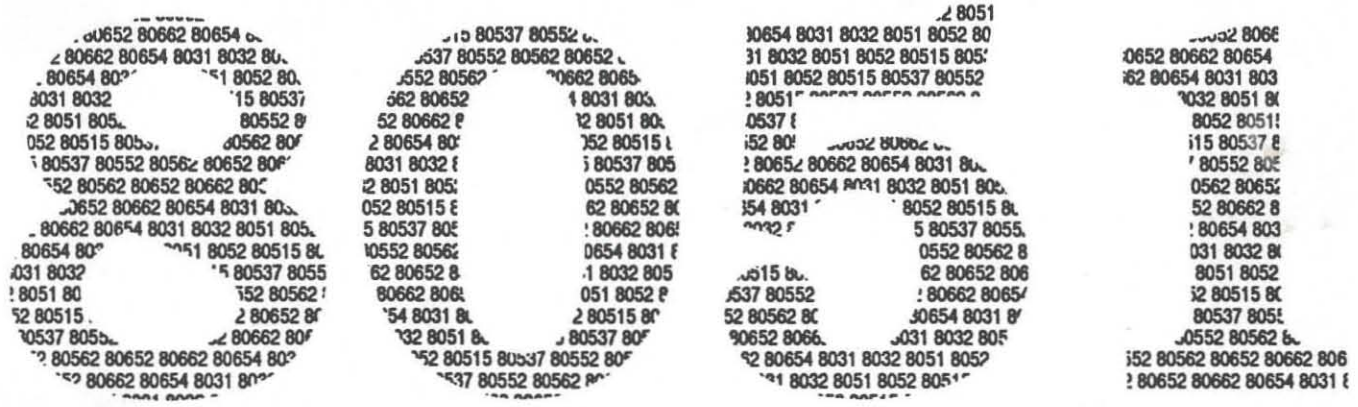
PC-NEWS

Das offizielle Mitteilungsblatt
des

PCC-TGM

(Personal Computer Club - Technologisches Gewerbe-Museum)

Mikro-Controller-Familie



Sammelbestellungen

AT-286, AT-386SX, AT-386
9-, 24-NADELDRUCKER
NOTEBOOK-386SX
LASERDRUCKER

Wenn unser E-CLUB

kein Hit wäre ...

Mit diesem Ausweis seid Ihr „live" dabei. Der ERSTE-Club ist ein Club, der vieles bietet, was junge Leute interessiert.

Der Clou bei der Sache: Ihr bezahlt keinen Mitgliedsbeitrag und kommt trotzdem in den Genuß ganz besonders günstiger Freizeitangebote, die es für ERSTE-Club-Mitglieder gibt.

Ob Rock oder Klassik, ob Theater oder Kabarett, ob Kino oder Sport – im monatlichen ERSTE-Club-Programm findet Ihr eine Menge von aktuellen Hinweisen und Veranstaltungstips. Es liegt in jeder Filiale der ERSTEN gratis für Euch bereit.

Wenn Ihr Eure Freizeit voll genießen wollt, dann genießt sie im ERSTE-Club-Stil.

DIE ERSTE
Nehmen Sie uns beim Namen

Wenn Euer erstes Geld bei uns nicht erstklassig aufgehoben wäre ...

Es ist nicht immer leicht, mit dem ersten eigenen Geld auch gleich das Richtige zu machen. Wer rechtzeitig beginnt, kleine Beträge auf die Seite zu legen, kann sich große Wünsche leichter erfüllen.
Ob Ihr vor dem Schritt ins Berufsleben oder vor dem Start ins Studium steht, der ERSTE-Jugendbetreuer hat ein umfassendes Informationspaket für Euch. Den ERSTEN-Jugendbetreuer findet Ihr in jeder Filiale der ERSTEN.
Er kennt sich bei Euren finanziellen Problemen aus. Von der günstigsten Sparform bis zu Tips für die ersten eigenen vier Wände.

DIE ERSTE
Nehmen Sie uns beim Namen

INHALT

CLUBTEIL

PREISLISTEN
INHALT 2
Liebe Clubmitglieder! 3
Mitgliederbefragung 3
Neue Diskettenpreise 3
Eingeschweißte Tabellen 3
Vorschau auf die PC-NEWS-25 3
Fehlerberichtigung PC-NEWS-24 3
Redaktionstermine 3
N E U R A L E N E T Z E - Wifi-Kurs 3
Seminare 3
BAZAR 4
Zum Druckbild der PC-NEWS 4
BTX: Satellitenbilder für den PC 4
BTX-Tip 4

VERZEICHNISSE
Verzeichnisse detailliert 48
Verzeichnisse tabellarisch 55

PREISLISTEN
Sonderpreisliste nds 57
Sonderpreisliste excon 59

SAMMELBESTELLUNGEN
Sammelbestellung (286,386SX,386) 67
Sammelbestellung (NOTEBOOK,LASERDRUCKER) . . . 68

Fallweise eingestreute Schmunzeltexte wurden von Sepp Melchart gesammelt.

BEITRAGSTEIL

Zu diesem Heft 5

805x-PROGRAMMENTWICKLUNG-ASSEMBLER
A S M 5 1 - Assemblerdirektiven 6
Externe Programmmodule in 805x-Programmen . . . 9
Die Bedienung des Assemblers ASM51 14

805X-PROGRAMMENTWICKLUNG-C-COMPILER
ENTWICKLUNGSSYSTEM FÜR 80C552 16

805X-PROGRAMMENTWICKLUNG-SIMULATION
AVSIM 51 21
INTEL 8051 LINKER MAP TO AVOCET AVSIM51 SYMBOL FORMAT CONVERTER 27

805X-ASSEMBLER/DISASSEMBLER
A3-Flachbettplotter 28
8052 - INLINE - ASSEMBLER 28
8052 - DISASSEMBLER 32

8052
Schnittstelle PC <-> 8052 36

805X-EXPERIMENTIERSYSTEME
Einsatz eines 8031-Experimetiersystems im Unterricht 40
80C32-Minimodul von nds als universeller Problemlöser 41

805X-KONSTRUKTIONEN
Am Anfang war das Chaos 43
HARDWARE-ERWEITERUNG zum mPROFI-51 . . 43
Drucksensor mit 80C522 46

Bezugsbedingungen:	Clubleistung für Mitglieder des PCC-TGM.
Auflage dieser Ausgabe:	1600 Stück.
Impressum:	Medieninhaber: PCC-TGM (Personal-Computer-Club-TGM), Wexstraße 21, Postfach 59, 1202 Wien.
Telefon/Anrufbeantworter:	(0222)/35 23 980
Bürozeiten:	Mi: 19.00-20.30, Fr: 9.00-12.00h (Frau Jelinek ,Herr Leeb)
Mailbox:	(0222)/602 10 36 (8-N-1), 9600, 4800, 2400, 1200 bit/s
BTX:	Leitseite *5645# Mitteilungen an 912222584 über Seite *941#
Grundlegende Richtung:	Auf Anwendungen im Unterricht bezogene Informationen über Personal-Computer-Systeme. Berichte über Veranstaltungen des Vereins. Beratung der Vereinsmitglieder gemäß der Statuten des PCCTGM.
Layout und Satz:	Word 5.5
Druck:	Erste Österreichische Spar-Casse, 1010 Wien
Erscheinungsort:	Wien, September 1991
Redaktion:	Franz Fiala, Siccardsburggasse 4/1/22, 1100 Wien.
Text und Programme:	Diskette TGM-DSK-169.
Beitragskennzeichnung:	Name, Firma, Abteilung, TGM-Diskette:Dateiname, Literaturhinweis. Nicht namentlich gekennzeichnete Beiträge stammen von der Redaktion.
Kopien:	Mit Quellenangabe gerne gestattet. Zwei Belegexemplare erbeten.

Zum Titelbild: Wenn man heute von Mikro-Controllern spricht, meint man meist einen Vertreter der Familie 8051! 8051 steht stellvertretend für eine große Zahl verschieden ausgestatteter Chips, von denen hier einige angegeben sind: 8031 8032 8051 8052 80515 80537 80552 80562 80652 80662 80654... Sie unterscheiden sich vor allem durch: Größe der internen/externen Speicher, Anzahl der Timer/Ports, Anzahl der seriellen Schnittstellen, AD/DA-Wandler und andere Eigenschaften, wie Sockelart, Temperaturbereich, Stromaufnahme. Sie besitzen dann weitere Kennbuchstaben zur Unterscheidung, wie: 8051AHP, 8051AHWP, 8051BH, 80C51... Dieser Mikrocontroller-Familie gemeinsam ist der Befehlssatz des 8051/8031 und daher hat sie auch ihren Namen. Alle Beiträge in diesem Heft beziehen sich auf diese Mikro-Controller-Familie.

Die 'Schrift in der Schrift' entstand nicht durch Ausschneiden, sondern durch das nebenstehende PostScript-Programm, welches, so wie es da steht, in WORD eingebunden werden kann.

NEWS24T.PS

```
% Programm 8051-Titelbild

/Bookman-Demi findfont 200 scalefont setfont
50 330 moveto
% (8051) show
(8051) true charpath clip newpath

/Helvetica-Narrow findfont 8 scalefont setfont
0.0 setgray
-600 300 moveto
25 {
        gsave
        6 {
                (8031 8032 8051 8052 80515 ) show
                (80537 80552 80562 80652 80662 80654 ) show
        } repeat
        grestore
        27 8 rmoveto
} repeat

showpage
```

Liebe Clubmitglieder!

Unsere heutige Ausgabe enthält ausschließlich Beiträge über die Mikrocontroller-Familie 805x. Alle anderen Beiträge erscheinen in der Dezember-Ausgabe (siehe Vorschau auf PC-NEWS-25).

Mitgliederbefragung

Die Diskettengutscheine wurden an alle Teilnehmer versendet. Die Ergebnisse: Anzahl der Rücksendungen: Mitglieder (54 von ca.1500), Betreuer (61 von ca.85). Mitgliederergebnisse (jeweils in %): **Frage 1:** 1:46, 2:13, 3:2, 4:17, 5:22, 6:0; **Frage 2:** 1:33, 2:9, 3:32, 4:4, 5:22; **Frage 3:** 1:37, 2:32, 3:7, 4:2, 5: 22; **Frage 4:** 1:43, 2:11, 3:4, 4:0, 5:20, 6:22; **Frage 5:** 1:44, 2:13, 3:19, 4:0, 5:24; **Frage 6:** 1:15, 2:13, 3:2, 4:2, 5:69, 6:0; **Frage 7:** 1:56, 2:6, 3:26, 4:2, 5:6, 6:5, **Frage 8:** 1:22, 2:74, 3:4; **Frage 9:** 1:52, 2:6, 3:28, 4:4, 5:7, 6:4. Die jeweils letzte Antwort einer Frage (z.B. Frage 2 Antwort 5) ist immer "keine Antwort". Wir danken für die Teilnahme.

Neue Diskettenpreise

Die Diskettenpreise wurden ab 1.Juli gesenkt: 5,25"/360k:40,-, 5,25"/1,2MB:50,-, 3,5"/720kB:50,-, 3,5"/1,44MB:60,-

Eingeschweißte Tabellen

Während des Sommers sind wieder viele Bestellungen für die eigeschweißten Übersichtstabellen in Clubbüro eingelangt. Da unsere Lehrmittelstelle während des Sommers nicht geöffnet ist, werden diese Bestellungen im September erledigt.

Alle Tabellen sind jetzt auch im A4-Format erhältlich. Preise A5/A4: S 15,-/S 25,-

In der heutigen Ausgabe haben wir diese Reihe erweitert. Wir bitten um Vorschläge für weitere Programmkurzbeschreibungen.

Berichtigung: TGM-TAB-001: Bei der Zeichensatztabelle sind bei den voll ausgefüllten Rahmen im Mittelteil der ersten Seite, bei den mittleren Feldern des rechten Rahmens die Codes 221/222 statt 219/219 eingetragen. Bei der Neuauflage der Tabellen ist dieser Fehler schon behoben. TGM-TAB-007: WORD hat einen Fehler im Druckausgabeprogramm. Bei proportionalen Fonts wird an Stelle des Zeichens 196(horizontaler Strich) das Zeichen 45(Bindestrich) an den Drucker gesendet. Daher ist diese Zeichenstelle ohne weitere Hilfsmittel nicht darstellbar.

Die Tabellen TGM-TAB-001 (PC-Zeichensatz, korrigiert), TGM-TAB-005 (Zeichensätze am Lasedrucker), TGM-TAB-006 (Corel-Draw) werden erst in den nächsten Wochen gefertigt und daher auch etwas später versendet.

Vorschau auf die PC-NEWS-25

- CELIZA : Ein geduldiger Psychiater im Source-Code
- Residenter Schnittstellenmonitor für COM1 oder COM2
- Umrechnungstabelle für Zeichenbreiten
- C++, eine Einführung (Teil 6 und 7)
- MSDOS-5.0: deutsch
- Formel-Setzen mit MS-Win-Word
- Lötaugen (Padstacks) in PCCARDS
- UROFLOWMETRIE
- AT-Interface, selbst geschnitzt, Teil 2: Software

Wenn Sie auf einen besonderen Beitrag schon gewartet haben, rufen Sie in der Redaktion an, wir schicken Ihnen einen Vorabdruck.

Fehlerberichtigung PC-NEWS-24

auf Seite 73, Pkt.4.1.1 magisches Quadrat 3x3, rechte untere Ecke: falsch 4, richtig 6. magisches Quadrat 4x4, linke untere Ecke: falsch 18, richtig 16.

Redaktionstermine

PC-NEWS 5/91 Heft 25	04. 11. 1991

NEURALE NETZE - Wifi-Kurs

NEUER WIFI-KURS AB Herbst 1991

Ziel: Der Teilnehmer soll Grundkenntnisse über den medizinischen Hintergrund, sowie über Aufbau und Funktion von neuralen Netzen in der künstlichen Intelligenz vermittelt bekommen, um selbständig Netze entwerfen bzw. deren Einsatzmöglichkeiten beurteilen zu können.

Teilnehmer: Personen mit Grundkenntnissen über die Bedienung eines Personal Computers und aus Digitaltechnik. Inhalt: Medizinischer Hintergrund - Prinzip des Konnektionismus - Netzarchitekturen (Perceptron, Feed-Forward-Netze, Assoziationsnetze u.a.) - Fallstudie "Bilderkennung" - Lernmechanismen - Training von neuralen Netzen - Realisation (Simulation, Workstations, neurale Hardware) - Einsatzgebiete und Anwendungsbeispiele - praktische Übungen mit Simulationssoftware und neuraler Hardware.

Allgemeines: 24 Lehreinheiten (davon 12 LE Labor) 1 x je Woche 4 LE 17.45

Kursort: TGM, 1200 Wien, Wexstraße 19-23

Teilnehmeranzahl: 9 - 12 pro Kurs

Kosten: Bitte am WIFI-Wien telefonisch erfragen (34 66 22/570)

Anmeldung: ab August 1991 am WIFI-Wien

Vortragende: D.I.Norbert Bartos (Theorie und Labor) Sabine Zimmermann (Labor)

Seminare

Für das kommende Schuljahr sind wieder Seminare geplant. Jedenfalls stattfinden sollen Seminare für Anfänger, WORD und ACAD. Termine können zur Zeit noch nicht angegeben werden. Wir ersuchen alle Interessenten uns eine vorläufige Anmeldung auf Postkarte/Anrufbeantworter zu schicken.

Haben Sie weitere Wünsche und Anregungen zu allgemein interessanten Themen, so bitten wir Sie ebenfalls um eine kurze schriftliche (Postkarte genügt) oder telefonische Mitteilung (Telefon Wien 35 23 98 0 Anrufbeantworter). Wir werden uns dann bemühen, einen fachlich kompetenten Kollegen für ein Seminar zu gewinnen.

Syrovatka

BAZAR

Zu verkaufen:

MELCHART Sepp, Tel. 0222/94 21 36:
- Floppycontroller 2 x 360k für XT 80,-
- MFM-Controller 500,-
  2 x Harddisk + 2 x Floppy für AT oder 386
- Schnittstellenkarte 1 x Parallel + 1 x Seriell 180,-
- Hercules-Grafikkarte mit Druckerschnittstelle 80,-
  (Probleme mit Word 5)
- 8 Stück RAM-Chips 4164 (64k x 1Bit, 150 ns) . . . . à 10,-

PACHER Christoph, Tel. 02287/4235:
- 1 Floppylaufwerk 360kB (Original IBM)
- CGA-Farbmonitor Philips + CGA-Grafikkarte

ECKEL Bernhard, Tel. 0222/216 04 79:
- 1 Festplatte 20 MB MFM + Floppy/Harddisk-Controller für AT

GRZECHENIA Thomas, Tel. 0222/948 70 23:
- CGA-Grafikkarte (Siemens, ungebraucht)
- CGA-Monitor monochrom (Siemens, ungebraucht)

HASSA Christian, Tel. 0222/85 99 82:
- 1 Festplatte . . . . . VB 16.000,-
  ESDI (Maxtor), 320 MB + Controller
- 1 Floppylaufwerk 360 kB
- 1 Grafikkarte Super-EGA (800x600, 16 Farben)
- 1 EGA-Farbmonitor

HUBER Ernst, Tel. 02243/84 602
- AT 8 MHz (100% IBM-AT !!! kompatibel) . . . VB 6.500,-
  2.6MB RAM (640kB+2MB Extended), EGA(256kB)+ CGA-Karte, 1.2MB FD, 360k FD, Genis-Maus GM6
- TVM Multisync MD11, 800*600 . . . . . VB 1.500,-
  leicht defekt aber funktionsfähig, müßte reparierbar sein: Der Bildschirm flackert nach dem Einschalten (verliert manche Farben), stabilisiert sich nach einiger Zeit.

Zum Druckbild der PC-NEWS
(TGM-DSK-169: POSTSCR.INI, POSTSCR.DBS)

Nachdem sich der PC-Club im Frühjahr einen PostScript-Drucker gekauft hat, war es auch an der Zeit, die zusätzlichen Möglichkeiten des PostScript-Druckertreibers auszunutzen.

Na, sehr aufregend ist das bei WORD noch nicht.: **Schattierter Hintergrund und skalierbare Schriften.** Die vielen anderen Möglichkeiten von PostScript werden leider nicht ausgenutzt.

Und die neuen Schriftarten? `Symbol` heißt der Zeichensatz, der statt lateinischen, griechische Zeichen enthält: (αβχδ...ABXΔ...) Zum Beispiel weiß ich jetzt, daß man in Griechenland, wo ja das griechische Alphabet herzukommen hat, weder fett noch kursiv und schon gar nicht beides schreibt, denn PostScript sieht das nicht vor! `ZapfDingbats` heißt ein Allerlei aus Symbolen: [abc ABC] wird zu [✪✪✱ ✡✢✣]. Schön, aber leider unsystematisch. Zur Not könnte man damit einige Zeichen in den Lücken des Bildschirmzeichenbereichs 0..31 füllen (z.B: ♣♦♥♠ u.a.). Es gibt aber zum Beispiel das Dreieck mit Spitze nach oben ▲ und unten ▼, nicht aber jene mit Spitze nach links und rechts.

Und der Aufwand? Von wegen "einfach nur Druckertreiber tauschen": Im Prinzip ja, doch fehlt, wie so oft, der IBM-Zeichensatz. Gerade bei Struktogrammen, wie in der heutigen Ausgabe, ein lästiger Mangel. Die Schriftart `LineDraw` kann zwar die Liniengrafikzeichen aber wieder nur in einer Größe darstellen (12pt). Alle Liniengrafikzeichen aus einem mit Text gemischen Bild (Diagramm, Struktogramm usw.) müssen händisch auf `LineDraw` umformatiert werden. (Und bei 12pt werden die Grafiken zu groß).

Lösung: 2 Tage Kampf mit dem 'PostScript-Drachen' und ein Kompromiß: Skalierbares `LineDraw` und (fast) vollständiger IBM-Zeichensatz in zwei Größen: 7+12pt. (Es fehlen die Zeichen von 1-31, sowie das Zeichen 169; das Zeichen Pt(158) wurde aus zwei Zeichen zusammengesetzt.) Für Interessenten ist dieser modifizierte WORD-Druckertreiber auf der TGM-DSK-169 enthalten und heißt POSTSCR.DBS und POSTSCR.INI.

Früher hatten wir für Programmlistings die Schrift `LinePrinter-9pt` verwendet. Da diese Schriftart in PostScript nicht exisitiert, wurde sie durch die einzige Schrift, die in PostScript mit festen Abständen existiert, `Courier` ersetzt. `Courier` hat aber den Nachteil, daß es sich bei gleicher Größe wie der Fließtext (`TimesRoman`) nicht ausreichend von diesem abhebt. Außerdem ist `Courier` ist relativ breiter als `LinePrinter`, daher mußte die Schriftgröße von 9 auf 7 pt reduziert werden, um etwa auf dieselbe Schreibdichte zu kommen. Frage an die Leser: Wer kann eine skalierbare Schriftart mit festen Zeichenabständen, ähnlich `LinePrinter` für PostScript besorgen oder eine Bezugsquelle angeben? Oder Wer hat ein kleines PostScript-Programm, mit dem eine eingebaute Schrift, z.B. HelveticaNarrow mit festen Abständen erscheint?

BTX: Satellitenbilder für den PC

Das Bundesamt für Zivilluftfahrt bietet über BTX auf der Seite *2310732# fünf mal täglich ein aktuelles Satellitenbild an. Dabei handelt es sich um einen Ausschnitt aus den EUMETSAT-Bildern, die auch im Fernsehen gezeigt werden.

Bisher konnten diese Bilder aufgrund der speziellen Codierung nur vom Mupid und Amiga angezeigt werden. Deshalb wurde im Auftrag des BAZ ein Anzeigeprogramm für Personal-Computer entwickelt.

Dieses Programm ist nun fertiggestellt worden und steht ab Anfang September allen Interessenten gratis zur Verfügung. Ab diesem Zeitpunkt wird die Public-Domain-Software unter dem Namen BTXSAT als Telesoftware im BTX-System angeboten werden und kann mit DECODIX ab der Version 1.16 geladen werden. *BTX-Grüße Hans Hornbostel (2:313/1.8)*

BTX-Tip

Für eine Anmeldung zu BTX (Erlangung einer persönlichen BTX-Kennung) ist es nicht mehr erforderlich einen ganzen Telefonanschluß zu haben. Jeder kann, sogar ganz ohne Telefon, BTX-Teilnehmer werden. Die einmalige Anmeldegebühr ist allerdings nicht zu umgehen. Dafür erhält man die regelmäßig erscheinende Zeitschrift BTX-Journal, sowie einmal jährlich den BTX-Führer von Professor Maurer (und fallweise einen BTX-Telebrief als Werbesendung). Wenn man am BTX-Dienst nicht teilnimmt entstehen keinerlei weitere Kosten. Gezahlt wird nur die Telefongebühr während einer BTX-Session.

