MTX User-Club Deutschland

Info 41
28.02.1991

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

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

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

Verpflichtungen: Einsendung unseres Anmeldeformulars.

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

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

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

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

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

Kontaktadressen:

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

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

Telefon-Sprechzeiten
Bis zu meinem Umzug im März unter (040) 200 87 04 kannst Du es jeden Abend versuchen. Danach Herbert zur Nedden: Do 18 - 22 Uhr, Sa 9 - 14 Uhr

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

Preis für dieses Info: DM 8,80

Wettbewerb (Herbert zur Nedden, 2000)
Siehe letztes Info: Bewertungsfunktion für UISQE.

Clubtreffen (Herbert zur Nedden, 2000)
Termin: 29/30. Juni oder ein Wochenende im July; Ort: Hotel Kückenmühle, 3003 Ronneberg; Übernachtung incl. Frühstück DM 55.- pro Person im Doppelzimmer. Wenn Du Interesse hast, bitte anmelden und ggf. Termineinschränkungen nennen. Meldeschluß: 18. März.

Kontostand (Herbert zur Nedden, 2000)
Eine rote Markierung auf dem Umschlag bedeutet, daß er zu niedrig ist.

Anzeigetexte samt Absender bitte schriftich an Herbert zur Nedden!

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

Herbert zur Nedden, Sonnenau 2, 2000 Hamburg 76, 040 - 2008704:
--> Interessiert Dich einer der von mir angebotenen Posten: Mach ein Angebot!
Meine hier genannten Preise sind nicht unbedingt unumstößlich!
(Neue/geänderte Posten haben einen * statt des - vorne weg)

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

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

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

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

Hauke Ahrensfeld, St.-Georg-Str. 6, 3100 Celle, 05141-23490:
MTX, 192kB, 2x 80Spur-Lw., Hardware-Uhr, erw. 80Zeichen-Karte, neues 80Z-EPROM, im XT-Gehäuse mit externem Netzteil, 'große' Tastatur, Monitor (bernstein), RAm 6.1, NW, SC, dBASE: Alles zusammen FP DM 450,-
Monitor Highscreen TP200 P39, FP DM 80.-, evtl. mit Monitorarm
18x 4164 RAMs: DM 25,-

Liebe Leserin, lieber Leser,

Mein Umzug nach Siek ist nun bald fällig - vermutlich 23./24. März! Meine neue Telefonnummer kenne ich schon: (04107) 99 00.

Die Farbe Pink des Deckblatts des Infos kommt daher, daß das grüne Papier nicht mehr hergestellt wird - Pastellfarben sind in. Die Druckerei hat daher in ihrem Bestand nachgeschaut, welche Farbe sie verwenden kann und landete beim Pink, da die anderen Farben eigentlich zu blaß sind - sie fallen nicht auf.

Info auf Diskette wird es (erst mal) nicht geben! Mittlerweile habe ich noch einige NEINs samt Begründungen erhalten. Einige waren begeistert; gelegentlich änderte sich diese Begeisterung auch, denn jedesmal den Rechner anschmeißen zu müssen ... Außerdem liest sich das Info viel bequemer (und sicherlich unter weniger Protest der besseren Hälfte) im Wohnzimmersessel oder der U-Bahn lesen. Also ist meist eh Ausdrucken angesagt. Und dabei gibt's Probleme mit all den Grafik-Einlagen. Ich hätte zwar auch eine Version des Infos, die mit 0-8-15-Grafik (also den Zeichen +, - und ! auskommt) bereitgestellt, aber dann ist der Ausdruck eher traurig. Mir kommt das insofern entgegen, daß es auch einiges an Zeit kostet, so an die 100 Disketten im Format 03 zu erstellen und einzutüten. Das Papier-Info ist da für mich Pflegeleichter: Ich bringe der Druckerei die Druckvorlage und die fertigen Umschläge - das Eintüten wird mir so abgenommen.

Hartmut Traber meinte übrigens in einem Leserbrief:
(Und die anderen Leichen?, ca. +-100 Mitglieder hat der Club, schreiben tun vielleicht 10, nützliche Programme kommen von vielleicht 5, fühle Dich doch mal endlich angesprochen und teile Deine Aktivitäten mit!) Ich könnte mir vorstellen, daß dann mancher Info-Bezieher die Kiste aus dem Keller holt und wieder anfängt! Bitte, bitte !

Das Clubtreffen findet statt! Leider kam im April eine Messe (Hannover Messe Industrie) dazwischen, so daß der erste mögliche Termin entweder der 23./24. März oder Ende Juni ist. Da ich am erstgenannten Termin jedoch umzuziehen plane, bleibt 29/30. Juni oder ein Wochenende im July. Der Ort des Treffens ist das Hotel Kückenmühle, 3003 Ronneberg. Eine Übernachtung incl. Frühstück kostet DM 55.- pro Person im Doppelzimmer. Wenn Du kommen möchtest, laß es mich bitte wissen. Die erhaltenen Anmeldungen betrachte ich alle nur als 'Vielleicht-Meldung', da sich der Termin doch erheblich verschoben hat! Sollte Dir der 29/30. Juni oder ein Wochenende im July nicht passen, laß mich das bitte ebenfalls wissen. Ich möchte versuchen, einen Termin zu finden, der möglichst allen Interessenten paßt. Meldungen, die ich nach dem 18. März erhalte kommen insofern zu spät, als ich dann die Termineinschränkungen nicht mehr beachten und für Bereitstellung eine Übernachtungsmöglichkeit nicht garantieren kann.

Übrigens zieht auch Olaf Krumnow bald um. Ab spätestens 1. April wohnt er in der August-Bebel-Allee 102c in 2050 Hamburg 80. Seine neue Telefonnummer kennt er noch nicht.

Euer

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

Korrektur & Nachtrag

RAM 6.x (Herbert Oppmann, 8520)
Zum Booten: ich glaube Jay Sage's Firma heißt "Echelon Inc." und nicht "Echolon Inc.".

Dateiextensions (Herbert Oppmann, 8520)
hier noch ein paar Ergänzungen:
Dateien, die Programm-Sources enthalten
.H C Header
.FCT Turbo Pascal Function
.MOD Modula-2 (Implementation) Module
.DEF Modula-2 Definition Module
Dateien, die Dokumentation enthalten:
.CMT Assembler-Doku anhand von Kommentaren
Dateien, die Dateien enthalten
.ZIP Archiv
.ARC Archiv MSDOS-Software
.ARK Archiv CP/M-Software
Dateien, die Listings enthalten
.SYM Symboldatei (Assembler, Modula-2)
.MSY Modula-2 Symboldatei
Sonstige Dateien
.O Object
.OBJ Object
.MCD Modula-2 Compiled
.MRL Modula-2 .REL

Fragen & Antworten

F: (Rolf Kirchhoff, 8520)
Wie kann man einen Streamer an unseren MTX anschließen, an dem Streamer ist ein 50-poliges Kabel, meiner Vermutung nach ähnlich einem 8"-Laufwerk.

F: (Rolf Kirchhoff, 8520)
Habt Ihr schon mal nachgedacht, ob es nicht möglich ist, eine HRG-Karte aus dem PC-Bereich an unsere schwarze Kiste anzuschließen.
A: (Herbert zur Nedden, 2000)
Grundsätzlich haben wir schon an den Einsatz von PC-Karten gedacht: siehe OMTI-Festplattencontroller. Wenn schon eine HRG-Karte, dann bitte etwas vernünftiges, also VGA o.ä.? Nur wer erstellt den passenden Treiber (mögl. Edicta-kompatibel)?

F: (Rolf Kirchhoff, 8520)
Wie kann ich aus meiner 80-Zeichen-Karte für den Farbmonitorausgang TTL-Level herstellen? Versuche dem LS06 (E3) einen LS04 huckepack aufzusetzen und diese Signale zu verwenden brachte keinen Erfolg.
A: (Herbert zur Nedden, 2000)
Evtl. mit einem HC04?

F: (Christian Ohly, 6200)
Gibt's in unserer PD-Sammlung Programme zum Notenschreiben und für Astronomie?

H a r d w a r e: IBM-Tastatur

IBM-Tastatur für den MTX

(Eckhard Hellmich, 3070)

Liebe MTX-Clubmitglieder,

vor einiger Zeit lernte ich den absoluten MTX-Freak Claudio kennen und mußte mich von der Leistungsfähigkeit des MTX überzeugen lassen. Nach einigen (besser: aufwendigen und nächtelangen) Versuchen konnte ich die MTX-80-Zeichen-Karte und einige andere Dinge in mein ECB-System integrieren, so daß ich nach und nach eine ganze Menge MTX-Software auf meiner Kiste zum Laufen bringen konnte.
Mittlerweile liefert eine IBM-kompatible Tastatur meiner CPU die Dinge, die ich ihr beibringen möchte. Das zugehörige Interface nebst Treibersoftware möchte hiermit als Newcomer im Club vorstellen . Nach HzN's Auskunft müßte der Treiber auch in RAM 6x einzubauen sein. Aber das bleibt den MTX-Usern überlassen.

Das Interface parallelisiert die seriellen Tastaturdaten einer IBM-XT-kompatiblen Tastatur und liefert sie an einen Z80-PIO-Baustein, der die Tastaturdaten per Interrupt für die CPU bereitstellt.

Die obengenannte Tastatur sendet immer dann ein "Datum", wenn eine Taste gedrückt bzw. losgelassen wird. Bei gedrückter Taste ist Bit 7 = 0, im anderen Fall 1. Die erste Codeform heißt "Makecode", die andere "Breakcode". Die restlichen Bits entsprechen der Platznummer der Taste auf dem Tastenfeld. Die ESC-Taste hat z.B. die Tastennummer 1, die Funktionstaste 10 die Tastennummer 68, so daß sich für diese Tasten folgende Codes ergeben:

	Makecode	Breakcode
ESC:	00000001	10000001
FT10:	01000100	11000100

Für die Tastenkombination Shift-Esc müssen also insgesamt vier Codes ausgewertet werden: Makecode Shift, Makecode ESC, Breakcode ESC, Breakcode Shift.
Dieser Code wird zusammen mit einem zusätzlichen Startbit (aktiv high) an Pin 2 des 5-poligen DIN-Tastatursteckers bereitgestellt. Die Pins des DIN-Steckers sind numeriert, so daß Irrtümer ausgeschlossen sind.

