR      Z,F0EF  ;JA, 2. TABELLE NEHMEN
F0BE  CD1FF1            CALL    F11F    ;PHYSIKALISCHER SCAN
F0C1  C8                RET     Z       ;NICHTS GEDR., FERTIG
F0C2  57                LD      D,A     ;TABELLEN-OFFSET
F0C3  0198F1            LD      BC,F198 ;UNSHIFT-TABELLE
F0C6  CD8FF1            CALL    F18F    ;CODE AUS NACH A UND (F001)
F0C9  47                LD      B,A
F0CA  FE7F              CP      7F
F0CC  2803              JR      Z,F0D1
F0CE  FE20              CP      20      ;CODE<#20 ODER =#7F ?
F0D0  3F                CCF
F0D1  3A03F0    SSCA10: LD      A,(F003)
F0D4  3804              JR      C,F0DA  ;KEIN STEUERCODE
F0D6  CB57              BIT     2,A     ;ZIFFERNBLOCK EINGESCH.?
F0D8  2012              JR      NZ,F0EC ;JA, SHIFT-TAB. BENUTZEN
F0DA  CB7F      SSCA20: BIT     7,A     ;ALPHA LOCK EINGESCH.?
F0DC  78                LD      A,B     ;CODE AUS DER TABELLE
F0DD  281A              JR      Z,F0F9  ;NEIN, OK
F0DF  213AF2            LD      HL,F23A ;ALPHA-TABELLE
                                        ;0=BUCHST., 1=SONST
F0E2  5A                LD      E,D     ;ALTER TABELLEN-OFFSET
F0E3  1600              LD      D,00
F0E5  19                ADD     HL,DE
F0E6  7E                LD      A,(HL)  ;1 ODER 0 AUS DER TABELLE
F0E7  53                LD      D,E     ;OFFSET WIEDER NACH D
F0E8  A7                AND     A       ;BUCHSTABE?
F0E9  78                LD      A,B     ;ASCII-CODE NACH A
F0EA  200E              JR      NZ,F0FA ;KEIN BUCHST., OK
F0EC  7A        SSCA30: LD      A,D     ;OFFSET FÜR SHIFT-TABELLE
F0ED  1804              JR      F0F3    ;CODE AUS SHIFT-TAB HOLEN

                        ;SHIFT IST GEDRÜCKT, SCAN FÜR SHIFT-TABELLE

F0EF  CD1FF1    SSCA40: CALL    F11F    ;PYS. SCAN
F0F2  C8                RET     Z       ;NICHTS GEDR., FERTIG
F0F3  01E9F1    SSCA50: LD      BC,F1E9 ;SHIFT-TABELLE
F0F6  CD8FF1            CALL    F18F    ;CODE AUS TAB+OFF
                                        ;NACH A UND (F001)
F0F9  47        SSCA60: LD      B,A     ;ASCII-CODE
F0FA  FEA0      SSCA70: CP      A0      ;= ALPHA-LOCK TASTE ?
F0FC  200A              JR      NZ,F108 ;NEIN, WEITER
F0FE  3A03F0            LD      A,(F003)
F101  EE80              XOR     80      ;JA, BIT UMSCHALTEN
F103  3203F0            LD      (F003),A
F106  AF                XOR     A       ;LIEFERT NAT. NICHTS
F107  C9                RET

F108  FE1D      SSCA80: CP      1D      ;TASTE=PAGE?
F10A  C0                RET     NZ      ;NEIN,FERTIG
F10B  5F                LD      E,A     ;JA, MERKEN
F10C  3A03F0            LD      A,(F003)
F10F  CB6F              BIT     5,A     ;PAGE/SCROLL-FLAG
F111  2801              JR      Z,F114  ;PAGE
F113  1D                DEC     E       ;CODE=1C: SCROLL
F114  EE20      SSCA90: XOR     20      ;FLAG UMSCHALTEN
F116  3203F0            LD      (F003),A
F119  7B                LD      A,E     ;CODE NACH A
F11A  3201F0            LD      (F001),A        ;UND SPEICHERN
F11D  A7                AND     A       ;NZ SETZEN
F11E  C9                RET
```

```
                        ;PHYSIKALISCHER ABFRAGE DER TASTATUR-LEITUNGEN
                        ;NZ: WAS GEDR. UND OFFSET FUR TABELLE IN A

F11F  0608      PSCAN:  LD      B,08    ;8 REIHEN ABFRAGEN
F121  AF                XOR     A       ;A=0, RESET CARRY
F122  4F                LD      C,A     ;SPALTENZAHLER=0
F123  3D                DEC     A       ;A=FF
F124  17        PSCA10: RLA             ;NULL-BIT VON RECHTS DURCH
F125  F5                PUSH    AF
F126  3200F0            LD      (F000),A        ;SCAN-BYTE MERKEN
F129  D305              OUT     (05),A
F12B  DB05              IN      A,(05)
F12D  FEFF              CP      FF
F12F  2823              JR      Z,F154  ;NICHTS GEDR., ALSO IN 6
F131  F5                PUSH    AF
F132  3E02              LD      A,02    ;SHIFT-REIHE?
F134  B8                CP      B
F135  2005              JR      NZ,F13C ;NEIN, OK
F137  F1                POP     AF      ;JA, SHIFT UNGEDRUCKT MACHEN
F138  CBF7              SET     6,A
F13A  1809              JR      F14
5
F13C  3E06      PSCA20: LD      A,06    ;CTRL-REIHE?
F13E  B8                CP      B
F13F  2803              JR      Z,F144  ;JA CTRL UNGEDR. MACHEN
F141  F1                POP     AF      ;ALTES IN 5
F142  1807              JR      F14B    ;AUSWERTEN

F144  F1        PSCA30: POP     AF
F145  CBC7      PSCA40: SET     0,A     ;SHIFT ODER CTRL UNGEDRUCKT
                                        ;MACHEN, DAMIT DIESE (HIER)
                                        ;NICHT AUSGEWERTET WERDEN
F147  FEFF              CP      FF      ;SONST NICHTS GEDRUCKT?
F149  2809              JR      Z,F154  ;NEIN, ALSO IN 6 TESTEN

F14B  D1        PSCA50: POP     DE      ;STACK KORRIGIEREN
F14C  0E00              LD      C,00    ;ZAHLER=0
F14E  0F        PSCA60: RRCA
F14F  3017              JR      NC,F168 ;C=BITNR DES NULL-BITS
F151  0C                INC     C
F152  18FA              JR      F14E

F154  DB06      PSCA70: IN      A,(06)
F156  E603              AND     03      ;NUR BITS 0 UND 1 GULTIG
F158  FE03              CP      03
F15A  2807              JR      Z,F163  ;NICHTS GEDRUCKT, NACHSTE REIHE
F15C  A7                AND     A       ;??????
F15D  C607              ADD     A,07    ;ERGIBT BITNR 8 OD. 9
                                        ;DAS IST EIN FEHLER...., FALLS
                                        ;FALLS Z.B. SPACE UND F4 GLEICH-
                                        ;ZEITIG GEDRUCKT, GIBT DAS A=7
F15F  4F                LD      C,A
F160  F1                POP     AF
F161  1805              JR      F168    ;OFFSET BERECHNEN

F163  F1        PSCA80: POP     AF      ;ALTES SCAN-BYTE WIEDER
F164  10BE              DJNZ    F124    ;8 REIHEN ABGEFRAGT?
F166  0E00              LD      C,00    ;NICHT GEDRUCKT, B=C=0

F168  79        PSCA90: LD      A,C     ;OFFSET=8*C+B
F169  87                ADD     A,A
F16A  87                ADD     A,A
F16B  87                ADD     A,A
F16C  80                ADD     A,B
```

```
F16D  2104F0            LD      HL,F004 ;REPEAT-TIMER
F170  2008              JR      NZ,F17A ;WAS GEDRUCKT
F172  3688              LD      (HL),88 ;NICHTS GEDRUCKT, RESET TIMER
F174  23                INC     HL
F175  3600              LD      (HL),00 ;..AUF #8800
F177  2B                DEC     HL
F178  2808              JR      Z,F182  ;HIER IMMER Z

F17A  CBBE      FSCA91: RES     7,(HL)  ;WAS GEDR.: BIT 7=0
F17C  47                LD      B,A     ;NEUEN OFFSET MERKEN
F17D  3A02F0            LD      A,(F002)        ;UND MIT ALTEM VERGL.
F180  B8                CP      B
F181  78                LD      A,B     ;NEUEN
F182  3202F0    FSCA92: LD      (F002),A        ;ALS ALTEN ABSPEICHERN
F185  C8                RET     Z       ;GLEICH WIE VORHER, UNGULTIG
F186  F5                PUSH    AF
F187  AF                XOR     A
F188  BE                CP      (HL)    ;HIGH-BYTE DES TIMERS=0 ?
F189  2002              JR      NZ,F18D ;NEIN, OK
F18B  CBCE              SET     1,(HL)  ;SET 0,(HL) VIEL SCHNELLER!
                                        ;JA, TIMER NEU STARTEN
F18D  F1        FSCA97: POP     AF
F18E  C9                RET

                ;CODE AUS TABELLE HOLEN NACH A UND (F001)
                ;EING: A=OFFSET, BC=TABELLENANFANG

F18F  2600      GTCODE: LD      H,00
F191  6F                LD      L,A
F192  09                ADD     HL,BC
F193  7E                LD      A,(HL)
F194  3201F0            LD      (F001),A
F197  C9                RET