Zu diesem Heft

Die vorliegende Ausgabe der PCNEWS ist dem Thema Mikro-Controller, der Familie 805x gewidmet. Der Grund für die Zusammenfassung von Wissenswertem über dieses hardwarenahe Spezialgebiet war unsere **Bausatzaktion** µPROFI, bei der sich mehr als 80 Mitglieder beteiligt haben. Außerdem sind viele Beiträge unmittelbar für den FTKL-Unterricht geeignet, sind also möglicherweise gerade zum Schulbeginn aktuell. Da der Umfang ganzer Programme zu groß ist, um abgedruckt zu werden, können Sie zu vielen Beiträgen die Originaldokumentation als TGM-LIT-xxx zusätzlich bestellen.

Für alle Leser, denen dieses Thema noch neu ist, hier eine kurze Einführung:

Was ist ein Mikro-Controller?

Es ist ein Mikro-Prozessor mit spezialisiertem Befehlssatz für Hardwaresteuerungen, der neben der eigentlichen CPU auch RAM, (EP)ROM, IO, TIMER und Interrupt-Steuerung mit auf dem Chip integriert hat. Es gibt zwar 8-, 16- und mehr-Bit Mikro-Controller, wir beschränken uns hier aber auf die 8-Bit-Familie der 805x-Chips. Wir finden solche Prozessoren auch in der PC-Tastatur und im AT auch als Tastaturcontroller auf der Hauptplatine. Auch in vielen Druckern sind diese Controller vertreten. Natürlich auch in allen Steuerungen, die bisher mit festverdrahteter Logik ausgeführt waren, werden heute nur mehr mit Mikro-Controllern ausgerüstet. Die Hardware ist mit verhältnismäßig wenig Fehlerquellen erstellbar, und der Ablauf ist durch die Mikro-Controller-Software auch später im Betrieb veränderbar, korrigierbar.

Was hat das mit dem PC zu tun?

Zunächst nichts; aber man benötigt für die Entwicklung von Programmen für Mikro-Controller Werkzeuge, die eine automatisierte Kode-Erzeugung ermöglichen. Man könnte dazu einen Mikro-Controller als Basis für einen Entwicklungsrechner heranziehen: So, wie man auf dem PC Programme für den 8086 entwickelt, würde man auf diesem 'PC51' Programme für den 8051 entwickeln. Allerdings müßte man alles, was man auf dem PC schon hat, wie BIOS, Betriebssystem, Editor, Assembler, Compiler usw. noch einmal in der Assemblersprache des 8051 verfassen. Da dies sehr aufwendig wäre, benützt man den bereits vorhandenen PC als Betriebsmittel und übersetzt 51-er-Programme mit eigenen Assemblern.

Programmentwicklung

Am TGM benutzen wir den **Assembler ASM51**(INTEL) und den **Assembler X8051**(2500AD), für den es auch den C-Compiler **C8051**(2500AD) gibt. Der durch den C-Compiler generierte Code ist aber alles andere als speicherplatzsparend, sollte daher nur sparsam eingesetzt werden.

Der **Assembler X8051** hat den Vorteil, daß er auf den C-Compiler abgestimmt ist. Außerdem kann man vom selben Erzeuger AD2500 auch noch Assembler für andere Prozessoren kaufen kann, etwa für 68000, Z80 o.ä., die dieselbe Syntax hätten. Benutzt man aber keinen C-Compiler, dann kann ebenso der **Assembler ASM51** von INTEL verwendet werden.

Debuggen

Ein weiteres Problem ist das Debuggen der so entstandenen Binärprogramme, denn: welches Programm läuft schon auf Anhieb? Hier gibt es einige Lösungswege:

1. Simulation: Das Binärprogramm läuft in einem Simulator am PC, der die Architektur des 8051-Prozessors nachbildet. Vorteil: Dieses Verfahren erfordert keinerlei zusätzliche Hardware und läßt uns die Funktion, wie bei Debuggern üblich, im Einzelschritt oder dauernd, am Bildschirm mitlesen. Nachteil: Man kann nicht alle Fehlerarten, insbesondere die des Echtzeitbetriebs und Besonderheiten des Hardwareaufbaus, testen.

2. ROM-Simulator: Ein im endgültigen Zustand als ROM (oder EPROM) ausgeführter Speicherbereich wird als RAM geschaltet, welches vom PC her ladbar ist. Vorteile: Ein ROM-Simulator kann für verschiedenste Mikro-Controller angewendet werden. Das lästige Brennen und Löschen von EPROMS entfällt. Am Mikro-Controller muß keine eigene Software betrieben werden. Nachteil: Nur 'go'-, 'no-go'-Prüfung möglich, kein Einzelschritt oder anderes Debugging.

3. Eigene Hardware mit Lader (µPROFI)**:** Darunter versteht man eine eigens angefertigte Hardware (etwa µPROFI oder Keil-Experimentierplatine), die mit einem Betriebsprogramm im PC und ein einem Boot-Eprom der Platine in der Lage ist, Binär- oder Hexdateien vom oder zum PC zu übertragen und ausführen zu lassen. Vorteil: Programm läuft in der zukünftigen Hardware. Nachteil: Hardware muß zunächst mit dem Lader laufen, bevor mit dem Testen der eigentlichen Anwenderprogramme begonnen werden kann. Für jede neue Hardware müßte dann auch ein modifiziertes Ladeprogramm geschrieben werden (daher hält man für die eigenen Versuche die Hardware immer gleich und baut nur etwas dazu).

4. Emulator: Ein Emulator ist eine 1:1-Nachbildung der zu emulierenden CPU, hier der 8051-CPU inklusive aller TIMER, des Speichers und der IO-Ports. Der Emulator steckt in der zukünftigen Hardware und ersetzt dort die CPU. Vorteil: Alle Testmöglichkeiten für beliebige Hardware. Nachteil: Aufwendig, da sowohl Hardware im PC(Steckkarte) als auch Hardware für den eigentlichen Emulatorkopf erforderlich ist.

Es ist naheliegend, daß die 4 beschriebenen Methoden in dieser Reihenfolge auch kostenintensiver werden. Wir liegen mit dem µPROFI gut, denn da wir die Basis-Hardware des µPROFI nicht verändern - und für Unterrichtszwecke ist das nicht notwendig - können wir Hardwareergänzungen anbringen, die immer mit demselben Treiberprogramm testbar bleiben.

Welche Programme benötigt man für die vorgestellten Beiträge?

Programm	Erzeuger	Beitrag
ASM51	INTEL	Assembler
X8051	2500AD	Assembler
C8051	2500AD	C-Compiler
AVSIM	AVOCET	8051-Simulator
TURBO-PASCAL	BORLAND	Assembler/Disassembler
FSD51	SCHARL	µPROFI-Debugger
MT	KEIL	KEIL-Monitor
EMUL51	NOHAU	C-Compiler

Wenn Sie zu den Beiträgen und zum Arbeiten mit diesen Programmen weitere Fragen haben, wenden Sie sich an das Clubbüro, an den Autor oder an die Redaktion.

A S M 5 1 - Assemblerdirektiven

Grete PESCHEK, TGM, NT/EL

1. Allgemeines

Ein Assemblerprogramm kann aus drei verschiedenen Typen von Assembleranweisungen bestehen:

- **Maschinenbefehl**: wird direkt durch Maschinencode ersetzt, entspricht einem vom Prozessor direkt durchführbaren Befehl.
- **Assemblerdirektive**: wird verwendet um die Programmstruktur und Symbole zu definieren; erzeugen einen nicht ausführbaren Code.
- **Assembler controls**: Assemblierungsmode wird gesetzt.

Segmente, Module, Programme

Ein Segment ist ein Block von Befehlen oder Daten im Speicher. Ein Segment kann

- verschiebbar, **relocatable** oder
- nicht verschiebbar, **absolute**

sein. Ein **verschiebbares Segment** hat eine Namen, einen Typ und eventuell andere Attribute. Segmente mit gleichen Namen, aber von verschiedenen Modulen, sind Teile des selben Segments und werden **partielle Segmente** genannt. Partielle Segmente werden mit Segmenten durch den Binder(Linker) RL51 verbunden. Ein **absolutes Segment** hat keinen Namen und kann mit keinem anderen Segment kombiniert werden.

Ein **Modul** besteht aus einem oder mehreren Segmenten oder partiellen Segmenten. Dem Modul wird durch den User ein Name zugewiesen. Ein **Objektfile** enthält einen oder mehrere Module.

Ein **Programm** besteht aus einem einzigen absoluten Modul, der aus allen absoluten und verschiebbaren Segmenten zusammengemischt ist.

2. Operanden und Ausdrücke

Operanden:

Allgemein ist eine Befehlszeile wie folgt aufgebaut:

```
[label:] Mnemonic [operand][,operand][,operand]  [;comment]
```

Jeder Operand ist aus einer der folgenden Klasse:
- Spezielles Assemblersymbol, reserviertes Wort
- Indirekte Adresse, Inhalt eines Registers ist Datenadresse
- Immediate \
- Datenadresse \ Numerischer Ausdruck
- Bitadresse / Symbolisch oder
- Codeadresse /direkt als Wert berechenbar

Die fünf Adressbereiche haben folgende vordefinierte Namen:

-	DATA	Directly addressable data address space
-	BIT	Bit address space
-	XDATA	External data address space
-	CODE	Code address space
-	IDATA	Indirectly addressable data space

Berechnung von Ausdrücken zur Übersetzungszeit

Ein Ausdruck ist eine Kombination von Zahlen, Zeicheketten, Symbolen und Operatoren, dessen Wert letzlich in einer 16-Bit Zahl abgelegt werden kann.

Zahlen:	B	Binary
	O,Q	Octal
	D oder nichts	Decimal
	H	Hexadecimal

3. Assembler Direktiven

(1) Symboldefinitionen

```
SEGMENT, EQU, SET, DATA, IDATA, XDATA, BIT, CODE
```

Mit Hilfe von Symboldefinitionen können Namen für Segmente, Register, Zahlen und Adressen vereinbart werden. Keine dieser Direktiven hat ein Labelfeld. Außer der SET-Direktive kann keine, wenn einmal gesetzt, verändert werden.

SEGMENT-Direktive

```
relocatable_segment_name SEGMENT segment type [relocation type]
```

Segmenttyp:

```
CODE/XDATA/DATA/IDATA/BIT
```

Relocationtyp:

PAGE Segment beginnt an einer 256-Byte Grenze, INPAGE muß innerhalb einer Seite liegen (nur für CODE, XDATA), INBLOCK muß innerhalb eines Block (2048-Byte) liegen, BITADDRESSABLE innerhalb vom bitadressierbaren Bereich auf Bytegrenze, UNIT Grenze auf Unit ausgerichtet, das ist beim Bit-Segment ein Bit, sonst Byte. Der UNIT-Typ wird als Defaultrelocationtyp verwendet. z.B.:

```
PART1   SEGMENT   CODE
```

EQU-Direktive

```
symbol_name EQU expression
```

oder

```
symbol_name EQU special_assembler symbol
```

Weist dem Symbolnamen einen Ausdruck zu. Der so vereinbarte Symbolname kann als symbolische Adresse in einem Daten-, Code-, Bit- oder externen Adreßsegment abhängig von Typ des Segmentes in dem es vereinbart wurde, verwendet werden. Ist der Wert des Ausdrucks eine Zahl, so kann dieses Symbol irgendwo verwendet werden.

Folgende speziellen Assemblersymbole können durch die EQU-Direktive neu vereinbart werden: A, R0, R1, R2, R3, R4, R5, R6, R7. z.B.:

```
REG1    EQU   R1
```

REG1 kann anstelle von R1 verwendet werden

```
ANZAHL  EQU   27
```

Statt der Zahl 27 kann das Symbol ANZAHL verwendet werden.

SET-Direktive

```
symbol_name SET expression

symbol_name SET special_assembler symbol
```

Ähnlich EQU, jedoch kann der Symbolname später neuerlich, duch eine weitere SET-Direktive definiert werden. z.B:

```
ZAEHLER   SET   0           ; Initialisierung des Zählers
ZAEHLER   SET   ZAEHLER+1   ; Inkrementieren des Zählers
```

BIT-Direktive

```
symbol_name BIT bit_address
```

Es wird einer Adresse im bitadressierbarem Speicher ein symbolischer Name zugeordnet. Zuordnung darf nicht durch eine weitere BIT-Direktive gändert werden. z.B.:

```
RSEG   DATA_SEG       ; relokatibles bitadressierbares
                      ; Segment
FLAGS:     DS     1
   ..
ALARM      BIT    FLAGS.0        ; Bit 0 im Byte FLAGS
LICHT_EIN  BIT    ALARM+1        ; Bit 1 im Byte FLAGS
DREHE_EIN  BIT    060H           ; absolutes Bit
```

DATA-Direktive

```
symbol_name  DATA  expression
```

Ordnet einer Adresse im internen Datenspeicher einen symbolischen Namen zu. Ergibt der Ausdruck eine Wert größer als 127, so wird ein symbolischer Name für ein Register vereinbart. Die Zuordnung darf nicht durch eine weitere DATA-Direktive im Programm verändert werden. z.B:

```
TABELLE   DATA   70H
TAB_END   DATA   7FH
```

XDATA-Direktive

```
symbol_name   XDATA   expression
```

Ordnet einer Adresse im externen Datenspeicher einen symbolischen Namen zu. Die Zuordnung darf nicht durch eine weitere XDATA-Direktive im Programm verändert werden.

IDATA-Direktive

```
symbol_name   IDATA   expression
```

Ordnet einer indirekten internen Adresse einen symbolischen Namen zu. Beim 8051 darf der Wert des Ausdrucks nicht größer als 127 sein. Die Zuordnung darf nicht durch eine weitere IDATA-Direktive im Programm verändert werden. z.B:

```
BUFFER      IDATA   60H
BUFFER_LEN  EQU     20H
BUFFER_END  IDATA   BUFFER+BUFFER_LEN-1
```

CODE

```
symbol_name   CODE   expression
```

Ordnet einer Adresse in Programmspeicher einen symbolischen Namen zu. Die Zuordnung darf nicht durch eine weitere CODE-Direktive im Programm aufgehoben werden. z.B.:

```
RESTART   CODE   00H
UPR1      CODE   70H
```

(2) Speicherinitialisierung und Speicherreservierung

```
DS, DBIT, DB, DW
```

DS-Direktive

```
[label:] DS   expression
```

Reservierung eines Bytebereiches. In jedem Segment, ausser in einem Bitsegment anwendbar.

DBIT-Direktive

```
[label:] DBIT expression
```

Vereinbarung von einem Bitbereich. Nur im bitadressierbaren Speicher möglich.

DB-Direktive

```
[label:] DB   expression list
```

Initialisierung eines Byte-Bereiches in einem Code-Segment. z.B.:

```
PRIM:  DB  1,2,3,5,7,11,13,17,19,23
ALTER: DB  'ERICH',22,'HANS',23
```

DW-Direktive

```
[label:] DW   expression
```

Initialisierung eines Wort-Bereiches in einem Code-Segment. z.B.:

```
ZWEIP:  DW  1024,2048,4096
SPRUNG: DW  UPR1,UPR2,UPR3
```

(3) Programm Binden (Program Linkage)

PUBLIC-Direktive

```
PUBLIC  list_of_names
```

Muß angegeben werden, wenn ein verwendetes Symbol außerhalb des zu assemblierenden Moduls definiert ist. z.B.:

```
PUBLIC Upr1_Name,Upr2_name
PUBLIC Anzahl1,Anzahl2
```

```
EXTRN-Direktive

EXTRN [segment_type(list_of_symbol_names)], ...
```

Mit der EXTRN-Direktive wird ein Bezug zwischen Symbolen, die in anderen Modulen definiert wurden und in diesem Modul verwendet werden, hergestellt. z.B.:

```
EXTRN  CODE(Mod_A,Mod_B)
```

NAME-Direktive

```
NAME module_name
```

Gibt dem aktuellen Modul eine Bezeichnung.

(4) Assembler Status Steuerung

END-Direktive

```
END
```

Darf keinen Label haben. Ist die letzte Anweisung in einem Programm.

ORG-Direktive

```
ORG expression
```

Durch die ORG-Direktive wird der Adreßzähler des Assemblers neu gesetzt. Nachfolgende Befehle beginnen mit der angegebenen Adresse. Direktive darf keinen Label haben.

(5) Segment Selection Directive

```
RSEG segment_name
```

Gibt an, daß nachfolgende Befehle im verschiebbaren Segment `segment_name` liegen.

```
CSEG/XSEG/DSEG/ISEG/BSEG   AT absolute_address
```

Gibt an, daß nachfolgende Befehle im absoluten Segment ab der Adresse `absolute_address` liegen. z.B.:

```
DATA_SEG1   SEGMENT  DATA      ; verschiebbares Datensegment

CODE_SEG1   SEGMENT  CODE      ; verschiebbares Codesegment

            BSEG  AT 70H       ; absolutes Bitsegment

DECIMAL_MODE: DBIT       1     ; absolutes Bit

CHAR_MODE:    DBIT       1

          RSEG      DATA_SEG1 ; selektiert das verschiebbare
                              ; Datensegment

TOTAL1:     DS         1
COUNT1:     DS         1

          RSEG      CODE_SEG1 ; selektiert das verschiebbare
                              ; Codesegment

BEGIN_CODE:
```

COMPUTER-BREVIER

Computer unser, der du bist im Rechenzentrum,
geheiligt werde dein Bildschirm.
Deine Vernetzung komme, dein COMMAND.COM geschehe,
wie im Hauptspeicher, so auf der Festplatte.
Unser tägliches Reset gib uns heute,
und vergib uns unsere Eingabefehler,
wie auch wir vergeben unseren schlampigen Programmierern.
Und führe uns nicht in die Endlosschleife,
sondern erlöse uns von allen Viren.
Denn dein ist die Floppy und der Drucker und der Joystick.
In Ewigkeit. ENTER.

(Gesehen in einem Klassenraum von "Arthur F.", modifiziert von Sepp Melchart)

Externe Programmmodule in 805x-Programmen

Dieter Reiermann, TGM, NT/EL TGM-DSK-169 EXTMOD*.*

Im Schulbetrieb werden meist kleine Programme aus wenigen Zeilen Assemblercode geschrieben. In der Praxis werden Mikroprozessorprogramme, wie man sich leicht vorstellen kann, wesentlich länger. Wenn die Hardware mit externen Speicherbausteinen ausgerüstet ist, wird im Anwendungsbereich der Mikroprozessoren der 805x- Familie nicht unbedingt Speicherplatzmangel herrschen müssen.

Man kann daher etwas verschwenderischer mit dem Speicherplatz umgehen und das Programm streng modular und gut strukturiert entwickeln. Wenn allerdings entgültig die entsprechende (P)ROM-Version eines 805x- Mikroprozessors eingesetzt wird, kann man sich unter Umständen solche allzu "sauberen" Methoden nicht leisten. Es stehen ja dann nur bis zu 8kByte zur Verfügung. Ich möchte im folgenden die Möglichkeiten der modularen Programmentwicklung mit Hilfe der beiden im Schulbetrieb verwendeten 8051-Assembler: ASM51 (Intel) und X8051 (2500AD Software) vergleichen.

Modulare Assemblerprogramme

Was ist eigentlich ein Progammodul?

Ein Programmodul ist ein Programmbaustein. Er kann sowohl Teil eines größeren Programmodules oder Teil des Gesamtprogrammes sein. Er kann in einer größeren Quelldatei stehen (interner Modul) oder eine eigene Quelldatei bilden (externer Modul).

Wozu wird man ein Programm aus externen Modulen aufbauen?

Die einzelnen Bausteine eines Programmes sollen getrennt kodiert und assembliert werden, um nicht zu große Quelldateien entstehen zu lassen (Fehleranfälligkeit).

Wie kann man ein Programm in Module aufteilen?

Programmodule sollen zusammengehörige Funktionen repräsentieren, sie sollen "große funktionelle Stärke" haben.

z.B.: Alle Programmschritte, die zur Funktion "Anzeige" gehören, werden in einem Modul zusammengefaßt ("große Stärke")

z.B.: Alle Eingänge einer Steuerung werden durch ein Modul zusammengefaßt. Hier ist die Funktionalität gering, da nicht alle Eingänge der gleichen Funktion zugehören. Es können Programmschritte nach vielen Kriterien zu Modulen zusammengefaßt werden (gleiche Befehlsschritte, zeitliche oder ablaufbezogene Zusammengehörigkeit, oder, wie beim zweiten Beispiel, datenstrombezogene Zusammengehörigkeit). Nur, wenn die Befehlsschritte eines Moduls zu einem (!) funktionellen Ziel führen, hat dieses Modul hohe funktionale Stärke und ist daher leicht änderbar, austauschbar und wartbar.

Eine weitere Hilfestellung bei der modularen Zerlegung eines Programms bietet das hierarchische Baumdiagramm. Wie in einem Organisationsschema beispielsweise eines Betriebes werden in Kästchen die einzelnen Module in ihrem hierarchischen Bezug zueinander dargestellt [Abb.1]

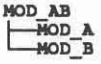


Abb.1: Hierarchie der Unterprogramme

Wie können Daten an ein Modul übergeben werden?

Daten werden in Mikroprozessorprogrammen über Speicherplätze übergeben. Man spricht auch von **Parametern**. Grundsätzlich können bestimmte Speicheradressen im RAM oder der Stack als Übergabespeicher dienen. Für den 8051 empfiehlt sich eher, die Parameter in einem Speicherbereich mit ihren Namen (=Adressen) zu übergeben . Die Übergabe über den Stack ist etwas fehleranfälliger, denn dabei werden ja nicht die Adressen, sondern nur eine Startadresse, die Reihenfolge und die Parameteranzahl übergeben. Das führt naturgemäß zu einer höheren Fehleranfälligkeit. Für kleine Programme wird diese Methode auch nicht verwendet.

Assemblieren mit dem ASM51 und RL51

Im folgenden Programmbeispiel sehen Sie die modulare Aufteilung und die Parameterübergabe. Die dabei verwendeten Assemblerdirektiven werden im Beitrag **8051-Assemblerdirektiven** zusammenfassend dargestellt.

Im ersten Assemblerlisting wird ein Rumpfhauptprogramm `Mod_AB` mit den wichtigsten Assemblerdirektiven für die Übergabe von Parametern (Daten und Unterprogrammadressen) dargestellt. Das zweite Modul `Mod_A` zeigt, wie Daten an das Hauptprogramm übergeben werden. Das dritte Modul `Mod_B` empfängt Daten vom aufrufenden Hauptprogramm.

Die drei Module wurden als ASCII-Quelldateien (geschrieben mit WORD 5.0) assembliert:

```
ASM51 extmodx.asm
```

Die daraus entstandenen Objekt-Dateien wurden zu einem Gesamtprogramm gelinkt:

```
RL51 extmod1.obj,extmod2.obj,extmod3.obj
```

Das Ergebnis war eine Binärdatei `extmod1` und eine Linkermap `extmod1.m51`, in der die gesamte Adressbelegung für den gewählten Mikroprozessor zu finden ist.