Pin-Belegung des Tastatursteckers

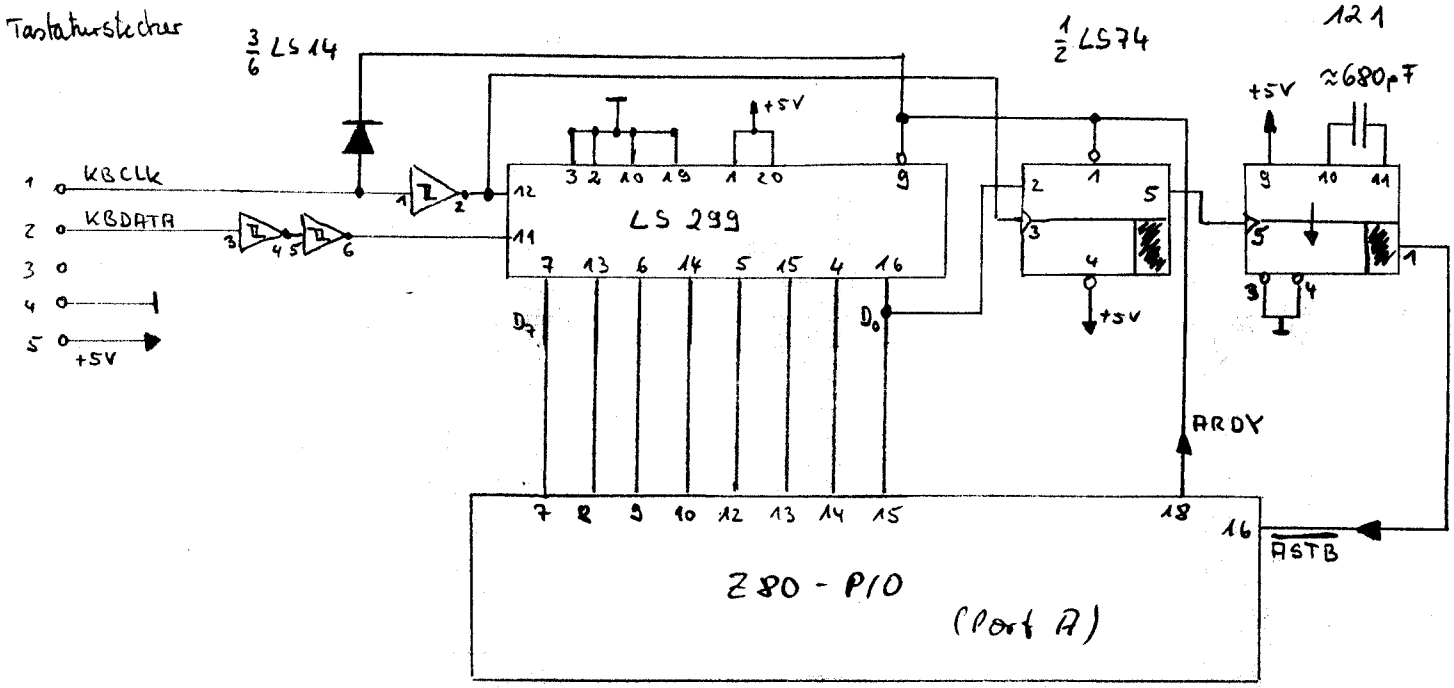
1:	KBCLK	*Datentakt von der Tastatur*
2:	KBDATA	*Daten von der Tastatur*
3:	KBRESET	(Habe ich nie benötigt)
4:	Masse	
5:	+ 5 Volt	

Die Tastatur erzeugt ihren eigenen Datentakt an PIN 1. Mit diesem Takt können die seriellen Daten über ein Schieberegister in parallele Daten umgewandelt werden. Näheres zu diesem Bitramsch findet man in der c't:

1.) Heft 10/85, c't 86 UNICARD
2.) Heft 6/88, Artikelserie über PC-Tastaturen

Wenn man die Schaltungen in den Artikeln studiert, scheint ein Widerspruch zur oben angegebenen Steckerbelegung zu existieren. Ich kann dazu nur soviel sagen, daß ich die hier angegebene Belegung am Commodore PC-10-II mit einer Peacock MF-II-Tastatur getestet habe. - Die Anschlüsse der c't-Tastaturen sind mir schleierhaft.
Der Artikel 1 ist die prinzipielle Grundlage für die folgende Parallelisierungsschaltung.

H a r d w a r e: IBM-Tastatur



Das IC 74LS299 ist als Universalschieberegister zwar etwas überdimensioniert, aber es lag gerade in meiner Bastelkiste herum. Der 74LS164 müßte auch funktionieren (Anschluß siehe Artikel 1 aus c't).

Funktionsweise der Schaltung in Stichworten:
- Startbit wird ins Flip-Flop geschoben und triggert Monoflop
- Monoflop erzeugt Strobesignal für PIO, Übernahme des Schieberegisterinhalts in Port A der Z80-PIO
- PIO setzt Readysignal solange auf Null, bis Byte eingelesen worden ist
- Readysignal "resettet" LS299 und LS74, über die Diode wird die Tastatur gesperrt.

Von den Betriebsarten der Z80-PIO wird hier der Modus 1 gewählt. Es handelt sich um ein interruptgesteuertes Einlesen eines Datenbytes. Das IC 74121 ist dabei der wesentliche Bestandteil zur Generierung dieses Interrupts. Nachdem die Tastaturdaten stabil an den Ausgängen des Schieberegisters anstehen, werden sie während der Low-Phase des Strobesignals in die PIO übertragen. Die ansteigende Flanke des Strobesignals setzt die Readyleitung des PIO-Ports A und die "Interruptanforderungsleitung" der Z80-PIO auf Null. Dadurch wird die Tastatur für weitere Datensendungen gesperrt, die Register der Parallelisierungsschaltung für die nächste Eingabe zurückgesetzt und ein Interrupt erzeugt. Liest die CPU aufgrund dieses Interrupts das Datenregister A der PIO, geht die Readyleitung auf 1, und die Schaltung kann das nächste Byte empfangen.
Damit die PIO diese Dienste leistet, muß sie entsprechend programmiert werden. Dazu werden insgesamt drei Bytes an den entsprechenden Controlport der PIO gesendet und der Interruptvektor des RAM xx Systems in geeigneter Weise gesetzt:

```
BASE    EQU     010H            ; für meine c't-IO-Karte
;
PIO1    EQU     BASE+08H
PIO1A   EQU     PIO1+0
PIO1AC  EQU     PIO1+1
```

H a r d w a r e: IBM-Tastatur

```
IntPio1a EQU    0F098H          ; Interruptvektor

                ; Hier holt sich die CPU die Startadresse für ein Programm, das
                ; den Interrupt der PIO 1A bedient.
                ; Das High-Byte dieser Adresse  (0F0h) steht im Interruptregister
                ; der CPU gemäß RAM 4x. Das Low-Byte (98h) muß die PIO liefern.

INIT:   LD      HL, IntTast     ; Bei der Adresse IntTast beginnt die
        LD      (IntPio1a), HL  ; eigentliche Tastaturinterruptroutine
                                ; der Interruptvektor muß auf IntTast
                                ; zeigen
                                ; siehe dazu "IBMINT. MAC"
        LD      HL, PIO1T       ; Tabellenstart der Initialisierungs-
                                ; befehle laden (für PIO 1)
        CALL    INITX
        IN      A,(PIO1A)       ; Dummy-Read des IBM-Tastaturports
                                ; sonst keine neuen Tastaturinterrupts
                                ; möglich
        IM2                     ;
        EI
        RETI                    ; ggf. Portbausteine aus Interrupt holen

INITX:  LD      A,(HL)
        OR      A
        RET     Z
        LD      B, A
        INC     HL
        LD      C,(HL)
        INC     HL
        OTIR
        JR      INITX

PIO1T:  DB      03, PIO1AC      ; 3 Bytes zum ControlPort der Pio-1A
        DB      LOW INTPIO1A    ; INTERRUPTVEKTOR LADEN
        DB      01001111B       ; Modus I anwählen
        DB      10000111B       ; Enable Interrupt
        DB      0               ; Ende der Tabelle für PIO 1
```

Software:

Das Programm IBMINT initialisiert die PIO und bringt eine Interruptroutine in den Commonbereich ab C000. Der wesentliche Teil des Tastaturtreibers (IBM.MAC) wird nach A000 verschoben. Die ganze Sache ist so formuliert, daß man für den Hauptteil des Treibers eine beliebige Bank wählen könnte (für RAM x.x-sichere Programmierer). Portadressen und Speicherstellen sind ggf. im Quelltext anzupassen. Der Treiber stellt in drei Bytes ab C000 die gedrückte Tastennummer, den Tastaturstatus und das gelieferte ASCII-Zeichen zur Verfügung. IBMDEMO (.PAS) zeigt diese drei Bytes auf dem Bildschirm an. Dazu muß ein COM-File mit Endadresse A000 erzeugt werden. Alle Programme sind PD und können bei mir (oder bei HzN) angefordert werden (bitte formatierte Diskette im NDR-Format und frankierte Diskettenbox o.ä. beilegen). --> KLICK.015

Das wars fürs erste, hier noch meine Adresse:

Eckhard Hellmich
Dobben 47d
3070 NIENBURG TEL.: 05021/18127

Z C P R 3.3: Aliase

Projektverwaltung II

(Herbert zur Nedden, 2000)

In Info 39 hat Olaf Krumnow dargestellt, wie er sich Programmier-Projekte von Diskette holt und sichert. Dazu hat er drei Aliase erstellt:

PROJEKT otto	*Sucht Projekt otto auf D: und holt es runter, falls da.*
PROBACK	*Sichert die geänderten Dateien des akt. Projekts zurück auf D:, so die Diskette nicht gewechselt wurde.*
PROENDE	*PROBACK und danach aufräumen.*

Damit das so funktioniert, merkt er sich in einer Shell-Variablen, von wo er welches Projekt wohin geholt hat. Da mich diese Lösung überzeugt hat, machte ich mich daran, die erforderlichen Programme zusammenzusuchen und die Aliase auszutesten.
Dabei trat bei mir ein kleines Problem auf: Die Shell-Variablen werden in der Datei SH.VAR gesichert, wobei diese Datei im Wurzelverzeichnis steht. Dummerweise ist das bei mir mein schreibgeschütztes Boot-Laufwerk. Also überlegte ich, wohin ich die Informationen noch sichern könne und ward fündig: Im Environment in einem der vier System-Dateinamen.