                ; IN-REIHEN 8 UND 9 WEGEN DER ABFRAGE-LOGIK VERTAUSCHT

F198            UNSHIF: DB      00      ;'NICHTS-GEDR.-BYTE

                        ;OUTBIT 7  6  5  4  3  2  1  0

                IN0:    DB      79,00,61,A0,71,00,1B,31
                        ;       y  SL a  AL q  CT ES 1
                IN1:    DB      63,78,64,73,65,77,32,33
                        ;       c  x  d  s  e  w  2  3
                IN2:    DB      62,76,67,66,74,72,34,35
                        ;       b  v  g  f  t  r  4  5
                IN3:    DB      6D,6E,6A,68,75,7A,36,37
                        ;       m  n  j  h  u  z  6  7
                IN4:    DB      2E,2C,6C,6B,6F,69,38,39
                        ;       .  ,  l  k  o  i  8  9
                IN5:    DB      3D,2D,7B,7C,7D,70,30,7E
                        ;       =  -  ä  ö  ü  p  0  ß
                IN6:    DB      15,00,0D,23,0A,2B,5E,3C
                        ;       IN SR CR #  LF +  ^  <
                IN7:    DB      0C,0A,1A,19,08,0B,05,1D
                        ;       CL Dn HM Ri Le Up EO FG
                IN9:    DB      83,87,82,86,85,81,84,80
                        ;       F4 F8 F3 F7 F6 F2 F5 F1
                IN8:    DB      20,00,00,00,7F,09,08,03
                        ;       SP           DL TB BS BK
```

```
SHIFT:  DB      00
        DB      59,00,41,A0,51,00,1B,21
        ;       Y  SL A  AL Q  CT ES !
        DB      43,58,44,53,45,57,22,40
        ;       C  X  D  S  E  W  "  §
        DB      42,56,47,46,54,52,24,25
        ;       B  V  G  F  T  R  $  %
        DB      4D,4E,4A,48,55,5A,26,2F
        ;       M  N  J  H  U  Z  &  /
        DB      3A,3B,4C,4B,4F,49,28,29
        ;       :  ;  L  K  O  I  (  )
        DB      3D,5F,5B,5C,5D,50,30,3F
        ;       =  _  Ä  Ö  Ü  P  0  ?
        DB      30,00,0D,60,0A,2A,27,3E
        ;       0  SR CR  ´ LF  *  ' >
        DB      0D,2E,32,33,31,35,38,37
        ;       CR .  2  3  1  5  8  7
        DB      8B,8F,8A,8E,8D,89,8C,88
        ;       F4 F8 F3 F7 F6 F2 F5 F1
        DB      20,00,00,00,36,34,08,39
        ;       SP        6  4  BS 9

ALPHA:  DB      1
        DB      0,1,0,1,0,1,1,1
        ;       Y   A   Q
        DB      0,0,0,0,0,0,1,1
        ;       C X D S E W
        DB      0,0,0,0,0,0,1,1
        ;       B V G F T R
        DB      0,0,0,0,0,0,1,1
        ;       M N J H U Z
        DB      1,1,0,0,0,0,1,1
        ;           L K O I
F263    DB      1,1,1,1,1,0,1,1
        ;           Ä Ö Ü P         BESSER FÜR A80: 0
        DB      1,1,1,1,1,1,1,1
        DB      1,1,1,1,1,1,1,1
        DB      1,1,1,1,1,1,1,1
        DB      1,1,1,1,1,1,1,1

        END
```

Schnelles Laden und Auslesen der 80-Zeichen-Karte

Die folgenden Assembler-Routinen dienen dazu, das Video-RAM der 80-Zeichen-Karte in den Hauptspeicher einzulesen oder den Inhalt des Hauptspeichers sehr schnell auf den Schirm zu bringen. Ersteres kann man natürlich sehr einfach in eine Screencopy-Routine umwandeln. Beide Programme laufen sowohl unter normalem CP/M-Betrieb als auch im FDX-BASIC (CALL #CE0 ==> B nach Centronics).

Die Attribute werden bei beiden Routinen nicht berücksichtigt, d.h. falls Grafikzeichen o.ä. auf dem Schirm sind, ist der Inhalt nicht eindeutig!

(Bernd Preusing)

1. Auslesen des Schirms:

```
STORAM:   LD   A,#0C      ;Reg. 12 des V-Controllers anwählen
          OUT  (#38),A
          IN   A,(#39)    ;Low-Byte der VRAM-Startadresse
          LD   E,A
          LD   A,#0D      ;Register 13
          OUT  (#38),A
          IN   A,(#39)    ;High-Byte der Startadresse
          LD   D,A
;
          LD   HL,#A000   ;lies die Daten nach #A000..#A77F
          LD   C,#32      ;ASCII-Port
          LD   B,24       ;24 Zeilen lesen
LINE:     PUSH BC         ;Zeilenzähler merken
          LD   B,80       ;80 Zeichen pro Zeile
CHAR:     LD   A,D        ;VRAM-Adresse anlegen
          AND  #07        ;Bits 3..7 = 0 setzen (READ)
          OUT  (#31),A
          LD   A,E
          OUT  (#30),A
          INI             ;Zeichen nach (HL), HL+1, B-1
(für Screencopy statt INI: IN B,(C)/CALL #0CE0/DEC B)
          INC  DE         ;nächste VRAM-Adresse
          JR   NZ,CHAR    ;80 Zeichen fertig?
(für Screencopy: LD B,#0D/CALL #0CE0/LD B,#0A/CALL #0CE0)
          POP  BC         ;Zeilenzähler holen (B)
          DJNZ LINE       ;24 Zeilen fertig?
          RET
```

2. Beschreiben des Schirms:

```
RAMTOS:   LD   A,#0C
          .
          bis incl. CHAR: genau wie im 1. Programm
          .
CHAR:     LD   A,D
          AND  #07
          OR   #C0        ;nur Zeichen schreiben (Attr. bleibt!!)
          OUT  (#31),A
          OTI             ;Zeichen bei (HL), HL+1, B-1
          LD   A,E
          OUT  (#30),A    ;Low-Byte VRAM-Adr. und STRDBE
          INC  DE         ;nächste VRAM-Adresse
          JR   NZ,CHAR    ;diese Zeile fertig?
          POP  BC
          DJNZ LINE       ;24 Zeilen fertig?
          RET
```

DISPLAY/ENLARGE

Folgendes Programm verwandelt den VS 4 in einen 40-Zeichen-Schirm (DISPLAY) oder stellt beliebige Zeichenketten vergrößert dar (ENLARGE). Hier die Beschreibung.

DISPLAY

Dieser Teil arbeitet mit Cursor-Positionen. Die Numerierung ist dabei die gleiche wie auf einem Textschirm. Jedes Zeichen ist sechs Punkte breit. Da 6*40 nur 240 und nicht 256 ist, habe ich es so gemacht, daß links und rechts je ein acht Punkte (also ein Zeichen auf dem normalen Grafikschirm) breiter Rand frei bleibt. Der Cursor wird positioniert, indem man seine Zeile in die Variable CSRH und seine Spalte in CSRV hineinschreibt. PAPER- und INK- Farben werden in die gleichnamigen Variablen hineingeschrieben. Die Variable TEXT besteht aus 80 Bytes, in die die abzubildende Zeichenkette geladen werden muß. Als Schlußmarkierung nimmt man #FF. Die Cursorposition erhöht sich automatisch und wird nach 39,23 auf 0,0 gesetzt. Taucht im Text ein CHR$(0) auf, dann wird der folgende Text invertiert dargestellt. Ein weiteres CHR$(0) hebt diesen Effekt wieder auf.

ENLARGE

Hiermit können Zeichenketten auf dem Grafikschirm vergrößert dargestellt werden. Der Faktor der horizontalen Vergrößerung muß in MAGH geladen werden, der der vertikalen in MAGV. Dieser Programmteil arbeitet nicht mit Cursorpositionen, sondern mit Plotpositionen. Dabei liegt Punkt 0,0 in der linken oberen Ecke und nicht, wie normalerweise, in der linken unteren Ecke. In PLOTH und PLOTV lädt man die Plotposition, die dem linken oberen Punkt des ersten abzubildenden Zeichens entsprechen soll (als Textspeicher dient wieder TEXT). Überschreiten die abzubildenden Zeichen den rechten Bildschirmrand, dann kommen sie links auf gleicher Höhe wieder herein. Auch hier kann mittels CHR$(0) die Invertierung umgeschaltet werden.

FLAGS

Beide Programmteile machen von FLAGS Gebrauch. Dieses Byte enthält fünf Flags mit folgender Bedeutung.

BIT	BEZEICHNUNG	MODUS	FUNKTION
0	INV	D/E	Wenn gesetzt, werden die Zeichen invertiert ausgegeben. Umschaltbar durch CHR$(0) im Text.
1	SMALL/LARGE	E	Wenn gesetzt, ist die Breite eines Zeichens ein Vielfaches von acht Punkten, sonst von sechs.
2	COLOUR	D/E	Wenn gesetzt, wird Farbe ausgegeben, sonst nicht. Ohne Farbe ist die Ausgabegeschwindigkeit wesentlich höher.
3	COPY	D	Wenn gesetzt, und MODE 0 ist, dann wird ein 40-Zeichen-Textschirm auf den Grafikschirm kopiert.
7	MODE		Wenn gesetzt, führt das Programm ENLARGE aus, sonst DISPLAY.

TESTPROGRAMM

Das Testprogramm wird mit GOTO 5000 gestartet. Es fragt zuerst nach den FLAGS. Sie müssen binär eingegeben werden (also z.B. 00000101 für farbige, invertierte Textdarstellung). Abhängig von Flags werden alle anderen möglichen Parameter abgefragt, und zwar stets paarweise (außer natürlich FLAGS und TEXT); man gibt also immer zwei Werte ein. Auf CURSOR? kann man demnach z.B. mit 1,2 antworten. Oder man drückt RET, um nichts zu verändern. Bei COLOURS? zuerst PAPER, dann INK eingeben. Sonst immer erst den horizontalen Faktor (Koordinaten oder Vergrößerung), und dann den vertikalen eingeben.

Beim Eintippen des Programms können die Zeilen null bis vier weggelassen werden. Dann muß Zeile 5020 aber lauten: 5020 LET VR=16394-16384*(PEEK(64122)=0). Diese Zeile erkennt, ob der BASIC-Bereich bei #4000 oder bei #8000 beginnt. Wer keine Zeit oder Lust zum Abtippen hat, bekommt das Programm von mir für 20.- DM auf Kassette. Dann bitte angeben, wo der BASIC-Bereich beginnt.

Andreas Viebke