Was kommt tatsächlich dabei an Maschinencode heraus? Das kann man sich z.B. mit dem Simulator `AVSIM51` anschauen. Allerdings muß vorher aus der Binärdatei eine Intel-HEX-Datei gemacht werden:

```
OH extmod51
```

und dann

```
AVSIM51 ALP'extmod1.HEX'
```

EXTMOD1.LST

```
MCS-51 MACRO ASSEMBLER    MOD_AB                                                07/02/91   PAGE    1


DOS 4.0 (033-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN EXTMOD1.OBJ
ASSEMBLER INVOKED BY:  D:\8051\ASM51.EXE EXTMOD1.ASM


LOC  OBJ             LINE     SOURCE

                        1     ;Beispiel eines übergeordneten Moduls MOD_AB,
                        2     ;z.Bsp. eines Hauptprogrammes, das von einem
                        3     ;Modul MOD_A Daten bekommt und an ein Modul
                        4     ;MOD_B zur weiteren Bearbeitung übergibt.Die
                        5     ;untergeordneten Module werden über CALL aufgerufen.
                        6     ;VERSION ASM51
                        7     ;================================================
                        8 +1  $TITLE  (Hauptmodul);Name erscheint am Kopf
                        9                         ;jeder Seite des Ausdrucks
                       10     NAME    Mod_AB     ;Name des Moduls
                       11     ;***********************************************
                       12     DataSeg SEGMENT DATA;Interner Datenbereich
----                   13             DSEG AT 50H; Ab Adresse 50H im
                       14                         ;internen RAM definiert
0050                   15     Status: DS 1       ;1 Byte reserviert für
                       16                         ;lokalen Parameter
                       17     ;***********************************************
                       18     DatXSeg SEGMENT XDATA;Externer Datenbereich
                       19     PUBLIC xdat_AB     ;Diese Daten werden in Mod_AB
                       20                         ;definiert
----                   21           XSEG AT 0c000H;Externer Datenbereich
C000                   22     xdat_AB:DS 10     ;Reservierung für 10 Bytes
                       23     EXTRN XDATA(xdat_A);Diese Daten werden im Mod_A
                       24                        ;definiert (ext.Speicher)
                       25     ;***********************************************
                       26     CodeSeg SEGMENT CODE;Bereich für Befehle
                       27     EXTRN   CODE (Mod_A,Mod_B);Adressen von
                       28                               ;Mod_A u. Mod_B
----                   29             CSEG AT 0         ;Beginn des Code-Teils
                       30     ;***********************************************
0030                   31             ORG 30H           ;Start nach den Inter-
                       32                               ;ruptvektoren
0030 120000    F       33     Mod_AB: CALL Mod_A        ;Aufruf von Mod_A
0033 75F00A            34             MOV B,#10         ;Schleifenzähler
0036 900000    F       35     Weiter: MOV DPTR,#xdat_A  ;Zeiger auf Daten
                       36                               ;von Mod_A
0039 E0                37             MOVX A,@DPTR      ;Daten über Akku..
003A 90C000            38             MOV DPTR,#xdat_AB ;Zeiger auf Daten
                       39                               ;von Mod_AB
003D F0                40             MOVX  @DPTR,A     ;..nach xdat_AB
003E D5F0F5            41             DJNZ B,Weiter     ;10 Bytes
0041 120000    F       42             CALL Mod_B        ;Aufruf von Mod_B
0044 7550FF            43             MOV  Status,#0ffH ;auf Addresse Status
                       44                               ;wird 1 Byte geladen
                       45                               ;Selber Name, aber
                       46                               ;andere Adresse als in
                       47                               ;Mod_B
                       48             END

MCS-51 MACRO ASSEMBLER    Hauptmodul                                            07/02/91   PAGE    2

SYMBOL TABLE LISTING
------ ----- -------


N A M E     T Y P E  V A L U E      A T T R I B U T E S

B. . . . .  D ADDR   00F0H   A
CODESEG. .  C SEG    0000H           REL=UNIT
DATASEG. .  D SEG    0000H           REL=UNIT
DATXSEG. .  X SEG    0000H           REL=UNIT
MOD_A. . .  C ADDR   ----      EXT
MOD_AB . .  C ADDR   0030H   A
MOD_B. . .  C ADDR   ----      EXT
STATUS . .  D ADDR   0050H   A
WEITER . .  C ADDR   0036H   A
XDAT_A . .  X ADDR   ----      EXT
XDAT_AB. .  X ADDR   C000H   A PUB


REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

EXTMOD2.LST

```
                          1     ;Beispiel eines untergeordneten Moduls Mod_A,
                          2     ;z.Bsp. eines Unterprogrammes, das an ein
                          3     ;Modul übergeornetes Modul Mod_AB Daten abgibt.
                          4     ;VERSION ASM51
                          5     ;*********************************************
                          6 +1  $TITLE  (Subr.Mod_A) ;Name erscheint am Kopf
                          7                          ;jeder Seite des Ausdrucks
                          8     NAME    Mod_A        ;Name des Moduls
                          9     ;*********************************************
                         10     DatXSeg SEGMENT XDATA;Externer Datenbereich
                         11     PUBLIC xdat_A        ;Diese Daten werden in Mod_AB
                         12                          ;definiert
----                     13             RSEG DatXSeg ;verschieblicher Speicher-
                         14                          ;bereich für Mod_A
0000                     15     xdat_A: DS 10        ;Reservierung für 10 Bytes
                         16     ;*********************************************
                         17     CodeSeg SEGMENT CODE ;Bereich für Befehle
                         18     PUBLIC Mod_A         ;Startadresse für Mod_A
                         19                          ;hier definiert
----                     20             RSEG CodeSeg ;verschieblicher Codebereich
                         21                          ;für Linker
                         22     ;*********************************************
0000 C0E0                23     Mod_A:  PUSH ACC
0002 C0D0                24             PUSH PSW     ;Sichern von Akku u.PSW
0004 C083                25             PUSH DPH
0006 C082                26             PUSH DPL     ;Sichern von DPTR
0008 900000   F          27             MOV DPTR,#xdat_A ;Zeiger auf Daten
                         28     ;Hier stehen weitere Befehle zur Erzeugung der
                         29     ;Daten für das Modul Mod_AB
000B D082                30             POP DPL      ;Zurückholen der
000D D083                31             POP DPH      ;zwischengespeicherten
000F D0D0                32             POP PSW      ;Register
0011 D0E0                33             POP ACC
0013 22                  34             RET
                         35             END

SYMBOL TABLE LISTING
------ ----- -------

N A M E     T Y P E   V A L U E       A T T R I B U T E S

ACC. . . .  D ADDR    00E0H   A
CODESEG. .  C SEG     0014H           REL=UNIT
DATXSEG. .  X SEG     000AH           REL=UNIT
DPH. . . .  D ADDR    0083H   A
DPL. . . .  D ADDR    0082H   A
MOD_A. . .  C ADDR    0000H   R PUB   SEG=CODESEG
PSW. . . .  D ADDR    00D0H   A
XDAT_A . .  X ADDR    0000H   R PUB   SEG=DATXSEG
```

EXTMOD3.LST

```
LOC  OBJ            LINE     SOURCE

                       1     ;Beispiel eines untergeordneten Moduls Mod_B,
                       2     ;z.Bsp. eines Unterprogrammes, das von einem
                       3     ;übergeorneten Modul Mod_AB Daten empf^ngt.
                       4     ;VERSION ASM51
                       5     ;*********************************************
                       6 +1  $TITLE  (Subr.Mod_B) ;Name erscheint am Kopf
                       7                          ;jeder Seite des Ausdrucks
                       8     NAME    Mod_B        ;Name des Moduls
                       9     ;*********************************************
                      10     EXTRN XDATA(xdat_AB) ;Daten definiert in Mod_AB
                      11     DataSeg SEGMENT DATA ;Bereich für internen Daten-
                      12                          ;speicher
----                  13             RSEG DataSeg ;verschieblicher Beginn
0000                  14     Status: DS 1         ;1 Byte-Reservierung für
                      15                          ;lokalen Parameter
                      16     ;*********************************************
                      17     CodeSeg SEGMENT CODE ;Bereich für Befehle
                      18     PUBLIC Mod_B         ;Hier ist die Startadresse
                      19                          ;des Mod_B definiert
----                  20             RSEG CodeSeg ;Verschieblich für Linker
                      21     ;*********************************************
0000 C0E0             22     Mod_B:  PUSH ACC
0002 C0D0             23             PUSH PSW     ;Sichern von Akku u.PSW
0004 C083             24             PUSH DPH
0006 C082             25             PUSH DPL     ;Sichern von DPTR
0008 900000   F       26             MOV DPTR,#xdat_AB ;Zeiger auf Daten
                      27     ;Hier stehen weitere Befehle zur Verarbeitung
                      28     ;der von Mod_AB empfangenen Daten
000B 750055   F       29             MOV Status,#01010101B;nicht identisch mit
                      30                          ;Status in Mod_AB
                      31                          ;Adresse Status
000E D082             32             POP DPL      ;Zurückholen der
0010 D083             33             POP DPH      ;zwischengespeicherten
0012 D0D0             34             POP PSW      ;Register
0014 D0E0             35             POP ACC
0016 22               36             RET
                      37             END

SYMBOL TABLE LISTING
------ ----- -------

N A M E     T Y P E   V A L U E       A T T R I B U T E S

ACC. . . .  D ADDR    00E0H   A
CODESEG. .  C SEG     0017H           REL=UNIT
DATASEG. .  D SEG     0001H           REL=UNIT
DPH. . . .  D ADDR    0083H   A
DPL. . . .  D ADDR    0082H   A
MOD_B. . .  C ADDR    0000H   R PUB   SEG=CODESEG
PSW. . . .  D ADDR    00D0H   A
STATUS . .  D ADDR    0000H   R       SEG=DATASEG
XDAT_AB. .  X ADDR    ----      EXT
```

Assemblieren mit X8051 und LINK (mit LIB) von 2500AD

Da Assembler und Linker von 2500AD aus einem C-Compilerpaket kommen, kann besonders gut mit Hilfe von Objektprogramm-Bibliotheken gearbeitet werden. Die Anpassung an den individuellen Prozessor aus der 8051-Familie erscheint mir bei dieser Software etwas schwieriger. Grundsätzlich gibt es die Möglichkeit mit Assembler und Linker ähnlich wie bei Intel zu arbeiten. Vorsicht: MODULE- und ENDMOD-Direktiven veranlassen den Assembler nämlich, ein Objektfile in gepckter Form `*.pak` zu erstellen, das nur vom Librarian, dem Bibliotheksverwaltungsprogramm, gelesen werden kann. Die Listings der drei Module zeigen diese Direktiven mit vorangestellten ";". Absolute Adressangaben werden erst beim Linkerlauf eingegeben.

Zur Assemblierung wird die Kommandozeile

```
X8051 extmodxa -d
```

eingegeben (`-d` für Listing auf Disk). Aus `extmodxa.asm` entstehen `extmodxa.obj,extmodxa.lst`.

Nach Aufruf von LINK wird nach den Objektdateien (Inputfiles) und den zu ihren Speicherbereichen gehörigen Adressoffsets gefragt. Die Optionen des Linkers werden D und für den Test mit dem AVSIM51 H heißen (D für Linker-map, H für Intel-Hex-Format). Der Linker produziert daher ein `extmod1a.hex` und ein `extmod1a.map`.

EXTMOD1A.MAP

```
Global Symbol Name  Global Value Global Filename

Mod_A               0047         extmod2a.obj
Mod_B               005B         extmod3a.obj
xdat_A              C00A         extmod2a.obj
xdat_AB             C000         extmod1a.obj
*************************************************
*          L O A D      M A P                   *
*************************************************
*    Section Name   Startad.  Endad.   Size     *
*************************************************
* extmod1a.obj                                  *
*    CODE           0030      0046     0017     *
* extmod2a.obj                                  *
*    CODE           0047      005A     0014     *
* extmod3a.obj                                  *
*    CODE           005B      0071     0017     *
*************************************************
Linker Output Filename : extmod1a.hex
Disk Listing  Filename : extmod1a.map
Symbol Table  Filename : extmod1a.sym /2500 A.D.
Link Errors :  0      Output Format :  Intel Hex
```

Das Intel-Hex-File extmod1a.hex kann dann direkt in das Simulatorprogramm AVSIM51 geladen werden.

Wenn man das 2500AD Entwicklungssystem noch besser nützen will, kann das Bibliotheksverwaltungsprogramm LIB benützt werden. Bedienung ganz einfach: Zuerst Quelldateien mit Moduldefinition (MODULE und ENDMOD-Direktiven) erzeugen, Assemblieren. Nun entstehen statt `*.obj` `*.pak` Dateien. LIB aufrufen. Mit NEW neue Bibliotheksdatei erzeugen, mit ADD Module aus den `*.pak` Dateien eintragen. Eventuell mit LIST und STAT die Modulparameter anschauen. Mit `j` und `k` kann man durch die Modulliste scrollen. Mit QUITT oder EXIT aus LIB aussteigen. Im Linker kann nun die Bibliotheksdatei vor dem Linken angegeben werden. Referenzen werden automatisch aufgelöst.

EXTMOD1A.LST

```
 1                                .TITLE  Hauptmodul
 2                                        ;Name erscheint am Kopf
 3                                        ;jeder Seite des Ausdrucks
 4                                .COMMENT ;Beispiel eines übergeordneten Moduls
 5                                MOD_AB,z.Bsp. eines Hauptprogrammes, das von einem
 6                                Modul MOD_A Daten bekommt und an ein Modul
 7                                MOD_B zur weiteren Bearbeitung übergibt.Die
 8                                untergeordneten Module werden über CALL aufgerufen.
 9                                VERSION X8051;
11
12                                ;.MODULE  Mod_AB    ;Modulbestimmung für die
13                                                    ;Verwendung in "Librarian"-
14                                                    ;Bibliothekverwaltungsprg.
15                                ;*********************************************
16  0000                                  DATA        ;Datenbereich
17  0050                                  ORG 50H     ;Start ab 50H, weitere
18                                                    ;Offset kann beim Linken angegeben werden
20  0050                          Status: DS 1        ;1 Byte reserviert für lokalen Parameter
22                                ;*********************************************
23  0000                          DatXSeg SECTION     ;selbstdef. Speicherbereich
24                                                    ;hier Daten, eingebettet in den
25                                                    ;vordefinierten Bereich "DATA"
26                                .PUBLIC xdat_AB      ;Diese Daten werden in Mod_AB
27                                                    ;definiert, die Offset 0C000H
28                                                    ;wird beim Linken angegeben
29  0000                          xdat_AB:DS 10       ;Reservierung für 10 Bytes
30                                .EXTERN  xdat_A     ;Diese Daten werden im Mod_A
31                                                    ;definiert (ext.Speicher)
32                                        ENDS        ;Ende des selbstdef. Speicher-
33                                                    ;bereiches
34                                ;*********************************************
35  0000                                  CODE        ;vordef. Bereich für Befehle
36                                .EXTERN  Mod_A,Mod_B
37                                                    ;Adressen von Mod_A u. Mod_B
38                                ;*********************************************
39  0030                                  ORG 30H             ;Start nach den Inter-
40                                                            ;ruptvektoren
41  0030  12 00 00                Mod_AB: CALL Mod_A          ;Aufruf von Mod_A
42  0033  75 F0 0A                        MOV B,#10           ;Schleifenzähler
43  0036  90 00 00                Weiter: MOV DPTR,#xdat_A    ;Zeiger auf Daten
44                                                            ;von Mod_A
45  0039  E0                              MOVX A,@DPTR        ;Daten über Akku..
46  003A  90 00 00                        MOV DPTR,#xdat_AB   ;Zeiger auf Daten von Mod_AB
48  003D  F0                              MOVX  @DPTR,A       ;..nach xdat_AB
49  003E  D5 F0 F5                        DJNZ B,Weiter       ;10 Bytes
50  0041  12 00 00                        CALL Mod_B          ;Aufruf von Mod_B
51  0044  75 50 FF                        MOV  <Status,#0ffH ;auf Addresse Status
52                                                            ;wird 1 Byte geladen
53                                                            ;Selber Name, aber
54                                                            ;andere Adresse als in
55                                                            ;Mod_B
56                                ;       ENDMOD
57  0047                                  END
```

EXTMOD2A.LST

```
1                                       .TITLE  Subr.Mod_A
2                                               ;Name erscheint am Kopf
3                                               ;jeder Seite des Ausdrucks
4                                       ;Beispiel eines untergeordneten Moduls Mod_A,
5                                       ;z.Bsp. eines Unterprogrammes, das an ein
6                                       ;Modul übergeordnetes Modul Mod_AB Daten abgibt.
7                                       ;VERSION X8051
8                                       ;=============================================
9
10                                      ;.MODULE Mod_A          ;Modulbestimmung, für die
11                                                              ;Verwendung im "Librarian"-
12                                                              ;Bibliotheksverwaltungsprg.
13                                      ;*********************************************
14  0000                                        DATA            ;Datenbereich
15  0000                                DatXSeg SECTION         ;selbstdefinierter
16                                                              ;Datenbereich, Speicherfest-
17                                                              ;legung wird beim Linken an-
18                                                              ;gegeben
19                                      .PUBLIC xdat_A          ;Diese Daten werden in Mod_A
20                                                              ;definiert
21  0000                                xdat_A: DS 10           ;Reservierung für 10 Bytes
22                                              ENDS            ;Ende des selbstdefinierten
23                                                              ;Datenbereiches
24                                      ;*********************************************
25  0000                                        CODE            ;Bereich für Befehle
26                                      .PUBLIC Mod_A           ;Startadresse für Mod_A
27                                                              ;wird beim Linken angegeben
28                                      ;*********************************************
29  0000  C0 E0                         Mod_A:  PUSH ACC
30  0002  C0 D0                                 PUSH PSW        ;Sichern von Akku u.PSW
31  0004  C0 83                                 PUSH DPH
32  0006  C0 82                                 PUSH DPL        ;Sichern von DPTR
33  0008  90 00 00                              MOV DPTR,#xdat_A ;Zeiger auf Daten
34                                      ;Hier stehen weitere Befehle zur Erzeugung der
35                                      ;Daten für das Modul Mod_AB
36  000B  D0 82                                 POP DPL         ;Zurückholen der
37  000D  D0 83                                 POP DPH         ;zwischengespeicherten
38  000F  D0 D0                                 POP PSW         ;Register
39  0011  D0 E0                                 POP ACC
40  0013  22                                    RET
41                                              ;ENDMOD
42  0014                                        END
```

EXTMOD3A.LST

```
1                                       .TITLE  Subr.Mod_B
2                                               ;Name erscheint am Kopf
3                                               ;jeder Seite des Ausdrucks
4                                       .COMMENT ;Beispiel eines untergeordneten
5                                       Moduls Mod_B,z.Bsp. eines Unterprogrammes,
6                                       das von einem übergeordneten Modul Mod_AB
7                                       Daten empfängt.
8                                       VERSION ASM51
9                                       ============================================;
10                                      ;.MODULE Mod_B          ;Modulbestimmung für die
11                                                              ;Verwendung im "Librarian"
12                                                              ;Bibliotheksverwaltungsprg.
13                                      ;*********************************************
14  0000                                        DATA            ;vordefinierter Speicher-
15                                                              ;bereich,Adressfestlegung
16                                                              ;beim Linken
17  0000                                DatXSeg SECTION         ;selbstdef. Datenbereich
18                                                              ;eingebettet in den vordef.
19                                                              ;Speicherbereich "DATA",
20                                                              ;Adressfestlegung beim
21                                                              ;Linken
22                                      .EXTERN  xdat_AB         ;Daten definiert in Mod_AB
23                                              ENDS            ;Ende des selbstdefinierten
24                                                              ;Speicherbereichs
25  0000                                Status: DS 1            ;1 Byte-Reservierung für
26                                                              ;lokalen Parameter im
27                                                              ;vordefinierten Speicherber.
28                                                              ;"DATA"
29                                      ;*********************************************
30  0000                                        CODE            ;vordef.Bereich für Befehle
31                                                              ;Offset wird beim Linken an-
32                                                              ;gegeben
33                                      .PUBLIC Mod_B            ;Hier ist die Startadresse
34                                                              ;des Mod_B definiert
35                                      ;*********************************************
36  0000  C0 E0                         Mod_B:  PUSH ACC
37  0002  C0 D0                                 PUSH PSW        ;Sichern von Akku u.PSW
38  0004  C0 83                                 PUSH DPH
39  0006  C0 82                                 PUSH DPL        ;Sichern von DPTR
40  0008  90 00 00                              MOV DPTR,#xdat_AB ;Zeiger auf Daten
41                                      ;Hier stehen weitere Befehle zur Verarbeitung
42                                      ;der von Mod_AB empfangenen Daten
43  000B  75 00 55                              MOV <Status,#01010101B
44                                                              ;nicht identisch mit
45                                                              ;Status in Mod_AB
46  000E  D0 82                                 POP DPL         ;Zurückholen der
47  0010  D0 83                                 POP DPH         ;zwischengespeicherten
48  0012  D0 D0                                 POP PSW         ;Register
49  0014  D0 E0                                 POP ACC
50  0016  22                                    RET
51                                              ;ENDMOD
52  0017                                        END
```

Die Bedienung des Assemblers ASM51

Wolfgang SCHARL, TGM, NT/EL

Für viele, die sich im Zuge des Unterrichts oder privat im Rahmen der Clubaktion mit dem 8051 Controller auseinandersetzen, stellt sich natürlich das Problem, selbstgeschriebene Programme zu assemblieren und zu linken. In diesem Zusammenhang haben sich einige immer wiederkehrende Fragen herauskristallisiert, die ich an Hand eines kleinen Beispiels - speziell für Anfänger - darstellen möchte.

Grundsätzlich seien Syntax und Befehlsvorrat des ASM51 hier einmal vorausgesetzt (Mnemonics 8051: TGM-LIT-003, Assemblerdirektiven: in diesem Heft). Geschrieben werden Assemblerprogramme mit irgend einem Editor, der unformatierte ASCII Texte erstellen kann. Von Sidekick bis zum Textverarbeitungssystem ist so ziemlich alles, mehr oder weniger komfortabel, verwendbar. Bei Textsystemen wie Word muß darauf geachtet werden, daß die Textdatei "unformatiert" abgespeichert wird.

Eine derart erstellte Textdatei hat am Ende grundsätzlich ein ^Z als "end of file"-Markierung. Die irritiert den Assembler, und er reagiert mit einer Fehlermeldung. Das ist an sich belanglos, wenn Sie das stört, können Sie mit einem kleinen Hilfsprogramm (EOF-Killer) das ^Z aus der Datei herauslöschen. [Beim Editieren mit WORD wird kein ^Z an die Datei angefügt, es kann aber bei Bedarf mit ALT+NUM2,NUM6 generiert werden.]

Als Beispiel möchte ich das Demoprogramm TEST1.ASM, das auf der Diskette "µPROFI-Grundpaket" enthalten ist, verwenden. Dieses Programm ist auf der Diskette auf eine falsche Adresse gelinkt und läuft daher auch nicht.

TEST1.ASM