Bei mir wird einfach der Name des Projektes als System-Dateiname Nr. 4 gespeichert. Da ich mir jedoch nicht merke, wo die Daten herkamen, muß ich auch beim Sichern mittels LOCNDO nachschauen, wo das Projekt auf der Disk steht.

```
hol     if ~ex #$1;or in Holen $1;                    << geht noch weiter
        if $2=;locndo H:#$1 hol1 $$;else;locndo $TD1:#$1 hol1 $$;fi;fi
hol1    setfile 4 $1;qcc $TP1
```

Aufruf: HOL PROJEKT [Laufwerk:]

HOL prüft erst mal, ob das Projekt schon im akt. Directory steht, und falls dem so ist, wird gefragt, ob es wirklich nochmal geholt werden soll. Gebe ich kein Laufwerk mit an, sucht HOL auf H: (SRAM-Floppy), sonst auf dem angegebenen. Bei Treffer wird dann HOL1 aufgerufen, welches den Namen des Projektes mit vorangestelltem # als System-Datei Nr. 4 sichert und die gewünschten Dateien holt.

```
put     if ex $sn4;locndo H:$sn4 put1 $$ $1;else;echo ^GProjekt falsch;fi
put1    bak;qcc $TP1=$HB: /nos;if ~nul $2;locndo $TD1:$sn4 put2 $$ $TP1;fi
put2    qcc $TP1=$TP2 /nos
```

Aufruf: PUT [Laufwerk:]

PUT schaut nach, ob das im System-Datei Nr. 4 genannte Projekt im akt. Directory ist, und falls ja, wird es auf H: gesucht und dann auf den entsprechenden User geschoben. Wird zusätzlich ein Laufwerk mit angegeben, wird das Projekt auch noch auf diesem Laufwerk gesucht, und auch dorthin kopiert. Diese evtl. ablaufende zweite Kopieraktion wird von H: aus gemacht. Der Grund ist der, daß ich im ersten Lauf vom akt. Directory auf H: kopiere, dabei nur die geänderten Dateien mitnehme und anschließend das Archiv-Bit wieder setze. Und genauso kopiere ich dann von H: auf das angegebene Lauferk und setze auf H: die Archiv-Bits.

```
cmp     if ex $sn4;locndo H:$sn4 cmp1 $$;             << geht noch weiter
        if ~nul $1;locndo $TD1:$sn4 cmp1 $$;fi;else;echo ^GProjekt falsch;fi
cmp1    bak;qcc $TP1:=$HB: /~c
```

Aufruf: CMP [Laufwerk:]

CMP vergleicht die Dateien - nur so zur Sicherheit. Wie es funktioniert dürfte wohl klar sein, wenn Du Dir schon HOL und PUT genauer angesehen hast.

Z C P R 3.3: SHELL-Variablen

SHELL-Variablen unter ZCPR3

(Olaf Krumnow, 2050)

Was sind Shell-Variablen?

Im RAM 6.0-Handbuch werden sie gelegentlich erwähnt, die genaue Beschreibung ist aber der knappen Zeit zum Opfer gefallen, da wir RAM 6.0 unbedingt zum Clubtreffen 1990 fertig haben wollten. Deswegen will ich hier jetzt etwas über dieses starke Konzept der SHELL-Variablen schreiben, damit es vielleicht etwas häufiger genutzt werden kann und auch stärker von den Programmierern unterstützt wird.

Im Environment-Descriptor gibt es an Offset 47H einen Dateinamen der Shell-Variablen-Datei. Standardmäßig heißt diese Datei SH.VAR. Diese Datei muß nicht existieren oder extra angelegt werden. Sie wird von den Programmen, die sie nutzen, ggf. selbst erzeugt. Eine Konvention sagt, daß SH.VAR immer in einem Directory ROOT: oder (falls dieses oder Named Directories allegemein nicht existieren) im Wurzelverzeichnis des Pfades liegen muß, damit sie bei Bedarf auch gefunden wird und die Sucherei nicht so ausartet. Das Laufwerk, auf dem SH.VAR liegt, sollte nicht schreibgeschützt sein, da diese Datei häufig geändert wird.

Wie ist die Datei jetzt aufgebaut?

Jede Shell-Variable besteht aus genau acht Zeichen, die notfalls mit Leerzeichen aufgefüllt werden (man sollte nur Großbuchstaben für den Variablennamen benutzen; ich weiß nicht, ob es Probleme gibt, wenn Kleinschreibung auftaucht). Direkt daran (ohne Trennzeichen) folgt der Inhalt der Variablen, der durch eine binäre NULL beendet wird (Hier können theoretisch alle Zeichen außer NULL und 1AH benutzt werden, also auch andere Steuerzeichen, Groß-Klein usw.). Gleich darauf folgt die nächste Variable. Ganz zum Schluß folgt ein 1AH (Achtung: die letzte Variablendefinition ist trotzdem mit NULL beendet!). Die Variablen stehen ohne Reihenfolge oder Sortierung in dieser Datei. Eine Datei kann also so aussehen:

```
TESTVAR TEST<00>
8ZEICHENHallo<00>
OLAF    Olaf Krumnow<00>
NUMMER  17<00>
LETZTE  DAS ENDE<00><1A>
```

Was kann ich damit anfangen?

Bis jetzt recht wenig. Leider werden Shell-Variablen nur von sehr wenigen Programmen unterstützt und genutzt, obwohl viele mächtige Anwendungen denkbar wären. Es gibt ein Utility namens SHVAR, mit dem die Variablen gesetzt, gelöscht, geändert oder einfach nur deren Inhalt angezeigt werden kann. Ferner kann RESOLVE den Inhalt einer Shell-Variablen in eine Kommandozeile einbauen. Mit diesen Utilities (und der Hilfe einiger anderer) kann man sich bereits das Leben vereinfachen, wie ich auch in meinem Artikel 'Programmierprojekte' (Info 39-10) gezeigt habe. Außerdem verwalte ich meine CD- und MC-Sammlung auch mit Unterstützung von Shell-Variablen (überhaupt benutze ich mit Ausnahme eines Druckprogramms dafür nur simple Utilities, die auf CLUB- oder KLICK-Disketten erschienen waren).

Z C P R 3.3: SHELL-Variablen

Wer UNIX oder VMS kennt, der weiß, welches Potential hinter Betriebssytem-Variablen steckt. Unter VMS (ein Multiuser-Multitasking-Betriebssystem für die VAX) werden praktisch alle wichtigen Programme über solche Variablen an die eigenen Bedürfnisse angepaßt. So kann für den MODULA-Compiler in Variablen angegeben werden, wo er seine Bibliotheken findet, entlang welchen Pfades er zu importierende Module sucht, und vieles mehr.

Ich habe bereits in Anfängen versucht, mit Hilfe von Shell-Variablen Programme flexibler zu gestalten. So berücksichtigt XRUN 2.5 Definitinen in der Shell-Variablen-Datei und startet evtl. andere als die im Programm selber installierten Kommandozeilen. Das ist auch ein Schritt hin in Richtung Bedienungssicherheit. Wenn mehrere Leute an dem Rechner arbeiten (und ich weiß von einigen, daß das so ist), so können für diese Benutzer einige Kommandos gesperrt oder verändert werden, damit kein Schaden angerichtet werden kann. Durch einfache Änderung des Verzeichnisses ROOT: (bzw. Änderung des Pfades) kann so jeder Benutzer seine eigene SH.VAR-Datei erhalten, mit seinen individuellen Rechten.

Überhaupt könnten viele Programme dadurch, daß sie Installationen oder Einstellungen aus Shell-Variable lesen, viel flexibler werden und sich ohne komplizierte Neuinstallation an viele Gegebenheiten anpassen (man denke nur daran, daß ein Alias ja sehr leicht ROOT: umdefinieren kann).

Der nächste Schritt wäre, daß ZCPR selbst die Shell-Variablen stärker unterstützt. So könnte bereits ZCPR Ersetzungen in Eingaben durchführen, also die Variable durch ihre Definition ersetzen. Das ist sogar rekursiv denkbar, so daß eine Definition selber wieder Variablen erhält, die dann ersetzt werden. So können dann beispielsweise komplizierte Kommandoeingaben in Shell-Variablen gespeichert werden, die dann nur durch den Namen der Variablen abgerufen werden (und das auch über das Abschalten des Rechners hinaus). Mit dem Konzept von K2DOS (müßte auch in diesem Info stehen) ist es ja problemlos denkbar, den ZCPR zu vergrößern (was im Environment auch schon vorgesehen ist; unter ZCPR 3.4 wird der CCP auch nicht mehr von den Systemspuren geladen, sondern ist eine Datei, die beliebig groß sein kann; leider ist ZCPR 3.4 ein kommerzielles Produkt). Warum sollte dieser Gedanke also nicht weitergedacht werden?