```
0 REM    ******************************
1 REM    *     DISPLAY AND ENLARGE    *
2 REM    *   FOR THE GRAPHIC SCREEN.  *
3 REM    *  (C) 1985  ANDREAS VIEBKE  *
4 REM    ******************************
10 CODE

40CF          JP PROGRAM
40D2 CSRH:    DB 0
40D3 CSRV:    DB 0
40D4 MAGH:    DB 1
40D5 MAGV:    DB 2
40D6 PLOTH:   DB 0
40D7 PLOTV:   DB 0
40D8 PAPER:   DB 5
40D9 INK:     DB 15
40DA FLAGS:   DB #04
40DB CHRBTS:  DS 8
40E3 TWOCHR:  DS 16
40F3 TEXT:    DS 80
4143          DB #FF
4144 PROGRAM:LD A,(FLAGS)
4147          BIT 7,A
4149          JP NZ,ENLARGE
414C          BIT 3,A
414E          JP NZ,COPY
4151 DISPLAY:CALL CTEST
4154          LD HL,ABYTE
4157          LD (HL),#FC
4159          LD HL,TEXT
415C CONT:    CALL GETONE
415F          RET Z
4160          PUSH HL
4161          CALL CALC
4164          POP HL
4165          PUSH HL
4166          CALL CINC
4169          POP HL
416A          INC HL
416B          JR CONT
416D CINC:    LD HL,CSRH
4170          INC (HL)
4171 CTEST:   LD HL,CSRH
4174          LD A,(HL)
4175          CP 40
4177          JR NC,INCV
4179          INC HL
417A OTHER:   LD A,(HL)
417B          CP 24
417D          RET C
417E          LD (HL),0
4180          RET
4181 INCV:    LD (HL),0
4183          INC HL
4184          INC (HL)
4185          JR OTHER
4187          RET
4188 GETONE:  LD A,(HL)
4189          INC A
418A          RET Z
418B          DEC A
418C          JP Z,SWITCH
418F          RES 7,A
4191          PUSH HL
4192          LD L,A
4193          LD H,0
4195          ADD HL,HL
4196          ADD HL,HL
4197          ADD HL,HL
4198          LD DE,#1800
419B          ADD HL,DE
419C          CALL OUTHL
419F          LD HL,CHRBTS
41A2          LD B,8
41A4          LD A,(FLAGS)
41A7          LD C,A
41A8 GETBTS:  IN A,(1)
41AA          BIT 0,C
41AC          JR Z,NOINV
41AE          CPL
41AF NOINV:   DB #E6
41B0 ABYTE:   DB 0
41B1          LD (HL),A
41B2          INC HL
41B3          DJNZ GETBTS
41B5          EI
41B6          INC B
41B7          POP HL
41B8          RET
41B9 CALC:    LD A,(CSRV)
41BC          LD H,A
41BD          LD A,(CSRH)
41C0          ADD A,A
41C1          LD L,A
41C2          ADD A,A
41C3          ADD A,L
41C4          ADD A,8
41C6          LD C,A
41C7          AND #F8
41C9          LD L,A
41CA          CALL OUTHL
41CD          LD DE,TWOCHR
41D0          LD B,16
41D2 FETCH:   IN A,(1)
41D4          LD (DE),A
41D5          INC DE
41D6          DJNZ FETCH
41D8          EI
41D9          PUSH HL
41DA          LD HL,TWOCHR
41DD          PUSH HL
41DE          POP IX
41E0          LD DE,CHRBTS
41E3          LD A,C
41E4          LD BC,#0808
41E7          AND 7
41E9          JR Z,ZERO
41EB          CP 6
41ED          JR Z,SIX
41EF          CP 4
41F1          JR Z,FOUR
41F3 TWO:     LD A,(HL)
41F4          AND #C0
41F6          LD (HL),A
41F7          LD A,(DE)
41F8          RRCA
41F9          RRCA
41FA          OR (HL)
41FB          LD (HL),A
41FC          INC HL
41FD          INC DE
41FE          DJNZ TWO
4200          JR OUT
4202 ZERO:    LD A,(HL)
4203          AND 3
4205          LD (HL),A
4206          LD A,(DE)
4207          OR (HL)
4208          LD (HL),A
4209          INC HL
420A          INC DE
420B          DJNZ ZERO
420D          JR OUT
420F SIX:     LD A,(IX+0)
4212          AND #FC
4214          LD (IX+0),A
4217          LD A,(DE)
4218          AND #C0
421A          RLCA
421B          RLCA
421C          OR (IX+0)
421F          LD (IX+0),A
4222          LD A,(IX+8)
4225          AND #0F
4227          LD (IX+8),A
422A          LD A,(DE)
422B          AND #3C
422D          CALL OP
4230          DJNZ SIX
4232          RLC C
4234          JR OUT
4236 FOUR:    LD A,(IX+0)
4239          AND #F0
```

```
423B          LD (IX+0),A
423E          LD A,(DE)
423F          AND #F0
4241          RRCA
4242          RRCA
4243          RRCA
4244          RRCA
4245          OR (IX+0)
4248          LD (IX+0),A
424B          LD A,(IX+8)
424E          AND #3F
4250          LD (IX+8),A
4253          LD A,(DE)
4254          AND #0C
4256          RLCA
4257          RLCA
4258          CALL OP
425B          DJNZ FOUR
425D          RLC C
425F OUT:     POP HL
4260          SET 6,H
4262          CALL OUTHL
4265          LD DE,TWOCHR
4268          LD B,C
4269 OUTL:    LD A,(DE)
426A          OUT (1),A
426C          INC DE
426D          DJNZ OUTL
426F          LD A,(FLAGS)
4272          BIT 2,A
4274          JR Z,END
4276          SET 5,H
4278          CALL OUTHL
427B          PUSH HL
427C          CALL COLOUR
427F          POP HL
4280          LD B,C
4281 COL:     OUT (1),A
4283          NOP
4284          NOP
4285          DJNZ COL
4287 END:     EI
4288          RET
4289 COPY:    LD HL,0
428C          LD (CSRH),HL
428F          LD HL,#1C00
4292          LD C,12
4294 COPYL:   CALL OUTHL
4297          PUSH HL
4298          LD HL,TEXT
429B          LD B,80
429D COPYLL:  IN A,(1)
429F          LD (HL),A
42A0          INC HL
42A1          DJNZ COPYLL
42A3          EI
42A4          PUSH BC
42A5          CALL DISPLAY
42A8          POP BC
42A9          POP HL
42AA          LD DE,80
42AD          ADD HL,DE
42AE          DEC C
42AF          JR NZ,COPYL
42B1          RET
42B2 OUTHL:   LD A,L
42B3          DI
42B4          OUT (2),A
42B6          LD A,H
42B7          OUT (2),A
42B9          RET
42BA OUTDE:   PUSH AF
42BB          EX DE,HL
42BC          CALL OUTHL
42BF          EX DE,HL
42C0          POP AF
42C1          RET
42C2 COLOUR:  LD A,(INK)
42C5          RLCA
42C6          RLCA
42C7          RLCA
42C8          RLCA
42C9          LD L,A
42CA          LD A,(PAPER)
42CD          AND #0F
42CF          OR L
42D0          RET
42D1 OP:      RLCA
42D2          RLCA
42D3          OR (IX+8)
42D6          LD (IX+8),A
42D9          INC DE
42DA          INC IX
42DC          RET
42DD SWITCH:  LD A,(FLAGS)
42E0          XOR 1
42E2          LD (FLAGS),A
42E5          INC HL
42E6          JP GETONE
42E9 ENLARGE: LD HL,ABYTE
42EC          LD (HL),#FF
42EE          LD HL,TEXT
42F1 NEXT:    CALL GETONE
42F4          RET Z
42F5          PUSH HL
42F6          CALL LARGE
42F9          POP HL
42FA          INC HL
42FB          JR NEXT
42FD LARGE:   LD BC,(MAGH)
4301          LD DE,(PLOTH)
4305          PUSH DE
4306 AGAIN:   PUSH BC
4307          PUSH DE
4308          LD HL,CHRBTS
430B          LD C,8
430D DOWN:    LD A,(FLAGS)
4310          AND 2
4312          ADD A,6
4314          LD B,A
4315          PUSH DE
4316 DOWNL:   LD A,(FLAGS)
4319          BIT 1,A
431B          LD A,B
431C          JR Z,INC
431E          DEC A
431F          DEC A
4320 INC:     INC A
4321          RLCA
4322          RLCA
4323          RLCA
4324          OR #46
4326          LD (TBYTE),A
4329          DB #CB
432A TBYTE:   DB 0
432B          LD A,#C7
432D          JR NZ,SET
432F          LD A,#87
4331 SET:     PUSH BC
4332          PUSH HL
4333          LD BC,(MAGH)
4337 SETL:    PUSH AF
4338          CALL PLOT
433B          POP AF
433C          INC E
433D          DEC C
433E          JR NZ,SETL
4340          PUSH AF
4341          LD A,E
4342          LD (PLOTH),A
4345          POP AF
4346          POP HL
4347 SETEND:  POP BC
4348          DJNZ DOWNL
434A          POP DE
434B          LD A,(MAGV)
434E          ADD A,D
434F          LD D,A
4350          INC HL
4351          DEC C
4352          JR NZ,DOWN
4354          POP DE
4355          INC D
4356          POP BC
4357          INC HL
4358          DJNZ AGAIN
435A          POP DE
435B          RET
435C PLOT:    PUSH HL
435D          PUSH DE
435E          LD HL,SBYTE
4361          LD (HL),A
4362          LD A,#FF
4364          SUB E
```

