MTX User-Club Deutschland

Info 28
30.07.1988

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

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

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

Verpflichtungen: Einsendung unseres Anmeldeformulars.

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

Club-Info, unser Blatt, verschicken wir ca. 8-wöchentlich. Inhalt ist alles was uns über den MTX/FDX (ohne Copyright) in die Hände fällt. Es kostet nicht über DM 12.- (90 Seiten) je Exemplar. Jeder kann dazu Beiträge liefern und hier gratis Kleinanzeigen veröffentlichen.

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

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

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

Kontaktadressen: (nach PLZ geordnet)

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

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

Inhaltsverzeichnis

Preis für dieses Info: DM

Redaktionsschluß für Info 29: 4. September 1988 oder auf dem Clubtreffen.

Liebe Leserin, lieber Leser,

dies ist mal wieder ein recht dünnes Info! Die Urlaubszeit macht sich doch irgendwie bemerkbar. Ich wollte und konnte nicht länger mit diesem Info warten, damit der Hinweis auf das nächste **NORDISCHE CLUBTREFFEN** am Samstag, den 10. September 1988 in Hemmingen bei Hannover (siehe Seite 2) noch rechtzeitig die Runde macht.

Die sicherlich lang ersehnte aktuelle Mitgliederliste ist jetzt endlich anbei. Ich habe damit etwas gewartet, da ich verschiedentlich im 1. Halbjahr 1988 Postkarten auf die Reise geschickt habe, um herauszufinden, wer noch dabei ist und Infos haben möchte. Das Resultat war recht drastisch: 10% der angeschriebenen Mitglieder sagten JA. Weitere 2% zuckelten dann noch nach. Daher ist die Mitgliederliste in diesem Info vielleicht überraschend kurz - sie enthält nur 'MTX-ler(innen), die noch voll dabei sind'. Übrigens, hast Du schon den Briefumschlag weggeworfen ? Was macht eigentlich Dein Kontostand ? Langt's für das nächste Info, d.h. sind noch DM 12.- drauf ?

Ich hatte mich übrigens mal wieder an Memotech in England gewendet, um Informationen zu erhalten, und wurde - wie das Mal davor - mit einem Pamphlet des englischen Memotech-Clubs beglückt. Als eine Errungenschaft fand ich in diesem Blatt den Hinweis, daß SuperCalc-Install im Installations-Menü auf den Buchstaben X hin, der nicht angezeigt wird, in interessante Installationsbereiche vordringt. Genauere Informationen hierzu findet Ihr im Info 13, Seite 36! Naja, vielleicht kommt Memotech's Export-Price-List ja noch irgendwann bei mir an!

Da war noch das wohl in Vergessenheit geratene Spiel, in dem es darum ging auf einem Spielbrett durch geschicktes Verschieben von Steinen eine bestimmte Stellung zu erreichen. Außer Jan Bredereke hat niemand eine Lösung gefunden! Damit war's das!

Auf mehrfachen Wunsch verschicke ich nun Public-Domain Disketten nicht nur im Format 03 (bzw. 09 für KLICK) 1B (5 1/4") oder 1C (3 1/2"), sowie auch auf den kleinen 3 1/2"-Scheibchen. SDX-Besitzer können natürlich die PD's auch auf Format 07 erhalten, da die SDX standardmäßig mit 07 arbeitet. Genaueres ist in den Angebots-Listen zu finden. Seht auch mal die Kleinanzeigen durch - es könnte sich lohnen.

Euer Herbert

In eigener Sache: Inhaltsverzeichnisse (Peter Würfel,7262)

Wie Euch Herbert ja schon im letzten Info verraten hat, bin ich jetzt fürs Info-Inhaltsverzeichnis zuständig. Da ich in Zukunft nicht nur die jeweils neuen Infos ins Verzeichnis aufnehmen, sondern auch peu a peu den einen oder anderen Bereich besser(?) strukturieren möchte, hab ich folgende Bitte:

Wenn es Euch mal wieder so geht, wie mir ab und zu, daß Ihr genau wißt, da hab ich doch mal den Artikel gelesen, und der müßte doch unter dem Stichwort zu finden sein, aber wo isser denn? Und dann geht die Blätter- und Sucherei erst richtig los!

Also wenns mal wieder soweit war, und Ihr dann doch endlich fündig geworden seit, dann schickt mir doch ne Postkarte mit folgenden Angaben:

-Info Nr. u. Seite
-Titel
-Euren Vorschlag für den Eintrag in Spalte 'Zweck' im Inhaltsverzeichnis
(das Stichwort unter dem der Artikel eigentlich (auch) erscheinen sollte.)

Ich werde dann Eure Vorschläge in die nächste Ausgabe des Inhaltsverzeichnisses aufnehmen.

Ich hoffe auf viele Tips,

Peter

C l u b t r e f f e n

```
 CCCC  L      U  U  BBBB   TTTTT  RRRR   EEEEE  FFFFF  FFFFF  EEEEE  N   N
C      L      U  U  B   B    T    R   R  E      F      F      E      NN
C      L      U  U  BBBB     T    RRRR   EEE    FFFFF  FFFFF  EEE    N N N
C      L      U  U  B   B    T    R  R   E      F      F      E      N  NN
 CCCC  LLLLL   UUU   BBBB     T    R   R  EEEEE  F      F      EEEEE  N   N
```

Untertitel: **Viele viele Memotechniker und keine Raubkopien!**

Termin Samstag, 10.. September 1988, ca. 10/11 Uhr bis ca. 18/19 Uhr
(Mittagessen ca. 13 Uhr)

Ort 3005 Hemmingen (bei Hannover)

Themen Alles um den MTX, und was sonst noch so zur Sprache kommt

Nachspiel Höchstwahrscheinlich gemütliches Zusammensitzen.

Christian wird versuchen einen Raum incl. Mittagessen (wie das letzte Mal) für ca. DM 25-35.- pro Teilnehmer zu bekommen. Je mehr Teilnehmer, desto billiger pro Person!

Wenn Du Lust hast, dort hinzufahren, dann schick bitte Christian Löhrmann (Anschrift siehe Deckblatt des Infos) umgehend eine Postkarte, jedoch spätestens bis zum 20. August 1988 damit er Deine Teilnahme einplanen kann.

Eigentlich haben wir nichts gegen unangemeldete Gäste, aber dann ist das mit dem Essen und dem Raum so ein Problem.

Den genauen Ort und Termin sowie einen Umgebungsplan schickt Euch Christian rechtzeitig zu!

Folgende Informationen sollten auf der Postkarte stehen:

1. Name, Anschrift, Telefon
2. Ich komme mit ?? Personen, nämlich .....
   (bitte diejenigen, mit denen Ihr kommt namentlich nennen, damit nicht aus Versehen Doppel-Anmeldungen passieren!)
3. Ich möchte (nicht) in Hemmnigen übernachten.
   (Christian kann dann ggf. versuchen ein Hotel-Zimmer o.ä. zu reservieren)
4. Ich möchte gerne sehen, mich interessiert
5. Ich möchte gerne vorführen
6. Ich möchte erst nach dem Mittagessen kommen (Spart Geld für Mittagessen)

Fahrgemeinschaften:
Wenn Ihr aus einer der u.g. Ecken kommt, wendet Euch doch mal an die genannten. (Die Postkarte an Christian ist trotzdem erforderlich!)

- München: Hans-Henning Herder
- Coburg: Dr. Holger Göbel
- Stuttgart: Peter Würfel ??
- Wuppertal: Manfred Flume
- Hamburg: Herbert zur Nedden

L e s e r b r i e f: Jan Bredereke, 2000

Jan Bredereke, Bauernvogtkoppel 65a, 2 HH 65

MTX-User-Club
c/o
Herbert zur Nedden
Sonnenau 2

2000 Hamburg 76

Hallo Herbert!

Anbei einige Artikel für das nächste Info.

Viele Grüße Jan

Antwort zur Frage zu NW von Hartmut Traber in Info 27-3:
(Jan Bredereke, 2000)

Wenn ich die Frage recht verstanden habe, stört sich Hartmut daran, daß sich der Text auf dem Bildschirm nach rechts verschiebt, wenn man Drucksteuerzeichen in den Text einfügt, wie z.B. Fettdruck mit der Schirmcode ^B.
Bei diesem Problem gibt es eine ganz einfache Abhilfe: Mit dem Befehl ^O^D wird die Darstellung der Drucksteuerzeichen abgeschaltet, um die Spalten in eine Flucht zu bekommen. Die hellere Schrift bei Fettdruck usw. bleiben natürlich auf dem Schirm. Mit einem weiteren ^O^D kann man die Darstellung der Steuerzeichen wieder einschalten. Die jeweilige Einstellung gilt natürlich nur bis zum Verlassen von Newword. Ich glaube, die Voreinstellung bei Aufruf von NW läßt sich auch irgendwo mit NWINSTAL ändern.

Zum Leserbrief von Stefan Hößler in Info 27-42:
(Jan Bredereke, 2000)

Die Fehler, die Stefans gepatchtes Newword zusammen mit NWSCHIRM macht, treten bei mir nicht auf. Die Textfetzen, die beim Blättern auf dem Schirm bleiben, stammen vermutlich daher, daß die Newword-interne Tabelle mit den Zeilenlängen zerstört wird. Als ich NWSCHIRM entwickelte, trat dieser Fehler bei mir auf, als ich mit NWSCHIRM Newword vorgaukelte, diese Tabellen seien für mehr Bildschirmzeilen ausgelegt, als dies beim Programmstart der Fall war. Damit überlappten sich natürlich verschiedene Tabellen, und der Fehler trat auf. Seitdem sorgt der Patch, der in Newword eingebaut wird, dafür, daß Newword beim Programmstart daran glaubt, daß der Schirm 110 mal 70 Zeichen groß ist, womit die Tabellen groß genug angelegt werden.
Dies kannst man übrigens einfach überprüfen: Wenn beim Programmstart von Newword zum erstenmal die Kopfzeile mit den vielen Unterstrichen gezeichnet wird, erscheint sie für einen Sekundenbruchteil in der Länge 110 auf dem Schirm, so daß eine Reihe von Unterstrichen in der zweiten Zeile aufblitzt. Ist dies nicht der Fall, so muß beim Patchen mit Herberts Patchprogramm eine ganze Menge schiefgegangen sein. (Falsche Newword-Version, PD-Diskette defekt, ...?)
Weshalb sich meine Klick-Programme KWecker und Weck-Aus bei Stefan nicht laden lassen, weiß ich nicht. Beide benutzen denselben Klix-Lader, und irgendetwas muß bei Stefan in RAM4 so seltsam aussehen, daß der Lader sich daran verschluckt.

L e s e r b r i e f: Peter Würfel, 7262

Dämmrige und dunkle Stunden

Macke in datum1.inc von club.024:

Wenn ich dieses Programm anwende, zeigt sich, daß die Wochentage des Februar 1988 falsch berechnet werden!