```
NAME    Test1
C_Test1 SEGMENT CODE
        RSEG C_Test1
Start:  mov     a, #01h         ;ACC lowest Bit setzen
        clrc    c               ;Carry löschen
Loop:   rlc     a               ;Accu Bit nach links rotieren
        jnc     Loop            ;solange nach Loop bis Carry
1
Ende:   clr     c               ;Carry löschen
END
```

Für den µPROFI ist es erforderlich, die Programme im Downloadbereich auf der Adresse a000h beginnen zu lassen. Dies kann mit dem Befehl ORG 0a000h am Beginn des Programmcodes erfolgen. Diese Methode hat aber den Nachteil, daß ein Verschieben des Progammes im Adreßbereich, sowie das Zusammenlinken mehrerer Programmodule nicht möglich ist. Man sollte daher stets einen Segmentcode definieren und die Startadresse dann beim Linken festlegen.

Der erste Schritt ist jetzt das Assemblieren des Programmes mit folgender Eingabe:

```
asm51 test1.asm db ep(test1.err)
```

Die einzelnen Teile dieser Kommandozeile haben folgende Bedeutung:

`asm51`	ruft den Assembler auf, der natürlich im aktuellen Verzeichnis oder im Suchpfad zu finden sein muß.
`test1.asm`	ist der Name des Sourcecodes unseres Programms.
`db`	weist den Assembler an, ein Symbolfile zu erstellen. Mit diesem Symbolfile können Sie später im FSD51 die Labels Ihres Programmes einblenden (so Sie welche verwendet haben).
`ep(test1.err)`	weist den Assembler an, ein Errorfile mit dem Namen TEST1.ERR zu erstellen. In diesem steht nur der Sourcecode mit ev. Fehlern. Alles andere wird unterdrückt. Sehr hilfreich zur Fehlersuche, besonders bei längeren Programmen.

Nun sollten zusätzlich zur Datei TEST1.ASM folgende Dateien entstanden sein:

TEST1.OBJ

TEST1.OBJ ist als Textdatei nicht darstellbar.

TEST1.ERR

```
DOS 3.30 (033-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN TEST1.OBJ
ASSEMBLER INVOKED BY:  ASM51.EXE TEST1.ASM DB EP(TEST1.ERR)

LOC  OBJ             LINE     SOURCE
REGISTER BANK(S) USED: 0
ASSEMBLY COMPLETE, NO ERRORS FOUND
```

TEST1.LST

```
MCS-51 MACRO ASSEMBLER    TEST1                                  03/21/91   PAGE    1
DOS 3.30 (033-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN TEST1.OBJ
ASSEMBLER INVOKED BY:  ASM51.EXE TEST1.ASM DB EP(TEST1.ERR)

LOC  OBJ             LINE     SOURCE
                        1     NAME    Test1
                        2
                        3     C_Test1 SEGMENT CODE
----                    4             RSEG C_Test1
                        5
                        6
0000 7401               7     Start:  mov     a, #01h         ;ACC lowest Bit setzen
0002 C3                 8             clr     c               ;Carry l^schen
0003 33                 9     Loop:   rlc     a               ;Accu Bit nach links rotieren
0004 50FD              10             jnc     Loop            ;solange nach Loop bis Carry 1
0006 C3                11     Ende:   clr     c               ;Carry l^schen
                       12
                       13     END
```

```
MCS-51 MACRO ASSEMBLER    TEST1                                  03/21/91   PAGE    2
SYMBOL TABLE LISTING
------ ----- -------
N A M E     T Y P E   V A L U E      A T T R I B U T E S
C_TEST1. .  C SEG     0007H          REL=UNIT
ENDE . . .  C ADDR    0006H   R      SEG=C_TEST1
LOOP . . .  C ADDR    0003H   R      SEG=C_TEST1
START. . .  C ADDR    0000H   R      SEG=C_TEST1
TEST1. . .    ----    ----
REGISTER BANK(S) USED: 0
ASSEMBLY COMPLETE, NO ERRORS FOUND
```

Als Nächstes wird auf die richtige Adresse gelinkt. Die Startadresse für den µPROFI ist a000h . Es ist dabei zu beachten, daß Hexzahlen immer mit einer Ziffer beginnen müssen, sonst werden sie von Assembler und Linker nicht erkannt (daher 0a000h statt a000h). Die Kommandozeile für den Linker sieht daher so aus:

```
rl51 test1.obj to test1.bin code(c_test1(0a000h))
```

Nun sind folgende Dateien entstanden:

TEST1.BIN

TEST1.BIN ist wieder als Textdatei nicht darstellbar. Weiters entsteht die Symboldatei:

TEST1.M51

```
RL51                                                    PAGE   1
DATE : 03/21/91
DOS 3.30 (033-N)
MCS-51 RELOCATOR AND LINKER V3.0, INVOKED BY:
RL51.EXE TEST1.OBJ TO TEST1.BIN CODE(C_TEST1(0A000H))
INPUT MODULES INCLUDED
  TEST1.OBJ(TEST1)

LINK MAP FOR TEST1.BIN(TEST1)
TYPE    BASE      LENGTH    RELOCATION   SEGMENT NAME
----    ----      ------    ----------   ------------
REG     0000H     0008H                  "REG BANK 0"
        0000H     A000H                  *** GAP ***
CODE    A000H     0007H     UNIT         C_TEST1

SYMBOL TABLE FOR TEST1.BIN(TEST1)
VALUE           TYPE            NAME
-----           ----            ----
-------         MODULE          TEST1
C:A000H         SEGMENT         C_TEST1
C:A006H         SYMBOL          ENDE
C:A003H         SYMBOL          LOOP
C:A000H         SYMBOL          START
-------         ENDMOD          TEST1
```

Die Binärdatei TEST1.BIN könnte nun so wie sie ist in ein Eprom geschrieben werden und müßte funktionieren. Die meisten Programmiergeräte und auch unser FSD51 erwarten diese Datei aber in einem besonderen, fehlersicheren Format genannt "Intel-Hex". Dieses Format wird mit dem Programm OH erzeugt:

```
oh test1.bin
```

es entsteht nun ein

TEST1.HEX

```
:07A000007401C33350FDC3DE
:00000001FF
```

Diese Datei kann nun mit dem FSD51 in den µPROFI geladen und mit Singlestep oder Run ausgeführt werden. Wird auch das Symbolfile TEST1.M51 in den FSD51 geladen, so erscheinen zur besseren Übersicht auch die Labels am Bildschirm.

Die weitere Vorgangsweise ist im Benutzerhandbuch des µPROFI51 ab Seite 25 ausführlich dargestellt.

Zusammenfassung

zum Assemblieren und Linken des Programmes TEST1.ASM sind folgende Eingaben erforderlich:

```
asm51 test1.asm db ep(test1.err)

rl51 test1.obj to test1.bin code(c_test1(0a000h))

oh test1.bin
```

Das erste Gesetz des wissenschaftlichen Fortschritts

Variante 1: Der Fortschritt einer Wissenschaft ist daran zu messen, in welchem Tempo sich Ausnahmen zu früheren Regeln ansammeln.
Variante 2: Ausnahmen sind grundsätzlich zahlreicher als Regeln.
Variante 3: Von allen anerkannten Ausnahmen gibt es Ausnahmen.
Variante 4: Wenn man die Ausnahmen endlich im Griff hat, erinnert sich niemand mehr der Regeln, für die sie angeblich gelten.

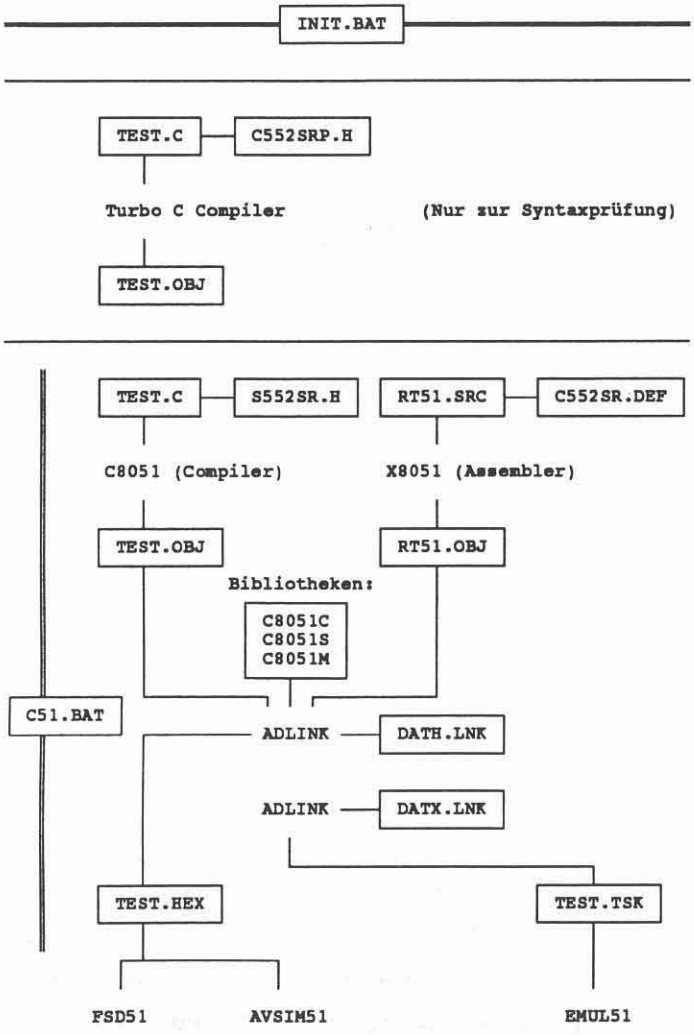
ENTWICKLUNGSSYSTEM FÜR 80C552

Gregor POKORNY,TGM,NT/EL,0SL90 TGM-DSK-169:\C*.*

TGM-interne Details sind kursiv gedruckt.

Ziel dieser Zusammenstellung war es, verschiedene, schon bestehende Softwarepakete, so miteinander zu verbinden, daß ein Entwickeln und Testen von C und Assemblerprogrammen für den 552 möglichst einfach und zweckmäßig wird. Die Programme im einzelnen: Compiler **C8051**, Assembler **X8051**, Linker **LINK**, Simulator **AVSIM51**, Debugger **FSD51**, Emulator **EMUL51**. Eine Anpassung an andere Mikocontrollertypen ist möglich.

Diagramm



Erläuterung zum Diagramm

Bevor man mit dem C-Compiler arbeiten kann, müssen am PC einige Dinge eingestellt werden. Diese Aufgabe erledigt INIT.BAT.

Das Diagramm zeigt die Entstehung eines Programms für den 552. Zuerst wird ein C-Programm geschrieben, in welches das File C552SR.H inkludiert werden muß, damit die Namen aller im 552 befindlichen Register im C-Programm bekannt sind. Dieses C-Programm schreibt man am besten im Turbo C++-Editor, da dieser Syntax durch Compilieren in ein Objektfile prüft, bevor eine Compilierung mit dem Compiler für den 552 durchgeführt wird. Diese Syntaxprüfung ist sinnvoll, weil der C-Compiler wesentlich schneller als der 552-Compiler ist und, weil manche Fehler vom 552-Compiler nicht erkannt werden, und die Gefahr besteht, daß das Programm dann nicht ordnungsgemäß läuft. Allerdings muß für diese Compilierung das File C552SRP.H inkludiert werden, da der Turbo-C-Compiler z.B. das Wort NEAR nicht versteht. (Dieses Wort sagt dem 552-Compiler, daß diese Variable im internen RAM abgelegt werden soll.) Wenn man vergißt diese Änderung wieder rückgängig zu machen bevor man den 552 Compiler startet, dann wird z.B. statt einer Ausgabe der PWM auf den entsprechenden Port, dieser Wert irgendwo im externen RAM abgelegt, wo er natürlich keine Wirkung hat. Beim Compilieren entsteht ein File mit der Erweiterung .OBJ.

Damit ein C-Programm am 552 ordnungsgemäß laufen kann, ist das Assemblerprogramm RT51.SRC notwendig, das einige Vorbereitungen trifft. In diesem Programm werden hauptsächlich Variablen definiert und initialisiert. Zum Definieren der Variablen wird das File C552SR.DEF in RT51.SRC eingebunden. Nach dem Assemblieren mit X8051.EXE entsteht ebenfalls ein File mit der Erweiterung OBJ. Alle diese OBJ-Files werden danach mit dem Linker verbunden. Zusätzlich können auch noch Programmbibliotheken eingebunden werden. Der Linker braucht dazu einige Informationen, die entweder händisch eingegeben, oder als File angebunden werden können. Hier können auch Optionen eingegeben werden. Diese sind unter anderem auch dafür verantwortlich, welches Format das Outputfile haben wird. Hier ist es vorgesehen, daß diese Informationen angebunden werden (DATH.LNK und DATX.LNK). Da für die verschiedenen Testmöglichkeiten verschiedene Outputfiles benötigt werden, muß der Linker zweimal gestartet werden. Einmal, um ein Outputfile im Intel-Hex-Format, und einmal, um ein binäres File zu erzeugen.

Der AVSIM51 braucht zum Beispiel ein File im Intel-Hex-Format. Die genaue Bedienung des AVSIM51 ist dem gleichnamigen Beitrag weiter hinten zu entnehmen. Der AVSIM51 ist allerdings mit Vorsicht zu genießen, da sich manche Dinge zwar simulieren lassen, aber dann in Wirklichkeit doch nicht funktionieren. Leider war es mir nicht möglich einen solchen Fehler genau zu lokalisieren, da die C-Programme nach dem Compilieren nicht mehr gut nachvollziehbar sind.

Der FSD51 verlangt auch ein File im Intel-Hex-Format. Außerdem ist zum Testen ein Prozessorprint nötig. Das hat den Vorteil, daß es sich dabei um keine Simulation handelt, hat aber den Nachteil, daß man sich auch öfter mit Hardwareproblemen herumschlagen muß.

Der EMUL51 braucht ein File im BIN-Format. Er ist ein sehr komplexes Hard- und Softwarepaket, und bietet neben der besseren Emulation auch die Möglichkeit, die serielle Schnittstelle zu benutzen, was beim FSD51 nur bedingt möglich ist, da er diese selbst zur Datenübertragung mit dem PC braucht. Außerdem können damit auch Programme ohne Prozessorprint getestet werden. Der einzige Nachteil ist, daß man eine umfangreiche, englische Anleitung zu studieren ist.

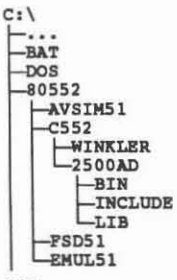
Installation

Die Programme sind samt Unterverzeichnissen mit backup *aus DOS 3.30 auf HD Disketten abgespeichert, und können mit* restore a: c:/s *wieder auf der Festplatte C: installiert werden. Es werden dann alle Verzeichnisse wiederhergestellt. Sollten die Programme auf einem anderen Laufwerk installiert werden,*

so muß diese Änderung in den Batchdateien berücksichtigt werden.

Weiters muß die Datei SETPATH geschrieben werden und, z.B. im Verzeichnis BAT, abgelegt werden. In AUTOEXEC.BAT sollte dann auch statt der Angabe eines Pfades die Datei SETPATH aufgerufen werden. (Siehe auch Erläuterung zu SETPATH und Ausdruck von SETPATH). Außerdem muß ein virtuelles Laufwerk mit dem Namen e: installiert werden. Um C-Programme zweckmäßig editieren zu können, ist es auch zweckmäßig Turbo-C zu installieren.

Systemkonfiguration




```
C:\80552:                       INIT.BAT

C:\80552\AVSIM51:               AVSIM51.EXE, AVSIM51.HLP,
                                AVSIM51.OVR, AVSIM51.REG, GMENU.COM,
                                M.MNU, SIM.BAT, SIM.CMD, SIM.SYM

C:\80552\C552:                  C51.BAT, C552SR.DEF, C552SR.H,
                                C552SRP.H, DATH.LNK, DATX.LNK,
                                RT51.OBJ, RT51.SRC, TEST.C,
                                TEST.HEX, TEST.MAP, TEST.OBJ,
                                TEST.SYM, TEST.TSK

C:\80552\C552\WINKLER:          AADC.C, AADCG.C, CADC.C, DADC.C,
                                FIRST.C, MADC.C, PIOLD.C, PIREG.C,
                                PIREG_L.C, PIREG_R.C, PIREG_RL.C,
                                RADC.C, TADC.C, TADCG.C, TESTADC.C,
                                TM2.C, TM2GUT.C, VADC.C

C:\80552\C552\2500AD\BIN:       ADLINK.EXE, C8051.EXE, C8051CG.EXE,
                                C8051P1.EXE, C8051PA.EXE, LIB.EXE,
                                MP.EXE, X8051.EXE.

C:\80552\C552\2500AD\INCLUDE:     C8051FI.SRC, C8051IFI.SRC,
                                C8051RT.SRC, C8051IIO.H, C8051IN.H,
                                C8051IO.H, C8051SR.H, ERROR.H,
                                MATH.H.

C:\80552\C552\2500AD\LIB:       C8051C.SRC, C8051IC.SRC,
                                C8051C.LIB, C8051IC.LIB,
                                C8051IM.LIB, C8051IP.LIB,
                                C8051IS.LIB, C8051M.LIB, C8051P.LIB,
                                C8051S.LIB.

C:\80552\FSD51:                 FSD51.EXE, GESCHW.M51, TEST.SET.

C:\80552\EMUL51:                80552.STR, 80552.SYM, EM.BAT,
                                EMUL31.HLP, EMUL31.NDX, EMUL31.EXE,
                                I751, PRIDOC.EXE, R751, SHOW51.COM,
                                TEST, TEST.A03, TEST.C, TEST.SRC,
                                TEST1.C.
```

FSD51

Einleitung

Es handelt sich bei dieser Dokumentation um eine Kurzbeschreibung des FSD51 zum Testen eines C-Programms. Da ein C-Programm nach dem Compilieren ohnedies als unlesbar zu bezeichnen ist, wird in diesem Fall der FSD51 im Prinzip nur zum Starten des Programms verwendet. Wichtig ist auch noch, da es sich bei dieser Version des FSD51 um eine spezielle Version handelt, die seinerzeit im Zusammenhang mit dem Robbie entwickelt wurde. Außerdem ist das dazugehörige EPROM nur auf dem alten 100-er-Print verwendbar, da einige Adreßleitungen ausgekreuzt sind. Da momentan noch kein anderer Print zu haben ist, wird eben dieser Print noch immer zu Testzwecken verwendet. Sollte ein neuer Prozessorprint fertiggestellt werden, sollte man sich auch eine neue Version des FSD51 mit dem dazugehörigen EPROM besorgen.

Hardware

Die Hardware besteht aus dem Prozessorprint, und einem Pegelwandler für die serielle Schnittstelle. Der Prozessorprint muß mit 5V versorgt werden und über den Pegelwandler an eine der seriellen Schnittstellen des PC's angeschlossen werden.

Software

Der FSD51 wird mit FSD51 TEST.SET gestartet. <spc> drücken, um den Begrüßungsbildschirm zu verlassen. Danach kommt der Hauptbildschirm, in dem immer ein Fenster heller erscheint. Ein Wechsel der Fenster ist mit <home> oder <end> möglich. Jetzt bestehen zwei Möglichkeiten: Entweder Der FSD übertrgt brav interne und externe Daten oder die Schnittstelle ist falsch eingestellt, und der FSD meldet sich mit `Empfänger nicht bereit` In diesem Fall kann mit F3 Setup die Schnittstelle geändert werden. Es empfiehlt sich auch das Setup gleich zu speichern, da sonst derselbe Fehler beim nächsten Mal wieder auftritt.

Laden eines Programms

Mit F2 File, und File download wählen. Dann erscheint der Name, der zuletzt eingegeben wurde, oder der Name, der im Setup abgespeichert war. Beim Speichern des Setups wird nämlich der Name, der hier gerade steht mitgespeichert, was den Vorteil hat, daß man beim Laden des Programms zumeist nur den Menüpunkt anwählt und <Enter> betätigt. Jetzt wird das Programm ins RAM des Prozessorprints geladen, und interne und externe Daten übertragen.

Starten des Programms

Zuerst muß der PC an die richtige Stelle gesetzt werden. Das geschieht im Fenster links oben. Die Adresse muß mit <Enter> bestätigt werden. Jetzt kann das Programm mit F6 gestartet werden. Unterbrochen kann es jetzt nur noch mit einem Hardwarereset werden, da es vollkommen unabhängig vom PC läuft. Man könnte sogar die serielle Schnittstelle abstecken, ohne daß das Programm in seinem Ablauf beeinträchtigt würde.

Verlassen des FSD

Der FSD kann mit F10 wieder verlassen werden. Das Programm am Prozessorprint läuft aber weiter.

Beschreibung der Files

INIT.BAT

```
Aufruf: INIT


call setpath
e:
md c552
cd c552
xcopy c:\80552\c552\*.* e:
set emul31=c:\80552\emul51
set include=e:\c552
set lib=c:\80552\c552\2500ad\lib
```

In diesem File werden Vorbereitungen für die Bearbeitung eines Programms getroffen. Voraussetzung für die Funktion ist, daß zuvor ein virtuelles Laufwerk E: installiert wurde. Zuerst wird mit der BAT-Datei SETPATH der Pfad wiederhergestellt, falls er, zum Beispiel durch einen Netzeinstieg, verändert worden ist. Dann wird auf E: ein Verzeichnis mit dem Namen \C552 angelegt, in das alle zu bearbeitenden Files kopiert werden. Dieser Aufwand ist deshalb gerechtfertigt, weil der C-Compiler eine sehr langsame Angelegenheit ist, die damit etwas beschleunigt wird. Am Schluß werden noch drei Pfadangaben ins Enviroment geschrieben, damit Compiler und Linker wissen, wo sie Include- und Lib-Dateien suchen sollen, und damit der EMUL51 funktioniert.

SETPATH.BAT

```
PATH=C:\BAT;C:\DOS;D:\ASM51;C:\80552\c552\2500AD\BIN;C:\TOOLS
;C:\TOOLS\NU;c:\tc\bin
```

Diese Datei setzt den Pfad. Sie wird von INIT.BAT aufgerufen und ist so gedacht, daß sie ganz allgemein verwendet wird, z.B. auch von AUTOEXEC.BAT. Wichtig für diese Software ist nur, daß dabei der Pfad C:\80552\C552\2500AD\BIN vorkommt. SETPATH.BAT ist dafür nur ein Beispiel.

C51.BAT