Wie programmiert man mit Shell-Variablen?

Es gibt da eine Sammlung von Routinen in Assembler (Z3VARS.LBR), die die Benutzung von Shell-Variablen unterstützen. Damit können Shell-Variablen gesucht, gelesen, geändert, erzeugt und gespeichert werden. Das ist alles, was man dazu braucht. Die Dokumentation (in Englisch) ist dabei.

Außerdem werde ich für TURBO-PASCAL bald einige Routinen herausbringen, die zur Programmierung von ZCPR-Features genutzt werden können (hoffentlich schon zum nächsten Info).

Für andere Programmiersprachen sollte es nicht allzu schwer sein, sich die Sachen selber zu programmieren. Einige Hinweise stehen ja bereits in diesem Artikel, ansonsten sollte man sich die Dokumentation zu Z3VARS mal durchlesen.

Ich kann nur hoffen, daß Shell-Variablen bald stärker unterstützt werden, denn ihre denkbaren Anwendungen sind sehr vielfältig, wie dieser Artikel hoffentlich verdeutlicht hat. Als Aufruf an alle Programmierer: Checkt Eure Programme, ob der Einsatz von Shell-Variablen sinnvoll ist und bringt neue Versionen raus!!

A s s e m b l e r: Macro

Bit-Macro (Claudio Romanazzi, 3070)

```
 1 Test Macro   RSB
 2      FTyp aset 0
 3      FBit aset 0
 4      FOffset aset 0
 5      Flag FTyp,RSB,FBit,FOffset
 6      endM
 7
 8
 9 Flag Macro FTyp,RSB,FBit,FOffset
10      local RSB1
11      RSB1 aset RSB+2                 ; Parameter zurechtrücken
12      if RSB1 > 3
13       RSB1 aset RSB1-3               ; das heißt Bit
14      endif
15      RSB1 aset RSB1 shl 6            ; in Position schieben
16      FBit aset FBit shl 3            ; Bitnummer in richtig Position
17
18      if FTyp > 5                     ; gilt nur für ix und iy
19       if FTyp = 6                    ; unterscheide ix und iy
20        dw 0cbddh                     ; ix 1. + 2. Byte eintragen
21       else
22        dw 0cbfdh                     ; iy 1. + 2. Byte eintragen
23       endif
24
25       db low FOffset                 ; 3. Byte, Offset eintragen
26       FTyp aset 0000000000000110b    ; 6 = indizierte Register
27                                      ; für gemeinsamen Ausgang
28      else                            ; jetzt die normalen Register
29       db     0cbh                    ; erstes Byte
30      endif
31
32      db low (RSB1 or FBit or FTyp)   ; jetzt letztes Byte konstruieren
33      endm
```

Was sind denn das für Hyroglyphen wird ein nicht so gut der Machinensprache Mächtiger bemerken. Vor der Antwort soll aber die Vorgeschichte kommen. Als Herbert sein geniales Macro 'Platz' vorstellte, dachte ich mir, es müßte doch hunderte anderer genialer Macros geben, die Ärger vermeiden helfen, Kompliziertes vereinfachen oder das Leben angenehmer machen. Da ich ja nicht erst seit gestern in Maschinensprache programmiere, nahm ich mir meine Quellen vor und suchte die Dinge einfacher zu machen. Doch nur unwichtiges, eben des Veröffentlichens nicht würdig, kam heraus. Doch kommt Zeit kommt Macro.

Als ich jetzt begann, meinen Wortmanager zu überarbeiten, fand ich mich in einem Dschungel von Flags (für den Laien: das sind Wegweiser für den Programmablauf) wieder. Ich benutzte damals sechs IX-Register bitweise (für den Laien: jedes Register hat acht Bits, von denen jedes zu Entscheidungszwecken herangezogen werden kann). Heute habe ich keine Ahnung mehr davon, welches Bit was bedeutet. Ich hatte zwar hervorragend dokumentiert, doch bei Quellcode (für den Laien: das ist das, was der Programmierer schreibt) von 150 Kilobytes dauernd nachzuschauen, ist äußerst nervig. Da kam ich auf die Idee, den Flags Namen zugeben.

Jedes Flag wird zunächst in einem eigenen Macro definiert. Es müssen dafür drei Parameter gesetzt werden.

A s s e m b l e r: Macro

1) der Typ
Der Typ, zur Unterscheidung von anderen Typen hier FTyp genannt, wird aus folgender Tabelle entnommen:

0) b-Register
1) c-Register
2) d-Register
3) e-Register
4) h-Register
5) l-Register
6) ix-Register
7) iy-Register

Diese Werte sind natürlich nicht zufällig gewählt. Sie entsprechen dem im Steuerbyte der Z80-Maschinensprache enthaltenen Code (bis auf Code 6 und 7, aber die werden ja sowieso gesondert behandelt). Will man also das c-Register haben, wird FTyp = 1 gesetzt.

2) das Bit
Das Bit wird hier aus den gleichen Gründen wie oben FBit genannt und ist wohl selbsterklärend. Will man Bit Nummer 5 haben, so wird FBit = 5 gesetzt

3) der Offset
Der Offset wird nur für die Index-Register gebraucht. Da wir aber alles einfach halten wollen (wirklich?), wird der Offset immer gesetzt. Wird er nicht gebraucht, so ist der Wert zwar beliebig, sollte aber aus logistischen Gründen auf 0 gesetzt werden. Soll aber ein Index-Register benutzt werden, so bezeichnet FOffset (für den Laien: Offset bezeichnet die Entfernung des Flags vom Grundwert, z.B. Grundwert ist 100, der Offset 5, so ist das Flag, das ich meine, in Byte 105 enthalten) eben besagter Offset.

Aufgerufen wird das eben erstellte Macro mit einem Parmeter RSB. R bedeutet Reset Bit (also lösche in evtl. bestehendes Bit), S bedeutet Set Bit (also setze ein Bit) und B ist gleich Bit (bedeutet also 'zeig mir den Inhalt dieses Flags'). Aus logistischen Gründen ist Res = 0, Set = 1 und Bit = 2 gesetzt. Natürlich dürfen keine Buchstaben, sondern nur Zahlen als Parameter übergeben werden.

Oben in Zeile 1 beginnt das Macro Test. wie man unschwer feststellen kann, betrifft dieses Macro das Register ? Bit ? (? bedeutet, sieh selber in den Tabellen oder im Text nach).

Kommen wir jetzt zur eigentlichen Auswertung. Damit kein ellenlanger Text entsteht habe ich die unübersichtliche Lösung gewählt, was die definierten Bytes angeht und die übersichtliche, was die Übersichlichkeit angeht. Der Kommentar hinter den Zeilen spricht eigentlich für sich. Ich will jedoch erklären, was da eigentlich gemacht wird:

Der Maschinen-Code für die Bit-Befehle des Z80 (unser Lieblingsprozessor) beginnt immer mit OCBH (H bedeutet Hex). Das ist jedoch nicht richtig für die Index-Register. Deshalb werden sie extra abgefragt und dann ihr Extra-Code vorangestellt. Wie man sieht, folgt danach auch das oben angesprochene OCBH. Damit die Werte in der richtigen Reihenfolge im Ram stehen, müssen sie (klingt paradox) verkehrtherum eingegeben werden. Bitbefehle der Index-Register benötigen aber auch noch den Offset. Der wird jetzt, wo wir noch in der Bedingung der Index-Register sind (Zeile 25) als Byte angehängt. Danach vereinigen sich die Stränge wieder. Es folgt das letzte Byte. Es setzt sich aus drei Komponenten zusammen, dem Code für res, set oder bit in den Bits 7 und 6 (das wurde in den

A s s e m b l e r: Macro

Zeilen 10 - 15 erledigt), dem Code für das Bit, der wegen seiner Stellung im besagten letzten Byte mit 8 multipliziert werden muß. Das macht auch ein dreimaliges Linksschieben (für den Laien: jedes 'Linksschieben' gleicht einer Multiplikation mit 2. Macht man das dreimal, so hat man mit 8 multipliziert) und dem Code des Registers. Die Index-Register haben einen gemeinsamen Code, nämlich den Code 6 (Zeile 26). In der Zeile 32 wird dieses letzte Byte zusammengesetzt und übergeben.

Ja und nun, wird mancher fragen. Ich jedenfalls nehme meine Flags und definiere sie als Macros. Im Text verwende ich die logischen Bezeichner (auch Namen genannt), die auch etwas aussagen und versehe sie mit dem logischen Parameter. Meine Macros machen dann was ich will und ich bins zufrieden. Punktum.

Ach ja, wer das nicht abtippen will, findet die beiden Macros auf PD CLUB.057.

Anwendungsbeispiel (Herbert zur Nedden, 2000)

```
; MACROS, die die Flags definieren

Flag1   Macro   RSB                     ; Bit 0 von C
        FTyp aset 1
        FBit aset 0
        FOffset aset 0
        Flag FTyp,RSB,FBit,FOffset
        endM

Flag2   Macro   RSB                     ; Bit 3 von C
        FTyp aset 1
        FBit aset 3
        FOffset aset 0
        Flag FTyp,RSB,FBit,FOffset
        endM

Flag3   Macro   RSB                     ; Bit 2 von (ix+10)
        FTyp aset 6
        FBit aset 2
        FOffset aset 10
        Flag FTyp,RSB,FBit,FOffset
        endM

; MACRO zur Codegenerierung

Flag    Macro FTyp,RSB,FBit,FOffset
        ... s.o. ...
        endm

; PROGRAMM

        Flag1   1                       ; Setze Flag 1
        Flag2   0                       ; Lösche Flag 2
        Flag3   2                       ; Prüfe Flag 3
```

A s s e m b l e r: Macro

Bit-Macro (Herbert zur Nedden, 2000)

Claudio, nimm's mir bitte nicht übel: Deine Bit-Macro-Idee finde ich einfach toll, aber die Bedienung zu unpraktisch. Daß ich wissen muß, daß Register B die Null und das Setze die Eins ist. Ich kann doch dieses Wissen auch dem Macro übereignen. Außerdem fehlen A und (HL). Also habe ich etwas probiert und folgendes Alternativ-Macro geschaffen, welches sogar ungültige Parameter anmeckert.