Anm.d.HzN.: Claudio Romanazzi schickte mir einen Korrektur-Hinweis zu einem Artikel aus der HöRZU. Daraus ist für den Februar folgendes in Sachen 29. zu entnehmen:
Jedes durch vier teilbare Jahr ist ein Schaltjahr, mit Ausnahme der vollen Jahrhunderte, deren erste beiden Ziffern sich **nicht** durch vier teilen lassen. D.h. im Jahr 2000 hat mein Bruder Geburtstag!

Macken(?) in Moni2:

Wenn man den Pascal-Patch von Info 22-11 mit Moni2 durchführen möchte, muß das zu patchende TURBO.OVR zuerst in TURBO.COM umbenannt werden (natürlich darf dann das 'richtige' TURBO.COM nicht auf diesem Laufwerk sein), dann patchen und anschließend wieder zurück'ren'amen. Versuche ich mit Moni2 TURBO.OVR aufzurufen und zu patchen, scheint auch alles zu klappen, nur das Ergebnis funktioniert nicht: Turbo-Aufruf führt zu Absturz.

Die Assembler-Eingabe 'LD A,C3' wird nicht akzeptiert.

Anm.d.HzN.: Man nehme 'LD A,0C3'.
Hex-Zahlen, die mit einem Buchstaben beginnen, also mit A, B, ... oder F müssen eine Null vorangestellt bekommen. Jeder vernünftige Z80-Assembler verlangt selbiges.

Die Assembler-Eingabe 'SBC HL,DE' (z.B. Info 7-34) führt zum Absturz von Moni (stilles Verharren des Cursors und keine Tastatureingabe außer SHIFT ESC mehr möglich)

Anm.d.HzN.: Du hast evtl. einen alten MONI! MONI 2.03 vom 12.02.87 hat damit keine Probleme.

Was nutzt der schönste Turbo-Patch?

Turbo wurde lt. Info 22-11 gepatcht; wenn ich nun z.B. XDIR aufrufe, wird das Programm zwar gestartet, das Directory angezeigt, aber dann wird der Bildschirm sofort gelöscht und ich bin wieder in Pascal. Da ich nicht so schnell gucken kann, was tun?

Anm.d.HzN.: Terminal-UnInit-String mit TINST herausnehmen.

Eine MackÜküküe im Tastaturtreiber von RAM 4.2 (?):

SHIFT+_+LINE FEED ergibt _Ü_Ü_Ü_Ü usw

Anm.d.HzN.: Nein, RAM4 ist unschuldig! Das liegt an der Hardware der Tastatur! Es geht einfach nicht immer 3 Tasten eindeutig zu erkennen, da Memotech die Tastatur ohne eine eigentlich erforderliche Dioden (je Taster eine Diode) gelötet hat. Mit 77 Dioden und der entsprecheden Zeit (77 Leiterbahnen auftrennen ...) kann dem abgeholfen werden.

L e s e r b r i e f: Peter Würfel, 7262

Ein Wenn-mans-nicht-weiß-dBASE-Dämmerstündchen:

dBASE sendet, wenn eine Zeile 'voll' ist sofort ein LF an den Drucker. Im folgenden Beispiel ist der Drucker z.B. auf PICA compressed eingestellt (d.h. 132 Zeichen/Zeile):

```
STOR 'abc...(mehr als 132 Zeichen)...' TO irgendwas
SET PRINT ON
? $(irgendwas,1,132)
? $(irgendwas,1,132)
```

Das ergibt drei (bzw.4) Zeilen; zwischen dem ersten und zweiten 'irgendwas' prangt eine Leerzeile.

Zwei MENU.COM Schattenstündle

MENU.CPR darf nicht größer als 16k sein! Wers trotzdem riskiert, hat nicht mal mehr mit SHIFT+ESC und Warmboot 'ne Chance; endlich hat er mal nen Grund die zwei Reset-Tasten zu bedienen. Was nützen mir da die versprochenen 255 Bildschirme? (Aber vielleicht findet sich ein Kräck, der dem Programm diese Einschränkung austreibt!)

Um dem Betriebssystem einen Diskettenwechsel mitzuteilen, geb ich ^C ein; in Menu führt das jedoch zum Ausstieg. Und nun das Schattige: wenn ich von Menu aus xdir aufrufe, wird mir ein directory angezeigt (was denn sonst). Wenn ich dann (zurück in einem Menue-Bildschirm) erneut xdir aufrufe, aber in der Zwischenzeit die Diskette auf diesem LW gewechselt habe, zeigt sich manchmal das Inhaltsverzeichnis der neuen Diskette, manchmal nur das der vorhergehenden Diskette; ein System, warum und wann das eine und wann das andere, ist nicht zu erkennen.
Wie ich mir beholfen habe? Bevor ich xdir aufrufe, rufe ich grundsätzlich cfig4 auf (nutzts auch nix so schads auch nix): das meldet mir alle LWerke als neu.

Ein(e) Info-Dämmer-...(monat/jahr):

dBASE-Tips in Holländisch, und das einem Südlicht wie mir (nicht jede Birne ist 'ne Leuchte). Wie wärs mit 'ner Übersetzung (für Ostfriesen soll ja Holländisch nicht all zu schwierig sein und den ersten Beitrag den es mal in Schwyzerdytsch geben sollte, verpflichte ich mich dann zu übersetzen!)

Zwei dSPWW (PWW=Peterwürfelwissen)

1. In einem dBASE-Programm fand ich folgenden Druckerbefehl:
   ```
   CHR(27)+CHR(108)+CHR(6)
   ```
   Was soll das?

2. Club.024 PRTNEU : was ist daran fehlerbereinigt?

L e s e r b r i e f: Wolfgang Kühn, 2399

Her mit dem Schampanjer !

(Literatur-Tip von Peter Würfel, 7262, in Info 27 - 49)
Bei mir klappt's mit dem RESET-Befehl: Ich habe die Schneider-Version 2.41 von dBaseII. dBase mit Overlay befindet sich in Laufwerk B:. Die Dateien sind in Laufwerk C:. Installiert wurde dBase so, daß es sein Overlay auch in B: findet. Ich rufe es von C: aus auf mit: C>B:dbase<RET>.
Nach Gebrauch einer Datei (auf C:) mit anschließendem .use will ich eine Datei auf einer anderen Diskette bearbeiten. Also die neue Diskette in C: 'rein (natürlich vorher die alte 'raus), und dann: .reset C:. Dabei muß das "C" groß geschrieben werden. So läßt sich die neue Datei ohne R/O z.B. erweitern. PROST!!!

ZCPR-Kommando GET

Es enthält wohl einen Bug, denn lade ich mit get 100 file.typ ein File nach Adresse 100H und gebe danach dir ein, so entsteht irgendwie Müll (zum Ausprobieren!). Gemerkt habe ich es zufällig: nach einem GET-Befehl wollte ich zwei Files mit ERA file.* löschen. Auf dem Bildschirm wurden nun ein File richtig und ein wichtiges Datenfile von mir als gelöscht angezeigt. Welch ein Schreck! Aber zum Glück wurden die Richtigen gelöscht, nur eben nicht angezeigt. Hat jemand schon diesen Bug behoben?

STAR NL-10

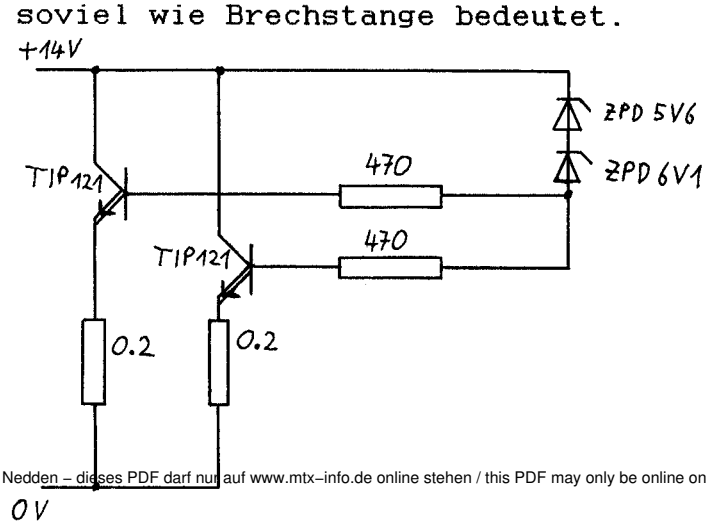
Hat noch jemand diesen Drucker? Ich habe eine Druck-Routine für Wordstar-Texte in Turbo-Pascal geschrieben, mit der man die besonderen Features dieses Druckers einfach ausnutzen kann (ohne große Klimmzüge wie Patches etc.). Z.B. Tiefstellen oder Hochstellen oder schnelles Unterstreichen oder Elite und sonst noch einiges mehr. Die Routine arbeitet wesentlich schneller als das WS-Drucker-overlay. Sie kann bei Bedarf leicht geändert werden. Jetzt ist Elite wieder aus.
Auch die Zeilenhöhe wird jetzt richtig (Das geht z.B. auch.

bearbeitet (stufenlos einstellbar in 1/48 Zoll-Abstaänden.)
Ferner habe ich auf der Rückseite des Centronics-Interfaces einen kleinen Schalter, mit dem ich bei Bedarf auf IBM-Mode umschalten kann (von wegen MS-Dos-Texte etc.). Benötigt werden für diesen Umbau nur ein Eprom 27256, ein Widerstand und ein Schalter. Wenn Interesse besteht: 04638/1266 lautet meine Telefonnummer.