```
4365          AND 7
4367          RLCA
4368          RLCA
4369          RLCA
436A          OR (HL)
436B          LD (HL),A
436C          LD A,E
436D          AND #F8
436F          LD E,A
4370          LD A,D
4371          CP 192
4373          JR C,CARRY
4375          SUB 192
4377          LD D,A
4378 CARRY:   AND 7
437A          ADD A,E
437B          LD E,A
437C          SRL D
437E          SRL D
4380          SRL D
4382          CALL OUTDE
4385          IN A,(1)
4387          DB #CB
4388 SBYTE:   DB 0
4389          SET 6,D
438B          CALL OUTDE
438E          OUT (1),A
4390          EI
4391          LD A,(FLAGS)
4394          BIT 2,A
4396          JR Z,PLTEND
4398          SET 5,D
439A          CALL COLOUR
439D          CALL OUTDE
43A0          OUT (1),A
43A2          EI
43A3 PLTEND:  POP DE
43A4          POP HL
43A5          RET

Symbols:

20 RETURN
```

```
4000 REM ******************************
4010 REM *        SAVE-ROUTINE        *
4020 REM ******************************
4050 REM
4060 CLEAR : SAVE "VS 4:DISP/ENLA"
4070 REM
4080 REM ******************************
4090 REM *    TESTING MACHINE CODE    *
4100 REM ******************************
5000 CLEAR : VS 5: CLS : VS 4: CLS
5010 CRVS 5,0,1,0,39,24,40: PAPER 5
5020 LET VR=16594-16384*(PEEK(64122)=0)
5030 DIM TEXT$(90): LET FL$="00000000"
5040 VS 5: PRINT : LET FL=0
5050 INPUT "FLAGS? ";FL$
5060 IF FL$="" THEN  GOTO 5100
5070 LET FL=0: FOR I=0 TO 7
5080 LET FL=FL-2^I*(FL$(8-I)="1")
5090 NEXT : POKE VR+8,FL
5100 IF FL$(5)="1" THEN  GOTO 5500
5110 INPUT "TEXT? ";T$: LET LE=LEN(T$)
5120 IF LE>80 THEN  LET LE=80
5130 IF LE=0 THEN  GOTO 5170
5140 FOR I=1 TO LE
5150 POKE VR+I+32,ASC(T$(I)): NEXT
5160 POKE VR+I+32,255
5170 IF FL$(6)<>"1" THEN  GOTO 5210
5180 INPUT "COLOURS? ";P$,I$
5190 IF P$<>"" THEN  POKE VR+6,VAL(P$)
5200 IF I$<>"" THEN  POKE VR+7,VAL(I$)
5210 IF FL$(1)="1" THEN  GOTO 5270
5220 INPUT "CURSOR? ";P$,I$
5230 IF P$<>"" THEN  POKE VR,VAL(P$)
5240 IF I$<>"" THEN  POKE VR+1,VAL(I$)
5250 VS 4: PAUSE 1000: GOSUB 10
5260 DSI : GOTO 5040
5270 INPUT "PLOT? ";P$,I$
5280 IF P$<>"" THEN  POKE VR+4,VAL(P$)
5290 IF I$<>"" THEN  POKE VR+5,VAL(I$)
5300 INPUT "MAGNIFICATION? ";P$,I$
5310 IF P$<>"" THEN  POKE VR+2,VAL(P$)
5320 IF I$<>"" THEN  POKE VR+3,VAL(I$)
5330 GOTO 5250
5500 IF FL$(1)<>"1" THEN  GOTO 5520
5510 PRINT "COPY in mode E?": GOTO 5040
5520 IF FL$(6)<>"1" THEN  GOTO 5560
5530 INPUT "COLOURS? ";P$,I$
5540 IF P$<>"" THEN  POKE VR+6,VAL(P$)
5550 IF I$<>"" THEN  POKE VR+7,VAL(I$)
5560 PRINT : PRINT "Fill the screen"
5570 PRINT "to see how COPY"
5580 PRINT "works.";CHR$(30): PRINT
5590 PAUSE 1000: DSI : VS 4: PAUSE 1000
5600 GOSUB 10: DSI : GOTO 5040
5610 REM ******************************
5620 REM *            END             *
5630 REM ******************************
```

Ausgabe auf 80-Zeichen-Karte (Herbert Herberg)

```
20 CODE

8007 AUSG80: LD H,0
8009         LD A,(Y)
800C         LD L,A      ; HL = y-Position
800D         SLA L       ; *2
800F         SLA L       ; *4
8011         SLA L       ; *8
8013         ADD HL,HL   ; *16
8014         LD D,H
8015         LD E,L      ; DE =  16* y-Position
8016         ADD HL,HL   ; *32
8017         ADD HL,HL   ; *64
8018         ADD HL,DE   ; HL  = 80* y-Position
8019         LD D,0
801B         LD A,(X)
801E         LD E,A      ; DE =  x-Position
801F         ADD HL,DE   ; HL = x +  80*y
8020         LD C,#39    ; Post für Einlesen ders  Offset für 6845
8022         LD A,12     ; Register 12
8024         OUT (#38),A         ; Anwählen
8026         IN D,(C)    ; Einlesen  High-Byte Offset
8028         INC A       ; Register 13
8029         OUT (#38),A         ; Anwählen
802B         IN E,(C)    ; Low-Byte Offset
802D         ADD HL,DE   ; HL  = Offset + x + 80*y
802E         XOR A       ; A = 0
802F         LD (ETXT),A         ; Ende des Textes durch Null markieren
8032         LD BC,TEXT  ; BC = Text-Adresse
8035 LOOP:   LD A,(BC)   ; Zeichen holen
8036         OR A        ; Prüfen ob Null (d.h. Ende)
8037         JR Z,ENDE   ; Fertig wenn Null
8039         OUT (#32),A         ; Zeichen ausgeben
803B         LD A,(ATTR)         ; Attribut holen
803E         OUT (#33),A         ; Ausgeben
8040         LD A,H      ; High-Byte Adresse
8041         AND #07     ; Nur Bit 0-2
8043         OR #E0      ; Beide RAM's
8045         OUT (#31),A         ; Adresse und Write-Enable
8047         LD A,L      ; Low-Byte Adresse
8048         OUT (#30),A         ; Adresse und Strobe
804A         INC BC      ; Nächstes Zeichen
804B         INC HL      ; Nächste Adresse
804C         JR LOOP     ; Weiter
804E ENDE:   RET
804F X:      DB 2        ; X-Position
8050 Y:      DB 2        ; Y-Position
8051 TEXT:   DB "MTX User-Club Deutschland"
806A         DB "          (Herbert Herberg)"
8084 ETXT:   DB 0
8085 ATTR:   DB #44      ; Hell und Blinken
8086         RET