Die Registerangabe kann A, B, C, D, E, H, L, X für IX, Y für IY und M für (HL) lauten. Das M habe ich aus der 8080-Syntax geklaut. Entsprechend wird für Setzen ein S, für Rücksetzen ein R und für Testen ein B angegeben. Der Offset ist optional und wird, so er fehlt mit 0 angenommen - er wird allerdings nicht explizit auf Gültigkeit geprüft, da die Verwendung im DB-Befehl bei ungültiger Angabe eh einen Fehler produziert.

Die Definition von Flags erfolgt mit dem Macro:

```
FlagDef MACRO   Name,Register,BitNummer,Offset
&Name   MACRO   RSB
        Flag    Register,RSB,BitNummer,Offset
        ENDM
        ENDM
```

Wobei Name Name des Flags
Register eines der Zeichen A, B, C, D, E, H, L, X, Y oder M ist
Bit-Nummer eine Zahl zwischen 0 und 7 ist
Offset das Nämliche für IX/IY, also Byte mit Vorzeichen in Bit 7
(Offset ist optional und wird bei Fehlen auf 0 gesetzt)

Beispiele:

```
        FlagDef Flag1,C,0               ; Bit 0 von C
        FlagDef Flag2,X,2,10            ; Bit 2 von (ix+10)
        FlagDef Flag3,Y,1               ; Bit 1 von (iy)
        FlagDef Flag4,M,7               ; Bit 7 von (hl)
```

Mit einer FlagDef-Zeile wird das Macro generiert, welches das eigentliche Flag ähnlich Claudio's Lösung dann definiert. Z.B. erzeugt

```
        FlagDef Otto,A,0
```

den Code

```
Otto    MACRO   RSB
        Flag    A,RSB,0
        ENDM
```

welches beim Aufruf dann via dem Macro Flag dann seinerseits den gewünschten Code erzeugt. Das erinnert doch irgend wie an 'Box in a Box in a Box'.

Das Programm könnte dann enthalten:

```
        Flag1   S                       ; Setze Flag 1
        Flag2   R                       ; Lösche Flag 2
        Flag3   B                       ; Prüfe Flag 3
        Flag4   X                       ; liefert einen Fehler!
```

A s s e m b l e r: Macro

Das Macro, welches dann den Code generiert:

```
Flag  Macro FReg,RSB,FBit,FOffset
      local RSB1,FBit1,FReg1,FOffset1

      ; Der Code eines Set-, Res- und
      ; Bit-Befehls beginnt mit 0CBh,
      ; für IX bzw. IY jedoch mit
      ; 0DDh, 0CBh, Offset bzw.
      ; 0FDh, 0CBh, Offset.
      ; Danach kommt das Byte, in
      ; dem Set/Res/Bit, das Regis-
      ; ter und das Bit codiert sind.


      ; RSB auswerten/prüfen
      ; (RSB = Set/Reset/Bit)

      ; Vermute Fehler
      RSB1 aset 0ffh

      if '&RSB' = 'S' or '&RSB' = 's'
       ; Set
       RSB1 aset 11000000b
      endif
      if '&RSB' = 'R' or '&RSB' = 'r'
       ; Reset
       RSB1 aset 10000000b
      endif
      if '&RSB' = 'B' or '&RSB' = 'b'
       ; Bit
       RSB1 aset 01000000b
       endif
      if RSB1 = 0ffh
ERR:   Flag mit ungültiger Option RSB
      endif

      ; Bit-Nummer auswerten/prüfen

      if (FBit < 0) or (FBit > 7)
ERR:   Flag mit ungültigem Bit FBit
      endif
      ; Bitnummer in richtig Position
       FBit1 aset FBit shl 3

      ; optionaler Offset

      ; Ist  FOffset  angegeben
      if  NUL  FOffset
       ; wenn nicht, dann nimm 0
       FOffset1  aset  0
      else
       ; wenn ja, dann nimm ihn
       FOffset1 aset FOffset
      endif

      ; Register auswerten/prüfen

      ; Vermute Fehler
      FReg1 aset 0ffh

      if '&FReg' = 'X' or '&FReg' = 'x'
       FReg1 aset 6
       ; IX: Präfix 0DDh
       db      0ddh,0cbh,FOffset1
      endif
      if '&FReg' = 'Y' or '&FReg' = 'y'
       FReg1 aset 6
       ; IY: Präfix 0FDh
       db      0fdh,0cbh,FOffset1
      endif
      if '&FReg' = 'B' or '&FReg' = 'b'
       FReg1 aset 0
       db      0cbh
      endif
      if '&FReg' = 'C' or '&FReg' = 'c'
       FReg1 aset 1
       db      0cbh
      endif
      if '&FReg' = 'D' or '&FReg' = 'd'
       FReg1 aset 2
       db      0cbh
      endif
      if '&FReg' = 'E' or '&FReg' = 'e'
       FReg1 aset 3
       db      0cbh
      endif
      if '&FReg' = 'H' or '&FReg' = 'h'
       FReg1 aset 4
       db      0cbh
      endif
      if '&FReg' = 'L' or '&FReg' = 'l'
       FReg1 aset 5
       db      0cbh
      endif
      if '&FReg' = 'M' or '&FReg' = 'm'
       ; M = (HL)
       FReg1 aset 6
       db      0cbh
      endif
      if FReg1 = 0ffh
ERR:   Flag mit ungültigem Reg. FReg
      endif

      ; das interessante Byte erzeugen
      db low (RSB1 or FBit1 or FReg1)
      endm
```

L e s e r b r i e f: Claudio Romanazzi, 3070

Über Sylvester habe ich mir einmal grundsätzliche Gedanken gemacht, die ich dem Leserforum zur Diskussion stellen will:

Bis jetzt ist, soweit ich verstanden habe, die Belegungstabelle des aktuellen Laufwerks immer auf Bank 0. Mir wäre es ganz lieb, wenn das komplett auf eine Bank verbannt wird. Zur CP/M-Kompatibilität (ich wüßte nicht ein vernünftiges Programm, das gerade diese Sache direkt aus dem Ram holt) könnte man einen Zeiger auf 0 setzen, damit sowas nicht abstürzt. Sinn des ganzen ist es, größere Platten mit größeren Partitionierungen anzuschließen. Wenn ich erst mal voll in die Grafik einsteige und dann mit 256 Farben arbeite, brauche ich pro Bild 1 Meg, d.h. 20 Bilder = 1 Festplattenkapazität heutiger Größe. Daraus folgt, ich brauche größere Platten, oder noch anders ausgedrückt, die Belegungstabellen sollten sich ohne größere Beschränkungen richtig breitmachen können. P2DOS kann ja, wenn ich mich recht erinnere bis 16 Meg adressieren. Mich hindert also nur die leidige Bank-0-Geschichte am Kauf oder am zukünftigen Kauf einer Platte mit 'vernünftiger' Kapazität, **ohne die Systemgröße kleinerzumachen!**. Optische Platten könne heute schon Gigas und warum sollte ich nicht (Finanzen vorausgesetzt) so ein Gerät anschaffen. So wie es jetzt ist, kann man sogar noch das eine oder andere Byte im High Memory abstauben. Ist die Belegungstabelle nicht ca. 600 B lang, oder war das Toam ? Selbst Toam ist einsparbar, wenn man einen Pointer setzt, der auf einen freien Platz auf einer anderen Bank zeigt. Als Ketzer könnte man sogar das gesamte BDOS auf eine andere Bank verlegen. Aus welchem kühnen Grunde sollte ich das nicht tun ? Das ist aber alles (erst mal!) nicht so wichtig, wichtiger erscheint mir die Verlegung der Belegungstabelle von Bank 0 auf Bank x<>0.

Und eine weitere Sache: mir wäre es ganz recht, Laufwerk A ganz abzuschaffen. Zwar habe ich das aufgerüstete Ram, doch der Klicker werden immer mehr, der Cache frißt Platz etc. Selbst wenn man sich auf die persönlich wichtigsten Klicks beschränkt, bleiben am Ende nur so 2-300 Kb übrig. Das wird heute jedoch keiner mehr als 'genügend' Laufwerkskapazität akzeptieren, oder anders ausgedrückt, das restliche Ram ist verloren und liegt brach, wird nicht benutzt. Auch darüber sprach ich mit Olaf und er meinte Sub.Com legt seine Daten immer auf A ab. Das war jedoch das einzige Programm, das ihm einfiel. Als weiters Argument fiel ihm die Systemspur ein. Das halte ich aber auch nicht für stichhaltig, weil sie durchaus in RamX eingelesen, oder vom Bootlaufwerk, soweit nonremovable, abgelesen werden kann. Syscopy6 ist bestimmt auch nicht CPM-kompatibel. Ich stelle mir vor, beim Hochfahren von RamX in ein zu installierendes Laufwerk einzuloggen, und dabei dieses Laufwerk zu A zu machen. Das würde die Sub-Geschichte abfangen. Die jetztigen Directory-Spuren und die Laufwerksspuren entfallen ersatzlos, was wieder Platz schafft. Besonders trickreich wäre es natürlich, das ganze wählbar zu machen.

Und ganz grundsätzlich noch: Ich glaube, wir sollten uns nicht scheuen, CPM und ZCPR weiter zu entwickeln. Wenn wir gute Arbeit leisten, wird 'the Community' uns folgen. Nicht zuletzt haben wir auch von deren Arbeit profitiert. Dabei sollten wir aber durchaus im Auge behalten, daß die technische Entwicklung weitergeht und uns nicht ausschließen, indem wir an Übeholtem festzuhalten versuchen. Das kostet, wie gesehen am MTX-User-Club Deutschland, Mitglieder und folgerichtig auch Gehirnmasse der Programmierer, Anwender und nicht zuletzt der Kritiker. Ich fände es schade, die Entwicklung und die Chance zu verpassen, die uns, in der Position der Macher, gerade jetzt ins Haus steht. In diesem Sinne möchte ich dazu aufrufen, weitere Ideen beizusteuern, nicht zuletzt, um über die Konkurrenz weiterhin erfolgreich die Nase rümpfen zu können.