Bräter im FDX

(Anleitung von Hagen Wenzek, 5300, in Info 13 - 16)

Auf der Rückseite der Slot-Karte in der FDX sitzt ein Lastwiderstand an der 12V-Versorgungsspannung. Durch seine große Wärmeentwicklung sind er selbst und die Platine meist ganz schön braun bis schwarz. Hagen Wenzek empfahl nun, diesen Widerstand sofort zu entfernen. Ich habe es nicht sofort getan, denn:
Memotech hat diesen Widerstand als Belastung in jene FDX's eingelötet, die keine Harddisk und keine Silicon-Disk besitzen. Er dient also nur als Belastung für den 12V-Zweig. Wird er entfernt und laufen die beiden Floppy's auch nicht, so steigt die Spannung auf etwas über 14V an. Das mag aber der Floppy-Controller FD1791 laut Datenblatt gar nicht gerne. Außerdem geht diese Spannung in die Regelung des Schaltnetzteils ein. Die Folge ist, daß dieses nun herunterregelt (natürlich auch die 5V!). Und das ergibt dann wiederum neue Schwierigkeiten wie z.B. Boot-Probleme, Aufhänger etc.. Ähnlich verhält sich das Schaltnetzteil, wenn es durch Ändern von R24 bzw. R26 hochgeregelt wird (Info 18 - 37, Info 21 - 34 und Info 25 - 29). Werden die 5V hochgedreht, steigen die 12V unkontrolliert wesentlich höher an. Das ist nun mal der Nachteil bei primär getakteten Schaltnetzteilen mit mehr als einer zu regelnden Größe. Ich habe das Prinzip der Grundbelastung beibehalten bzw. an eine andere Stelle verlagert und brauche die 5V nun nicht zu verstellen, denn der Einfluß der 5V auf die Regelung ist wesentlicher größer als der der 12V. Dazu habe auch ich den Lastwiderstand an der Slot-Karte entfernt und die 12V über das Tastaturkabel (die CPU-Platine 'hängt' in der FDX) in die Tastatur geführt. Dort habe ich an den Stellen, wo die Handballen beim Schreiben aufliegen (natürlich auf der Innenseite der Tastatur), zwei Leistungsdarlingtons TIP 121 eingebaut und entsprechend mit 4 Widerständen und zwei Zenerdioden verdrahtet. In der FDX liegen nun etwa 12.5V an und außerdem habe ich keine eiskalten, verkrampften Handballen mehr. Für die Darlington's können natürlich auch andere andere Typen wie z.B. BD 675, BD 695 oder TIP 140 (oder ist TIP 145 der NPN-Typ?) eingebaut werden. Je nach Zuleitungskabel / Transistor / Netzteil können die Zenerdioden eventuell etwas variert werden (zusammen >=11.5V). Auch die Gegenkopplungswiderstände 0.2 Ohm können bei Bedarf etwas verändert werden. Auf alle Fälle sollte die Schaltung erstmal an einem externen Netzgerät ausprobiert werden: bei 13V am Eingang sollten nicht mehr als 0.5 bis 0.8 Ampere fließen.
Übrigens heißt die crawbar im Artikel Info 25 - 29 richtig crowbar mit o, was soviel wie Brechstange bedeutet.



H a r d w a r e: Laufwerke

Automatische Umschaltung für Laufwerke (Herbert zur Nedden, 2000)

Im Info 26, Seite 29 war ein Artikel mit eben dieser Überschrift von Michael Kessler, den ich aufgrund eines Telefonats mit Michael aufgenommen habe.

Da ich mehrfach gefragt wurde, was damit überhaupt gemeint ist, möchte ich an dieser Stelle meinen Fehler 'Kanppheit' wieder gutmachen:

Die TEAC FD55 GFV und GFR-Laufwerke können ja zwischen dem Normal-Density und dem High-Density Modus umgeschaltet werden. Im ND-Modus verhalten sie sich wie die altbekannten Single/Double-Density 5"-Laufwerke (z.B. Formate 03, 07, 09, 0A, bzw. IBM 360kB, 720kB), während sie sich im HD-Modus wie 8"-er geben (z.B. Formate 10, 13, 1B, IBM 1.2MB). Das EPSON SD 580 bietet auch diese Option.

Auch das neue TEAC FD135 HFN kann zwischen ND- und HD-Modus umgeschaltet werden, wobei es allerdings nicht wie die o.g. Laufwerke im HD-Modus mit 360, sondern 300 Umdrehungen/Minute drehen, und so im HD-Modus die Formate 1C und IBM-PS/2 1.44MB unterstützen. Es leistet also das gleiche wie ein TEAC FD55 GFV, welches mit der im Info 26 beschriebenen Motor-Umschaltung auf 300 Umdrehungen im HD-Modus gesetzt wird.

Bislang war es eigentlich die gängige Praxis, die Umschaltung zwischen ND- und HD-Modus mit einem Schalter zu realisieren - schließlich ist das einfacher zu machen, als Löterei auf der Floppy-Controller-Platine, aber dafür muß man immer wieder Hand anlegen.

Die TEAC-Laufwerke verwenden den Pin 2 des Floppy-Daten-Flachbandkabels für die Umschaltung ND/HD-Modus. Vermutlich ist das beim EPSON SD 580 ähnlich. Unser Floppy-Controller hingegen verwendet diesen Pin 2 nicht - er liegt auf 0 Volt.

Was also liegt näher, als das auf dem Floppy-Controller vorhandene Signal mit dem dieser zwischen dem ND- und HD-Modus umgeschaltet wird irgendwie auf Pin 2 des Floppy-Datenkabels zu legen, und so den manüllen Umschalter für die Laufwerke überflüssig zu machen.

Und genau das war der Inhalt von Michaels Artikel:

Auch auf dem Floppy-Controller hat Memotech einen kleinen dSM-Anfall wirken lassen: Der Stecker für das 34-polige Floppy-Daten-Flachbandkabel ist verkehrtherum durchnummeriert. Pin 2 des Kabels heißt bei Memotech 33. In der nun folgenden Bastelanleitung beziehe ich mich auf Memotech's Notation, d.h. Ihr müßt Euch direkt an der Platine, und nicht am Kabel orientieren.

Zuerst muß (nach dem Ausbau der Floppy-Controller-Platine antürlich) die Masse-Verbindung vom Pin 33 des 34-poligen Floppy-Anschlusses auf der Platinen-Unterseite abgetrennt werden. Da der kleine Transistor, der neben dem Pin 33 sitzt so seinen Masse-Anschluß verliert, muß dieser Masse-Anschluß neu gelegt werden, und zwar natürlich so, daß Pin 33 nicht wieder an Masse kommt. Bitte überzeuge Dich mit einem Meßgerät davon, daß Pin 33 keinen Masse-Kontakt mehr hat.

Jetzt muß lediglich der o.g. Pin 33 an Pin 12 des IC 5A (74 LS 273) auf der Floppy-Controller-Platine angeschlossen werden. HALTSTOPP! Falls Du (jemals) ein TEAC FD 135 HFN in Deine FDX einsetzen willst, mußt Du etwas mehr tun: Du mußt die Leitung von Pin 12/IC 5A an Pin 33/Floppy-Anschluß durch einen Inverter führen, d.h.:

Pin 12/IC 5A --> Inverter (z.B. 74 LS 04) --> Pin 33/Floppy-Anschluß.

(Den 74LS04 habe ich huckepack auf den 74LS273 gelötet.)

H a r d w a r e: Laufwerke

Wenn das geschafft ist, wird noch der evtl. noch am Laufwerk vorhandene Schalter entfernt, und die LG, HG, I, II, IS - Brücken wie folgt gesteckt (mit 'Inverter' ist der o.g. Inverter zwischen IC 5A und Floppy-Anschluß gemeint!

		ohne Inverter	mit Inverter
'Neues' Signal:		0 Volt = HD 5 Volt = ND	0 Volt = ND 5 Volt = HD
TEAC FD55 GFV:	zu	I, LG	I, HG
	auf	II, HG	II, LG
TEAC FD55 GFR:	zu	IS, LG	IS
	auf	I, II	I, II, LG
TEAC FD135 HFN:		unzulässig	hat diese Jumper nicht

Für das TEAC FD55 GFV hat diese Lösung leider einen Haken, nämlich den, daß jede Umdrehungsgeschwindigkeitsumschaltung, und die kommt am laufenden Band, wenn das Laufwerk im ND-Modus unter RAM4 läuft (da immer im HD-Modus gesteppt wird), dazu führt, daß der Kopf klappern könnte. Abhilfe: Jumper U2 sollte stecken, U1 nicht. Am besten sollten HL und IU auch offen sein. Dieser Tip ist ohne Gewehr, da ich es nicht testen konnte.

Strapse für TEAC GFV/GFR/HFN-Betrieb und automatischer Umschaltung mit Inverter

Alle: einer von D0 (B:), D1 (C:), D2 (D:) oder D4 (E:)

GFV: zu: I, HG, U2, RY, RE, FG
auf: II, LG, HL, U1, IU, ML

GFR: zu: IS, U1, RY, FG
auf: I, II, LG, HS, HL, IU, U0, DC2, E2

HFN: zu: RY, FG
auf: MS, IR (=IL), RY-DC (Quer-Jumper)

Strapse für TEAC GFV/GFR/HFN-Betrieb und manueller Umschaltung

Alle: einer von D0 (B:), D1 (C:), D2 (D:) oder D4 (E:)

GFV: zu: I, U2, RY, RE, FG
auf: II, HL, U1, IU, ML
Schalter: HG <-> LG

GFR: zu: IS, U1, RY, FG
auf: I, II, HS, HL, IU, U0, DC2, E2
Schalter: LG

HFN: zu: RY, FG
auf: MS, IR (=IL), RY-DC (Quer-Jumper)
Schalter: Pin 2 Flachbandanschluß (= Pin 33 gem. Memotech)

H a r d w a r e: Interrupts

Noch ein Kukucksei von Memotech: (Herbert zur Nedden, 2000)

Auf der RS232-Karte sind die Leitungen INT, BUSREQ und WAIT - so wie es sich gehört - über Dioden auf den Z80-CPU-Bus geführt worden. Das hat den einfachen Grund, daß so sichergestellt wird, daß von dort aus diese Leitungen auf 0 Volt gezogen, d.h. aktiviert werden können, jedoch ein Pegel von 5 Volt nicht stört. Wären diese Dioden nicht da, so würden die Bustreiber auf der RS232-Karte diese Leitungen normalerweise auf 5 Volt hochzeihen, und z.B. ein Interrupt-Request des CTC, der dafür die INT-Leitung auf 0 Volt zieht, hätte Probleme - er müßte gegen diese 5 Volt ansteuern. Da er sehr dicht an der CPU sitzt, würde das möglicherweise funktionieren. Der DART hingegen hätte kaum Chancen, da er neben dem Bustreiber sitzt, und ein Bustreiber mehr Power hat!

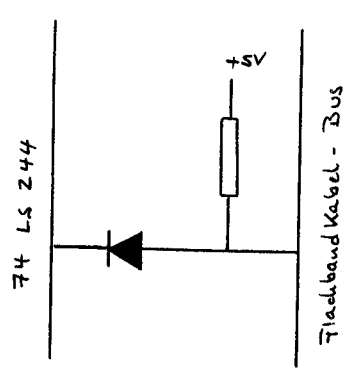
Übrigens sind aus diesen Gründen die INT-Ausgänge der Z80-Bausteine auch Tristate-Ausgänge, d.h. sie liefern im inaktiven Zustand nichts!

So weit so gut, wären da nicht die Folgen von Memotech's Dioden-Geiz in der FDX im Weg. Die Leitungen INT, BUSREQ und WAIT, die aus der FDX kommen, werden fest auf den FDX-MTX-Bus (60-poliges Flachbandkabel) gelegt, d.h. daß außer der FDX hier eigentlich niemand etwas zu melden haben darf - so Memotech! Wie nun, wenn auf den ECB-Bus eine Karte mit WAIT (weil zu langsam) oder INT (weil sie es eilig haben darf) operieren will ? Geht nicht! FDX regiert den Bus schon, und wenn eine ECB-Karte dagegen anstinken will ....