Symbols:
AUSG80  8007    Y       8050
X       804F    ETXT    8084
TEXT    8051    LOOP    8035
ENDE    804E    ATTR    8085
```

WAS IST PIP ? Übertragen von Eta&Herbert Gollnik

Wir hoffen, man/frau kommt klar mit dem Dargestellten und ihr habt eine kleine Hilfe damit.

Programm zum Datenaustausch mit den am Computer angeschlossenen Geräten.Dabei sind zwei Betriebsarten möglich:

1. Der Befehl PIP, wird mit Dateinamen verknüpft womit PIP in den Arbeitsspeicher geht, den Befehl ausführt um anschließend zurrück zum CP/M zu gehen.
2. Hier wird PIP in den Arbeitsspeicher geladen von wo aus es dann die eingegebenen Befehle abarbeitet ohne in CP/M zurückzugehen.Auf diese Art können mehrere PIP-Befehle bearbeitet werden.

Syntax:

1. A>**PIP** NEUER.TYP=ALTER.TYP (RET)

2. A>**PIP** (RET)
   * jetzt 1.PIP Kommando eingeben (RET)
   * 2. dto. (RET)
   * 3. dto. (RET)
   * soll kein weiterer Befehl eingegeben werden die Ausführung mit Leerzeile u.(RET) beenden!

Die Optionen von PIP müssen in [ ], d.h. Ä Ü, und nicht in ( ) wie in diesem Text! Anm. d. Red

USING PIP TO COPY DISC FILES (Mit PIP Diskettendateien kopieren)

Die Syntax der PIP-Befehle ist so, daß grundsätzlich der Dateiname der Neuen Datei -also jener die entstehen soll - direkt nach dem PIP-Befehl eingegeben werden muß. Nach dem "=" Zeichen kommt dann der Name der bereits bestehenden Datei.
Ebenso muß dem jeweiligen Namen die Diskettenstation -von wo nach wo kopiert werden soll- vorangestellt sein.

BEISPIEL: A>**PIP** NEUEDAT.TYP=ALTEDAT.TYP

1.Kopieren einer Datei vom logischen Laufwerk A -das Laufw. von dem aus PIP aufgerufen wurde- zum Laufwerk C. Pysikalisch also vom Laufwerk B: zum Laufwerk C:

A>**PIP C:**NEUE.TYP=ALTE.TYP (Befehlsaufbau wie 1.,s.o.)
oder
A>**PIP** (RET)
*__C:__NEUE.TYP=ALTE.TYP (Befehlsaufbau wie 2.,s.o.)
*(RET)

Falls gewünscht, kann hierbei eine Namensänderung erfolgen.

2.Wie erstens, jedoch mit Benennung von Laufwerk B

A>**PIP C:**NEUE.TYP=**B:**ALTE.TYP

wobei hier die Diskette in Laufwerk B den PIP-Befehl als Datei enthalten muß. Ebenso muß ein eventuellen Diskettenwechsel in B mit ^C mitgeteilt worden sein.

3.Soll die Neue Datei ohne Namensänderung übertragen werden,kann der Befehl verkürzt eingegeben werden:

A>**PIP C:**=ALTE.TYP

4.Sollen mehrere Dateien gleichen TYPs kopiert werden:

A>**PIP C:**=*.TXT d.h.Kopiere alle TXT-Dateien nach C
4.Kopieren von gesamten Disketten

A>**PIP C:**=*.* d.h. alles was auf der Diskette im log.Laufwerk A ist wird auf die Diskette im Laufwerk C übertragen.
ACHTUNG: Systemdateien werden hierbei nicht kopiert sondern nur was über DIR angezeigt wird. Siehe auch weiter unten!

A>**PIP B:**=**C:***.* das gleiche, jedoch von Laufw.C nach B

5.Ablauf eines Kopiervorgangs:

Grundsätzlich läuft das Kopieren mit PIP in zwei Schritten ab. Hieraus ergeben sich Konsequenzen hinsichtlich des Platzbedarfs auf der Zieldiskette.
1.PIP baut beim Kopieren auf der Zieldiskette zunächst eine Zwischendatei auf.Diese wird vor der endgültigen Eröffnung überprüft und mit dem Orginal verglichen.
2.Eröffnen der neu eingerichteten Datei mit Namenseintrag im Inhaltsverzeichnis (DIR) der Zieldiskette.
Daraus folgt allerdings, daß auf der Zieldiskette der **doppelte Speicherplatz** vorhanden sein muß! Es ist daher angebracht mit dem STAT-Befehl eventuell den Speicherplatz bzw.Speicherbedarf zu überprüfen.

USING PIP TO COPY BETWEEN PERIPHERALS (Kopieren mit PIP bzw.Austausch mit den am Computer angeschlossenen Geräten)

Hier geht es um den Datenaustausch zwischen Diskettendateien (FILES) und z.B. Drucker,Bildschirm,Telephonmodem,Tastatur usw. Man kann damit zum Beispiel die Tastatur mit dem Drucker so verbinden, daß dieser wie eine Schreibmaschine läuft.

Die Peripheriegeräte haben dabei Befehlsnamen:

CON : Tastatur bzw. Bildschirm

LST : Drucker,Plotter usw.

RDR : Geräte zum Einlesen von Daten z.B.Telefonmodem, Kartenleser, Lochstreifenleser

PUN : Geräte zum Auslesen, also Senden von Daten z.B. Telefonmodem in die andere Richtung.

A>**PIP LST:**=DATEINAME.TYP — stellt eine Verbindung mit dem Drukker her.Auf diese Weise kann auch eine Neue Datei erstellt werden.

A>**PIP LST:**=**C:**NAME.TYP — schickt die Daten der bezeichneten Datei vom Laufw.C auf den Drucker und macht eine Hartkopie.

A>**PIP CON:**=DATEINAME.TYP — schickt die Daten auf den Bildschirm d.h sie werden angezeigt und können sofern man/frau noch kein Brett davor hat, angesehen werden.

A>**PIP** DATEINAME.TYP =**CON** — eröffnet eine Datei entsprechenden Namens und speichert das über die Tastatur Eingegebene ab.So kann zum Beispiel recht einfach ein Brief geschrieben werden.
Ist man/frau fertig, mit **^Z** beenden.

A>**PIP LST:**=**CON** — so wird der Computer zur Schreibmaschine. Mit **^Z** wird beendet.

A>**PIP CON:**=**RDR** — Schaltet eine Verbindung zwischen Bildschirm und Telefonmodem, was über den Draht kommt wird auf dem Bildschirm angezeigt. (o.g.entsp.)

A>**PIP PUN:**=**CON** — Verbindet Ausgabengerät (Telefonmodem) mit Tastatur/Bildschirm das heißt, alles was angezeigt wird wird auch in den Draht gedrückt und an einen gewillten Empfänger übertragen.

A>**PIP PUN:**=DATEINAME.TYP — schickt bezeichnete Datei übers Telefon ab.

A>**PIP** DATEINAME.TYP=**RDR** — Schafft eine Verbindung zwischen zwei Computern.Der Empfänger muß RDR eingeben.

A>**PIP PUN:**=DATEINAME.TYP — schafft Verbindung zwischen zwei Computern, wobei hier der Sender gemeint ist.

OPTIONS FOR CHANGING FILES (Verändern v.Textdateien während des Kopierens)

Hirbei sind die Optionen Buchstaben, welche der Quellendatei hinten angestellt werden. Mehrere Optionen in einer Klammer sind möglich.Die Optionen können in Verbindung mit den oben beschriebenen Befehlen benutzt werden.

D löscht Buchstaben
F Seitenformat veränd.
N Zeilennummerierung
Q Kopieren beenden
T Tabulator setzen
Z Paritätsbit ändern

E Bildschirm Echo
L alles in Kleinbuchstaben
P Seitenformat
S Kopieren starten
U Nur Großbuchstaben

D verändern der Zeilenbreite während des Kopiervorgangs:
A>**PIP** **B:**=**C:**DATEINAME.TYP**(D80)** ==> 80 Zeichen/Zeile

E Kopieren auf Bildschirm überwachen:
A>**PIP** **B:**=**C:**DATEINAME.TYP**(E)**

F Spezialcode für den Drucker, dieser wird zum nächsten Blatt gezwungen. Wichtig wegen der Abreißperforation von Endlospapier:
A>**PIP** **C:**ARBEITSKOPI.TYP=ORGIALDATEI.TYP**(F)**

L alles in Kleinbuchstaben ausdrucken:
A>**PIP** **C:**NEUE.TYP=**B:**ALTE.TYP**(L)**

N bzw.**N2** Zeilennummerierung N= 1,2,3,4....n
bzw.N2=001,002....n :
A>**PIP** VERSION1.ASM=ORGINAL.ASM**(N)**bzw.**(N2)**

P bzw.**Pn** Seitenform auf Kopie,d.h. wieviel Zeilen pro Seite
P= 60; Pn =n Zeilen:
A>**PIP** **LST:**=DATEINAME.TYP**(P)** ==60 Zeilen/Seite

Q Suchwort ^Z macht eine Kopie bis zu einem bestimmten Wort:
A>**PIP** NEUER.TEX=TOTALE.TEX**(Q** Suchwort ^Z**)**

Der Kopiervorgang geht bis zum Antreffen des Suchwortes Kopiert.Hierbei kann auch ein ganzer Satz als Suchwort eingegeben werden.
Damit auch Kleinbuchstaben eingegeben werden können den Befehl mit Variation 2.(erste Seite) eingeben.

S Startpunkt^Z hier wird ab dem eingegebenen Startwort kopiert: d.h.,die Datei wird gelesen und kopiert ab dem definierten Wort (s.a.oben):
A>**PIP** NEUER.TEX=GANZER.TEX**(S** Startwort ^Z**)**

Von = bis Kopieren:
A>**PIP** NEUER.TYP=GANZER.TYP**(S** suchwort^Z **Q**suchw.^Z**)**

Tn linker Seitenrand bei n Zeichen: (am Absatzanfang einrücken)
A>PIP NEUE.TYP=ALTE.TYP(T5) =linker Rand 5 Zeichen nach rechts.

U Ausdruck nur in Großbuchstaben
A>PIP CON:=C:NAME.TYP(U)

Z Paritätsbit auf 0 bzw.1 setzen für alle Daten einer Datei:
A>PIP NAME.TYP=NAME.TYP(Z)
dreht den jeweiligen 8.Paritätsbit um.

OPTIOS FOR GENERAL COPYING (Befehle fürs allg.Kopieren)

Die folgenden Optionen können mit den obigen Befehlen zusammen verwendet werden. Dabei können mehrere Befehle in die Klammer.

B Blockkopieren **G** Quellendatei anderer Benutzer ansprechen

U Kopie prüfen **W** Überschreiben **R** Lesen von Systemdateien

B Pip kopiert blockweise z.B. 16kB. dies kann mit (B) abgestellt werden wenn auf Bänder kopiert werden soll.
A>PIP ALLES.TYP=RDR:(B)
Die Obergrenze stellt der Arbeitsspeicherumfang dar!

Gn spricht Quellendatei anderer Benutzer an
A>PIP A:=NAME.TYP(G2) =eine Datei des Benutzer 2 wird gelesen.

R Erstellen bzw. Lesen von Systemdateien:
A>PIP C:=NAME.TYP(R) die Datei wird nach C übertragen und dort als Sytemdatei abgelegt.Daraus folgt allerdings, daß die Datei nicht im DIR angezeigt wird. Fügt man beim allgemeinen Kopieren ein (R) an die Quelle,an werden die Systemdateien mit übertragen.

W Lesen von geschützten Dateien.
A>PIP C:=NAME.TYP(R)

V Überprüfen des Kopiervorgangs: (ist Grundsätzlich sinnvoll)
A>PIP C:=*.*(V)

W Überschreiben von R/O Dateien.
A>PIP C:=NAME.TYP(W)

SPECIAL USES FOR PIP (Spezialanwendungen von PIP)

In Folgenden werden spezielle Möglichkeiten von PIP aufgezeigt.

1.Vernüpfen von Dateien zu einer gemeinsamen:

A>**PIP** SAMMELDAT.TYP=DATEI1,DATEI2,DATEI3,DATEI4....

2.Wie 1. jedoch von unterschiedlichen Disketten bzw. Laufwerken:

A>**PIP C:**SAMMELDAT.TYP=**B:**DATEI1.TYP,**C:**DATEI2.TYP usw.

3. Anhängen von Dateien an eine bestehende:

A>**PIP** SAMMELDAT.TEX=SAMMELDAT,DATEI1,DATEI2 usw.

SAMMELN VON DATENDATEIEN (Nicht Text sonder numerische Dateien)

1.Für Sequentielle Dateien vom TYP "O"

A>**PIP** NEUE.DAT=DATA1(O),DATA2(O),DATA3(O) usw.

2.Sammeln von HEX-Dateien

A>**PIP** NEUE.HEX=DATEI1.HEX(I),DATEI2.HEX(I),DATEI3.HEX(I) usw.

PRüFEN VON HEX-DATEIEN

A>**PIP** DATEI.HEX=**RDR:(H)** dabei ist**(H)** die Prüfoption

SPEZIALWERKZEUGE BZW. HILFSMITTEL

BEFEHL	:	RESULTAT
EOF	:	Dateiende Beispiel: A>**PIP PUN:**NEUE.TYP**,EOF** Überträgt bis zum Dateiende EOF
NUL	:	Sendet 40 Nullen zu Beginn und Ende A>**PIP PUN:=NUL:C:**NAME.TYP**,NUL:** Vor und nach der Datei wird ein aus Nullen bestehender Vor -bzw.Nach-spann geschrieben.
PRN	:	Drucken von Dateien wie bei LST,jedoch mit Tabulatorzwischenraum nach Zeilen-nummer Beispiel:A>**PIP PRN:**=DATA.ASM
INP	:	Benutzen von externen Programmen zusammen mit PIP Beispiel:A>**PIP** NAME.PRO=**INP:**
OUT	:	Beispiel:A>**PIP OUT:**=CON

R O M - G E T (Herbert Herberg) Seite 1