S o f t w a r e: MTX-Menu 1.0

MTXMENU 1.0 - Das Menü-Programm für Ram6 (Claudio Romanazzi, 3070)

Jetzt ist es also fertig. Irgendwann im Mai, glaube ich, habe ich die erste Version an Peter Würfel geschickt. Ca. 15 Disketten sind hin- und hergegangen. Mindestens 10 Telefonate (jeder!) sind nötig gewesen. Dabei betrafen die Themen nicht nur die Fehlersuche, sondern auch neue Ideen, deren Parameter und Ausführung. Entstanden ist ein Programm, das nicht nur äußerst kompakt (4Kb) ist, sondern auch unglaublich viele Eigenschaften, bei größtmöglicher Freiheit für den Programmierer des Service-Files, enthält. Wer in der Lage ist 96 Zeichen horizontal darzustellen, der kann in den Genuß von Peters Menü kommen. Die Eigenschaften unserer 80-Zeichen-Karte (hier 96) werden hier voll ausgeschöpft. Außerdem kann man einiges über die Bedienung von MTXMENU lernen, denn Erklärungen sind nun einmal trocken. Diese Erklärungen befinden sich in einem Doc, welches die (gigantische?) Größe von 66 Kb hat, nicht wenig für ein 4Kb-File. Die Grundversion des Docs habe ich Peter zugeschickt. Damals waren das noch ca. 16 Kb. Im Laufe unserer Zusammenarbeit hat er dann den 'Rest' dazugeschrieben. Deshalb:

MTXMENU 1.0 (c) CR & PW 1.1991

Tja, all das hat natürlich seinen Preis. Es kostet -nichts-! Neben anderen PD-Programmen befindet sich MTXMENU auf KLICK.013.

MTXMENU 1.0 (Peter Würfel, 7262)

MTXMENU ist eine konsequente Weiterentwicklung des MENU.COM, das vielen schon von RAM 4 her bekannt ist. Genauso wie dieses Menu, besteht auch MTXMENU aus dem eigentlichen Programm MMENU6.COM und einem Service-file MTXMENU.80Z (beim altem MENU hieß das MENU.CPR), in dem die einzelnen Menü-Bildschirme und die Kommandozeilen festgelegt sind.

In Info 38 hatte ich ja noch gemeint, Claudio könne es sich sparen, ein neues MENU zu schreiben. Und als ich dann aus dem Urlaub zurückkam, lag nicht nur dieses Info bei meiner Post, sondern auch eine Diskette von Claudio mit einem neuen MENU. Und dann begann eine intensive Zusammenarbeit: ich testete, meckerte und wünschte, und Claudio programmierte. Vielen Dank Claudio! Das was dabei rauskam - MTXMENU 1.0 - kann sich wirklich sehen lassen!

Was kann das neue MTXMENU ?

- MTXMENU kann selbstverständlich alles, was das 'alte' MENU konnte.
- MTXMENU läßt dem Programmierer freie Hand bei der Gestaltung der Menü-Bildschirme. Der MTX+++-Bildschirmtreiber kann vollständig genutzt werden.
- MTXMENU hat eine definierte Schnittstelle zu HRG-Bildschirm.
- MTXMENU ist eine Shell.
- MTXMENU macht das Programm BOOTER überflüssig, da man seine Bootzeile im Service-file ablegen kann.
- MTXMENU ist nur 4k groß, hat aber eine Dokumentation von 66k, in der, an zahlreichen Beispielen ausführlich dargestellt wird, wie ein Service-file programmiert werden kann.
- MTXMENU unterstützt Systemvariable.
- MTXMENU ermöglicht mehrere Benutzereingaben in einer Kommandozeile.
- MTXMENU kann Kommandozeilen erzeugen, die viel länger als 204 Zeichen sind.
- MTXMENU stellt interne Variable zur Verfügung, mit denen sich der Ablauf des Programms steuern läßt.
- MTXMENU ermöglicht durch Einsatz von Variablen das Erstellen eines kompakten Service-files.
- MTXMENU ..... schaut es euch einfach mal an und probiert es aus!

S o f t w a r e: SLRROR 1.0

S L R R O R Version 1.0 (Herbert zur Nedden, 2000)

Als ich unlängst in der c't oder mc von einem Assembler las, der seine Fehlermeldungen in die Source-Datei stellt, war ich beeindruckt von der - wenn auch eigentlich banalen - so aber auch genialen Idee!

Ich fragte mich, wie ich das meinem SLR-Assembler beipulen könnte. Klar, einfach die Fehlermeldungen von dem Bildschirm einlesen und in die Datei bzw. Dateien einstellen.

Damit ich die Fehlermeldungen auch richtig erwische und insbesondere die netten 'Out of Range'-Meldungen von relativen Vorwärts-Sprüngen an die richtige Stelle packen kann, setzt **SLRROR** voraus, daß die Assemblierung mit 2-Pass, d.h. mit 2nd pass listing erfolgt.

SLRROR geht folgendermaßen vor:

- Bildschirm einlesen, dabei aber max. 80 Zeichen je Zeile berücksichtigen.
- Suche 'End of File Pass 1', da dahinter die erste Fehlermeldung steht und 'End of File Pass 2' oder eine Leerzeile - also das Ende des Elends -; die Zeilen dazwischen sind die jeweils zweizeiligen Fehlermeldungen.
- Hole aus den Fehlermeldungen die Dateinamen und Zeilennummern.
- Pack die Fehlermeldungen als Kommentar '; ERROR: ....' in die jeweiligen Dateien - und zwar direkt über die betroffene Zeilen.

Um z.B. ein Programm umzuwandeln habe ich folgende Befehlsfolge:
($1 = Aufrufparameter, d.h. Name des Programmes - schließlich ist dies bei mir ein Alias!)

```
SLRASM $1/RUF          Assemblieren als REL, EXTERNALs selbst de-
                       finieren, Full-Listing (= 2-Pass)
IF ER                  Wenn Fehler aufgetreten sind
   SLRROR              Rein in den Source damit!
   E $1                und MTX-Edit bemühen
ELSE                   Sollte es wider erwarten geklappt haben
   SLRNK /V,$1,%1/N/E  dann linken, aber dabei etwas erzählen
ENDIF                  Finito
```

Wenn in mehreren Source-Dateien Fehler sind, dann mußt Du von Hand MTX-Edit oder einen anderen Editor bemühen.

Klar, wenn ich Pech habe, meckert der Linker, daß ihm Labels fehlen, da ich schließlich beim Assemblieren sage, er soll sich die Externals selbst definieren, aber der Linker verrät nicht, wo die Dinger her kommen, so daß SLRROR dessen Meldungen auch nicht mag!

Wenn Du nicht so faul wie ich bist, und die Externals im Source sauber als solche angibst, dann solltest Du beim Assemblieren die Option U weglassen. Aber selbst Dann kann es passieren, daß der Linker meckert - dann fehlt i.a. eine Library.

Sollte **SLRROR** melden, daß ihm eine der 'End of File Pass ?'-Meldungen fehlt, so mußt Du Deinen SLRASM so installieren, daß er nach 10 Fehlern aufhört, damit die Meldungen auf dem Bildschirm stehen bleiben und mit /F aufrufen!

Viel Spaß

S o f t w a r e: K2DOS

K2DOS oder 63.25kB-System (Herbert zur Nedden, 2000)

Unlängst spielte ich mit dem Gedanken, zu versuchen, das P2DOS so wie unser BIOS zu banken, d.h. möglichst viel davon auf eine andere Bank zu packen. Daß Claudio in seinem Leserbrief auch diese Idee artikulierte ist a) Zufall, b) Telepatie, c) der Beweis der Qualität der Idee ... oder x) ???

Also spielte ich mit meiner Tastatur und nach dem einen oder anderen Mißerfolg kam dann der erhoffte Erfolg! Das P2DOS ist auf Bank 0 von 3.5 KiloByte auf ein popeliges PfundByte (1 Pfund = 1/2 Kilo!) geschrumpft. Weniger ist nicht möglich, da sowohl das P2DOS als auch das BIOS (also BIOS-Sprungleiste in RAM 6.x) auf einer Adresse der Form 0XX00h beginnen müssen. Beim 60k-System beginnt das BIOS bei 0E900h - also landete das P2DOS bei 0E800h.

Das gebankte P2DOS habe ich dann K2DOS getauft. Es läuft im KLIX-Heap. Damit die Chose auch Hand und Fuß hat, muß der ZCPR 3.3 natürlich auch weiter oben im Speicher laufen, damit auch ZEX etwas davon hat. Daher enthät das KLIX-Overlay neue Systemspuren, da diese schließlich den ZCPR 3.3 und das P2DOS (also das, was für Bank 0 übrig blieb) enthalten.

Was bringt's?

Da die meisten Programme das P2DOS nicht überschreiben - sie brauchen es halt für Diskettenzugriffe - gewinne ich 3.25 kB Platz. MTX-Edit kann nun z.B. über 50kB große Dateien verarzten. Jetzt kann ich in RAM 6.x endlich wieder etwas einbauen - unter einem 60k-System habe ich beim Linken keinen Platz mehr!

Problem - FEHLER

K2DOS in der Version 1.0 (KLICK.014) ist leider nicht ganz fehlerfrei! Das ist mit Version 1.1, die obendrein einen BDOS-Trace bietet und auf KLICK.015 kommt, behoben.