Abhilfe schafft natürlich auch hier die Dioden-Lösung. Dazu müssen drei Widerstände á je 1 kiloOhm und drei Dioden spendiert werden. 0-8-15-Dioden tun diesen Zweck ohne weiteres, Leisutngsdioden sind nicht sinnvoll! (1 z.B. 1N4148)

Die Leitungen WAIT, BUSREQ und INT, die an dem Flachbandkabel-Bus auf den Pin's 50, 51 und 53 liegen sind an dem LS 244 angeschlossen, der ganz am Rande der FDX-Interface-Platine sitzt, d.h. dorten, wo auch drei Widerstände nebeneinander ihr berechtigtes Dasein fristen.

Die 3 Verbindungen LS244 <-> Flachbandkabel-Bus(50, 51, 53) müssen aufgetrennt, und jeweils mit den Dioden wiederhergestellt werden. Dabei gehören die Kathoden - oder sind das die Anoden - naja, auf jeden Fall die Querbalken-Seite der Dioden, an das LS244-seitigen Ende der aufgetrennten Leitungen. Diese drei zu trennenden und diodisch zu überbrückenden Leitungen sind die drei, die auf der Platinenoberseite unter dem LS244 hindurch und neben den Widerständen verlaufen - ich glaube die entsprechenden Pin's des LS244 sind 12, 14, 16. Außerdem müssen die drei Widerstände jeweils zwischen einen der drei o.g. Pins 50, 51 und 53 des Flachbandkabel-Busses und 5 Volt gelötet werden.



H a r d w a r e: Interrupts

ECB-Bus und die Interrupts & Daisy-Chain (Herbert zur Nedden, 2000)

Die Interrupt Daisy-Chain:

Wie ich im Info 27 ausgeführt habe sollten Z80-Bausteine, die Interrupts erzeugen dürfen, in der sogenannten Interrupt Daisy-Chain (Chain = Kette) hintereinandergeschaltet werden, um so eine Prioritäten-Reihenfolge festzulegen. Dabei wird jeweils der IEI-Pin (Interrupt Enable In) des Bausteins mit höherer Priorität mit dem IEO-Pin (Interrupt Enable Out) des nachfolgenden Bausteins verbunden; der IEI-Pin des Bausteins höchster Interrupt-Priorität muß an 5 Volt gelegt werden. Über die Daisy-Chain teilen die Bausteine höherer Priorität denen niedrigerer Priorität mit, ob diese interrupten dürfen, warten müssen bis sie an der Reihe sind, oder ihre Interrupt-Bearbeitung von einem Baustein höherer Priorität interruptet (und damit vorübergehend zurückgestellt) wurde:

Ein Baustein höherer Proirität kann dann zwar die Interrupt-Bearbeitung eines Bausteins niedrigerer Priorität interrupten (unterbrechen), aber nicht umgekehrt - genau so, wie es sein sollte.

Nun zu den Hardware-Voraussetzungen auf dem ECB-Bus:

Damit ECB-Karten auf diese Daisy-Chain-Signale zugreifen können, gibt es auf dem ECB-Bus die beiden Signale IEI und IEO auf den Kontakten 11c bzw. 16c.

Auf einer ECB-Karte, die Interrupts unter Verwendung der Daisy-Chain auslösen soll, muß der IEI-Anschluß der Karte mit dem IEI-Pin des Interrupts auslösenden Bausteins, der auf der Karte die höchste Interrupt-Priorität hat, verbunden werden, und dessen IEO-Pin mit dem IEI-Pin des nachfolgenden solchen Bausteines, u.s.w., und der IEO-Pin des letzten dieser Bausteine auf der Karte mit dem IEO-Anschluß der Karte.

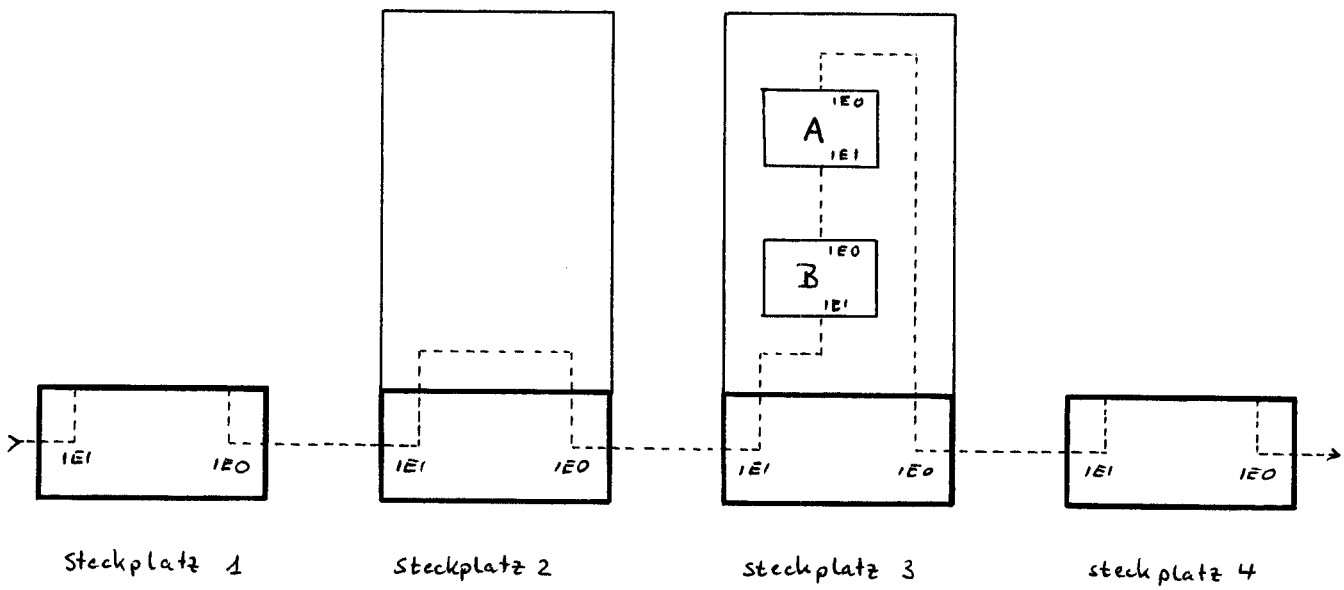
Eine ECB-Karte, die keine Interrupts auslöst, oder, aus welchen Gründen auch immer, unter Mißachtung der Daisy-Chain Interrupts auslösen soll (weil auf ihr nicht-Daisy-Chain-taugliche Bausteine sitzen) muß ihren IEI-Anschluß mit ihrem IEO-Anschluß verbinden, damit sie die Daisy-Chain nicht unterbricht.

Damit nun die Daisy-Chain auch von Karte zu Karte weiterläuft, muß der der IEO-Anschluß der ersten Karte mit dem IEI-Anschluß der zweiten Karte verbunden werden, der IEO-Anschluß der zweiten mit dem IEI-Anschluß der dritten Karte, u.s.w. D.h. die erste Karte hat die höchste Interrupt-Priorität auf dem Bus. Falls ein Baustein auf der 1. Karte interruptet, teilt er dieses ja an seinem IEO-Pin seiner Nachwelt, und damit über den IEO-Anschluß der Karte direkt der zweiten Karte, und damit auch den dahinterliegenden Karten mit.

Daher müssen bei einem für Interrupts ausgelegten ECB-Bus die IEO-Anschlüsse der einzelnen Steckplätze jeweils mit dem IEI-Anschluß des nächsten Steckplatzes verbunden sein. Der IEI-Anschluß der ersten Karte und der IEO-Anschluß der letzten Karte des ECB-Busses hingegen werden am Ende des Busses herausgeführt.

Werden mehrere ECB-Busse hintereinandergeschaltet, muß dann der IEO-Anschluß am Ende des ersten Busses mit dem IEI-Anschluß des folgenden Busses verbunden.

H a r d w a r e: Interrupts



In diesem Bild hat der Steckplatz 1 die höchste, und der Steckplatz 4 die niedrigste Interrupt-Priorität. Auf der Karte in Steckplatz 2 ist die Daisy-Chain direkt durchgeschleift (wie bei den meisten ECB-Karten). Auf der Karte im Steckplatz 3 sind zwei Bausteine A und B, die auf der Daisy-Chain sitzen, wobei B höhere Priorität als A hat.

Für die Hardware-Realisierung der Interrupt Daisy-Chain auf dem ECB-Bus ergeben sich folgende zwei Notwendigkeiten:

1. Die CPU-Platine muß in der Daisy-Chain die höchste Priorität haben, d.h. in der Regel entweder im ersten Stekplatz des ECB-Busses stecken, oder (wie beim MTX) vor dem ersen Steckplatz angeschlossen werden.

2. Der ECB-Bus sollte (für Daisy-Chain Interrupt-Karten muß) durchgehend vom ersten Steckplatz an mit Karten bestückt werden, da ein leerer Steckplatz die Daisy-Chain unterbricht. (Falls es erforderlich ist, einen Steckplatz frei zu lassen, kann auch ein ECB-Stecker, bei dem lediglich der IEI- mit dem IEO-Anschluß verkabelt ist, in den leeren Steckplatz gesteckt werden.)

Eigentlich ist diese ganze Sache wohl recht naheliegend, denn wie sollte sonst die Daisy-Chain aufgebaut werden ? Gute (und meist teure) ECB-Buskarten sind auch so ausgelegt: Die IEI- und IEO-Anschlüsse sind richtig von Steckplatz zu Steckplatz verdrahtet, d.h. die Daisy-Chain ist sauber 'durchgeschleift' (wie auf dem obigen 'Bildle').

Leider werden häufig ECB-Buskarten unter Mißachtung dieser Feinheit hergestellt, da das die Herstellung aufwendiger macht, und es auch andere Bus-Systeme gibt, die mit den gleichen Steckern arbeiten, aber diese IEI-IEO-Verdrahtung nicht mögen. Bei denen sind schlicht und ergreifend alle IEI-Anschlüsse miteinander verbunden - die IEO-Anschlüsse ebenfalls.

H a r d w a r e: Interrupts

Das hat zwei Folgen:

1. Alle Karten erhalten das selbe IEI-Signal, d.h. nix Daisy-Chain.

2. Da alle IEO-Anschlüsse verbunden sind, und die meisten ECB-Karten ihren IEI-Anschluß mit ihrem IEO-Anschluß verbinden, ist ein Kurzschluß für die Karten gegeben, die zwischen diesen Anschlüssen andere Bauteile haben.

Was tun sprach Zeus ?

Man nehme ein scharfes Messer (zum Auftrennen von Leiterbahnen), einen feinen Lötkolben, feinen Elektronik-Lötzinn, etwas isolierten Draht (keinen KuperLack-Draht!), eine Zange (zum Zerschneiden und Abisolieren des Drahtes), gutes Licht, Z-E-I-T, und ggf. Mut!