```
0 GOTO 3
1 CODE

800E          JP START
8011 PAGE:    DB 0
8012 VON:     DW 0
8014 BIS:     DW 0
8016 NACH:    DW 0
8018 LEN:     DW 0
801A CLEN:    DW 0        ; Länge CODE-Zeile
801C RLEN:    DW 0        ; Länge REM-Zeile
801E START:   LD HL,#C000
8021          LD DE,(#FAA7)      ; BASIC - Ende
8025          INC DE
8026          XOR A
8027          SBC HL,DE
8029          LD B,H
802A          LD C,L
802B          LD HL,(#FAA7)
802E          LD (HL),#20
8030          LDIR        ; Alles ab BASIC-Ende bis #C000 mit Blank füllen
8032          LD DE,(VON)
8036          LD HL,(BIS)
8039          XOR A
803A          SBC HL,DE
803C          INC HL
803D          LD (LEN),HL        ; Länge des Assembler-Segmentes
8040          LD A,D
8041          OR #60      ; für BASBOT=#8000: OR #A0
8043          LD D,A      ; DE=VON mit korrigiertem High Byte, so daß
8044          LD (NACH),DE       ; VON=#abcd -> NACH=#?bcd
8048          LD HL,(NACH)
804B          LD BC,(LEN)
804F          DEC HL      ; CODE - Zeile generieren
8050          LD (HL),B   ; BC = Länge des Codes
8051          DEC HL
8052          LD (HL),C
8053          DEC HL
8054          LD (HL),#C2        ; Token
8056          DEC HL
8057          LD (HL),0   ; Zeilennr. 100
8059          DEC HL
805A          LD (HL),100
805C          LD A,C
805D          ADD A,B
805F          JR NC,A8    ; Carry? (d.h. Übertrag bei Addition)
8061          INC B
8062 A8:      LD C,A
8063          LD (CLEN),BC       ; Zeilenlänge = Code + Header
8067          DEC HL
8068          LD (HL),B   ; Zeilenlänge
8069          DEC HL
806A          LD (HL),C
806B          DEC HL
806C          LD (HL),#FF        ; Ende REM - Zeile
806E          LD DE,(#FAA7)
8072          XOR A
8073          SBC HL,DE
8075          INC HL
8076          LD (RLEN),HL       ; Von Programmende bis CODE-Zeile: REM
8079          LD DE,(RLEN)
807D          LD HL,(#FAA7)
8080          LD (HL),E   ; Länge REM -  Ziele
8081          INC HL
8082          LD (HL),D
8083          INC HL
8084          LD (HL),90 ; Zeilennr. 90
8086          INC HL
8087          LD (HL),0
8089          INC HL
808A          LD (HL),#80        ; REM - Token
808C          LD HL,(#FAA7)
808F          LD DE,(RLEN)
```

R O M - G E T (Herbert Herberg) Seite 2

```
8093        ADD HL,DE
8094        LD DE,(CLEN)
8098        ADD HL,DE
8099        LD (#FAA4),HL      ; Ende + REM + CODE = neues Ende
809C        LD (#FAA7),HL
809F        LD (#FAAC),HL
80A2        DEC HL
80A3        LD (HL),#FF        ; Ende CODE - Zeile
80A5        LD HL,(VON)
80A8        LD DE,(NACH)
80AC        LD BC,(LEN)
80B0        LD A,(PAGE)        ; Seite
80B3        DI
80B4        OUT (0),A  ; für FDX: OUT (7),A
80B6        LDIR       ; Code holen
80B8        LD A,(#FAD2)       ; Korrekte Seite
80BB        OUT (0),A  ; für FDX:  OUT (7),A
80BD        EI
80BE        RET

Symbols:
PAGE    8011    VON     8012
BIS     8014    LEN     8018
NACH    8016    A8      8062
RLEN    801C    CLEN    801A
START   801E
```

```
2 STOP
3 CLS : CSR 8,2: PRINT "  R O M - G E T": GOSUB 30
4 CSR 6,6: INPUT "  Page ";P$: LET P$=RIGHT$("0"+P$,1): CSR 13,6: PRINT P$;"    "
5 CSR 6,8: INPUT "  Von  ";V$: LET V$=RIGHT$("0000"+V$,4): CSR 13,8: PRINT V$;"    "
6 CSR 6,10: INPUT "  Bis  ";B$: LET B$=RIGHT$("0000"+B$,4): CSR 13,10: PRINT B$;"    "
7 LET X$="4011": GOSUB 10: LET O=X: LET X$=P$+"0": GOSUB 10: LET P=X: LET X$=V$: GOSUB 10: LET V=X: LET X$=B$: GOSUB 10: LET B=X
8 POKE O,P: POKE O+1,INT(MOD(V,256)): POKE O+2,INT(V/256): POKE O+3,INT(MOD(B,256)): POKE O+4,INT(B/256)
9 GOTO 1
10 LET X=0
11 FOR I=1 TO LEN(X$)
12 LET X=X*16
13 LET Y=ASC(X$(I,1))-48: IF Y>9 THEN LET Y=Y-7: IF Y>15 THEN LET Y=Y-32
14 LET X=X+Y
15 NEXT
16 RETURN
20 CLEAR : SAVE "ROM-GET": CLEAR : VERIFY ""
30 CODE
```

```
86B3        LD HL,#0000        ; S i e h e   T e  x t !!!!!!
86B6        LD (#FAA4),HL
86B9        LD (#FAA7),HL
86BC        LD (#FAAC),HL
86BF        RET