Idee

Ich spiele schon wieder mit meinen Gedanken ('Er denkt ja schon wieder' laut der Feuerzangebowle): mit RAM 6.3 könnte das P2DOS bei 0F000h beginnen. Das wird hoffentlich möglich, ggf. indem ich etwas MTX-inkompatibel werde und die dämliche Tastaturtabelle aus dem Common entferne. Programme, die die eingelesene Taste damit vergleichen fallen dann halt auf die Nase. Aber das ist eigentlich nicht so schlimm, da seit RAM 4.x die Tastencodes des 10-er Blocks fest definiert sind, und daher direkt abgefragt werden können und sollten. Ich kenne kein Programm (außer alten für die Edicta-Grafikkarte) die das tun. Weiterhin wird der RCP verschwinden, damit ich auch diese 11 Sektoren verwenden kann. Das RCP diente ja der Erweiterung des ZCPR 3.3, da er in seiner Größe beschränkt ist, vor allem, damit er auf die Systemspuren paßt. Ist der ZCPR 3.3 jedoch im K2DOS oder einem Nachfolger dessen enthalten, ist die Länge wesentlich unkritischer! (Schließlich sieht ZCPR 3.3 auch CCPs unterschiedlicher Größe vor.)

Damit hätten wir dann ein 65.25kB-System! Klingt tierisch, nicht wahr?

S o f t w a r e: Krüpthogravieh / Wortstern-Flicken

K R Y P T O . P A S - 1.0 (Peter Würfel, 7262)

Ein Programm zum Ver- und Entschlüsseln von Dateien.

Zur Kodierprozedur schrieb c't: "Das Kodierverfahren ist so ausgewählt, daß es 'antimetrisch' funktioniert, so daß das zweimalige Verschlüsseln mit demselben Paßwort wieder die ursprüngliche Information zutage bringt. Als Verschlüsselungsoperation eignet sich dafür die einfache XOR-Verknüpfung. Um einen ausreichenden Schutz zu gewährleisten, werden nicht alle Bytes nach dem gleichen Muster verschlüsselt. Die Operationen wiederholen sich erst nach einer Anzahl von Bytes, die der doppelten Länge des verwendeten Codewortes entspricht. Da aber der mitlaufende Zähler über die Funktion lo(I*I) 'xort' wird, läßt sich so leicht keine Verschlüsselungsperiode erkennen."

Die Procedure 'Code' sowie die '0%-50%-100%'-Bildschirmanzeige unterliegt dem Copyright von Jürgen Baumgartl, da sie seinem Programm 'protect.pas' entnommen wurde, das er in c't 3/1986 veröffentlichte. Die Procedure 'GetDriveUser' stammt von Herbert zur Nedden aus seinem Programm 'filz.pas'. Die Proceduren zu einem Programm zusammengebaut (programmieren kann man ja sowas kaum nennen) hat:

Peter Würfel
7262 Althengstett
Datum : 03.02.91
Compiler : Turbo Pascal 3.0
Sonstiges : Voraussetzung ist der Bildschirmtreiber von RAM 6.x
PD-Disk : KLICK.015

WordStar 4.0 - Patches (Herbert zur Nedden, 2000)

Auf CLUB.055 sind meine aktuellen Patches für WordStar 4.0. Was gut und aufwendig ist, ist mit von der Partie. Sei es einfach mal die Anpassung an den Memotech, die IBM-Grafik via ^P^X oder gar echt kursive Zeichen auf dem Bildschirm oder nur einfach, daß beim Ändern einer Datei dessen Create-Zeit nicht verändert wird ... es ist dabei.

Da vermutlich jeder, der WS 4.0 hat auch RAM 6.1 hat, sind die Patches für eben dieses Betriebssystem ausgelegt. Anderenfalls mußt Du die Patches erst mal etwas ändern - wie ist jeweils vermerkt.

Etwas unklar ist mir immer noch, wo WS 4.0 warum seine Overlays sucht. Sind sie auf I3: dann solltest Du a) das Laufwerk I: als erstes gültiges Laufwerk in WS 4.0 installieren, b) User 3 als Overlay-User in WS 4.0 installieren und c) I3: in den ZCPR 3.3-Suchpfad aufnehmen.

Für Spellstar von WS 4.0, der Versuch von MicroPro etwas bereitzustellen, was mit Claudio Romanazzis WortManager konkurrieren könnte, ist eine Anpassung auf diese unsere Sprache deutsch samt Lexikon mit auf dieser PD.

Sogar Füßeln wird unterstützt: WS-NOTE, ein Fußnoten-Programm ist zu allem Überfluß auch noch auf der Scheibe.

T u r b o - P a s c a l und B D O S: Laufwerk/User

Laufwerk/User setzen (Herbert zur Nedden, 2000)

Peter Würfel bat mich um einen Artikel:

In 'krypto.inc' gibts die PROCEDURE GetDriveUser. Daß man damit z.B. User einloggen kann, was ja sonst von Pascal aus nicht geht, das hab ich kapiert. (Die Procedure stammt aus Deinem 'filz.pas'. (Was mir gerade bei meinen Programmierübungen in Pascal immer wieder gute Dienste geleistet hat)). Was ich nicht kapiere, das sind folgende Zeilen in diesem Programm:

```
Laufwerk.Drive:=chr(65+(Mem[4] and $f));
Laufwerk.User:=Mem[4] shr 4;
...
BDOS(13);
BDOS(32,Laufwerk.User);
BDOS(14,i);
BDOS(37,1 shl I);
```

Könntest Du für eines der nächsten Infos dazu mal was dazu schreiben? Ich glaube, daran hätten auch andere Interesse: Wie ist das mit BDOS... in dieser Procedur. Was hat das mit CP/M und was mit ZCPR3 zu tun? Wie geht man damit um? Und: Was kann man auf diesem Weg sonst noch tun?

Der Artikel:

Der CCP (und daher auch anstandshalber der ZCPR 3.3) legt das aktuelle Laufwerk und den aktuellen User an Adresse 4 (= 00004h) ab. In dem unteren Nibble (= Bit 0-3) steht das Laufwerk (0 = A:, ..., 15 = P:), im oberen (= Bit 4-7) der User (0 = 0, ..., 15=15). Eben dieses Byte ist übrigens der Grund, warum das Einloggen von Usern 16 bis 31 ein Problem darstellt!

```
Laufwerk.Drive:=chr(65+(Mem[4] and $f));
```
Holt Laufwerk von Adresse 4. Mit 'and $f' wird das obere Nibble wegmaskiert, damit nur Bit 0-3 überleben; die Addition von 65 macht aus der 0 den Ascii-Code von 'A' und chr(..) macht ein Zeichen aus dem Kram in der Klammer.
```
Laufwerk.User:=Mem[4] shr 4;
```
Holt den User von Adresse 4. Mit 'shl 4' wird die Information 4 Bit nach rechts geschoben, d.h. Bit 4-7 wandern in Bit 0-3.

```
BDOS(13);
```
BDOS-Funktion 13: Reset
```
BDOS(32,Laufwerk.User);
```
BDOS-Funktion 32: User selektieren
```
i := ord(Drive.Laufwerk) - ord('A');
BDOS(14,i);
```
BDOS-Funktion 14: Laufwerk selektieren
```
BDOS(37,1 shl I);
```
BDOS-Funktion 37: Ein einzelnes Laufwerk zurücksetzen

Ich gebe zu, daß BDOS(13) und dann noch BDOS(37,...) doppelt gemoppelt ist. Die letztere Variante setzt nur ein bestimmtes Laufwerk zurück, während die erstere gleiche alle zurücksetzt. Sind auf einem Laufwerk offene Dateien, solltest Du von dem Zurücksetzen aller Laufwerke Abstand nehmen. Das ist auch der Grund, warum DiJey (das Kopier-u.s.w.-Programm im KLIX-Heap) nur das explizite Zurücksetzen einzelner Laufwerke unterstützt.

H a r d w a r e: Probleme ?

Farbmonitor (Herbert Oppmann, 8520)

Seit kurzem bin ich im Besitz eines solchen. Ist riesengroß und leider RGB, so daß ich ihn nicht so ohne weiteres an die 40-Zeichen-Karte anschliessen kann. Am Anfang rätselte ich ganz schön rum, warum beim Booten und bei einigen MTX-Programmen einige Buchstaben mitten im Wort eine andere Farbe als die umgebenden Buchstaben haben. Bis ich dann auf dem Monochrom-Monitor bemerkte, daß das genau die Buchstaben mit Unterlängen in unterstrichenen Wörtern waren. Zugegeben, auf Monochrom sieht das besser aus, aber auf dem Farbmoni doch recht merkwürdig.

Rolf Kirchhoffs Festplatte (Herbert Oppmann, 8520)

Sie geht endlich. Yippie! Ich habe vor zwei Wochen endlich ein Manual zum OMTI 5520 auftreiben können und mit diesem Material war ich in der Lage, den Fehler näher zu analysieren. Also, der Controller meldete sich auf den gewünschten Ports und ich konnte auch Befehle absetzen. Allerdings meldete mir der Controller zu fast allen Befehlen $02 zurück, also Fehler. Auch der Befehl, mit dem man nachschauen kann, was genau der Fehler war, meldete Fehler. Na toll. Das sah nach einem Fehler im Datenbus aus, da das ganze Handshaking einwandfrei (deterministisch) funktionierte. Trotz der Tatsache, daß wir die Hardware schon ein dutzendmal gecheckt hatten, nochmal den Datenbus durchgemessen: einwandfrei. Ja zum ... Da mir nix mehr schlaues einfiel und ich schon an meiner Intelligenz zweifelte, klammerte ich mich an die schwache Hoffnung, daß vielleicht ein Bustreiber was bringen könnte. Also schnell eine Portdekodierung auf Papier entworfen und ran ans Löten (der Bustreiber darf ja nur Richtung MTX treiben, wenn der Controller lesend angesprochen wird). Nachdem ich einen LS245 draufgetan hatte, machte ich mich daran, den Datenbus aufzutrennen. Da ich nicht auf der Platine kratzen wollte, bot es sich an, am Slotverbinder die Drahtstücke zu entfernen, mit denen der Datenbus auf die Platine verbunden ist. Nachdem ich schon D0-D6 weggelötet hatte, machte ich mich über D7 und bekam fast einen Schreikrampf. Noch bevor ich den Lötkolben hinhielt, hatte ich diesen Draht in der Hand (bzw. Zange)! Der Draht war im Loch in der Platine abgebrochen! Von aussen absolut nicht zu sehen! Und wenn ich gemessen habe, habe ich natürlich die Prüfspitze auf den Kontakt des Slotverbinders gehalten und damit den Draht in das Loch gedrückt und damit die Verbindung hergestellt.