```
Aufruf: C51 test


e:
cd\c552
del %1.obj
c8051 %1
adlink dath
adlink datx
xcopy e:\c552\%1.* c:\80552\c552
```

Diese Datei löscht zuerst das OBJ-File, sonst würden bei nicht erfolgreicher Kompilierung die alten Files zusammengelinkt, was man dann leicht übersieht. Dann wird der Compiler gestartet (und hinterher zweimal der Linker), um ein HEX-File (für FSD51 und AVSIM51) und ein TSK-File (für EMUL51) zu erzeugen. Am Schluß werden noch alle Dateien, die mit dem Projekt zu tun haben auf die Festplatte zurückkopiert.

TEST.C

```
#include "c552sr.h"

extern struct special_function_bits SFB;
   int a=1;

main()
{
  while (1)
  {
  SFB.P1_5 = ~SFB.P1_5;
  }
}
```

TEST.C ist das eigentliche C-Programm (siehe Diagramm-Beschreibung). Damit man die Special-Function-Register bitadressieren kann, muß das `struct special_function_bits SFB` extern deklariert werden. Danach steht das eigentliche C-Programm für den 552. Das hier abgedruckte Programm invertiert nur ein Portbit, was beim Testen sehr schön am Oszi zu beobachten ist, und in jedem Programm zum Testen zweckmäßig ist.

TEST.HEX

```
:02A22500000136
:108C0000EE24FCFEEF34FFFF79007401C39970037A
:108C1000028C25E430950104F420E004C295800222
:108C2000D295028C08F5F0EE2404FEEF3400FFE547
:028C3000F02230
:0FA2270001A238A23800000000A338A3380000BD
:03A0000002A103B7
:03A0530002A20165
:03A0730002A21036
:10A1000002A18E7E7F7FFF743675F0A290A236C3C7
:10A110009582F8E5F09583F94860387436C3943633
:10A12000FA74A294A2FBEA2436FAEB34A2FB7C0078
:10A130007D808A828B83E493A3AA82AB838C828DF9
:10A1400083F0A3AC82AD83E8C39401F8E99400F9ED
:10A150004870DF743875F0A490A236C39582F8E594
:10A16000F09583F948600FE4F0A3E8C39401F8E99F
:10A170009400F94870F190A236747FC39400FE7485
:0EA18000FF9400FFF0A3EEF0128C0002A0008E
:10A18E0022C0D0C0E0E58124FDC0F0C082C083C0F3
:10A19E0000C001C002C003C004C005C006C007F85D
:10A1AE0090A236E0C0E0A3E0C0E0EEC39400FEF063
:10A1BE00EF9400FF90A236F0E6C0E008E6C0E02281
:10A1CE00EE2400FEEF3400FF90A236D0F0D0E0F087
:10A1DE00A3E5F0F0D007D006D005D004D003D0020E
:10A1EE00D001D000D083D082D0F0D0E0D0D0158175
:10A1FE00158132C0D0C0E053C5EF432480D0E0D0EB
:10A20E00D032C0D0C0E053CB7F53EAEF43C50843F2
:07A21E002401D0E0D0D03292
:02A00003A000BB
:00000001FF
```

Dieses File entsteht, beim Linken, und dient als Inputfile für den AVSIM51 und den FSD51. Am Anfang jeder Zeile steht immer ein Doppelpunkt, danach die Anzahl der Bytes, danach die Adresse, und dann die Daten. Dieses Wissen kann vor allem bei der Suche von Link-errors praktisch sein.

C552SR.H

```
/************************************************************/
/* 8051 C COMPILER SPECIAL FUNCTION REGISTER DECLARATIONS */
/************************************************************/

.asm
.chip 8051
.endasm

extern  near  char  ACC; /* power control       */
extern  near  char  B;   /* power control       */
extern  near  char  SP;  /* power control       */
extern  near  char  DPL; /* power control       */
extern  near  char  DPH; /* power control       */
extern  near  char  PSW; /* program status word */

extern  near  char  P0;     /* port 0               */
extern  near  char  P1;     /* port 1               */
extern  near  char  P2;     /* port 2               */
extern  near  char  P3;     /* port 3               */
extern  near  char  P4;     /* port 4               */
extern  near  char  P5;     /* port 5               */
extern  near  char  ADCON;  /* ADC control          */
extern  near  char  ADCH;   /* ADC high             */
extern  near  char  S0CON;  /* serial control       */
extern  near  char  S0BUF;  /* serial buffer        */
extern  near  char  TMOD;   /* timer mode           */
extern  near  char  TCON;   /* timer control        */
extern  near  char  PCON;   /* power control        */
extern  near  char  TM2CON; /* timer 2 control      */
extern  near  char  CTCON;  /* capture control      */

extern  near  char  S1CON;  /* serial 1 contorl     */
extern  near  char  S1STA;  /* serial 1 status      */
extern  near  char  S1DAT;  /* serial 1 data        */
extern  near  char  S1ADR;  /* serial 1 address     */

extern  near  char  TL0;    /* timer 0 low byte     */
extern  near  char  TH0;    /* timer 0 high byte    */
extern  near  char  TL1;    /* timer 1 low byte     */
extern  near  char  TH1;    /* timer 1 high byte    */
extern  near  char  TML2;   /* timer 2 low byte     */
extern  near  char  TMH2;   /* timer 2 high byte    */
extern  near  char  T3;     /* timer 3              */

extern  near  char  IP0;  /* interrupt priority control */
extern  near  char  IP1;  /* interrupt priority control */
extern  near  char  IEN0; /* interrupt enable control   */
extern  near  char  IEN1; /* interrupt enable control   */
extern  near  char  TM2IR;/* timer 2 inerrupt flags     */
extern  near  char  STE;  /* set enable                 */
extern  near  char  RTE;  /* reset/toggle enable        */

extern  near  char  PWMP; /* PWM prescaler              */
extern  near  char  PWM0; /* PWM 0                      */
extern  near  char  PWM1; /* PWM 1                      */

extern  near  char  SIGNAL;
extern  near  char  T2_OLD;

/************************************************************/
```

```
/* 8051 C COMPILER SPECIAL FUNCTION BIT DECLARATIONS */
/*******************************************************/

near struct special_function_bits
{                           /*  Bit Name   Bit Value */

/* P0  (Port 0)                                 */

unsigned int P0_0   : 1; /* bit 0   80h */
unsigned int P0_1   : 1; /* bit 1   81h */
unsigned int P0_2   : 1; /* bit 2   82h */
unsigned int P0_3   : 1; /* bit 3   83h */
unsigned int P0_4   : 1; /* bit 4   84h */
unsigned int P0_5   : 1; /* bit 5   85h */
unsigned int P0_6   : 1; /* bit 6   86h */
unsigned int P0_7   : 1; /* bit 7   87h */

/* TCON  (Timer Control)                                        */

unsigned int IT0: 1; /* input INT0/ transition activ.  88h  */
unsigned int IE0: 1; /* ext interr.request flag 0  89h     */
unsigned int IT1: 1; /* input INT1/ transition activ.  8ah  */
unsigned int IE1: 1; /* ext interr.request flag 1  8bh     */
unsigned int TR0: 1; /* counter 0 enable/disable       8ch  */
unsigned int TF0: 1; /* counter 0 interr.req & overfl 8dh  */
unsigned int TR1: 1; /* counter 1 enable/disable       8eh  */
unsigned int TF1: 1; /* counter 1 interr.req & overfl 8fh  */

/* P1  (Port 1)                      */

unsigned int P1_0: 1; /* bit 0  90h */
unsigned int P1_1: 1; /* bit 1  91h */
unsigned int P1_2: 1; /* bit 2  92h */
unsigned int P1_3: 1; /* bit 3  93h */
unsigned int P1_4: 1; /* bit 4  94h */
unsigned int P1_5: 1; /* bit 5  95h */
unsigned int P1_6: 1; /* bit 6  96h */
unsigned int P1_7: 1; /* bit 7  97h */

/* SCON   (Serial Port Control)                              */

unsigned int RI :1; /* reception complete int.flag    98h */
unsigned int TI :1; /* transmission complete int.flag 99h */
unsigned int RB8:1; /* received data bit 8            9ah */
unsigned int TB8:1; /* transmitter data bit 8         9bh */
unsigned int REN:1; /* receiver enable                9ch */
unsigned int SM2:1; /* conditional receiver enable    9dh */
unsigned int SM1:1; /* serial port operation mode lsb 9eh */
unsigned int SM0:1; /* serial port operation mode msb 9fh */

/* P2   (Port 2)           */

unsigned int P2_0: 1; /* bit 0 a0h */
unsigned int P2_1: 1; /* bit 1 a1h */
unsigned int P2_2: 1; /* bit 2 a2h */
unsigned int P2_3: 1; /* bit 3 a3h */
unsigned int P2_4: 1; /* bit 4 a4h */
unsigned int P2_5: 1; /* bit 5 a5h */
unsigned int P2_6: 1; /* bit 6 a6h */
unsigned int P2_7: 1; /* bit 7 a7h */

/* IE   (Interrupt Enable)                               */

unsigned int EX0: 1;  /* external request 0          a8h */
unsigned int ET0: 1;  /* internal timer/counter 0 a9h */
unsigned int EX1: 1;  /* external request 1          aah */
unsigned int ET1: 1;  /* internal timer/counter 1 abh */
unsigned int ES : 1;  /* internal serial port        ach */
unsigned int IE_5: 1; /* reserved                    adh */
unsigned int IE_6: 1; /* reserved                    aeh */
unsigned int EA  : 1; /* enable all                  afh */

/* P3   (Port 3)         */

unsigned int P3_0: 1; /* bit 0     b0h */
unsigned int P3_1: 1; /* bit 1     b1h */
unsigned int P3_2: 1; /* bit 2     b2h */
unsigned int P3_3: 1; /* bit 3     b3h */
unsigned int P3_4: 1; /* bit 4     b4h */
unsigned int P3_5: 1; /* bit 5     b5h */
unsigned int P3_6: 1; /* bit 6     b6h */
unsigned int P3_7: 1; /* bit 7     b7h */

/* IP   (Interrupt Priority)                             */

unsigned int PX0: 1;  /* external request 0          b8h */
unsigned int PT0: 1;  /* internal timer/counter 0 b9h */
unsigned int PX1: 1;  /* external request 1          bah */
unsigned int PT1: 1;  /* internal timer/counter 1 bbh */
unsigned int PS : 1;  /* internal serial port        bch */
unsigned int IP_5: 1; /* reserved                    bdh */
unsigned int IP_6: 1; /* reserved                    beh */
unsigned int IP_7: 1; /* reserved                    bfh */

/*      (Unimplemented)                        */

unsigned int C0H: 1; /* unimplemented  c0h */
unsigned int C1H: 1; /* unimplemented  c1h */
unsigned int C2H: 1; /* unimplemented  c2h */
unsigned int C3H: 1; /* unimplemented  c3h */
unsigned int C4H: 1; /* unimplemented  c4h */
unsigned int C5H: 1; /* unimplemented  c5h */
unsigned int C6H: 1; /* unimplemented  c6h */
unsigned int C7H: 1; /* unimplemented  c7h */
```

C552SR.DEF

```
*************************************************************
* 8051 C COMPILER SPECIAL FUNCTION REGISTER DEFINITIONS *
*************************************************************

spec_funct_regs:  .section   reg, offset 0, ref_only

         .globals  on

_ACC:         .reg     e0h   ;accumulator
_B:           .reg     f0h   ;B register
_PSW:         .reg     d0h   ;program status word
_SP:          .reg     81h   ;stack pointer
```

```
/*      (Unimplemented)                  */

unsigned int C8H: 1; /* unimplemented c8h */
unsigned int C9H: 1; /* unimplemented c9h */
unsigned int CAH: 1; /* unimplemented cah */
unsigned int CBH: 1; /* unimplemented cbh */
unsigned int CCH: 1; /* unimplemented cch */
unsigned int CDH: 1; /* unimplemented cdh */
unsigned int CEH: 1; /* unimplemented ceh */
unsigned int CFH: 1; /* unimplemented cfh */

/* PSW  (Processor Status Register)                             */

unsigned int P     : 1; /* parity                    d0h */
unsigned int PSW_1: 1; /* reserved                  d1h */
unsigned int OV    : 1; /* overflow                  d2h */
unsigned int RS0   : 1; /* register bank select 0 d3h */
unsigned int RS1   : 1; /* register bank select 1 d4h */
unsigned int F0    : 1; /* user flag 0               d5h */
unsigned int AC    : 1; /* auxiliary carry           d6h */
unsigned int CY    : 1; /* carry                     d7h */

/*       (Unimplemented)                                  */

unsigned int D8H: 1; /* unimplemented          d8h */
unsigned int D9H: 1; /* unimplemented          d9h */
unsigned int DAH: 1; /* unimplemented          dah */
unsigned int DBH: 1; /* unimplemented          dbh */
unsigned int DCH: 1; /* unimplemented          dch */
unsigned int DDH: 1; /* unimplemented          ddh */
unsigned int DEH: 1; /* unimplemented          deh */
unsigned int DFH: 1; /* unimplemented          dfh */

/* ACC   (Accumulator)                                */

unsigned int ACC_0: 1; /* accumulator bit 0 e0h */
unsigned int ACC_1: 1; /* accumulator bit 1 e1h */
unsigned int ACC_2: 1; /* accumulator bit 2 e2h */
unsigned int ACC_3: 1; /* accumulator bit 3 e3h */
unsigned int ACC_4: 1; /* accumulator bit 4 e4h */
unsigned int ACC_5: 1; /* accumulator bit 5 e5h */
unsigned int ACC_6: 1; /* accumulator bit 6 e6h */
unsigned int ACC_7: 1; /* accumulator bit 7 e7h */

/*       (Unimplemented)                  */

unsigned int E8H: 1; /* unimplemented e8h */
unsigned int E9H: 1; /* unimplemented e9h */
unsigned int EAH: 1; /* unimplemented eah */
unsigned int EBH: 1; /* unimplemented ebh */
unsigned int ECH: 1; /* unimplemented ech */
unsigned int EDH: 1; /* unimplemented edh */
unsigned int EEH: 1; /* unimplemented eeh */
unsigned int EFH: 1; /* unimplemented efh */

/* B   (B Register)                       */

unsigned int B_0: 1; /* unimplemented f0h */
unsigned int B_1: 1; /* unimplemented f1h */
unsigned int B_2: 1; /* unimplemented f2h */
unsigned int B_3: 1; /* unimplemented f3h */
unsigned int B_4: 1; /* unimplemented f4h */
unsigned int B_5: 1; /* unimplemented f5h */
unsigned int B_6: 1; /* unimplemented f6h */
unsigned int B_7: 1; /* unimplemented f7h */

/*       (Unimplemented)                  */
unsigned int F8H: 1; /* unimplemented f8h */
unsigned int F9H: 1; /* unimplemented f9h */
unsigned int FAH: 1; /* unimplemented fah */
unsigned int FBH: 1; /* unimplemented fbh */
unsigned int FCH: 1; /* unimplemented fch */
unsigned int FDH: 1; /* unimplemented fdh */
unsigned int FEH: 1; /* unimplemented feh */
unsigned int FFH: 1; /* unimplemented ffh */

};
```

In diesem File werden die Variablen, die mit dem 552 zu tun haben deklariert. Wenn Namen im 552-Handbuch etwas anders sind, dann empfiehlt es sich hier nachzusehen.

C552SRP.H

Dieses File ist dem C552SR.H sehr ähnlich, nur so abgeändert, daß es der Turbo-C-Compiler versteht (siehe Diagramm-Beschreibung). Diese Datei ist hier nicht abgedruckt, da sie inhaltlich praktisch gleich der Datei C552SR.H ist mit folgenden Abweichungen: Es fehlen die drei mit einem Punkt beginnenden Zeilen, zu Beginn von C552SR.H; es fehlt das Schlüsselwort `near`. Die Datei ist aber auf TGM-DSK-169 enthalten.

```
_DPL:         .reg     82h    ;data pointer low
_DPH:         .reg     83h    ;data pointer high

_P0:          .reg     80h    ;port 0
_P1:          .reg     90h    ;port 1
_P2:          .reg     a0h    ;port 2
_P3:          .reg     b0h    ;port 3
_P4:          .reg     c0h    ;port 4
_P5:          .reg     c4h    ;port 5
_ADCON:       .reg     c5h    ;ADC control
_ADCH:        .reg     c6h    ;ADC high

_S0CON:       .reg     98h    ;serial control
_S0BUF:       .reg     99h    ;serial buffer
_TMOD:        .reg     89h    ;timer mode
```

```
_TCON:          .reg    88h     ;timer control
_PCON:          .reg    87h     ;power control
_TM2CON:        .reg    eah     ;timer 2 control
_CTCON:         .reg    ebh     ;capture control

_S1CON:         .reg    d8h     ;serial 1 control
_S1STA:         .reg    d9h     ;serial 1 status
_S1DAT:         .reg    dah     ;serial 1 data
_S1ADR:         .reg    dbh     ;serial 1 address

_TL0:           .reg    8ah     ;timer 0 low byte
_TH0:           .reg    8ch     ;timer 0 high byte
_TL1:           .reg    8bh     ;timer 1 low byte
_TH1:           .reg    8dh     ;timer 1 high byte
_TML2:          .reg    ech     ;timer 2 low byte
_TMH2:          .reg    edh     ;timer 2 high byte
_T3:            .reg    ffh     ;timer 3
_STE:           .reg    eeh     ;set enable
_RTE:           .reg    efh     ;reset/toggle enable

_IP0:           .reg    b8h     ;interrupt priority control
_IP1:           .reg    f8h     ;interrupt priority control
_IEN0:          .reg    a8h     ;interrupt enable control
_IEN1:          .reg    e8h     ;interrupt enable control
_TM2IR:         .reg    cbh     ;timer 2 interrupt flags

_PWMP:          .reg    feh     ;PWM prescaler
_PWM0:          .reg    fch     ;PWM 0
_PWM1:          .reg    fdh     ;PWM 1

_SIGNAL:        .reg    24h
_T2_OLD:        .reg    25h

_SFB:           .reg    80h     ;special function bits
                                ;starting address

          .globals  off

          .ends

; User bit structures located in the internal RAM may
; be defined here, or in another file that is included
; in 'c8051rt.src'. Creating a user defined file will keep
; the additions from being overwritten if the compiler is
; reinstalled. If another file is created, be sure and copy
; the 'section' directive shown below.

; The actual structure template can be defined in
; 'c8051sr.h', or in a user defined include file. The
; structure 'special_functions_bits' can be used as an
; example. Be sure and put an underbar in front of the name
; that will be used when accessing the structure in C.

; For example, if a structure consisting of 5 bits is to be
; defined and placed at the top of the internal memory, the
; following could be used: (The semi-colons must be removed
; for the statements to take effect)

;user_defined_bits:  .section     bit, offset 7bh, ref_only

;          .globals  on

;_FIVE_BITS:     .ds     5

;          .globals  off

;          .ends

; Note that the section is linked to start at 7bh by the
; 'offset' modifier on the section definition statement. This
; number should be changed as additional bit structures are
; defined.  Alternatively, the 'offset' modifier can be
; removed and the section can be located using an 'org'.

          .end
```

In diesem File werden die Register des 552 definiert. Es wird in das Assemblerfile RT51.SRC beim Assemblieren eingebunden.

DATH.LNK

```
test
8c00
_
rt51
0
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
```

DATX.LNK

```
test
8c00
_
rt51
0
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
```

```
_                       _
test                    test
c8051c                  c8051c
c8051s                  c8051s
c8051m                  c8051m

_                       _
h                       x
```

DATH.LNK ist jenes File, das man dem Linker übergeben muß, damit er ein HEX-File erzeugt. Achtung! Es kann passieren, daß sich die Anzahl der einzugebenden Parameter ändert. Das passiert zum Beispiel, wenn man keine globalen Variablen im C-Programm definiert. Dann funktioniert das Linken nicht, solange man das LNK-File nicht ändert. Weiters muß bei Veränderung des Namens des C-Programms auch hier die Änderung vorgenommen werden. Zweckmäßiger ist es aber den Namen TEST.C beizubehalten und immer das Programm, das man momentan bearbeitet, in TEST.C umzubenenen.

DAT.LNK funktioniert genauso wie DATH.LNK, nur erzeugt der Linker dann kein HEX-, sondern ein TSK-File.

RT51.SRC

Nur auf Disk TGM-DSK-169. RT51.SRC ist das Assemblerfile, das vor dem C-Programm abgearbeitet wird (siehe Diagrammbeschreibung). Bemerkenswert ist dabei: Auf Page 2 unten wird C552SR.DEF eingebunden. Hier werden einige Variablen definiert. Weiters auf Page 3, mitte steht die Zeile `.org 0a000h`. Hier steht der Offset der Startadresse. Nachdem beim Linken für die Startadresse aber 0 angegeben wird, ist das gleichzeitig die Startadresse. Wirklich gestartet wird das Programm aber auf A103h. Den genauen Grund habe ich nicht erforscht. Es funktioniert einfach so. Weiters ist zu erwähnen, daß es sehr zweckmäßig ist, am Ende des RT51.SRC Interruptservice-Routinen zu schreiben, wie es Prof. Winkler schon gemacht hat (siehe Ende des Files). Achtung! Man darf danach nicht vergessen RT51.SRC neu zu assemblieren.

SIM.BAT

```
gmenu m
avsim51 bflsim.cmd
```

SIM.BAT ist die Datei, die man zum Starten des AVSIM51 verwendet. Beim AVSIM51 besteht die Möglichkeit, daß man Tasten, die man im Programm betätigen müßte, beim Aufruf als Buchstabenkombination anhängt. Weiters besteht im Programm die Möglichkeit ein File zu laden, das einige Tasten betätigt. Im SIM.BAT wird jetzt eine trickreiche Kombination dieser beiden Möglichkeiten angewendet, indem mit der Tastenkombination `bflsim.cmd` mit dem "b" der Prozessor ausgewählt wird, und danach das File `sim.cmd` nachgeladen wird, welches weitere Tasten betätigt.

SIM.CMD

```
lssim.sym
lpe:\c552\test.hex
a103
```

In diesem File sind die Tasten abgespeichert, die der AVSIM51 nach seinem Aufruf selbständig betätigen soll. In diesem Fall wird damit das HEX-File geladen und der PC auf a103h gestellt.

SIM.SYM

```
SG CODE  C:0000 FFFF
SG XDATA X:0000 FFFF
```

Aus diesem File bezieht der AVSIM51 Symbole und die Speicherkonfiguration.

AVSIM 51

Allgemeines

Kommandozeilenaufruf

```
(1) AVSIM51
```

Es meldet sich eine Auswahl über insgesamt 6 mögliche Prozessoren:

```
A:8051/8751, B:8052/8752, C:8031, D:8032, E:80C51, F:80C31
```

```
(2) AVSIM51 <argument>
```

wobei `<argument>` eine beliebige Folge von Tastenberührungen ist.

Beispiele:

`AVSIM51 A` `A` überspringt das erste Menu und wählt 8051.

`AVSIM51 AFLDEMO.CMD` `A` überspringt das erste Menu und wählt 8051. `F` Menu-Punkt commandFile. `L` Load Lädt die Kommando-Datei `DEMO.CMD`. Die Kommando-Datei enthält dann weitere Tastaturkommandos, die dann automatisch am Bildschirm ablaufen.