Symbols:
```

```
31 RETURN
```

A c h t u n g:

Die Assembler-Zeilen, deren Adresse unterstrichen sind,
müssen ggf. **geändert** werden!

MTX User Club Deutschland

Anschluß von 8"-Laufwerk am FDX (Siegfried Rauth)

Seit Januar besitze ich das FDX-System mit zwei 5 1/4"-Lauf-werken des Typs EPSON SD 521. Die im INFO 3 ausgeführten positiven Bemerkungen von Bernd Preusing hierüber kann ich nur unterstreichen. Sie sind schnell, leise und zuverlässig sodaß das Arbeiten damit Spaß macht.

Um vorhandene Software nutzen zu können, machte ich mich vor zwei Wochen daran als drittes ein 8" Shugart Laufwerk anzuschließen. Laut Technical Manual des FDX kein Problem..... Aber wie immer steckt auch hier der Teufel im Detail.

Nach dem Aufschrauben des schönen schwarzen Kastens und einem ersten ungläubigen Blick auf das Controllerboard herrschte auf dem Einbauplatz "J2" (hier sollte sich ein 50-poliger Stecker zum Anschluß der 8"-Laufwerke befinden) eine gähnende Leere. Auch von den so wunderschön beschriebenen 8-way DIL-switches keine Spur.Weiterhin fehlten die für das spätere Nachrüsten der RAM-Disc's erforderlichen Kartenstecker-Leisten auf der "Backplane".

Ein Anruf bei VOBIS in Aachen schaffte Klarheit. Die fehlenden Teile fielen bei MEMOTECH einer Sparwelle anheim. Zugunsten VOBIS muß ich sagen, daß man sich dort bereit erkärte mir das nur "halbbestückte" Controllerboard gegen ein voll-ständiges zu tauschen, womit mein Problem gelöst war. Offen blieb die Frage nach dem Zweck zweier ebenfalls auf der Platiene unbestückt gebliebenen Einbauplätzen für WesternDigital-IC's. Sollten sich hierdurch etwa Funktionseinbußen gegenüber dem im deutschen Handbuch zum MTX ausdrücklich erwähnten vollständigen Western Digital IC-Satz ergeben? Fragen über Fragen.....

Nachdem diese Hürde nun genommen war, war der Rest dann ein Kinderspiel. Stecker rein und ....denkste! Es steht zwar im Technical Manual unter der Rubrik "THE CONNECTORS" beschrieben, verblüffend bleibt es dennoch : die Belegung der Stecker ist der üblicherweise an den Diskettenlaufwerken verwendeten genau entge-gengesetzt!

Auch das Head-load Signal entsprach nicht der beschrie-benen Funktion. Der Head-load Magnet wurde permanent angesteuert, sodaß der Schreib/Lesekopf ständig auf der Diskette auflag. Um hier Abhilfe zu schaffen rangierte ich auf dem Laufwerk einfach die Headload- auf die Drive-Select-Leitung und schon war alles im Lot.

Aber nichtnur Wunder sondern auch Fehler gibt es immer wieder. Beim Einbau der Steckerleiste zum Anschluß externer 8"-Laufwerke auf der Gehäuserückseite des FDX fiel mir auf, daß dort die Beschriftung des Steckereinbauplatzes für "EXTERNAL 8" DISC DRIVES" mit dem für "BUS EXPANSION" vertauscht wurde. Die 50-polige Steckerleiste paßt genau in die unterste Gehäuseaus-sparung. Na auch gut!

Der Betrieb mit dem Laufwerk hat sich seitdem bewährt. Da das IBM 3740 Format entsprechend unserem Configurationscode 10 schlechthin das Standardformat für 8" Disketten ist gibt es nun mit der Beschaffung von Software keine Probleme mehr.

55

Binder A.

****
* Hallo-MTX-Freunde! *
****

Betrifft: Fehler im FDX-Basic
(INFO 3 Beitrag von Frank Dersewski)

Eine SEQUENTIELLE Datei muß immer bis zur letzten Eintragung gelesen werden!

Z.B.
```
10 REM ######### WERTE SICHERN ##########
20 CLS
30 DISC OPEN #1,"C:TEST.DAT","O"
40 DIM A$(10,20)
50 FOR B=1 TO 6
60 PRINT B;".te Eingabe :";:INPUT B$
70 LET A$(B)=B$
80 DISC PRINT #1,B$
90 NEXT B
100 DISC CLOSE #1
110 CLEAR:STOP
```

Falls Sie mit Daten arbeiten,deren längen festgelegt sind,ist es auf jedem Falle besser eine RANDOM Datei zu wählen.

Wenn Sie in ihrer SEQUENTIELLEN Datei die fünfte Eintragung lesen wollen,müßen Sie die gesammte Datei lesen und dabei gesondert die fünfte Eintragung in einer Variablen abspeichern.
Da dieses Verfahren sehr zeitraubend ist,lohnt es sich nur wenn Sie mit Daten arbeiten,deren längen unterschiedlich sind.
Ich arbeite an einem Wörterbuch Englisch-Deutsch. In dieser Datei habe ich zum Beispiel alle englischen Wörter,die mit AD beginnen (135 Stück)in einer File: WBUCH.AD gespeichert.

Nun wieder zurück zu Ihrem Problem.

```
500 REM ##### FUENFTE EINTRAGUNG LESEN ######
510 LET FIND$="?"
520 CLS
530 DISC OPEN #1,"TEST.DAT","I"
540 FOR B=1 TO 1000
550 IF EOF(1)=-1 THEN B=1001:GOTO 580
560 DISC INPUT #1,B$
570 IF B=5 THEN LET FIND$=B$
580 NEXT B
590 DISC CLOSE #1
600 PRINT "Fünfte Eintragung: ";FIND$:STOP
```

Dies ist alter FDXB (defekt bzgl RANDOM-Datei)

Frage:
* Wie wird mit dem FDX-Basic der 128/K Speicher angesprochen?
FDX-Basic oder MTX-Basic vertragen sich nicht mit der SILICON-DISC
* Liegt es an der alten FDX-Basic Version? Wer hat eine neue Version?

Antwort zu 128K : garnicht!!

Abspeicherung von Basic-Zeilen (Herbert Oppmann)

Wie eine Basic-Zeile im Groben aussieht, wurde in den Club-Infos bereits mehrmals beschrieben. Zur Erinnerung möchte ich es kurz noch einmal erwähnen: nach 2 Byte Zeilenlänge folgen 2 Byte Zeilennummer, dann n Byte Text, dann #FF (Endmarker).
Ich habe mich nun mit dem Text näher befaßt und einige Interessante Dinge herausgefunden. Kommen Schlüsselwörter vor, so werden sie als eine einzige Zahl (Token), alle anderen Zeichen mit ihrem ASCII-Code abgespeichert, und zwar in der Reihenfolge ihres Vorkommens in der Basic-Zeile.
Achtung: Rechenzeichen sind Tokens, außer wenn sie in Strings stehen. Spaces vor und nach Tokens werden nicht mitgespeichert.
Eine große Ausnahme zu dem vorher Gesagten bilden jedoch Assembler-Zeilen, die ich näher untersucht habe. Nach #C2 ("Code") folgen 2 Byte, die die Länge des ablauffähigen Assemblerprogramms angeben. Dann folgen n Byte ablauffähiges Programm, abgeschlossen mit #C9 (Ret). Jetzt kommen alle zusätzlichen Informationen, soweit vorhanden. Sie werden jeweils angeführt durch einen Marker, der ihre Art angibt (siehe Tabelle).

Marker Bed.	Weitere Bytes+Bedeutung
02 Label	2 Byte Position (*) des Labels im Asssprg. n Byte Text des Labels (+) 1 Byte #00 Bedeutung unbekannt 1 Byte Anz.der absoluten Aufrufe d.Labels (Call;Jp;Ld HL,..) entsprech.Anzahl:je 2 Byte Pos.(*), geordnet nach abfallenden Zahlen,d.h. umgekehrt wie im Programm 1 Byte Anz.der relativen Aufrufe des Labels (Jr, Jr NZ usw.) entspr.Anzahl:je 2 Byte Position (*) (wie oben)
00 Verschied. Texte(auch nichtdef. Labels)	n Byte Text (+) weitere Byte wie bei #02 (Label) Verwendung:z.B.bei Jp #0000 ist "#0000" so abgespeichert.Nicht verwendet bei DB,DW,DS,Kommentar.
03 DB (define Byte)	2 Byte Position (*) 1 Byte Anzahl der definierten Bytes 1 Byte Länge des Textes nach "DB" n Byte ASCII-Text,ohne Kommas,Ende jedes Datenbytes gekennzeichnet (+)
04 DW (define Word)	wie bei #03 (DB)
05 DS (define Space)	wie bei #03 (DB)
08 Kommentar	2 Byte Position (*) 1 Byte Länge des Kommentars+1 n Byte ASCII-Text + #FF

(*)Die Position wird so bestimmt: Erstes Byte des ablauffähigen Programms bekommt Nummer 0. Bei Sprüngen o.ä. wird bis einschl. zum ersten Byte des variablen Teils des Befehls, d.h. sozusagen

bis zum "nn" gezählt.
(+)Das Ende eines Textes wird durch Setzen des Bit 7 des Codes des letzten Buchstaben gekennzeichnet (beim ZX81 war der Buchstabe dann invertiert).

Wenn bei DB eine Zahl größer als 255 eingetragen und trotz der Warnung "out of range" aus dem Assembler gestiegen wird, so ist nur der Rest der Division dieser Zahl durch 256 im ablauffähigen Programm gespeichert (die ursprüngliche Zahl ist jedoch als Text noch in den Zusatzinformationen enthalten).

Die Zusatzinformationen werden in folgender Reihenfolge abgelegt: Zuerst alle Labels, dann alle Versch.Texte jeweils in der Reihenfolge ihres Auftretens im Programm. Dann folgen in der Reihenfolge der Eingabe alle weiteren Informationen. Abgeschlossen wird die Zeile dann ganz normal durch #FF (Endmarker).

Hardware-Erweiterungen für den MTX

(Klaus Muerling)

Da beim MTX der Systembus seitlich herausgeführt ist, lassen sich Hardware-Erweiterungen eigener Konstruktion anschließen.
Ein Problem ist der 60polige Stecker, den ich bisher noch nirgends fand. Ich habe mich deshalb mit einem 46poligen Stecker beholfen, der normalerweise für den ZX81 Verwendung findet. Diese 46 Pole reichen für die meisten Erweiterungen aus. Man kann diesen Stecker bei Fa. PROFISOFT, Osnabrück, Sutthauser Str. erhalten (13,-DM). Bei diesem Stecker lassen sich die Kontakte und der Führungssteg herausnehmen (**wichtig**) und auf die Belegung des Systembus des MTX anpassen.(Führungssteg in 5).
Der Z80 hat 255 I/O-Port-Adressen. Beim MTX-Grundgerät werden die I/O-Ports bis OFh benutzt, beim MTX mit Floppysystem bis Port 21h. Die oberen Portadressen stehen dem Benutzer für eigene Anwendungen zur Verfügung. Solche Anwendungen sind z. B.:

- Anschluß eines 3-fach I/O-Ports 8255 zur Ansteuerung von weiteren Bausteinen wie AD-Wandlern oder DA-Wandlern
- Mit einer solchen Erweiterung kann z. B. ein Plotter angesteuert werden, oder ein Grafik-Tableau, oder ein Poti-Kreuzknüppel-Joystick und vieles mehr
- Anschluß eines Counter-Timer-Bausteins wie z. B. den 8253 zur Erfassung von Zeiten, Frequenzen, Zählvorgängen. Hiermit lässt sich z. B. ein RTTY-Interface (Fernschreibumsetzer) realisieren, oder ein Frequenzgenerator oder -messer.
- Anschluß von externen Speichererweiterungen oder auch Sub-Prozessoren

Nun hat der MTX zwar schon einen eingabauten I/O-Port. Für viele Anwendungen werden aber mehrere Ports gleichzeitig sowie Leitungen für Steuer- und Kontrollbefehle gebraucht.

Den Stecker für den Systembus lötete ich auf eine Bastelplatine, auf der sich ein bidirektionaler Buspuffer-Baustein (74LS245) zur Pufferung des Datenbusses sowie die Dekodierung der I/O-Adresse (mit 74LS30 und 74LS04) befinden. Diese Einheit ist steckbar. Von dort gehe ich dann mit einem Flachbandkabel über Pfostensteckverbindung auf meine Erweiterungsplatine.

Im unten gezeigten Schaltbild benutze ich die Portadressen DCh bis DFh. Die Adressleitungen A2 bis A7 werden über das NAND-Gatter 74LS30 geführt, wobei A5 über 74LS04 invertiert wird. $\overline{\text{IORQ}}$ wird ebenfalls über 74LS04 invertiert zugeführt werden. Das Ausgangssignal des 74LS30 benutze ich als Chip-Select-Signal sowohl für den 8255 als auch als Enable für den 74LS245.
Angesprochen wird das I/O-Port über die Basic-Befehle OUT (Port),Wert bzw. INP (Port).Beispiel: OUT (223),128 oder LET A=INP(222).

Die Ports des 8255 lassen sich über das Kontrollregister mittels Statuswort verschieden konfigurieren. (In- oder OUT-Port). Das Port C lässt sich zusätzlich 4-Bit-weise als I- oder O-Port definieren.
Eine Tabelle der Steuerworte ist beigefügt.

Hardware-Erweiterungen für den MTX

Fortsetzung
(Klaus Muerling)

An die Ports lassen sich nun AD- oder DA-Wandler-Bausteine anschließen. Hier gibt es verschiedene Bausteine verschiedener Hersteller für 8- und 12-Bit-Wandlung. Ich benutze die Bausteine ZN 425 (AD-DA-Wandler) und ZN 427 (AD-Wandler) der Firma Ferranti. (8 Bit)

Nachstehend ein Schaltplan zur Ansteuerung zweier DA-Wandler ZN425. Mit dieser Schaltung steuere ich einen XY-Plotter an. Mit diesem kann ich Grafik zeichnen lassen. Ebenso kann ich über einen Opto-elektronischen Sensor Grafik von einer Vorlage abscannen und in den Grafik-Screen übertragen.

Zum Schaltbild: Die Spannungsversorgung 5 V nehme ich nicht aus dem MTX, sondern über ein eigenes Netzteil. Ich weiß nicht, wieviel Last extern das MTX-Netzteil verträgt. Der Datenbus ist gepuffert, um weitere Erweiterungen dranhängen zu können.
Der Reseteingang vom 8255 ist activ high!
Die Datenleitungen habe ich zusammengefasst gezeichnet, um eine bessere Übersicht zu haben.

Status-wort	Port A	Port B	Port C Bit 4...7	Port C Bit 0...3
128	out	out	out	out
129	out	out	out	in
130	out	in	out	out
131	out	in	out	in
136	out	out	in	out
137	out	out	in	in
138	out	in	in	out
139	out	in	in	in
144	in	out	out	out
145	in	out	out	in
146	in	in	out	out
147	in	in	out	in
152	in	out	in	out
153	in	out	in	in
154	in	in	in	out
155	in	in	in	in

Kommandoliste für das Steuerregister des 8255.

Ansteuerung

MTX-Port	PIO 8255
220	Port A
221	Port B
222	Port C
223	Kontrollreg.

Ausgabe von zwei Analogspannungen:

```
10 OUT 223,128   STATUSWORT;setzen von A B C = out
20 INPUT A       0 - 255 => 0 - 2.55 Volt
30 OUT 220,A     Analogspannung an Ausg. ZN 425 (1)
40 INPUT B       0 - 255 => 0 - 2.55 Volt
50 OUT 221,B     Analogspannung an Ausg. ZN 425 (2)
```

Pinout 74LS245

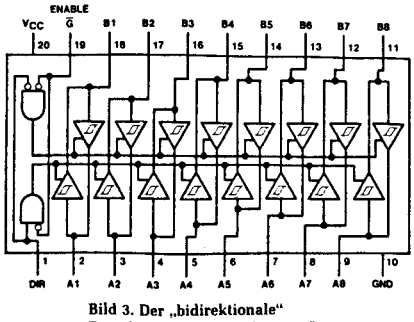


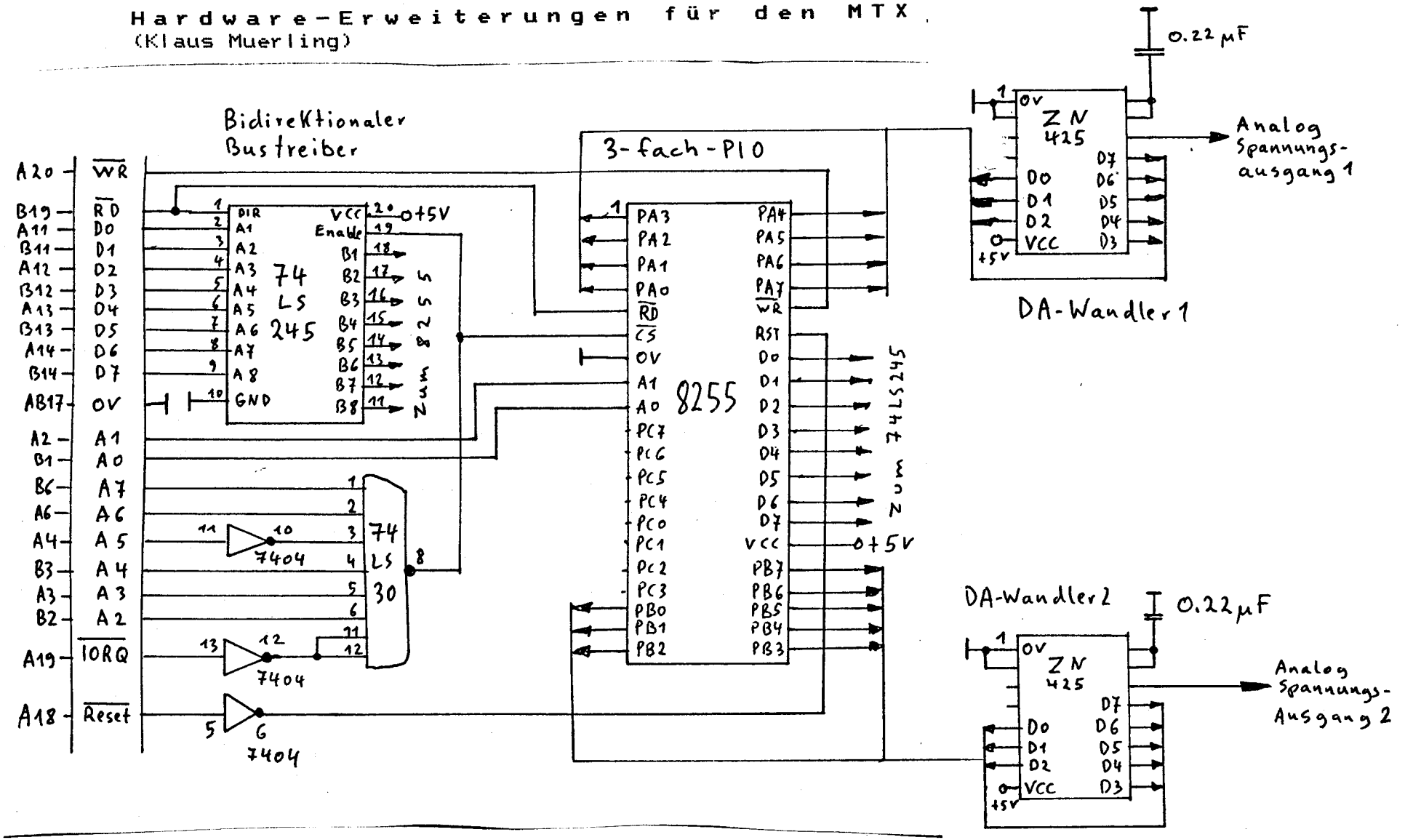
Bild 3. Der „bidirektionale" Datenbus-Puffer-Baustein 74 245

Wahrheitstabelle

Enable $\overline{G}$	Direction Control Dir.	Operation
L	L	B to A
L	H	A to B
H	X	Isolation

Hardware-Erweiterungen für den MTX

(Klaus Muerling)



Der Baustein ZN 425 lässt sich durch andere Beschaltung auch als Analog/Digital-Wandler einsetzen. Das Port C des 8255 verwende ich zur Ausgabe von Steuersignalen z. B. "Pen Down" beim XY-Plotter. Da ich eine externe Spannungsversorgung 5 V benutze, ist nur die Masseleitung zum Systembus geführt.

Der Schaltplan sieht unübersichtlich aus, da die Pins der IC's originalgetreu aufgezeichnet sind!

(Anm. d. Red.)