Aaaaaaaarrrrrrrgggggghhhhhh!!

13 Monate hat uns dieser Fehler gekostet, von Nerven und Telefoneinheiten ganz zu schweigen.

Hardware-Problem (Herbert Oppmann, 8520)

Ich muß meinen Rechner nach dem Einschalten erst ca. 2 Minuten laufen lassen, bevor ich Reset drücken und erfolgreich Booten kann. Wenn ich zu kurz warte, fängt er zwar an zu Booten, bleibt aber irgendwo mittendrin stehen. Das scheint von der Außentemperatur unabhängig zu sein. Hat jemand ne Ahnung, woran das liegen kann?

Anm.d.HzN: Evtl. Netzteil, Festplatte oder ein alter Taktgenerator (74S04 statt eines 74HC04) oder gar schlechte Steckkontakte. Anscheinend muß die Kiste erst mal warmlaufen.

H a r d w a r e: Festplatte

Leserbrief oder meine Geschichte von der HD-Installation! (Hartmut Traber 5270)

Im Januar fiel mir ein Funkschau-Heft vom November in die Hände. So geht das schonmal, darinnen eine Anzeige der Fa. Pollin (sollte man sich merken), HD-Laufwerke zum Superpreis.
Nach Anruf, Fragen usw. bestellte ich einfach eine HD.
Ich erhielt zum Preis von DM 240,-- incl. Versand, Verpackung:
1 3,5"-HD Miniscribe 8425, 21 MB, Steprate 28 ms,
zusätzlich ein Gehäuse aus Alu-Druckguß, teilbar (!), ca. 41 x 41 x 5,5, universell verwendbar, und Einbauten darin für die Zukunft. Darüberhinaus aber auch noch ein Netzteil 9,5 A bei 5 V, -5 V und -12 V jeweils ca. 0,2 A, bei +12 V 1,9 A, absolut passend (Abmessungen 16 x 16 x 5 incl. Lüfter (!).
Damit wandert diese HD mitsamt Netzteil bald in die FDX.
Die Vorgeschichte ist damit fast beendet.
Da ich eigentlich nur eine 3,5"=Platte erwartete, aber ein Riesenpaket (ca. 1,0 x 1,0 x 0,5 cbm) bekam, dachte ich erst an eine Falschlieferung! Und jetzt ist die Vorgeschichte beendet.
Den Omti-Controller, Gerhad Witzels Adapter und die Lötarbeiten für die Uhr hatte ich ja, na sagen wir mal, vor einem Jahr erledigt.
D. h. ich konnte einfach anschließen (!).
Das ging auch, keinerlei Abstürze, wie auch Hans Gras im Info 38 - 28 beschrieb.
Aber meine Probleme fingen dann schon mit dem falschen Port an:
Holger Hansen teilte mir alle Patch-Adressen mit, um das zumindest zu beheben und kündigte mir die Übersendung eines neuen HDSERV.CHN an.
Man ist ja so heiß mit neuer Hardware...
Folge: Ich als Nicht-Programmierer habe Turbo gelernt, und die im Info genannten Änderungen eingebaut---->es lief!
D. h. Port, Format für Omti 5510 mit 9 Sektoren usw. in hdserv.chn eingebaut, RAM61 mit fmt6 geändert, eigentlich sehr einfach, wenn man alles vorrätig hat.
Und das habe ich demnächst auf den Partitionen meiner HD.
Es läuft alles z. Zt. seit 5 Stunden. Steprate, Skew usw. muß ich noch testen.
Die Geschwindigkeit der Platte ist zwar auch beeindruckend, mir jedenfalls kommt die Speicherfähigkeit zugute.
Dank an HzN wegen der Angabe der Stepraten-Adresse unter RAM61, die geändert ist (9161h), an Holger, von dem ich trotzdem noch eine Diskette erhoffe und an Hans Gras, der seine Geschichte veröffentlichte und mir sicherlich viel Zeit erspart.
Und jetzt ist Freitag, 08.02. und 17.40 Uhr. Die Diskette mit Holgers neuem Programm HDFORMAT.Com und ...Chn war in der Post.
Die Programme Hdserv und Hdformat ergänzen sich, bis jetzt schmeiße ich noch keins von beiden weg (obwohl man sie ja eigentlich nur ein einziges Mal braucht!).
Einige Tests mit Steprate und Skew-Faktor habe ich auch schon hinter mir und will diese Ergebnisse hier nicht verschweigen:
Mein Rechner läuft noch nicht so schnell, vermutlich mit 6 MHz, ich habe jedenfalls diesen Quarz drin und die Steprate in RAM61 ist eingestellt auf 10 us.
Die Skew-Faktoren 2, 3, 4, 5, 6 habe ich ausprobiert und mit Faktor 5 statt 3 Minuten Formatierzeit 2:45 erreicht.
Da sollte vielleicht noch was zu holen sein, bitte teilt Erfahrungen mit. Im übrigen macht sich das im normalen Betrieb, beim Kopieren oder während der Arbeit fast garnicht bemerkbar, soweit ich das schon jetzt beurteilen kann.

R A M 6.x: Fragen und Probleme

Format 1C (Herbert Opmann, 8520)

Würde ich inzwischen nicht mehr so machen. Bei so großen Disketten bietet sich der Einsatz von Libraries an, und dann wäre es besser, wenn man 4 KByte Blöcke hätte und dafür weniger Direktory-Einträge.

Anm.d.HzN: Mich hat das Format auch schon dank seiner reichlich bemessenen Directory-Einträge etwas geärgert, da es dadurch langsamer als z.B. Format 1B ist. Das BDOS (also unser P2DOS) nudelt doch recht lange darin herum. Da andererseits die Umstellung von 1C auf ein neu zu schaffendes Format gleicher Bauart, jedoch mit 4kB- oder gar 8kB-Blöcken und weniger Directory-Einträgen eine nervige Arbeit ist, insbesondere, wenn man nur ein dafür geeignetes Laufwerk sein eigen nennen kann, habe ich für RAM 6.2 folgendes vor:
Der Cache von RAM 6.2 prüft beim Einlesen des Directory in den Cache, ob das dort entdeckte Format-ID mit dem Config-Code übereinstimmt und liefert ggf. eine Fehlermeldung. Nur so zur Sicherheit. Ob ein automatisches Umconfigurieren machbar ist, wage ich zu bezweifeln, da das bedeutet, daß die P2DOS-Disktabellen (ALV, CKV und Skew) geändert und damit an eine andere Adresse gelegt würden - und das P2DOS hat sich deren Adresse schon gemerkt!

Problem mit RAM 6 beim Drucken (Herbert Oppmann, 8520)

Ich habe noch keine Zeit gehabt, den Fehler systematisch einzugrenzen, aber er ist bereits mehrfach aufgetreten. Einmal habe ich mir die genaue Reihenfolge meiner Aktionen notiert:
- ich bin am Drucken
- ich breche das Druck-Programm mit <SHIFT>+<Esc> und W ab
- es kommt das Fenster "Seitenwechsel Centronics"
- da ich nicht weiter drucken möchte, gehe ich mit <SHIFT>+<BRK> in das Spoolermenü und leere den Spooler, sowie setze seine Größe auf 10 KByte (mein Defaultwert, war vorher größer)
- ich drücke <ESC> und komme wieder in das Fenster "Seitenwechsel Centronics"
- ich drücke <SHIFT>+<BS> und komme wieder auf die ZCPR-Kommandozeile
- jetzt kann ich jede beliebige Taste drücken: keine wird in der Kommandozeile angezeigt, alle rufen ein Piepsen hervor!
- es bleibt nur noch Reset

Ist so ein Fehler schon bekannt? Wenn nicht, könnte mal jemand bei der Fehlersuche helfen? Ich werde mich bei neuen Erkenntnissen melden.

Anm.d.HzN: Unter RAM 6.1 genügt ein KLICK-Warmboot und es läuft wieder.

zu Info 38-26 - Festplatten (Herbert Oppmann, 8520)

Vielen Dank für diesen sehr gut gemachten Artikel! Allerdings wage ich trotzdem eine kleine Kritik. Es geht um die "bad sectors", genauer um die auf der Festplatte oder dem Beipackzettel vermerkten "bad spots". Die in diesem Artikel beschriebene Methode zur Berechnung der Sektoren ist m.E. unzureichend. Wie beschrieben ist mit der Angabe "Bytes" die Anzahl der Bytes von Beginn der Spur an gemeint, an der die schlechte Stelle beginnt. Meines Wissens sind hier aber unformatierte Bytes gemeint (denn der Festplattenhersteller kann ja gar nicht wissen, welche Sektorgröße und damit auch welche Sektoranzahl ich verwenden möchte). Des weiteren wird die Platte ja mit einem Interleave formatiert, d.h. die physikalischen Sektoren auf der Spur werden nicht in der Reihenfolge 1-2-3-4-... angelegt, sondern z.B. bei Interleave 4 in der Reihenfolge 1-x-x-x-2-x-x-x-3-... (wobei x für andere, hier nicht näher interessierende Sektoren steht). Auch das ist vom Hersteller nicht vorhersehbar, und ist in der angegebenen Berechnung nicht berücksichtigt!