```
(3) SIM
```

Am einfachsten ruft man den Simulator mit der Batchdatei `SIM.BAT` auf. (Sie wählt eine 8031 CPU (ROM-less) als Simulationsobjekt.)

AVSIM-kompatible Dateien

Dateien im HEX-Format können durch den AVSIM bearbeitet werden. Sowohl der Assembler X8051 als auch der Assembler ASM51 sind dazu geeignet. Der Aufruf erfolgt wie folgt:

```
X8051/LINK      ASM <name>
ASM51/OH        ASS <name>
```

In beiden Fällen hat die Datei den Namen `<name>.ASM`. Soll auch eine Symboldatei <name>.SYM verwendet werden, muß auf das durch AVSIM vorgegebene SYM-Format geachtet werden.

Betriebsarten

Der Simulator unterscheidet zwischen **Kommando-Modus** (Menü) und **Display-Modus** (Screen). Mit `<Esc>` schaltet man zwischen Menü und Screen um. Um den Simulator zu verlassen, wählen Sie im Menü-Modus den Punkt `Quit` an. Danach geben Sie `E` für `Exit` ein. Die beiden Modi werden im Folgenden erläutert.

Kommando-Modus(Menü)

Mit `<Space>` schaltet man zwischen den beiden Menü-Balken um. Um zwischen den Menüpunkten zu wählen, benutzt man die Cursor-Tasten. Angewählt wird ein Menüpunkt entweder durch Cursor-Auswahl und `<Enter>` oder durch die entsprechende (großgeschriebene) Taste. (Auch die Menü-Punkte des verstecken Menü-Balkens.) Mit `<Ctrl-C>` kommt man jederzeit zum Hauptmenü. Mit `<Esc>` schaltet man jederzeit zwischen Menü und Screen um. Um Hilfe zu bekommen ruft man vom Hauptmenü aus den Menüpunkt Help auf. Es erscheint ein Untermenü, in dem man die 3 verschiedenen Helps aufrufen kann.

Display-Modus(Bildschirm)

Im linken Teil sieht man den disassemblierten Op-Code der nächsten auszuführenden Befehle. Im rechten Teil sind die Special-Function-Registers der zu simulierenden CPU abgebildet. Die beiden Dumpareas geben, umschaltbar (Pg-Up, Pg-Dn), einen Teil des internen RAMs wieder. Außerdem werden die Zustände der umschaltbaren Simulator - Funktionen angezeigt (siehe Funktionstasten).

Speicherbereiche

Das Festlegen des definierten Speicherbereichs erfolgt vor dem Aufrufen des Simulators und zwar durch die Datei SIM.SYM. Sie ist eine normale ASCII-DATEI, die wie folgt aufgebaut ist:

```
SG CODE C:0000 1000
SG DATA X:8000 A000
```

Die erste Zeile definiert das ROM (in obigem Beispiel von 0000 bis 1000 (Hexadezimal)). Die zweite Zeile definiert das RAM (in obigem Beispiel von 8000 bis A000 (Hexadezimal)). Startet man den Simulator mit der Batch-Datei, wird diese Datei automatisch geladen. Ansonsten muß man Sie im Load-Menü als SymbolTable laden. Definierter Speicherbereich wird bei einschalten mit 00h gefüllt, nicht definierter Speicherbereich mit FFh.

Hinweis: Zu großer definierter Speicher benötigt Platz auf der Diskette! ==> Zugriff dauert lange!

Ein Programm laden

Ein zu simulierendes Programm muß, um es in den Simulator zu laden im Intel-Hex-Format (Quelltext ==> Assembler(ASM51) ==> Objectcode_to_Hex-Conversion(OH) ==> Intel-Hex-File) sein. Vom Hauptmenü aus wählt man LOAD und dann PROGRAM.

Simulation der seriellen Schnittstelle

Die zu empfangenden Zeichen müssen in einer ASCII-Datei gespeichert sein. (Achtung: Auch Carrige-Return und Line-Feed gelten als Zeichen.)

Im Menü IO wählt man OPEN und gibt den Namen der Eingabe-Datei an. Dann den der Ausgabe-Datei. (Einer der Datei-Namen kann mit <ENTER> übersprungen werden.) Als Trigger verwendet man CYCLES. Anschließen gibt man an nach wievielen Zyklen ein Byte hereinkommmt. Bei Format gibt man MAPBYTE an. Als 'Expression' gibt man SBUF an. (Dort wird empfangenes Zeichen abgespeichert.) Um den IO Transfer zu aktivieren muß man im SET Menü mit CYCLES den Cycle-Counter einschalten.

Hinweis zu Assembler ASM51

Am Ende des Quelltextes, der mit einem Texteditor erstellt wird, muß das Assembler-Directive 'END' stehen, das von einem Strichpunkt ';' gefolgt wird. Hinter diesem Strichpunkt darf K E I N Zeichen mehr stehen. (Auch nicht Carrige-Return, Line-Feed oder ein Blank!)

Eingabeformat

Zur Berechnug von Werten bzw. zur Eingabe von Opcodes stehen zur Verfügung:

Operatoren

```
+ - @
()   to 4 levels
```

Zahlenbezeichner

```
Binary:  %111   or  111B
Octal:   @377   or  377Q
Hex:     $FF    or  FFH
```

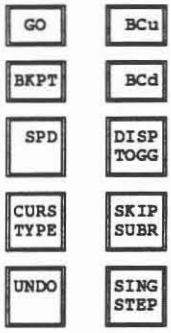
Operanden

```
CPU symbols - see file AVSIM51.REG
symbols     - user symbol table
$             Current value of PC
.             Value of previous expression
'x' or "x"    Character constant
digits        Default radix decimal
```

Tastenfunktionen

Funktionstasten

Sie arbeiten immer!



Kursorsteuerung

Mit dem Cursor springt man zwischen den Fenstern.

```
<CuLt>          1 Zeichen links
<CuRt>          1 Zeichen rechts
<CuUp>          1 Zeile aufwärts
<CuDn>          1 Zeile abwärts
<End>           Letztes Zeichen im Fester
<PgUp>          1 Fenster aufwärts
<PgDn>          1 Fenster abwärts
<Return>        Zurück zum letzten Fenster
<CTRL - CuLt>   1 Fenster links
<CTRL - CuRt>   1 Fenster rechts
```

Schnellkommandos

Zu den einzelnen Registern kommt man mit folgenden Tastenkombinationen:

```
<CTRL-A>        A Accumulator
<CTRL-B>        B Register B
<CTRL-C>        C Carry-Register
<CTRL-X>        AC Auxillary-Carry
<CTRL-F>        F0 Flag
<CTRL-O>        OV Overflow-Flag
<CTRL-P>        PC Program-Counter
<CTRL-S>        SP Stack-Pointer
<CTRL-R>        RB Registerbank
<ALT-0..ALT-7>  R0..R7 Register
<CTRL-D>        DP Data-Pointer
<CTRL-I>        Interrupt Enable
<CTRL-T>        TH Timer High
<ALT-A>         Data-Space 1
<ALT-B>         Data-Space 2
<ALT-P,ALT-Q>   P0..P2 Ports
<ALT-Y>         Cycles-Counter
```

```
<ALT-S>         SBUF Serial Buffer
<ALT-C>         SCON Serial Control
<CTRL-PgUp>     SCL Scroll-Mode
<F5>            SPD Speed
<F6>            DSP Display-Toggle
<F8>            SKP Skip-Subroutine
<F7>            CURSOR
```

Editieren

Die Werte der Register können durch überschreiben verändert werden. Außerdem mit +,- inkrementiert bzw. dekrementiert werden.

```
<+> /< ->       inc/dec byte/word/flag
<INS>           Toggle byte/nibble/bit
<CTRL-END>      clear to end of object
<CTRL-HOME>     clear entire object
```

Betriebsartsteuerung

```
<Esc>       Umschaltung zwischen Menü- und Screen-Modus
<F5>        SPEED - set simulation speed (lo/mid/hi)
<ALT-F5>    LABEL Toggle - LABEL: Addresses & operands are
                                  displayed symbolically
                           ADDR: No symbols in disassembly
<F6>        DISPLAY Toggle -  ON: Screen is updated after
                                  each instruction during GO
                            OFF: Only TRACED windows
                                  are updated
                                  until trap occurs
<ALT-F6>    TRACE Toggle -    ON: Window is updated even
                                  when display OFF
                            OFF: Window is updated only
                                  when display ON
<F7>        CURSOR TYPE - Hex / Ascii / Binary
               Cursor will move to preferred screen object
               type, if displayed as several types.
<F8>        SKIP Toggle - SKIP ON will Single Step across
               call opcodes (by setting a bkpt at the next
               instruction and GOing with display OFF).
<CTRL-PgUp> Toggle Scroll Mode
```

Programm-Ausführung

```
<F1>        GO - startet ein Programm
               ab dem Program-Counter bis
             * die die Taste nochmals gedrückt wird oder
             * ein Breakpoint erreicht wird oder
             * ein Zugriff auf eine nichtdefinierte Adresse
               erfolgt.
<F10>       Single-Step - führt einen einzelnen Befehl aus.
<F9>        UNDO - macht einen Befehl rückgängig
```

Breakpoint-Steuerung

```
<F2>        bewegt den Breakpoint Cursor aufwärts
<F4>        bewegt den Breakpoint Cursor abwärts
<F3>        Setzt dynamischen Breakpoint auf Brkpoint Cursor
```

Bildschirmbild

```
LABEL    OPERATION          8051/8751 AVSIM 8051 Simulator/Debugger       V1.22
RESET    no     memory      CPU REGISTERS         FLAGS     SCL SPD DSP SKP CURSOR
0001H    no     memory      C  Accumulator     AC F0 OV P OFF HI  ON  OFF  MENU
0002H    no     memory      0  00000000:00:Á    0  0  0  0          Cycles:
0003H    no     memory         addr       data
0004H    no     memory      PC:0000 »1 FF FF FF FF Timers TH/TL  TF/TR  G/T/M1/M0
0005H    no     memory      SP:  07 » 00 00 00 00   T0:  00 00   0  0  0 0  0  0
0006H    no     memory                00 00 00 00   T1:  00 00   0  0  0 0  0  0
0007H    no     memory      DP:0000 » FF FF FF FF
0008H    no     memory      R0:00:Á » 00:Á  RB:00 Ints A S T1 X1 T0 X0 Edg IT IE
0009H    no     memory      R1:00:Á » 00:Á   B:00  En  0 0 0  0  0  0   X0: 0  0
000AH    no     memory      R2:00   R4:00   R6:00  Pr    0 0  0  0  0   X1: 0  0
000BH    no     memory      R3:00   R5:00   R7:00 SBUF:  In   Out  PCON:0xxxxxxx
000CH    no     memory      Data Space                   00:Á 00:Á SCON:00000000
000DH    no     memory      0000  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ Ports
000EH    no     memory      0008  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ P0   11111111
000FH    no     memory      0010  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ FF:Á:11111111
0010H    no     memory      0018  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ P1   11111111
0011H    no     memory      Data Space                             FF:Á:11111111
0012H    no     memory      0020  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ P2   11111111
0013H    no     memory      0028  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ FF:Á:11111111
0014H    no     memory      0030  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ P3   11111111
0015H    no     memory      0038  00 00 00 00 00 00 00 00 ÁÁÁÁÁÁÁÁ FF:Á:11111111
Select Command- or use arrow-key
Dump Expression commandFile Help IOLoad --space--
```

Kommandozeile

Die folgende Übersicht zeigt den gesamten Menübaum des AVSIM51, die Einrückung gibt die entsprechende Ebene an. Rechts von der Kurzbezeichnung steht, durch Bindestriche getrennt, der Kurzkode, mit dem man das Kommando erreicht. Dabei bedeuten [...] eines aus, <n> Dateiname, <c> condition, <a> Adresse, <d> Daten, <e> expression.

Dump----------D

Einstellung des Datenbereiches, des in den zwei Datenfenstern eingestellt werden kann. Durch die Wahl `Indirect` können auch Datenbereiche im externen RAM oder ROM dargestellt werden.

Data Space 1 ----------D1

Absolute----------D1A<a>

Eingabe Startadresse

Indirect----------D1I<a>

Zeiger auf indizierte Speicherzelle

Data Space 2 ----------D2

Absolute----------D2A<a>

Eingabe Startadresse

Indirect----------D2I<a>

Zeiger auf indizierte Speicherzelle

Expression ----------E

Eingabe eines Ausdrucks in jene Bildschirmzelle, auf der im Augenblick der Kursor(PC) für steht (siehe ENTER EXPRESSION)

---

commandFile----------F

Man kann eine oft wiederkehrende Folge von Tastendrücken abspeichern (Makro) und dann wieder abrufen.

Load----------FL<n>

Wiedergabe einer gespeicherten Tastenfolge (Makro)

Open----------FO<n>

Eröffnen einer Makro-Datei für Tastendrucke. Jetzt führt man die zu sichernde Folge von Befehlen aus. Um das Makro zu sichern, wählt man wieder `commandFile` und dann `Close`. Hat man sich geirrt, kann man statt `Close Restart` wählen und ohne Eingabe eines neuen Namens von vorne beginnen.
Achtung: Es werden auch die Tastendrucke, die zum Speichern des Makros notwendig sind, aufgezeichnet. D.h. nachdem ein Makro aufgeführt wurde, befindet man sich im `commandFile`-Menü.

Close----------FC

Schließen der Datei für Tastendrucke

Restart----------FR

Neubeginn

Help----------H

Hilfebildschirme

Commands----------HC

Besonderheiten der Kommandozeile.

Display----------HD

Bildschirmsteuerung

Simulation----------HS

Funktionstastenbelegung

Avocet

Herstelleradresse, sowie Hinweise auf Aufrufparameter bezüglich Bildschirmkarte und Hinweise auf andere Produkte des Herstellers.

IO-File----------I

Der Inhalt einer Datei kann den Inhalt einer Prozessoradresse beeinflussen. Ebenso kann der Inhalt einer Prozessoradresse in eine Datei gespeichert werden. Der Dateizugriff (Speichern oder Lesen) erfolgt entweder bei jedem Zugriff durch einen Befehl oder nach einer einstellbaren Anzahl von Prozessortaktzyklen.

Open------**IO<n>[YN]<n>[CO]<a>[IO]**

Eingabedatei <n>, Wiederholen der Datei bei Dateiende [YN], Ausgabedatei <n>, Triggerung auf Taktzyklen oder Befehlszugriff [CO], Anbindung an adresse <a>, Richtung In/Out [IO].

Close----------IC

Load----------L

Laden von Dateien

Avocet ----------LA<n>

Data ----------LD<n>

Laden von RAM-Daten.

Program ----------LP<n>

name.HEX

Symbol-table ----------LS<n>

Laden einer Symbol-Tafel.

rOm ----------LO<n>

Memory ----------M

Bearbeitung des RAM-Speichers.

Clear ----------MC<a><a>

Startadresse, Stopadresse.

Fill ----------MF<a><a><d>

Startadresse, Stopadresse, Füllwert.

Move ----------MM<a><a><a>

Startadresse, Stopadresse, Zieladresse.

Search ----------MS<a><a><d>

Startadresse, Stopadresse, Suchbegriff.

searchNext ----------MN

Wiederholen der letzten Suche.

Patch----------P

Direkte Eingabe von Mnemonics

Patch Code ----------PP

Open ----------PO<n>

Datei zum Mitschreiben händisch assemblierter Zeilen.

Close ----------PC

Mitschreiben abschließen

Quit ----------Q

Exit ----------QE

Reset ----------R

Dient zum Rücksetzen verschiedener Funktionen. Unter anderem kann ein Reset der CPU simuliert werden, sowie der Cycles-Counter zurückgesetzt werden.

Cpu ----------RC

Disptrace ----------RD

cYcles ----------RY

Set ----------S

Hier können Breakpoints gesetzt, der Cycles-Counter ein/ausgeschaltet, die Chip-Konfiguration (P2 für Higher-Adressen) und andere Funktionen eingestellt werden.
Achtung: In diesem Unter-Menü gibt es 2 Menü-Balken(mit SPACE umschaltbar!).

Memory Map ----------SM

random-Access----------SMA

lower Address - higher Address

read-Only----------SMO

lower Address - higher Address

Passpoint ----------SP

R/W-point----------SPA

R/W-range----------SPB

Write-point----------SPC

Write-range----------SPD

opTions ----------ST

Page----------STP[YN]

Change external DataDrive with Port P2 from No to: [YN]

cYcles ----------SY

V-drive ----------SV

Set virtual memory

Set Breakpoint ----------S__

On Breakpoints, a delay can be specified by typing digits BEFORE selecting the Breakpoint TYPE. On sticky breakpoints, this delay is restored after each activation.

Conditional ----------S__C

Trap when condition match.

Indirect----------S__CI<e><c>

Mask----------S__CM[01x]

Range----------S__CR<e><e><c>

Value----------S__CV<e><c>

Dynamic ----------S__D

Cleared automatically upon Trap

R/W-point----------S__DA

R/W-range----------S__DB

Write-point----------S__DC

Write-range----------S__DD

Opcode ----------S__O<mnem>

Sticky----------S__S

R/W-point----------S__SA

R/W-range----------S__SB

Write-point----------S__SC

Write-range----------S__SD

setUp ----------U

Der AVSIM51 hat die Fähigkeit, ausgeführte Befehle rückgängig machen zu können. In diesem Menüpunkt legt man die Gedächtnislänge fest.

Undo ----------UU

View ----------V

Dient zum Anzeigen verschiedener Simulator-Zustände.

Bkpts ----------VB

```
   BKPT Addr  EndAddr  Access  Life     Delay/count
Condition
```

IO-files ----------VI

Memory-map ----------VM

```
MEMORY MAP          Label        Addr     EndAddr     Type
Code Space: 0000 to FFFF
Data Space: 0000 to 00FF
                    R0:          D:0000   007F
External Data Space: 0000 to FFFF
```

Opc-traps ----------VO

```
Opcode Traps:           Operation     Address Mode
                           ADD    A,R0
```

Passpts ----------VP

```
PASS Addr  EndAddr  Access          Count          Condition
```

Symbols ----------VS

Alpha ----------VSA

Symbol	Addr	Symbol	Addr	Symbol	Addr	Symbol	Addr
AC	B:00D6	PX0	B:00B8	R5	D:0005	SM1	B:009E
CY	B:00D7	PX1	B:00BA	R5"	D:0015	SM2	B:009D
EA	B:00AF	R0	D:0000	R5'	D:000D	T0	B:00B4
ES	B:00AC	R0"	D:0010	R5\	D:001D	T1	B:00B5
ET0	B:00A9	R0'	D:0008	R6	D:0006	T2	B:0090
ET1	B:00AB	R0\	D:0018	R6"	D:0016	T2EX	B:0091
EX0	B:00A8	R1	D:0001	R6'	D:000E	TB8	B:009B
EX1	B:00AA	R1"	D:0011	R6\	D:001E	TF0	B:008D
EXTI0	C:0003	R1'	D:0009	R7	D:0007	TF1	B:008F
EXTI1	C:0013	R1\	D:0019	R7"	D:0017	TI	B:0099
F0	B:00D5	R2	D:0002	R7'	D:000F	TIMER0	C:000B
IE0	B:0089	R2"	D:0012	R7\	D:001F	TIMER1	C:001B
IE1	B:008B	R2'	D:000A	RB8	B:009A	TIMER2	C:002B
INT0	B:00B2	R2\	D:001A	RD	B:00B7	TR0	B:008C
INT1	B:00B3	R3	D:0003	REN	B:009C	TR1	B:008E
IT0	B:0088	R3"	D:0013	RESET	C:0000	TXD	B:00B1
IT1	B:008A	R3'	D:000B	RI	B:0098	WR	B:00B6
OV	B:00D2	R3\	D:001B	RS0	B:00D3		
P	B:00D0	R4	D:0004	RS1	B:00D4		
PS	B:00BC	R4"	D:0014	RXD	B:00B0		
PT0	B:00B9	R4'	D:000C	SINT	C:0023		
PT1	B:00BB	R4\	D:001C	SM0	B:009F		

Registers ----------VSR

Symbol	Symbol	P2
ACC	SBUF	P2L
B	SBUFL	P3
CYCLES	SCON	P3L
DP	SP	PC
DPH	T2CON	PCON
DPL	TCON	PSW
IE	TH0	RCAP2H
IIP	TH1	RCAP2L
IP	TH2	
P0	TL0	
P0L	TL1	
P1	TL2	
P1L	TMOD	

Numbers ----------VSN

Code ----------VSC

Addr	Symbol
C:0000	RESET
C:0003	EXTI0
C:000B	TIMER0
C:0013	EXTI1
C:001B	TIMER1
C:0023	SINT
C:002B	TIMER2

Data ----------VSD

Addr	Symbol						
D:0000	R0	D:000A	R2'	D:0014	R4"	D:0016	R6"
D:0001	R1	D:000B	R3'	D:0015	R5"	D:0017	R7"
D:0002	R2	D:000C	R4'			D:0018	R0\
D:0003	R3	D:000D	R5'			D:0019	R1\
D:0004	R4	D:000E	R6'			D:001A	R2\
D:0005	R5	D:000F	R7'			D:001B	R3\
D:0006	R6	D:0010	R0"			D:001C	R4\
D:0007	R7	D:0011	R1"			D:001D	R5\
D:0008	R0	D:0012	R2"'			D:001E	R6\
D:0009	R1	D:0013	R3"'			D:001F	R7\

eXt ----------VSX

Bit ----------VSB

Addr	Symbol	Addr	Symbol
B:0088	IT0	B:00AC	ES
B:0089	IE0	B:00AF	EA
B:008A	IT1	B:00B0	RXD
B:008B	IE1	B:00B1	TXD
B:008C	TR0	B:00B2	INT0
B:008D	TF0	B:00B3	INT1
B:008E	TR1	B:00B4	T0
B:008F	TF1	B:00B5	T1
B:0090	T2	B:00B6	WR
B:0091	T2EX	B:00B7	RD
B:0098	RI	B:00B8	PX0
B:0099	TI	B:00B9	PT0
B:009A	RB8	B:00BA	PX1
B:009B	TB8	B:00BB	PT1
B:009C	REN	B:00BC	PS
B:009D	SM2	B:00D0	P
B:009E	SM1	B:00D2	OV
B:009F	SM0	B:00D3	RS0
B:00A8	EX0	B:00D4	RS1
B:00A9	ET0	B:00D5	F0
B:00AA	EX1	B:00D6	AC
B:00AB	ET1	B:00D7	CY

SFR ----------VSS

Addr	Symbol	Addr	Symbol
R:0080	P0	R:00CD	TH2
R:0081	SP	R:00D0	PSW
R:0082	DP	R:00E0	ACC
R:0083	DPH	R:00F0	B
R:0087	PCON		
R:0088	TCON		
R:0089	TMOD		
R:008A	TL0		
R:008B	TL1		
R:008C	TH0		
R:008D	TH1		
R:0090	P1		
R:0098	SCON		
R:0099	SBUF		
R:00A0	P2		
R:00A8	IE		
R:00B0	P3		
R:00B8	IP		
R:00C8	T2CON		
R:00CA	RCAP2L		
R:00CB	RCAP2H		
R:00CC	TL2		

eXecute ----------X

Ausführen eines einzelnen Simulator-Befehls. Folgende Opcode-Syntax ist einzuhalten:

```
{label:}  operation  {operands}  {; comments}
```

Dateien für den Simulator

Die mit '*' gekennzeichneten Dateien sind nachfolgend kurz beschrieben.