Mit diesen Hilfsmitteln werden die Leiterbahnen, die die IEI- und IEO-Anschlüsse verbinden VORSICHTIG aufgetrennt, und anschließend von Steckplatz 1 an (das ist der, in dem der ECB-Adapter steckt) bis zum vorletzten (im letzten steckt die Terminierung) der IEO-Anschluß mit den IEI-Anschluß des Folgesteckplatzes verbunden.

BusAcknowledge Daisy-Chain:

Übrigens gibt es noch eine zweite Daisy-Chain bei Z80-Systemen, die BusAcknowledge Daisy-Chain, die die gleiche Mimik für Busanforderungen, d.h. DMA-Zugriffe bereitstellt. Sie arbeitet mit BAI, BAO statt IEI, IEO. Diese wird jedoch beim Memotech in keinster Weise verwendet. Der DMA auf dem Floppy-Controller schert sich einen Feuchten um solche Dinge.

Wer auf dem ECB-Bus einen DMA-Controller unterbringen möchte sollte mal über diese Sache nachdenken. Falls der DMA-Controller nicht im Hintergrund arbeiten soll, was meines Erachtens auch eine recht komplexe Angelegenheit werden würde, zumal er eh die Diskettenzugriffe nicht stören darf, da diese so etwas garnicht abkönnen, ist die BusAcknowledge Daisy-Chain auch nicht erforderlich.
Falls ein DMA-Controller erkennen soll, ob gerade Diskettenzugriffe laufen, muß er das BUSREQ-Signal auf dem FDX-Bus abfragen - das BUSREQ-Signal auf dem Flachbandkabel ist dafür nicht geeignet.

Soll also eine BusAcknowledge Daisy-Chain aufgebaut werden, sollte das invertierte BUSREQ-Signal des FDX-Busses als BAI für den höchst-priorisierten DMA verwendet werden, damit so der Floppy-Controller Vorrang hat. Ob das allerdings richtig klappt ist nicht sicher.

Theorie versus Praxis:

Holger Göbel hat auf seinem so wie hier beschrieben vorbereiteten MTX/FDX/ECB-System eine CTC/SIO/PIO/PIO-Karte stecken, jedoch noch Probleme auftreten! Ob für die recht lange Daisy-Chain eine 'Look Ahead'-Mimik her muß, da die Durchlaufzeiten zu lang sind? Wenn er in der Interrupt-Routine Zeichen DIREKT (d.h. ohne BIOS ...) auf den Drucker ausgibt, klappt's - will er via BIOS ein Zeichen auf den Bildschirm ausgeben hapert's noch.

Wir bleiben jedenfalls am Ball, und wenn's klappt bist Du er erste, der's im Info liest - so die Post Dir wohlgesonnen ist, und Dein Guthaben bei mir ...

S o f t w a r e: Interrupts

Programmierung von Interrupt-Routinen für Z80-Bausteine im Interrupt Modus 2

(Herbert zur Nedden, 2000)

Auf die Interrupt-Modi 0 und 1 gehe ich nicht ein, da sie wesentlich inflexibler sind, RAM 4 Modus 2 verlangt (da verwendet), und auf dem MTX nicht vernünftig einsetzbar sind, und die IM0- und IM1-Interrupt-Routinen an bestimmten Adressen zwischen 0000h und 0100h beginnen müssen, was sich mit unserem Paging nicht vernünftig realisieren läßt.

Übrigens, wer unter RAM4 die Interrupts sperrt hat eine tote Tastatur! Daher hat Bernd in die meisten RAM4- und System-Funktionen ein EI eingebaut, d.h. ein DI ist meist nur von kurzer Dauer - oder alles hängt!

Im Interrupt-Modus 2 stellt der Baustein, der den Interrupt auslöst das untere Byte einer Adresse auf den Datenbus, wo es sich die Z80-CPU auch holt. Das obere Byte dieser Adresse entnimmt die Z80-CPU aus ihrem I-Register (das I steht für Interrupt). An der so zusammengesetzten Adresse muß dann die Adresse der Interrupt-Routine stehen, die die Z80-CPU mit einem CALL anspringt. Zusätzlich **sperrt** die Z80-CPU erst einmal weitere Interrupts!

In lachser Z80-Notation lautet die Antwort der Z80-CPU auf einen IM2-Interrupt:

```
   DI
   LD   L,Daten-Bus
   LD   H,I-Regiser
   CALL (HL)
```

(natürlich verwendet die Z80-CPU hierbei nicht das HL-Register!)

Das Byte, welches der Interrupt-Baustein auf den Datenbus stellt nenne ich
'untere Interrupt-Routinen-Adreß-Zeiger-Hälfte' (uIRAZH),
und den Inhalt des I-Registers
obere IRAZH (oIRAZH),
da beide zusammengenommen einen Zeiger (Pointer) auf die Adresse der Interrupt-Routine (IRAZ) bilden: An der Adresse
IRAZ = oIRAZH * 256 + uIRAZH
steht die Adresse der Interrupt-Routine.

Damit sieht die o.g. Antwort der Z80-CPU auf einen IM2-Interrupt wie folgt aus:

```
   DI
   LD   L,uIRAZH
   LD   H,oIRAZH
   CALL (HL)       ; HL = IRAZ
```

Die Interrupt-Routine muß mit einem **RETI** (Return from Interrupt) beendet werden, sollte alle verwendeten Register tunlichst retten, und, falls nicht unerwünscht, die Interrupts mittels EI wieder aktivieren. Bei kurzen Interrupt-Routinen ist es zu empfehlen, diese mit den Befehlen

```
   EI
   RETI
```

zu beenden; bei etwas längeren Interrupt-Routinen sollte der EI schon früher erfolgen.

S o f t w a r e: Interrupts

Beispiel des Interrupt-Handlings:

Was passiert eigentlich, wenn ein Interrupt druch einen Baustein ausgelöst wird, das Ganze 'mit Daisy-Chain läuft', und

- im I-Register der Z80-CPU 0F0h (= oIRAZH),

- im RAM unter anderem folgendes steht:

Adresse	Inhalt
0F000h	0E500h
0F020h	0E400h

- und A und B Z80-Bausteine auf der Daisy-Chain sind, die so initialisiert sind, daß sie Interrupts erzeugen dürfen, wobei B höhere Priorität als A hat ?

Nun kann z.B. folgendes passieren:

- A interruptet, und stellt als uIRAZH 00h auf den Datenbus (das er im Falle eines Interrupts die uIRAZH 00h liefern soll, wurde bei der Programmierung von A angegeben) und signalisiert an seinem IEO-Pin, daß er interruptet hat, und unterbindet somit Interrupts von Bausteinen mit niedrigerer Priorität (Daisy-Chain).

- Die Z80-CPU setzt diese uIRAZH 00h mit dem Inhalt ihres I-Registers (oIRAZH) zusammen, und erhält den IRAZ 0F000h. Im RAM steht dort die Adresse der zugehörigen Interrupt-Routine: 0E500h; die Z80-CPU führt daher einen CALL 0E500h aus.

- A beobachtet den Bus, und wartet darauf, daß die Z80-CPU den Befehl RETI ausführt, d.h. seine Interrupt-Routine beendet wurde.

- Die Z80-CPU beginnt mit der Abarbeitung der Interrupt-Routine von A. Diese Interrupt-Routine enthält zu Beginn den Befehl EI, um die automatische Interrupt-Sperre aufzuheben.

- B interruptet (B hat ja höhere Priorität als A), und stellt als uIRAZH 20h auf den Datenbus und signalisiert an seinem IEO-Pin, daß er interruptet hat.

- Damit stellt unter anderem A an seinem IEI-Pin fest, daß seine Interrupt-Anforderung unterbrochen wurde (Daisy-Chain!), und wartet jetzt darauf, daß der Interrupt höherer Priorität bearbeitet wurde, d.h. daß ihm an seinem IEI-Pin wieder 'Interrupt-Freigabe' signalisiert wird.

- Die Z80-CPU setzt die uIRAZH 02h mit dem Inhalt ihres I-Registers zusammen, und erhält den IRAZ 0F020h. Im RAM steht dort die Adresse der angeforderten Interrupt-Routine: 0E400h; die Z80-CPU führt also einen CALL 0E400h aus.

- B beobachtet nun den Bus, und wartet darauf, daß die Z80-CPU den Befehl RETI ausführt, d.h. seine Interrupt-Routine beendet wurde.

- Die Z80-CPU beginnt mit der Abarbeitung der Interrupt-Routine von B.

S o f t w a r e: Interrupts

- Die Interrupt-Routine von B wird mit den Befehlen EI und RETI beendet.

- B (dessen Interrupt damit bearbeitet ist) erkennt dies, und gibt damit an seinem IEO-Pin die Interrupts für Bausteine mit niedrigerer Priorität wieder frei (Daisy-Chain).

- Die (durch den Interrupt von B) unterbrochene Interrupt-Routine von A wird weiter durchlaufen.

- A stellt an seinem IEI-Pin fest, daß seine Interrupt-Anforderung wieder in Arbeit ist, und wartet jetzt wieder darauf, daß die Z80-CPU einen RETI ausführt.

- Die Interrupt-Routine von A wird nun mit dem Befehl RETI beendet. (Der EI-Befehl ist nicht erforderlich, da dieser schon zu Beginn dieser Interrupt-Routine abgesetzt wurde.)

- A (dessen Interrupt damit nun auch bearbeitet ist) erkennt dies, und gibt damit an seinem IEO-Pin die Interrupts für Bausteine mit niedrigerer Priorität wieder frei (Daisy-Chain).

- Die Z80-CPU fährt mit der unterbrochenen Arbeit fort.

Was zu beachten ist:

Interrupt-Routinen sollten kurz sein, alle verwendeten Register retten, in der Regel den Z80-Befehl EI enthalten (meistens als vorletzten Befehl), um die automatische Interrupt-Sperre wieder aufzuheben, und MÜSSEN mit dem Z80-Befehl RETI (statt RET) beendet werden, da die Z80-Bausteine während ihrer Interrupt-Bearbeitung den Bus beobachten, und erkennen, wenn die Z80-CPU den Befehl RETI ausführt, und damit die Beendigung ihrer Interrupt-Bearbeitung automatisch erkennen. Bedenkt auch, daß jeder PUSH und CALL zwei Bytes Platz auf dem Stack benötigt.

Die Adressen aller Interrupt-Routinen müssen im RAM abgelegt werden, wobei die Adressen, an denen die Adressen der Interrupt-Routinen stehen gerade sein (d.h. Bit 0 der uIRAZH ist null) und das selbe obere Byte (oIRAZH) haben müssen. Das Bit 0 der uIRAZH, und damit auch Bit 0 der IRAZ null ist, ist keine Einschränkung, da die Adressen der Interrupt-Routinen immer zwei Bytes belegen.