```
ASM        BAT        39 09-19-90   11:27a *
ASS        BAT       728 01-30-89   12:01a *
AVSIM      TXT     10693 04-08-91    1:16p <- hier eingearbeitet
AVSIM      DFV                             <- hier eingearbeitet
AVSIM51    EXE     94720 01-30-89   12:01a
AVSIM51    HLP      5938 01-30-89   12:01a <- hier eingearbeitet
AVSIM51    OVR     29370 01-30-89   12:01a
AVSIM51    REG      1192 04-16-91    8:41a <- hier eingearbeitet
DEMO       CMD       118 08-29-86    7:57p *
DEMO       MSG       313 08-29-86    7:57p *
DEMO1      ASM      4334 08-29-86    7:57p *
DEMO1      HEX       372 08-29-86    7:57p
DEMO1      SYM       333 08-29-86    7:57p
DEMO2      ASM      4639 10-16-90   10:12p *
DEMO2      HEX       424 10-16-90   10:12p
DEMO2      LST     11188 10-16-90   10:12p
DEMO2      OBJ       981 10-16-90   10:12p
DEMO2      SYM         1 10-16-90   10:12p
IOTRAN     DOC      3717 01-30-89   12:01a *
IOTRAN     EXE     16896 01-30-89   12:01a *
KILLEOF    COM     11659 12-20-87   11:28p *
PAR        DAT         5 10-24-90   12:25a *
README             343 10-17-90   12:33p
RUNME      BAT       852 01-30-89   12:01a *
RUN_DEMO   BAT       239 08-29-86    7:57p *
SIM        BAT        32 01-30-89   12:01a *
SIM        CMD        21 01-30-89   12:01a *
SIM        SYM        45 01-30-89   12:01a *
```

ASM.BAT

Assemblieren und Linken einer Datei mit dem Assembler/Linker von 2500AD. Aufruf: `ASM <name>`. `<name>` wird ohne Extension angegeben. Die Datei `<name>` hat die Extension `.ASM` und heißt daher `<name>.ASM`.

ASS.BAT

Assemblieren und Linken einer Datei mit `ASM51` und `OH`.

DEMO.CMD

Automatischer Ablauf des Simulators bei Aufruf mit `AVSIM51 AFLDEMO.CMD`. Es ist die Abfolge der Tastendrucke, die nötig sind, um das Programm `DEMO.ASM` auszuführen. Da es sich um Tastendrucke und nicht um ASCII-Zeichen handelt, ist die Datei nicht darstellbar.

DEMO.MSG

`DEMO.MSG` enthält einen ASCII-Text. Dieser Text wird durch das Programm `DEMO.ASM` über die serielle Schnittstelle eingespielt.

DEMO1.ASM

Originalversion des AVSIM-DEMO-Programms. (Übertragung einer Datei über die serielle Schnittstelle.)

DEMO2.ASM

Wie `DEMO1.ASM` aber modifiziert für den X8051-Assembler.

IOTRAN.DOC, IOTRAN.EXE

Programm zur Anpassung Datentransofmation an verschiedene Formate.

```
hb  - hex-to-byte      file.AH --> file.AB
hw  - hex-to-word      file.AH --> file.AW
hb2 - hex-to-2 byte    file.AH --> file.AB1 & file.AB2
hw2 - hex-to-2 word    file.AH --> file.AW1 & file.AW2
bh  - byte-to-hex      file.AB --> file.AH
wh  - word-to-hex      file.AW --> file.AH
b2h - 2 byte-to-hex    file.AB1 & file.AB2 --> file.AH
w2h - 2 word-to-hex    file.AW1 & file.AW2 --> file.AH
wd  - word-to-decimal     file.AW --> file.AD
b2d - 2 byte-to-decimal  file.AB1 & file.AB2 --> file.AD
w2d - 2 word-to-decimal  file.AW1 & file.AW2 --> file.AD
```

KILLEOF.COM

Löscht Dateieendemarkierung in einer Datei

PAR.DAT

Beispieldaten für die Anbindung an einen IO-Port.

RUNME.BAT

RUN_DEMO.BAT

Aufruf des Demo-Programms.

SIM.BAT

Aufruf von `AVSIM51`.

SIM.CMD

Abgespeicherte Folge von Tastendrucken.

SIM.SYM

Festlegung von Speicherbereichen.

Jedes Programm, in das sich ein Fehler einschleichen kann, wird auch einen enthalten. Folgerung: Jeder Fehler wird dort sitzen, wo er am spätesten entdeckt wird und den größtmöglichen Schaden anrichtet.

INTEL 8051 LINKER MAP TO AVOCET AVSIM51 SYMBOL FORMAT CONVERTER

W.MURTH TGM, NT/EL TGM-DSK-169: MAP2SYM.C,~.DOC,~.EXE

```
SYNTAX :  MAP2SYM [d:]sourcefile[.m51] </r>
          [d:]     drive and path of sourcefile
          [.m51]   extension of sourcefile is not needed
          </r>     view report during conversion
                   (when not set program runs a lot fa-
ster)

OUTPUT:   [d:]sourcefile.SYM
```

You get a symbolfile with the same drive, path and program name as your source file. The extension of the symbol file is .SYM and is automatically set by MAP2SYM.

WHY DO I NEED MAP2SYM ?

If you develop a 8051 program with INTEL MCS-51 Assembler and test your program with AVOCET AVSIM51 you don't see your labels in the AVSIM51. Also your external DATA and CODE space will not be recognized. When you want to have full advantage of the AVSIM51 you need MAP2SYM. This program will generate the informations needed by AVSIM51.

WHAT DO I HAVE TO NOTICE WHEN DEVELOPING PROGRAMS?

During development the source file(s) must have the assembler controls $DEBUG and $SYMBOLS switched on. Only if these options are set, MAP2SYM will work correctly. The only restriction is that at minimum one label must exist. If you dont't follow this MAP2SYM will work without stopping.

AVSIM51 has some restrictions in Labels they are

```
MOV DPTR,#label   "label" must be in the XDATA area, if it is
                  defined in CODE area
                  it will not be recognized
MOV Rn,#label     "label" must be defined with EQU
                  to be recognized
MOV A,label       "label" has to be a DATA address
MOV A,#label      "label" has to be defined with EQU
MOV Rn,#label     "label" das to be defined with EQU
```

These are the known effects of the AVSIM51, but there may be more equivalent effects. A future version of MAP2SYM may have a solution for this problem

HOW DO I LOAD PROGRAMS IN AVSIM51?

To load your 8051 program into AVSIM51 there must be a HEX file of your program. To load it choose the LOAD AVOCET option and enter the filename without any extension.

DO I HAVE ANY SUPPORT FOR THIS PROGRAM?

Sorry there is no support for this program. I tested it with INTEL MCS-51 and KEIL A51. One of the testfiles was the sample file from INTEL MCS-51 MACRO ASSEMBLER with relocatable segments. There were no errors or system failures. The computer with which this program was compiled was 100% virus free (NO Yankee Doodle !!!).

INTEL MCS-51 ist a registrated Trademark of INTEL Corporation.

AVSIM51 is a registrated Trademark of AVOCET SYSTEMS INC.

Version 1.31 released on 10th March 1991 by W.Murth

COPYRIGHT NOTES:

You can copy this program freely without any restrictions. But please copy also the documentation file (to prevent "What the hell does this program do?" or "Why does it work so catastrophically" or "Hmmm? What do i do wrong? After 54 runs I dont have the right output") AND the source code (!!!).

Please don't change the copyright notice in the header of the source and recompile it with your name instead.

Anmerkung: Das gleiche Problem tritt auch beim X8051-Assembler auf. Die vom Linker generierten Symboldateien sind dem AVSIM nicht unmittelbar zuzumuten. Das hier beschriebene Programm MAP2SYM könnte dazu verwendet werden, auch die Umformung der X8051-SYM-Dateien zu übernehmen. Wenn es jemand geschafft hat, bitte der Redaktion mitteilen.

Zur Schulabgängerniveaudebatte: EIN MODERNER LIEBESBRIEF

Halo hübsches Freulein!

Wie get es Tier? Mir get eß gut. Weil ich gestern zuflel, gelehrnt hab und zulang auf wahr, bin ich Häute ein plsschen mühde. Abba sonnst ß alles in Ortnung.

Tschau, dein Beerll.

A3-Flachbettplotter

Die folgenden Beiträge sind Auszüge aus einer umfangreichen Projektarbeit des Speziallehrgangs für Mikroelektronik aus dem Jahr 1986. Das Thema, ein A3-Flachbettplotter, wurde von den Schülern selbst gewählt; die Hardware haben sie selbst beigesteuert.

1986 verfügten wir am TGM noch über keine Entwicklungssoftware für die 8051-Familie. Daher war auch ein Assembler/Disassembler zu schreiben, sozusagen ein Abfallprodukt der gesamten Entwicklung. Das hätte den Gesamtaufwand für das halbjährige Projekt zu stark ansteigen lassen, daher wurde nach einer Möglichkeit gesucht, diese Hilfsmittel einfacher zu bekommen. Gefunden wurde in einem Chip-Spezial der in TURBO-PASCAL geschriebene Assembler INLASS, dessen Mnemonics durch einen Pseudobefehl umschaltbar waren. Daher war die Erweiterung auf einen weiteren Prozessortyp, hier auf den 8052, kein unüberwindliches Problem mehr. Während der Assemblerteil durch Modifikation eines vorhandenen Programms entstand, ist der Disassembler eine Eigenentwicklung. Den Abschluß bildet die Beschreibung der Kommunikation zwischen dem 8052 und dem PC.

Der 8052-BASIC hat den Vorteil eines eigenen BASIC-Interpreters, sodaß Programmentwicklung besonders wenig zusätzlichen Aufwandes bedarf. Siehe auch Beitrag 80C32-Minimodul.

Für Interessenten am Gesamtprojekt können wir weitere Unterlagen anbieten:

TGM-LIT-016:	*Auszug aus Chip-Spezial: INLASS-Assembler*
TGM-LIT-017:	*BASIC-Dokumentation für 8052, INTEL*
TGM-LIT-018:	*Datenblatt 8052-BASIC, INTEL*
TGM-LIT-019:	*Gesamtdokumentation Flachbettplotter, inklusive Schaltbild*

Der Mikrocontroller für den Plotter ist auf einer Europakarte 100x160 mm aufgebaut und enthält neben dem Controller 8052 selbst auch einen 32k EPROM-Speicher, 8k RAM und wegen der damit belegten Portleitungen auch einen zusätzlichen 8255 mit insgesamt 32 zusätzlichen IO-Leitungen. Die Platine ist im Prinzip nicht für den Plotter spezialisiert und kann auch für andere Steuerungsaufgaben verwendet werden, die mit dem 8052 gebaut werden sollen.

Auf Anfrage steht kostenlos ein fertiger Print im Europakartenformat 100x160 zur Verfügung. Sollten mehrere Mitglieder sich diesen Mikrocontroller aufbauen wollen, stehen auch die Filme für das Layout zur Verfügung.

8052 - INLINE - ASSEMBLER

Paul KOSTAL, Peter ULLRICH, TGM, NT/EL, SLME86

TGM-DSK-190
TGM-LIT-016,017,018,019

TGM-DSK-190: INLASS.PAS, PARAMSTR.BIB, SCANLINE.INC, ERROR.INC, SUCHBAUM.INC, BERECHNE.INC, HL1-Z80.INC, HL2-6502.INC, HL3-8080.INC, HL4A8086.INC, HL4B8086.INC, HL5-8052.INC, INL.INC, DOPASS.INC, INLOAD52.PAS

Von den Dateien auf TGM-DSK-190 wurden die Dateien DOPASS, INL, INLASS und INLOAD52 gegenüber dem Original in TGM-LIT-016 verändert. Alle anderen Dateien entsprechen dem Original.. Neu ist HL5-8052.INC. Für einen Vergleich finden Sie auch die jeweiligen Originale auf der Diskette: INLOAD.ORI, DOPASS.ORI, INL.ORI, INLASS.ORI und DOPASS.ORI. Die lauffähigen Versionen sind: INLASS.COM und INLOAD52.COM.

1. Allgemeines

Als Grundlage diente der Turbo-Assembler INLASS (TGM-LIT-016). Dieser ist in Turbo-Pascal geschrieben und daher leicht verständlich veränderbar. Der Assembler besteht aus einem Hauptprogramm und mehreren Include-Prozeduren für die Prozessoren Z80, 6502, 8080 und 8086. Diese sollen nun mit dem Prozessor 8052 erweitert werden. Die neue Prozedur erhielt den Filenamen HL5-8052.INC. Diese Prozedur behandelt nur die Umsetzung der Mnemonics in den entsprechenden OpCode. Die Behandlung der Pseudo-OpCodes sowie der Labels erfolgt ausschließich im Hauptprogramm.

Auf Grund des speziellen Adreßformates der Prozessorfamilie MCS-51 und, um die neue Prozedur in den bestehenden INLASS einzubinden, mußten auch die anderen Include-Files (Prozeduren) geringfügig verändert werden (siehe Punkt 8.6.). Das oben erwähnte Adressformat wird nicht wie üblich in der Reihenfolge "Low-Adress, High-Adress" sondern in der Reihen-folge "High-Adress, Low-Adress" im Speicher des MCS-51 abgelegt.

Da der INLASS ein relocatibles (=verschiebares) Programmfile erzeugt, kann dies mit dem Programm INLOAD52 in ein unverschiebbares COM-File umgewandelt werden. Dieses INLOAD ist gleich dem INLOAD für den Z80 (Datei INLOAD.ORI), nur daß hier wieder das andere Adreßformat des MCS-51 berücksichtigt wurde. (siehe 8.4.)

Die Bedienung und die Pseudo-OpCodes des INLASS sind der Beschreibung des INLASS zu entnehmen. Der Befehlsaufbau des MCS-51 und seine Handhabung sind dem Datenbuch (INTEL) des MCS-51 zu entnehmen.

2. Abweichungen von den 8052-Mnemonics

1. Die Befehle AJMP und ACALL wurden nicht berücksichtigt. Grund: Die Sprungadresse dieser Befehle ist teilweise mit dem Opcode verknüpft. Da aber im 1.Pass diese noch nicht bekannt ist und die Prozedur nur im 1.Pass durchlaufen wird, kann der OpCode für diese zwei Befehle nicht generiert werden.

2.Die Befehle LJMP und LCALL wurden auf JMP und CALL umgenannt. Grund: Wegen Punkt 1 ist nun eine Trennung zwischen A.. und L.. nicht nötig.

3. Der relative Sprungbefehl SJMP wurde in JR umbenannt.

4. Bei den Bit-Befehlen ANL und ORL wurde das Komplement-Zeichen "/" durch "!" ersetzt. Grund: Das Zeichen "/" ist vom Assembler schon als PseudoOpCode für die Division reserviert.

3. Reservierte Symbole

Folgende Symbole sind mit fixen Adressen belegt.

3.1. RAM-ADRESSEN

Symbol	Name	Adresse
ACC	Accumulator	0E0H
B	B-Register	0F0H
PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPH	Data Pointer High-Byte	83H
DPL	Data Pointer Low-Byte	82H
P0	Port 0	80H
P1	Port 1	90H
P2	Port 2	0A0H
P3	Port 3	0B0H
IP	Interrupt Priority Control	0B8H
IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
TCON	Timer/Counter Control	88H
T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 (High-Byte)	8CH
TL0	Timer/Counter 0 (Low-Byte)	8AH
TH1	Timer/Counter 1 (High-Byte)	8DH
TL1	Timer/Counter 1 (Low-Byte)	8BH
TH2	Timer/Counter 2 (High-Byte)	0CDH
TL2	Timer/Counter 2 (Low-Byte)	0CCH
RCAP2H	Timer/Counter 2 Capture Register (High-Byte)	0CBH
RCAP2L	Timer/Counter 2 Capture Register (Low-Byte)	0CAH
SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	97H

3.2. BIT-ADRESSEN

Symbole: P0, P1, P2, P3, IP, IE, TCON, T2CON, SCON, PSW, ACC, B (Name und Adresse siehe vorher)

Aufbau der Bit-Adresse: Symbol.x (x = 0 - 7)

3.3. DIREKTE BIT-ADRESSEN

ByteSymbol	Symbol	Name	Adresse
TCON.0	IT0	Interrupt 0 Typ-Controllbit	88H
TCON.1	IE0	Interrupt 0 Edge-Flag	89H
TCON.2	IT1	Interrupt 1 Typ-Controllbit	8AH
TCON.3	IE1	Interrupt 1 Edge-Flag	8BH
TCON.4	TR0	Timer 0 Run-Controllbit	8CH
TCON.5	TF0	Timer 0 Overflow-Flag	8DH
TCON.6	TR1	Timer 1 Run-Controllbit	8EH
TCON.7	TF1	Timer 1 Overflow-Flag	8FH
SCON.0	RI	Rceive Interrupt-Flag	98H
SCON.1	TI	Transmit Interrupt-Flag	99H
SCON.2	RB8	Received Bit 8	9AH
SCON.3	TB8	Transmitted Bit 8	9BH
SCON.4	REN	Serial Reception Enable	9CH
SCON.5	SM2	Serial Mode	9DH
SCON.6	SM1	Serial Mode	9EH
SCON.7	SM0	Serial Mode	9FH
IE.0	EX0	External Interrupt 0 Enable	0A8H
IE.1	ET0	Timer 0 Overflow-Interr. Enable	0A9H
IE.2	EX1	External Interrupt 1 Enable	0AAH
IE.3	ET1	Timer 1 Overflow-Interr. Enable	0ABH
IE.4	ES	Serial Port Interrupt Enable	0ACH
IE.5	ET2	Timer 2 Overflow-Interr. Enable	0ADH
IE.7	EA	All Interrupts Enable	0AFH
IP.0	PX0	External Interrupt 0 Priority	0B8H
IP.1	PT0	Timer 0 Interrupt Priority	0B9H
IP.2	PX1	External Interrupt 1 Priority	0BAH
IP.3	PT1	Timer 1 Interrupt Priority	0BBH
IP.4	PS	Serial Port Interrupt Priority	0BCH
IP.5	PT2	Timer 2 Interrupt Priority	0BDH
T2CON.0	CP/RL2	Capture/Reload Flag	0C8H
T2CON.1	C/T2	Timer/Counter Select	0C9H
T2CON.2	TR2	Timer 2 Run-Controllbit	0CAH
T2CON.3	EXEN2	External Timer 2 Enable-Flag	0CBH
T2CON.4	TCLK	Transmittclock-Flag	0CCH
T2CON.5	RCLK	Receiveclock-Flag	0CDH
T2CON.6	EXF2	Timer 2 External-Flag	0CEH
T2CON.7	TF2	Timer 2 Overflow-Flag	0CFH
PSW.0	P	Parity Flag	0D0H
PSW.1	---	- reserved -	----
PSW.2	OV	Overflow Flag	0D2H
PSW.3	RS0	Register-Bank Select	0D3H
PSW.4	RS1	Register-Bank Select	0D4H
PSW.5	F0	Flag 0	0D5H
PSW.6	AC	Auxiliary Carry Flag	0D6H
PSW.7	CY	Carry Flag	0D7H

Die reservierten Symbole können an jeder Stelle als Argument angegeben werden. Der Assembler setzt automatisch die richtige Adresse für das Symbol ein.

Einschränkung!!: Die reservierten Symbole können nicht in EQU-Anweisungen verwendet werden. Dies würde zu der Fehlerausgabe "Unbekanntes Label oder Symbol" führen.

```
Motor   EQU  P0
MOV     A,Motor     ->   ist nicht möglich

Motor   EQU  080H
MOV     A,Motor     ->   ist möglich

MOV     A,P0        ->   ist möglich
```

Grund: Im ersten Pass sind die Labels (auch EQU-Definitionen) noch nicht bekannt. Daher kann die Prozedur HL5-8052 noch nicht die zugehörigen Adreßwerte einsetzen, da sie z.B. das Symbol Motor nicht kennt. Erst im zweiten Pass sind die Labels bekannt. Jetzt wird aber die Prozedur nicht mehr aufgerufen, da alle Befehle schon übersetzt sind. Für das Hauptprogramm ist dann aber bei der Zuweisung von P0 zu Motor, P0 nicht bekannt.

4. INLOAD52

Dieses Programm ist ähnlich aufgebaut, wie das in der INLASS-Beschreibung beschriebene INLOAD für den Z80. INLOAD52 macht aus einem mit INLASS assemblierten 8052 Programm ein .COM-File mit Hexcodes ohne Trennzeichen. Dieses ist dann unverschiebbar, wenn absolute Sprünge verwendet wurden. Es wird daher nach Aufruf dieses Programms die Startadresse verlangt. Diese wird in hexadezimaler Form eingegeben ($xxxx). In dezimaler Form werden nur Zahlen bis 32767 akzeptiert. Wird nur Return eingegeben, so wird - wenn vorhanden - die mit ORG definierte Start-Adresse eingesetzt. Ansonsten wird $0000 als Startadresse verwendet.

5. Änderung der bestehenden Include-Files

Folgende Änderungen mußten vorgenommen werden, um die Prozedur HL5-8052 einzubinden. Die Zeilennummern sind dem Listing entnommen.