Wird ein Z80-Baustein für Interrupts vorbereitet, muß die Interrupt-Routine im RAM bereitgestellt, und deren Adresse an einer geeigneten Stelle (IRAZ) abgelegt werden. Falls schon Interrupt-Routinen-Adressen im RAM stehen, und demzufolge auch im I-Register die entsprechende oIRAZH steht, muß natürlich auch die neue IRAZ diese oIRAZH haben; anderenfalls (d.h. I-Register ist noch nicht initialisiert worden) muß die oIRAZH in das I-Register gestellt werden. Anschließend muß dem Z80-Baustein seine uIRAZH mitgeteilt, und sein 'Interrupt-Enable-Flag' gesetzt werden.

Da alle IRAZ die selbe oIRAZH haben, also in einem 256-Bytes langen RAM-Bereich liegen, werden die Adressen aller Interrupt-Routinen normalerweise in einer zusammenhängenden Tabelle abgelegt, die bis zu 128 Einträge enthalten kann
(1 Eintrag = 1 Adresse = 2 Bytes).

S o f t w a r e: Interrupts

Die meisten Z80-Bausteine können Interrupts aus verschiedenen Gründen liefern. Sie ersparen dem Anwender die Notwendigkeit, in der Interrupt-Routine einen Status-Wert vom Baustein einzulesen, um die Ursache für den Interrupt zu erfahren, indem sie nicht nur eine uIRAZH, sondern je nach Ursache verschiedene uIRAZH'n liefern können. Daher müssen je Baustein häufig mehrere Interrupt-Routinen bereitgestellt werden, und entsprechend auch mehrere Interrupt-Routinen-Adressen im RAM abgelegt werden.

Bei der Programmierung solcher Bausteine muß nicht für jede mögliche Interrupt-Ursache eine uIRAZH übergeben werden, da die Bausteine in der Regel die Bits 1-? der uIRAZH modifizieren, und so verschiedene uIRAZH'n, und damit mehrere aufeinanderfolgende IRAZ erzeugen. (Bit 0 der uIRAZH wird nie modifiziert, da es immer null ist).

Der Z80-CTC kann aus 4 Gründen einen Interrupt erzeugen (je nach dem, welcher seiner Kanäle 0 bis 3 der Auslöser ist), und überschreibt die Bits 1 und 2 seiner uIRAZH mit einer der Bitkombinationen 00, 01, 10 bzw. 11. Wurde dem CTC als Basis-uIRAZH der Wert X (Bit 0-2 von X sind null) übergeben, so liefert er im Falle eines Interrupts je nach auslösendem Kanal als uIRAZH einen der Werte X, X+2, X+4 oder X+6.
Demzufolge müssen die vier Interrupt-Routinen-Adressen direkt hintereinander im RAM ba einer Adresse, deren Bits 0-2 null sind, stehen.

Der Z80-DART benötigt sogar 8 Interrupt-Routinen, und modifiziert hierfür Bits 1-3 der uIRAZH, weshalb die zugehörigen acht Interrupt-Routinen-Adressen hintereinander im RAM, beginnend an einer Adresse, deren Bits 0-3 null sind, stehen müssen.

Bei gebankten Systemen (wie unserem) müssen die Interrupt-Routinen im nicht-gebankten Bereich liegen (zumindest beginnen und enden, falls sie die Interrupts gesperrt lassen, also insbesondere keine BDOS/BIOS/RAM4-Aufrufe absetzen), sowie ihre Adressen im nicht-gebankten Bereich stehen, da ein Interrupt nicht automatisch eine bestimmte Speicherbank anwählt. Falls die Interrupt-Routine die Bank umschaltet, muß sie vor ihrem RETI die Umschaltung wieder rückgängig machen. (Beim MTX ist der nicht-gebankte Bereich oberhalb von 0C000h.)

S o f t w a r e: Interrupts

```
Anmerkung zu RAM4: oIRAZH             = 0F0h (steht im I-Register)
                   DART-Basis-uIRAZH =  80h
                   CTC-Basis-uIRAZH  =  90h
d.h.
                   8 DART-IRAZ'n = 0F080h, 0F082h, ..., 0F08Eh
                   4 CTC-IRAZ'n  = 0F090h, 0F092h, ..., 0F096h

Etwas Z80-Code like:

; Interrupt-Routinen-Adressen, wie in RAM4 vorgesehen
;
          ORG  0F080h    ; Bit 0-3 der Adresse sind null
;
 DARTINT: DW   DARTINT0  ; Kanal A, Sendepuffer leer
          DW   DARTINT1  ; Kanal A, Externer Statuswechesel
          DW   DARTINT2  ; Kanal A, Zeichen im Empfangspuffer
          DW   DARTINT3  ; Kanal A, besonderer Empfangsstatus
          DW   DARTINT4  ; Kanal B, Sendepuffer leer
          DW   DARTINT5  ; Kanal B, Externer Statuswechesel
          DW   DARTINT6  ; Kanal B, Zeichen im Empfangspuffer
          DW   DARTINT7  ; Kanal B, besonderer Empfangsstatus
 CTCINT:  DW   CTCINT0   ; Kanal 0
          DW   CTCINT1   ; Kanal 1
          DW   CTCINT2   ; Kanal 2
          DW   CTCINT3   ; Kanal 3

DART, besonderer Empfangsstatus =
  Parity-Fehler, Empfangspuffer-Überlauf, CRC-Fehler, SDLC.

Die RAM4-Realität sieht allerdings etwas anders aus:
  Es sind nur die Kanäle 0 und 3 des CTC für Interrupts initialisiert worden!
  CTC-Kanal 1 und 2 liefern die DART-Baudraten-Takte
  Der DART ist nicht für Interrupts initialisiert.

; RAM4-Realität:
          ORG  0F080h
 DARTINT: DW   INTRET
          DW   INTRET
          DW   INTRET
          DW   INTRET
          DW   INTRET
          DW   INTRET
          DW   INTRET
          DW   INTRET
 CTCINT:  DW   CTCINT0   ; RAM4-Uhreninterrupt
          DW   INTRET    ; kein Interrupt: RS232-A-Takt
          DW   INTRET    ; kein Interrupt: RS232-B-Takt
          DW   CTCINT3   ; RAM4-Tastatur-Interrupt

; und

INTRET:   EI
          RETI
```

A s s e m b l e r: Cursortastenabfrage

(Herbert zur Nedden, 2000)

Gelegentlich steht man vor dem Problem, daß sich in einem Programm die Verwendung der Cursortasten nur so aufdrängt. Da überall anders belegt sein können, wird das schon interessant.

Die Belegung der Tasten variiert nicht nur von Land zu Land, sondern (insbesondere der Tasten des 10-er Blocks) auch noch von ... über ... zu:

- Memotech-(FDX)BASIC,
- Original Memotech CP/M,
- Memotech CP/M mit kleinen Patches,
- Memotech CP/M mit einem von Bernd Preusings Tastaturtreibern,
- RAM 4.x (als Funktionstasten),
- und was noch immer ...

Memotech-(FDX)BASIC lasse ich im Folgenden unter den Tisch fallen. Im Prinzip ist die Chose hier ähnlich, nur daß die Tastaturtabelle und der Einsprung in die Tastaturabfrage an anderer Stelle stehen.
(Ob sich jedoch hier doch noch einiges anders verhält, ????)

Die 'normale' Tastatur-Abfrage geht wie folgt vor:

1. Anforderung des Programmes an das BDOS.
2. Das BDOS leitet die Anforderung an das BIOS weiter.
3. Das BIOS macht einen CALL auf Adresse OFFD0h, um das nächste Zeichen von der Tastatur zu holen.
4. Zurück kommt der ASCII-Code, den das BIOS an den Aufrufer weitergibt.

   Dieser ASCII-Code der gedrückten Taste wird aus der sog. Tastaturtabelle entnommen. Wird zusätzlich die Ctrl-Taste gedrückt, wird der Code aus der Tastaturtabelle vorher noch entsprechend modifiziert, d.h. in den entsprechenden Ctrl-Code umgewandelt. Ist eine der Shift-Tasten gedrückt oder AlphaLock gesetzt, wird das insofern ausgewertet, als die Tastaturtabelle für jede Taste zwei Einträge enthält, nämlich den für ohne, und den für mit Shift.

   Wird Ctrl-M gedrückt, liefert CALL OFFD0h den Wert 0Dh, also tatsächlich Ctrl-M, und nicht etwa erst irgendeinen Code für Ctrl, und anschließend noch ein M.

Ist im System ein Funktionstasten-Decoder installiert, so ist dieser Bestandteil des BIOS:

Ein CALL OFFD0h liefert auch in diesem Fall den ASCII-Code der gedrückten Taste, der in der Tastaturtabelle steht, ggf. wegen Ctrl entsprechend modifiziert.

Nur wenn der BIOS- oder BDOS-Einsprung verwendet wird, werden die Funktionstasten dekodiert.

Wird F1 gedrückt, liefert ein CALL OFFD0h den Wert 80h (falls die Tastaturtabelle nicht mißhandelt wurde), während ein BIOS- oder BDOS-Aufruf bei aktivem Funktionstastendecoder statt der 80h den zu F1 gehörenden String zeichenweise liefert.

A s s e m b l e r: Cursortastenabfrage

Erfreulicherweise ist unter allen Systemen sichergestellt, daß die Tastaturtabelle immer an der selben Stelle im Speicher liegt, oder ?? Naja, zumindest in Deutschland und unter RAM 4.x beginnt sie an der Adresse **0F199h**.
Ich glaube, in Belgien liegt die Tabelle beim Original-MTX um ein Byte verschoben - was jedoch RAM4 in den Griff bekommt, indem es diese ggf. in die richtige Position bringt. Steht in der Speicherstelle 0F19Ch der Wert 0A0h, ist alles o.k., sonst ist die Tabelle vermutlich eine Speicherstelle zu weit oben, beginnt also bei 0F19Ah.

In der Tastaturtabelle ist der ASCII-Code abgelegt, den die einzelnen Tasten liefern sollen. Wer nun direkt auf die Tastaturabfrage über den Einsprung bei 0FFD0h zugreift, der auch immer dort steht, erhält stets diesen zur Taste gehörenden ASCII-Code aus der Tastaturtabelle, ggf. modifiziert wegen Ctrl.

Eventuell im System aktive Funktionstastendecoder stören hier nicht, da diese sich in die BIOS-Sprungtabelle einschleifen, jedoch den an 0FFD0h stehenden Sprung auf die Tastaturabfrage in seiner Wirkung unverändert lassen.

Will ich wissen, welche Taste gedrückt wurde, hole ich mir den ASCII-Code über CALL 0FFD0h, und vergleiche das Ergebnis mit dem zu der gewünschten Taste gehörenden Eintrag in der Tastaturtabelle. Sind beide Werte gleich, habe ich die Taste (vermutlich) erwischt.

Diese Methode ist zwar nicht 100%-ig, da evtl. verschiedene Tasten den selben ASCII-Code liefern, insbesondere unter Nicht-RAM4-Systemen einige Tasten vom Cursor-Block und Ctrl-Codes. Z.B. leifert der Pfeil-nach-unten, die LineFeed-Taste und Ctrl-J im Original-Memotech-Tastaturtreiber den selben ASCII-Code, nämlich 0Ah.

Daher ist dieses Verfahren eigentlich nur dann sinnvoll anwendbar, wenn solche Kollisionen nicht stören.

Sicherlich kann man die Tastaturtabelle erst einmal so patchen, daß keine Kollisionen auftreten können, aber das ist eigentlich ein rabiates Verfahren, und es muß beim Beenden des Programmes der Urzustand wieder hergestellt werden. (Unter RAM 4.x tritt dieses Problem nicht auf, da hier die Tasten des 10-er Blocks eigene ASCII-Codes liefern, weil auch diese Tasten Funktionstasten sind.)

Als andere Möglichkeit bietet es sich an, die Tastatur direkt mit Port-Befehlen abzufragen, d.h. einen Keyboard-Scan durchzuführen. Aber wer möchte schon solche Wege gehen ? (Unter RAM4 oder beim Einsatz eines Interrupt-Tastaturtreibers sollte man hiervon tunlichst Abstand nehmen, da diese die Tastatur in Interrupt-Zyklen abfragen, also der Tastatur-Interrupt während solcher Abfragen gesperrt werden müßte, damit dieser nicht dazwischenscannt - gefährlich!!)

Meistens geht es jedoch nur darum, die Cursor-Tasten irgendwie zu erfassen, wobei jedoch Ctrl-Codes (und andere evtl. auf dem Zehnerblock liegenden ASCII-Codes) nicht gefragt sind, da die erwarteten Kommandos entweder über die Tasten F1-F8, oder mit normalen Buchstaben-Tasten (A-Z) aktiviert werden.

Klar, daß solche Programme gefoppt werden können, indem man an Stelle der Cursortasten andere Tasten drückt, die den gleichen ASCII-Code liefern (z.B. eine 8 statt Shift-Pfeil-hoch beim Original-Memotech-Tastaturtreiber). Aber was macht's schon ? (Olaf's MONI arbeitet übrigens nach diesem Prinzip, und kann mit den Kollisionen gut leben.)

A s s e m b l e r: Cursortastenabfrage

Als Anwendungsbeispiel mag z.B. ein Auswahl-Menü dienen, bei dem mit den Cursortasten der gewünschte Auswahl-Punkt ermittelt, und dann z.B. durch Drücken der HOME-Taste aktiviert wird.

Folgendes Assembler-Unterprogramm liest ein Zeichen von der Tastatur ein, und fragt ab, ob es sich um eine der Pfeiltasten ohne Shift handelt. Ist eine der Tasten gedrückt, so wird eine entsprechende Routine angesprungen.

```
KbdScan:  call  OFFD0h      ; ASCII-Code im A-Register
          ld    hl,OF1D5h   ; Left = Pfeil links
          cp    (hl)        ; Vergleiche ASCII-Code mit Tabelle
          jp    z,DoLeft    ; Ja, Left ist gedrückt
          ld    hl,OF1D4h   ; Right = Pfeil rechts
          cp    (hl)        ; Vergleiche ASCII-Code mit Tabelle
          jp    z,DoRight   ; Ja, Right ist gedrückt
          ld    hl,OF1D6h   ; Up = Pfeil hoch
          cp    (hl)        ; Vergleiche ASCII-Code mit Tabelle
          jp    z,DoUp      ; Ja, Up ist gedrückt
          ld    hl,OF1D2h   ; Down = Pfeil runter
          cp    (hl)        ; Vergleiche ASCII-Code mit Tabelle
          jp    z,DoDown    ; Ja, Down ist gedrückt
          ret               ; ASCII-Code noch immer im A-Register
                            ; Zero-Flag nicht gesetzt!

DoLeft:   ; Pfeil-links wurde grdrückt
          ....
          xor   a
          ret               ; Zero-Flag gesetzt!
```

DoRight, DoUp, DoDown enstprechend!

Da die Unterprogramm-Zweige DoLeft, DoRight, DoUp und DoDown, die die Verarbeitung für die Pfeil-Tasten vor dem RETurn mittels XOR A das Zero-Flag setzen, kann das aufrufende Programm das Zero-Flag abfragen, um festzustellen, ob eine der Pfeil-Tasten gedrückt wurde, und so die Verarbeitung steuern, z.B. in der Form:

```
Loop:     .....
          call  KbdScan
          jp    z,Loop      ; es war eine Pfeiltaste,
                            ; also weitermachen
          ....
```

Für die Pfeil-Tasten mit/ohne Shift und für HOME sind die zugehörigen Einträge in der Tastaturtabelle:

Taste	Left	Right	Up	Down	HOME
o. Shift	OF1D5h	OF1D4h	OF1D6h	OF1D2h	OF1D3h
m. Shift	OF226h	OF225h	OF227h	OF223h	OF224h

Zu bemerken ist, daß der Eintrag zu einer Taste mit Shift stets 81 Bytes hinter dem Eintrag zu dieser Taste ohne Shift in der Tastaturtabelle steht.
Beachte: 81 = 51h, und OF1D5h + 51h = OF226h.

Im Info 4 hat Bernd Preusing ein disassembliertes Listing des Original-Memotech-Tastaturtreibers veröffentlicht. Aus diesem Info habe ich die Tastaturtabelle in verkürtzter Form (d.h. ohne die ASCII-Codes, die ja gerade der Stein des Anstoßes sind) übernommen:

A s s e m b l e r: Cursortastenabfrage

Deutsche Tastaturtabelle (Herbert zur Nedden, 2000)

Beachtet bitte, daß Tasten wie Shift (ShL, ShR), AlphaLock (AlLo) und Ctrl nicht mit der o.g. Mimik abgefragt werden können, da sie vom Tastaturtreiber (Aufruf CALL OFFD0h) schon umgesetzt werden.

Außerdem liefern einige Tasten mit/ohne Shift ohne RAM 4 die selben Werte (z.B. =, Esc, BS, Ret, LF), während sie unter RAM 4 andere Resultateliefern (/, KLICK-Aufruf, Del, LF, Repeat-Taste).

In der 2. Hälfte der Tabelle (mit Shift) habe ich die Tasten des 10-er-Blocks mit den selben Einträgen wie ohne Shift versehen. Im Original-Memotech-Treiber müßten hier eigentlich 0, 1, ..., 9, ., bzw. RET an den entsprechenden Stellen stehen.

ohne Shift

0F199h	y	ShL	a	AlLo	q	Ctrl	Esc	1
0F1A1h	c	x	d	s	e	w	2	3
0F1A9h	b	v	g	f	t	r	4	5
0F1B1h	m	n	j	h	u	z	6	7
0F1B9h	.	,	l	k	o	i	8	9
0F1C1h	=	-	ä	ö	ü	p	0	ß
0F1C9h	Ins	ShR	Ret	#	LF	+	^	<
0F1D1h	Cls	Down	HOME	Righ	Left	Up	Eol	Page
0F1D9h	F4	F8	F3	F7	F6	F2	F5	F1
0F1E1h	Spac				Del	Tab	BS	Brk

mit Shift

0F1EAh	Y	ShL	A	AlLo	Q	Ctrl	Esc	!
0F1F2h	C	X	D	S	E	W	"	§
0F1FAh	B	V	G	F	T	R	$	%
0F202h	M	N	J	H	U	Z	&	/
0F20Ah	:	;	L	K	O	I	(	)
0F212h	=	_	Ä	Ö	Ü	P	0	?
0F21Ah	Ins	ShR	Ret	'	LF	*	'	>
0F222h	Cls	Down	HOME	Righ	Left	Up	Eol	Page
0F22Ah	F4	F8	F3	F7	F6	F2	F5	F1
0F232h	Spac				Del	Tab	BS	Brk

S o f t w a r e: Korrektur

Fehler in PASCAL, BACKUP und COPY

(Holger Hansen, 3300)

Zuerst PASCAL.PAS:
Dies betrifft nur die von mir erweiterte Version von PASCAL, die den Commandstring weiterreicht an ein SUB, ZEX oder LU.
In Zeile 58 muß es heißen :

```
  COM:=COM+' '+paramstr[i];
 statt
  COM:=paramstr[i]+' '+COM;
```

Dieser Fehler führte dazu, daß die Kommandozeile in umgekehrter Reihenfolge an das folgende Programm weitergegeben wurde.

Nun BACKUP und COPY:
Bei der von mir bis jetzt ausgelieferten Version 1.63 bzw. 1.65 hat sich ein Fehler eingeschlichen. Er betrifft die Erkennung von Dateien die nur einen Directory-Eintrag belegen und genau 80 Records lang sind. Bei den angegebenen Versionen werden solche Dateien leider nicht erkannt. Um dieses zu beheben, müssen im Quelltext folgende Änderungen vorgenommen werden. Die Zeilenangaben sind nicht unbedingt als absolut anzusehen, aber +/- 5 Zeilen dürften sie richtig sein.

Für BACKUP:

Datei BACKUP.PAS

- Zeile 3: Versionsnummer von 1.63(1.65) nach 1.70 ändern

```
 alt:( const   meldung : string[80] = 'BACKUP   Version 1.63 (c) ... )
 bzw ( const   meldung : string[80] = 'BACKUP   Version 1.65 (c) ... )
 neu:( const   meldung : string[80] = 'BACKUP   Version 1.70 (c) ... )
```

- Zeile 96ff: Zwei Zeilen einfügen

```
 alt:  zuser:=0;
 neu:  gus:=0;
 neu:  zus:=0;
 alt:  new(qnamensfeld);
```

- Zeile 117ff: Das unterstrichene einfügen

```
 alt:   val(qu,qus,io);
 neu:   if (io<>0) or (qus<0) or (qus>31) then qus:=0;
 alt:   quser:=qus;
 alt:   write('Ziellaufwerk   (',high,'B',low,'): ');readln(ziel);
```

- Zeile 125ff: Das unterstrichene einfügen

```
 alt:   val(zu,zus,io);
 neu:   if (io<>0) or (zus<0) or (zus>31) then zus:=0;
 alt:   zuser:=zus;
 alt:   write('Dateiname    (',high,'*.*',low,'): ');readln(dateistring);
```

Datei BACKUP1.INC

- Zeile 93ff: Eine Zeile einfügen, eine Meldung einfügen

```
 alt: for sector:=1 to seczahl do
 alt: begin   (* Sektorzaehlung *)
 alt:  readsector;                              (* ein Sektor = 3 Dir Eintraege *)
 alt:  if buffer[0+3*32]<>chr($21) then         (*  4.Eintrag = Zeiteintrag     *)
 alt:   begin
 neu:    driveb:=char(drive+$41);
 neu:    writeln('Fehler Directory Laufwerk ',high,driveb,':',low,' hat
                  keine Zeiteintraege );
 alt:    delay(500);
```

S o f t w a r e: Korrektur / KLICK-PD's

```
  - Zeile 120ff: zusätzliche Bedingung einfügen
   alt: sum[zaehler]:=tage1*86400.+std*3600.+min*60.+sec;
   neu: if (hex(buffer[15+j*32])='80') and (hex(buffer[12+j*32])<>'00') then
           zaehler:=zaehler
   alt: else  zaehler:=zaehler+1;
```

Die Korekturen bei COPY sind einfacher:

Datei COPY.PAS:

-Zeile 3: Versionsnummer von 1.10 nach 1.20 ändern

Datei COPY.INC:

-Zeile 102 : zusätzliche Abbruchbedingung einfügen

```
 alt: namensfeld^[zaehler]:=namensfeld^[zaehler]+buffer[12+j*32]+buffer[15+j*32];
 neu: if (hex(buffer[15+j*32])='80') and (hex(buffer[12+j*32])<>'00') then
            zaehler:=zaehler
 alt: else zaehler:=zaehler+1;
```

analag sind die Änderungen in COPY-2.PAS und COPY-2.INC

Mit TURBO 3.0 übersetzen:
Nur als COM oder CHN File, dabei die Endadresse auf $BFF8 setzen!!
Wer keinen TURBO 3.0 Compiler besitzt, schickt bitte an mich eine Diskette. Ich sende demjenigen dann eine korrigierte BACKUP/COPY Version zu.

KLICK-PD's: Was ist da so drauf ? (Olaf Krumnow, 2050)

Auf vielfachen Wunsch einer einzelnen Person gebe ich hier den Inhalt der KLICK-PD's.

KLICK.001

Nun, da wären zunächst einmal die neuesten Versionen der Z80-Assembler-Bibliotheken KLIXLIB und MTXLIB, wobei es für KLIXLIB jetzt endlich auch eine Dokumentation gibt, nämlich KLIXLIB.HLP. Die Pascal-Bibliothek KLIX/K.INC ist auch noch mal dabei, auch mit HELP-File.

Ferner gibt es zwei alte Utilities in neuen Versionen, nämlich KILL und SIZE, sowie zwei neue Utilities, die sich SETSPOOL und SAVSPOOL nennen und sich mit dem Spooler beschäftigen.

Ein neues KLIX-Programm (INFO.COM) ist eine übersichtliche Zusammenfassung von KLIX-Variablen und -Sprüngen und den Bibliotheken KLIXLIB und KLIX/K.INC. Dann ist da noch eine Pascal-Bibliothek, die sich mit der Zeit unter RAM4 beschäftigt, mitsamt HELP-File. Last, but not least noch ein kleines Programm für Herbert, HERBY.COM, das einen kleinen Trick benutzt, um diese netten flackernden Buchstaben zu zaubern. Erst überlegen, dann in der Source nachgucken, wie's gemacht ist.

Least but not Last sind noch ein paar KLIX-Programme anbei: Ein Taschenrechner, dessen Fenster sogar über den normalen Bildschirm geschoben werden kann - an jede Stelle natürlich, Hardcopy, eine Kalenderanzeige und ein Display mit Portogebühren.

KLICK.002

Zuerst sind die angekündigten HELP-Files um und über KLICK und ähnliches anbei.

Als KLIX-Programme ist eine Tabelle mit den Hex- und Dezimalcodes aller Zeichen in allen drei Modi (Standard, Alternate, Grafic) sowie Erläuterung der Bildschirmcodes, erweiterte Version für RAM4.2 Codes (ASCII3) anbei.

Weiterhin ist eine neue Version von Hardcopy (Vers. 3) mit Pull-Down-Menüs und Edicta-Fähigkeit sowie ein erweiterter Kalender fürs KLICK mit drauf.

S o f t w a r e: KLICK-PD's

Um das Erstellen von KLIX-Programmen zu vereinfachen sind entsprechende Programme und SUBmit-Dateien dabei. Und für SYSLIB-Freunde ist auch die Version 3.6 dieser Standard-CP/M-Assembler-Bibliothek KLICK-tauglich drauf.

Wer wissen will, wie es um seinen KLIX-Heap bestellt ist, kann dies mit KSWEEP, einem NSWEEP-Artigen Programm für KLIX-Heap-Einträge erledigen.

KLICK.003
Auf der neuesten Diskette sind zwei große Themen und einige 'Kleinigkeiten' untergebracht.

Zunächst die 'Kleinigkeiten':

Da ist zunächst einmal alles, was zur Formatumschaltung im laufenden NewWord benötigt wird. Es stammt von Jan Bredereke bzw. basiert auf von ihm geschaffenen Grundlagen. Das Programm läuft (logischerweise) im KLICK und kommt mit NewWord 2.02, 2.16 und 2.17 zurecht. Das Patch-Programm NWPATCH.COM stammt von HzN und führt an NewWord noch einige weitere Patches (zusätzlich zu dem, der für die Formatumschaltung nötig ist) aus, falls gewünscht. Das Programm ist vollständig selbsterklärend.

Eine neue, fehlerbereinigte Version des bereits bekannten Programmes SETSPOOL. Es tauchten mal wieder Schwierigkeiten mit der Zahl $8000 auf, die TURBO-Pascal einfach nicht mag. Außerdem konnte es vorkommen, daß die alte Version den KLICK-Aufruf sperrte.

Ferner die neuste Version des bereits bekannten Programms KLIXINFO, jetzt mit den Informationen zu der Menüoberfläche OKMENU. Für diejenigen, die KLIXINFO noch nicht kennen: Dort werden alle RAM4-Sprünge und -Variablen, sowie alle Routinen aus KLIXLIB (Assembler und Turbo-Pascal) dokumentiert. Das Programm steht natürlich im Heap parat und kann immer dann aufgerufen werden, wenn man es braucht.

Das erste große Thema:

KLIX-Programmierung unter "C". Und dazu auch gleich ein sehr schönes Programm von Jan Bredereke, ein Wecker, der einen zu allen möglichen Terminen aus dem Programmieren reißt. Mit den beigelegten Prozeduren können KLIX-fähige C-Programme geschrieben werden. Aber es ist auch möglich, KLIX-Programme in Assembler zu schreiben, ohne MACRO-80 zu besitzen. Es ist ein Programm-Lader dabei, der mit ZMAC/ZLINK von der C-PD (CLUB.024 glaube ich) zusammenarbeitet.

Das zweite große Thema

letztlich ist OKMENU, das von Herbert schon angekündigte Menü-Unterprogramm. Eine Anwendung befand sich bereits auf der letzten PD, nämlich HARDCOPY 3.00. Diesmal ist alles dabei, was man benötigt, um eigene Menüs zu erstellen, natürlich auch ein ausführliches DOC. Möglich sind fast alle Arten von Menüs, von einfachen Auswahlmenüs über Tastenabfrage (wie in HELP) bis hin zu baumartig strukturierten Pull-Down-Menüs. Dabei können in einem Menübaum natürlich auch die verschiedensten Arten gemischt werden. Natürlich wird alles ausgenutzt, was RAM4 hergibt. Deshalb ist diese Version Voraussetzung für OKMENU. Außerdem wird Heap benötigt, wenn der Bildschirmhintergrund gerettet werden soll. Es sollte also ein etwas besserer Speicherausbau vorhanden sein (was für KLICK-Programme ja ohnehin nötig ist).

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

Umleitung des Tastaturtreibers unter TURBO-PASCAL 3.0 (Olaf Krumnow, 2050)

Neulich hatte ich das Problem, aus einer fertigen (von Herbert geschriebenen) Maskeneingabe heraus auf einen Tastendruck (SHIFT-RETURN) hin eine bestimmte Reaktion auszuführen. Im Normalfall muß die Eingaberoutine so umgeschrieben werden, daß der geforderte Tastendruck extra abgefragt wird. Das wollte ich jedoch vermeiden, um bei eventuellen Änderungen der Maskeneingabe seitens Herbert nicht jedesmal diese Änderungen durchführen zu müssen. Bei der Suche nach einer Lösung stieß ich auf das Kapitel 'Benutzergeschriebene I/O-Treiber' im Pascal-Handbuch (S. 272). Somit war die Lösung relativ einfach. Ich leitete die Tastatureingabe auf eine eigene Routine um, die sich ein Zeichen via BIOS einliest und prüft, ob es das geforderte Zeichen ist. Wenn nein, so wird das eingelesene Zeichen an den Aufrufer zurückgegeben. Anderenfalls kann ich die von mir gewünschte Routine aufrufen.

```
function OwnConIn: char;
const Aktiv: boolean = false; { Flag }
var c: char;
begin
     c := BIOS(2);              { Zeichen ueber das BIOS einlesen }
     if Aktiv or                { eigene Routine laeuft schon }
        (c <> #10) then         { nicht das geforderte Zeichen }
        OwnConIn := c
     else
        begin
        Aktiv := true;          { Flag setzen, um mehrfache Aufrufe zu vermeiden }
        { hier die eigene Routine aufrufen }
        Aktiv := false;         { Flag wieder loeschen }
        OwnConIn := #0;         { Irgendein unmoegliches Zeichen an den
                                  Aufrufer zurueckgeben }
       end
end; { OwnConIn }
```

Im Hauptprogramm muß dann nur noch der neue Treiber eingebunden werden. Dies geschieht mit **ConInPtr := addr(OwnConIn);**
Damit steht der neue Treiber überall im Programm zur Verfügung. Soll er nur an einigen Stellen zur Verfügung stehen, so kann entweder ein zusätzliches Flag eingeführt werden, das dann global bekannt ist und gesetzt wird, sobald der Aufruf des eigenen Treibers gestattet ist. Oder die Treiberzuweisung erfolgt nur dann, wenn der Treiber benutzt werden soll. Anschließend wird wieder die Adresse des alten Treibers gesetzt (vorher natürlich retten!).
Die vom neuen Treiber aufgerufene Routine ist in keinerlei Weise eingeschränkt. Sie kann ihrerseits auch selber die Tastatur abfragen, da ein mehrfacher Aufruf durch Benutzung des (lokalen) Flags unmöglich ist. Bei mir bietet die Routine an, diverse Strings auf eine simulierte Funktionstaste zu legen. In diesem Fall kann statt **OwnConIn := #0** bereits das erste Zeichen der F-Taste zurückgeliefert werden, indem nochmals das BIOS aufgerufen wird (**OwnConIn := BIOS(2)**).

Noch ein anderer Tip im Zusammenhang mit Funktionstasten unter TURBO-Pascal: Wenn ein Text auf einer F-Taste liegt und in einem TURBO-Programm abgerufen werden soll, so fehlt i.A. jedes zweite oder dritte Zeichen. Dies kann vermieden werden, indem die undokumentierte Variable **CBREAK** auf **false** gesetzt wird. Allerdings kann dann die Bildschirmausgabe solange nicht mit ^S angehalten werden. Wer das anschließend wieder möchte, setzt CBREAK wieder auf true.

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