```
1. File:   INLASS.PAS
   Zeile:  28
    " Prozessor:
(MPC_Z80,MPC_6502,MPC_8080,MPC_8086,MPC_8052);"
   Zeile:  40
    " {$I HL5-8052.INC} "

2.  File:   DOPASS.INC
    Zeile:  nw. 25 u. 26 löschen
    Zeile:  105
     " ELSE IF o='.8052' THEN PseudoOp:=p8052 "
    Zeile:  204
     " p8052: Prozessor := MPC_8052; "
    Zeile:  231
     " MPC_8052 : HexLine5(OPCODE,Arg1,Arg2,L)
```

Folgende Änderung mußte auf Grund des schon in Punkt 1. erwähnten speziellen Adreßformats des MCS-51 vorgenommen werden:

```
File:   INL.INC
Zeile:  79
  " IF fc=0 THEN IF Prozessor=MPC_8052 THEN "
  "   z:=z+'$'+HexByte(Hi(e))+'/$'+HexByte(Lo(e)) ELSE "
```

6. Die Schnittstelle zwischen Hauptprogramm und Prozedur

Die Prozedur (HL5-8052) erhält vom Hauptprogramm folgende Parameter:

```
Object
Argument1
Argument2
```

Die Prozedur gibt an das Hauptprogramm einen String mit dem übersetzten Hexcode zurück der folgende Form haben muß:

```
'$'+Hexcode+'/$'+Hexcode+.....
```

Soll ein Argument zurückgesendet werden, so muß nicht getestet werden, ob dies eine Zahl oder ein Symbol ist, sondern es muß lediglich angegeben werden, ob es, je nachdem wie es das Objekt verlangt, ein 8-bit Argument oder ein 16-bit Argument sein soll. Das Hauptprogramm prüft dann, ob das Argument dieser Forderung entspricht. Die Argumentgröße wird folgendermaßen gekennzeichnet:

```
/<   8-bit Argument
/    16-bit Argument
```

Ein 3-Byte Befehl `obj Arg1,Arg2` muß dann in folgender Form an das Hauptprogramm zurückgegeben werden:

```
'$'+Objcode+'/<'+Arg1+'/<'+Arg2
```

wobei die Reihenfolge von `Arg1` und `Arg2` vom Prozessor und dem jeweiligen Op-Code abhängt. Wird aber schon in der Prozedur sichergestellt, daß sich ein Argument um eine erlaubte Zahl handelt, so kann auch das Argument in der Form '$'+Hexcode zurückgesendet werden.

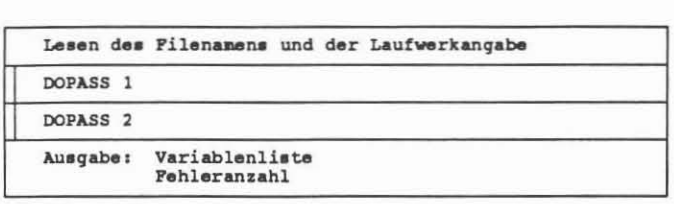
Relative Adressen (Labels) müssen mit

```
'/-'+Adresse bzw. Label
```

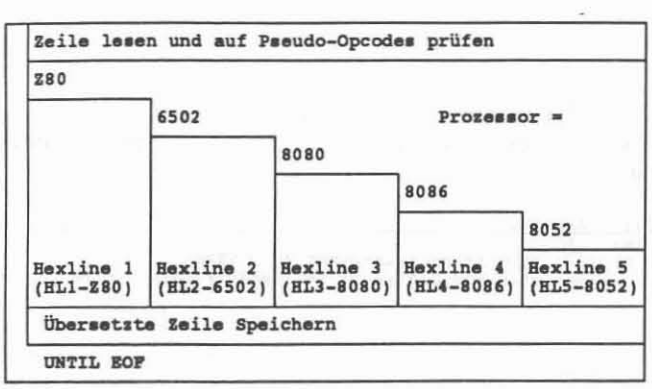
zurückgesendet werden. Absolute Adressen werden wie obige Argumente behandelt.

7. Struktogramme

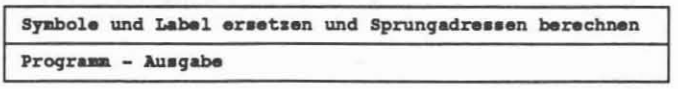
7.1 INLASS



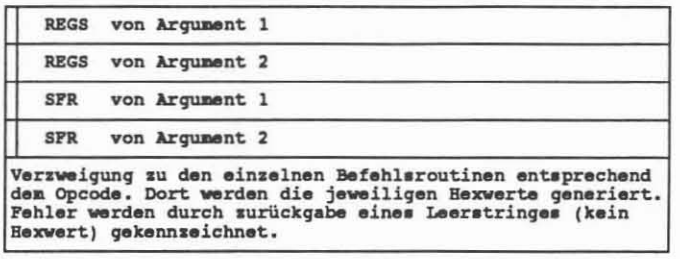
7.2 DOPASS1



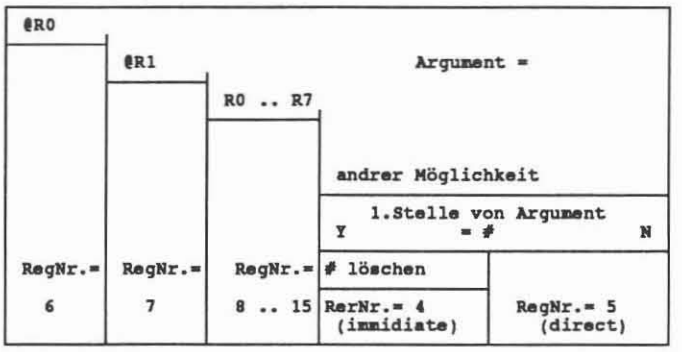
7.3 DOPASS2



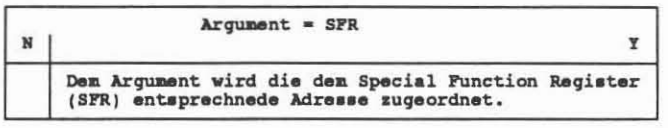
7.4 HEXLINE5



7.5 REGS



7.6 SFR



8. Programmbeschreibung HL5-8052

```
Eingangsvariablen:  o          ->  Mnemonics
                    a1,a2,(a3) ->  Argumente
Ausgangsvariable:   H          ->  Hexcodezeile
```

(F) bzw. (P) bei den Programmnamen bedeuten 'Funktion' und 'Prozedur'.

8.1 OPCodeNummer (F)

```
Aufruf: on:=OpCodeNummer(o)
```

Diese Funktion vergleicht die Eingangsvariable `o` mit den erlaubten Mnemonics des 8052. Bei Gleichheit wird der Variablen `on` je nach Mnemonics ein Wert zwischen 1 und 41 zugeordnet. Trifft kein Vergleich zu, dann wird `on` gleich Null (-> unbekanntes Mnemonics). Die Vergleichs-Mnemonics sind im Konstanten-Feld `OpCodeName` in alphabetischer Reihenfolge gespeichert. Dadurch kann eine sehr schnelle Vergleichsroutine verwendet werden, die mit größer/kleiner Vergleichen arbeitet. Es wird immer verglichen, in welcher Hälfte des Konstantenfeldes das gesuchte Mnemonics liegt. Diese Hälfte wird dann wieder halbiert, und wie vorher verfahren, bis das gesuchte Mnemonics gefunden ist.

8.2 SFR (F)

```
Aufruf: a:=SFR(a)
```

Diese Funktion vergleicht, ob das Argument ein reserviertes Symbol, d.h. ein Special Function Register ist. Das Argument `a` wird nacheinander mit den `SFR`-Namen verglichen. Bei Gleichheit wird die zugehörige Adresse aus dem Feld `SFRAdr` statt dem `SFRNamen` zurückgegeben.

8.3 Regs (P)

```
Aufruf: Regs(a,R)
```

Diese Prozedur prüft, ob das Argument ein Register, ein indirekter Registerbefehl, ein direktes oder ein immediate Argument ist.

```
R0 - R7         Register
@R0, @R1        indirekt
#xxxx           immediate
xxxx            direkt
```

Je nach den oben angeführten Fällen wird `R` ein bestimmter Wert zugeordnet. Die Werte ergaben sich aus dem Bitaufbau der OpCodes des 8052:

```
Argument        OpCode          Wert
-------------------------------------
R0 - R7         yyyy 1rrr   ->  8 - 15
@R0, @R1        yyyy 011r   ->  6, 7
#xxxx           yyyy 0100   ->  4
xxxx            yyyy 0101   ->  5
```

`yyyy` richtet sich nach dem OpCode. Dieser Bit-Aufbau ist nur bei manchen OpCodes anders, die dann gesondert behandelt werden müssen. Der Vorteil dieser Zuordnung ist, daß dann unabhängig vom Argument der High-Teil des OpCodes nur mit `R` verort werden muß. -> z.B. `$20 or R`. Dies wird z.B. in der folgenden Prozedur `AOP` verwendet. `xxxx` kann eine beliebige 8bit oder 16bit-Zahl sein oder ein Label oder Symbol.

8.4 AOP (A-Operation) (P)

```
Aufruf: AOP(Hex,H)
```

Diese Prozedur prüft zuerst, ob das 1.Argument der Akkumulator ist. Wenn ja, bildet sie nach vorher erwähnter Methode zusammen mit der übergebenen Zahl `Hex` den OpCode und legt diesen in `H` ab. Ist `R2` (Index2 für 2.Argument) aus vorheriger Prozedur kleiner 6 (-> #xxxx oder xxxx), dann muß noch das 2.Argument `xxxx` an `H` angehängt werden. Da die "A,"-Befehle nur mit 8bit-Argumenten arbeiten muß vor `xxxx` noch `/<` eingefügt werden (`xxxx` kann eine Zahl oder ein Symbol sein). Die verwendete Prozedur `HexByte(x)` erzeugt aus einer Integer einen String.

8.5 HLogik (P)

```
Aufruf: HLogik(Hex,H)
```

Diese Prozedur wird von den Logik-Befehlen `ANL`, `ORL` und `XRL` verwendet. Zuerst wird die Prozedur `AOP` aufgerufen, da diese auch für alle Logikbefehle mit `A` gilt. Lag kein `A`-Befehl vor, so sendet die Prozedur `AOP` einen Leerstring zurück `H=""`. Ist dies der Fall, so muß Argument 1 ein direct-Argument `R1=5` sein, ansonsten liegt eine unerlaubte Anweisung vor. Ist nun `R1=5`, so gibt es nur mehr 2 Möglichkeiten. Entweder Argument 2 ist `A` oder ein "immediate"-Argument. In beiden Fällen wird dann der Opcode wie in der Prozedur `AOP` gebildet und in `H` abgelegt.

8.6 HBit (P)

```
Aufruf: HBit(Hex,a,H)
```

Diese Prozedur prüft, ob das übergebene Argument a ein erlautes reserviertes Bitsymbol ist. In `Hex` wird wieder der Hexcode des Mnemonics übergeben, um dann damit den gesamten Opcode zu bilden und in `H` abzulegen. Der Befehl `pos('.',a)` sucht in dem String `a` nach dem Zeichen `"."` und übergibt die Zeichenstelle, an der dieses Zeichen gefunden wurde (nicht gefunden bedeutet 0). Die Stelle wird in `s` gespeichert.

1) Wurde ein "." gefunden (S>0) liegt eine reservierte Bit-Adresse vor:

```
Symbol.x            x ... 0 - 7
```

Der Befehl `copy(a,1,S)` kopiert vom String `a` `s` Zeichen ab dem 1.Zeichen in `b`. Dadurch ist nun in `b` nur das Symbol vorhanden ohne `.x`. Mit dem Befehl `delete(a,1,S)` wird ab dem 1.Zeichen `s` Zeichen von `a` weggeschnitten. Dies bedeutet, daß nur mehr `x` in `a` bleibt. Da `a` ein String ist, muß mit dem Befehl `val(a,bit,error)` daraus eine Integer-Zahl gemacht werden, um `x` als Zahl zu haben. Dabei wird in `bit` die Zahl übergeben und in `Error` der Fehler wie folgt übergeben:

```
String      bit   Error                 B ... Buchstaben
-------------------------------         Z ... Ziffern
Z           Z     0
B           0     1
BZ          0     1
ZB          Z     1.Stelle v. B
Z1BZ2       Z1    1.Stelle v. B
```

Nur wenn `Error` gleich Null ist und `bit` 0 - 7 ist, war die Syntax richtig. Es wird nun das Symbol `b` mit den Elementen der Konstante `BSFRName` verglichen. Trifft ein Vergleich zu, so wird zur Startadresse der `BSFR` ($80) die Stelle an der das Symbol gefunden wurde `s` mal 8 addiert, da jedes `BSFR` 8 Bit beinhaltet. Die Bitstelle wird durch `bit x` angegeben und wieder addiert. Das Ergebnis ist die Bitadresse für das Eingegebene Symbol `x`.

2) Wurde kein "." gefunden, so liegt eine direkte reservierte Bit-Adresse oder ein mit `EQU` definiertes Symbol vor. Im ersten Fall wird String `a` mit den Elementen der Konstanten `BitsFRName` verglichen. Bei Gleichheit wird das zugehörige Element der Konstanten `BitsFRAdr` zur OpCode-Bildung herangezogen. War kein Vergleich zutreffend, so liegt direkt eine Adresse vor, oder ein Symbol, das aber mit `EQU` definiert werden muß. Es wird daher in diesem Fall das Argument `a`, so wie es war, an das Hauptprogramm zurückgesendet, zusammen mit dem "<" damit eine 8-Bit Adresse verlangt wird.

8.7 Cbit (P)

```
Aufruf: Cbit(Hex,H)
```

Diese Prozedur wird von den Bit-Befehlen mit C (ANL, ORL) verwendet. Ist das 1.Argument nicht "C", dann liegt ein Logik-Befehl vor, und es wird die Prozedur HLogik aufgerufen, an die der Hexcode unverändert weitergegeben wird. Ist `a1` aber `c` dann wird `Argument2` untersucht, ob es ein `/` enthält. Dieses wird dann abgetrennt und zusammen mit dem dazugehörigen Hexcode die Prozedur `HBit` aufgerufen.

BEFEHLSBESONDERHEIT: Der Befehl CJNE benötigt 3 Argumente, da er wie folgt aufgebaut ist:

```
CJNE        a1,a2,a3
```

Das Hauptprogramm trennt zwar das erste Komma weg und zerteilt in `a1` und `a2`, aber prüft nicht, ob noch ein zweites Komma folgt. (Prüft nur auf ein folgendes Semikolon zur Kommentartrennung). Es wird daher `a2, a3` in dem `Argument2` vom Hauptprogramm übergeben. Es muß daher nun `a2` erst in `a2` und `a3` aufgeteilt werden. Dies geschieht dadurch, daß mit `pos` die Stelle des Kommas in `a2` gesucht wird und dann der Teil hinter dem Komma mit `copy` in `a3` kopiert wird und der hintere Teil mit dem Komma dann aus `a2` gelöscht wird.

8052 - DISASSEMBLER

Paul KOSTAL, Peter ULLRICH, TGM,NT,SLME86 DISAS52.PAS, DIASSEM.INC, DISAS52.COM

Mit DISASS51 kann von jedem beliebigen COM-File ein Hexdump erstellt werden oder dieses disassembliert werden, wenn es im MCS-51 Code geschrieben ist.

Der Aufruf erfolgt folgendermaßen:

```
DISAS-52  filename
```

Der Filename wird ohne Extension angegeben, da immer ein COM - File gelesen wird. Wird kein Filename angegeben, so wird dieser nachträglich vom Programm angefordert. Es können aufgrund der im Pascal verwendeten INTEGERS nur Programme bis zu einer Länge von 32767 Bytes eingelesen werden.

Als Nächstes muß die Adresse angegeben werden, auf die das eingelesene Programm disassembliert werden soll.

Folgendes Menü wird nun angezeigt:

```
                Disassembler-Befehle
                --------------------


D               Disassembliere ab Adresstand
D XXXX          Disassembliere ab Adresse xxxx
M               Hexdump ab Adresstand
M XXXX          Hexdump ab Adresse xxxx
^S              Anhalten/Weiter
RETURN          Stopt Funktion
PgUp            Seite vor
PgDn            Seite rückwärts
Home            Anfang
End             Ende
? od. a         Zeigt dieses Menü
QUIT            Ausstieg
```

Ist man in dem Menüpunkt Disassembliere, so kann man die Steuertasten PgUp, Home, End nur begrenzt verwenden:

Home: Der Disassembler beginnt nicht am Programmanfang zu disassembieren, sondern ab der Adresse, die beim Aufruf dieses Menüpunkts gerade aktuell war (oder die durch xxxx angegebene).

End: Der Disassembler springt nicht an das Programmende, sondern an die bis zu diesem Zeitpunkt höchste, vorgekommene Adresse (Diese wird durch Drücken von PgDn erhöht). Ein Verlassen dieses Menüpunktes löscht aber wieder diesen Wert.

PgUp: Diese Funktion wird nur solange ausgeführt, bis die Adresse der Home - Taste erreicht ist.

Grund für die eingeschränkte Verwendung der Steuertasten: Normalerweise kann ein Disassembler nicht zu kleineren Adressen hin disassemblieren, da die Byteanzahlen sowie der OpCode selbst hier nicht eindeutig sind. So würde der Befehl:

```
74H 00H MOV A,#0
```

verkehrt gelesen ein NOP ergeben.

Der Disassembler speichert daher alle ersten Adressen pro disassemblierter Seite ab. Die erste abgespeicherte Adresse wird für die Home-Taste und die letzte abgespeicherte Adresse für die End-Taste verwendet. Die PgUp-Taste nimmt jeweils die nächstkleinere abgespeicherte Adresse, solange bis die erste erreicht ist.

1. Reservierte Symbole

Der Disassembler DISASS52 hat alle Special-Function-Register des 8052 implementiert. Tritt eine 8bit-Datenadresse auf, die auf ein SFR zutrifft, so wird dieses SFR-Symbol als Kommentar in Klammer an den jeweiligen Befehl angehängt. Handelt es sich bei dem Befehl um einen Bit-Befehl, so wird als Kommentar die zugehörige Byteadresse und die Bitposition (0 - 7) ausgegeben oder, falls es sich um ein fixes Bitsymbol handelt, dieses zugeordnet.

2. Änderungen gegenüber den 8052-Mnemonics

Da dieses Programm zusammen mit dem Inlass-8052 entwickelt wurde, wurden hier fast die gleichen Änderungen vorgenommen:

```
1. LCALL und LJMP  ->  CALL und JMP
2. SJMP            ->  JR
```

Im Gegensatz zum Inlass wurde hier als Negationszeichen wieder das "/" verwendet und die Befehle ACALL und AJMP wieder berücksichtigt, da diese hier keine Schwierigkeiten bereiten.

3. Relative Sprünge

Alle relativen Sprungadressen und auch die ACALL- und AJMP-Adressen werden sofort berechnet und als absolute Adresse ausgegeben. Dies hat den Vorteil der besseren Übersicht, und man erspart sich das mühsame Berechnen der Adressen, um einen Programmverlauf zu verfolgen. Der relative Sprungwert ist außerdem auch aus den Hexcodes ersichtlich.

4. Beispielausdrucke

Das Beispielprogramm BEISP wurde folgendermaßen erstellt:

```
1. Texteditor    ->   BEISP.SOU
2. INLASS        ->   BEISP.PRN
                 ->   BEISP.INL
3. INLOAD52      ->   BEISP.COM
4. DISAS-52
```

BEISP.SOU

```
        .8052
STROUT  EQU     6
        ORG     3000H
;
        MOV     A,#STROUT       ;Ausgabe eines Strings
        MOV     R3,#HI(MLDG)    ;HighAdresse des Strings
        MOV     R1,#LO(MLDG)    ;LowAdresse des Strings
        CALL    30H             ;Befehl ausführen
        RET                     ;Zurück zum Basic
;
MLDG    DB  'Dies ist ein Beispiel'
        DB  00H
```

BEISP.PRN

```
PASS 1
0 Fehler aufgetreten.
PASS 2
*0000                    .8052
*0000 (0006)                STROUT    EQU 6
*0000                                ORG 3000H
*3000
*3000 /$74/$06                       MOV A,#STROUT      ;Ausgabe eines Strings
*3002 /$7B/$30                       MOV R3,#HI(MLDG)   ;HighAdresse des String
*3004 /$79/$0A                       MOV R1,#LO(MLDG)   ;LowAdresse des String
*3006 /$12/$00/$30                   CALL 30H           ;Befehl ausführen
*3009 /$22                           RET                ;Zurück zum Basic
*300A
*300A /$44/$69/$65/$73/$20/$69       MLDG     DB 'Dies ist ein Beispiel'
*3010 /$73/$74/$20/$65/$69/$6E
*3016 /$20/$42/$65/$69/$73/$70
*301C /$69/$65/$6C
*301F /$00                           DB 00H
0 Fehler aufgetreten.

MLDG      * 300A
STROUT    = 0006
```

Ausdrucke des DISAS-52

(Alle Zahlenwerte sind Hexadezimal!!)

1. Hexdump

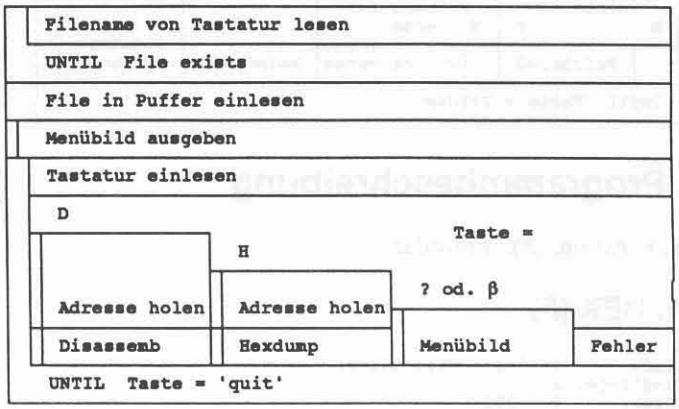
```
3000  74 06 7B 30 79 0A 12 00 30 22 44 69 65 73 20 69
      t  .. {  0  y  .. .. .. 0  "  D  i  e  s     i
3010  73 74 20 65 69 6E 20 42 65 69 73 70 69 65 6C 00
      s  t     e  i  n     B  e  i  s  p  i  e  l  ..
3020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
3030  00 00 00 00 00 00
      .. .. .. .. .. ..  .....
```

2. Disassembler

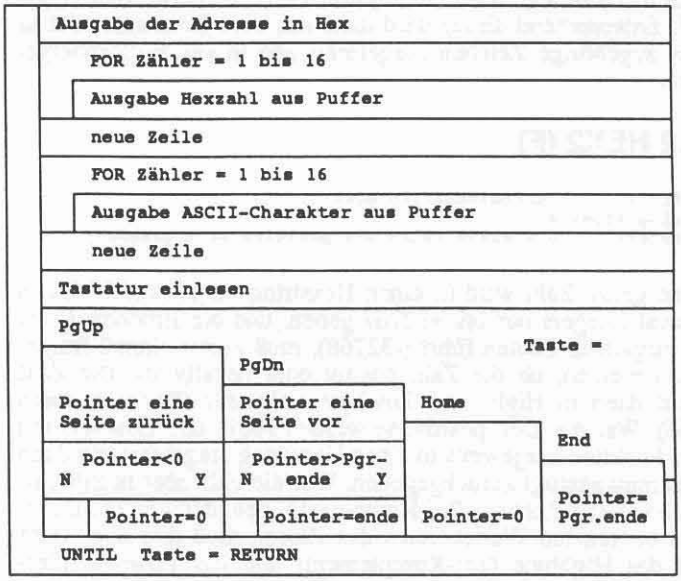
```
3000  74 06      MOV   A,#06                    t
3002  7B 30      MOV   R3,#30                   {0
3004  79 0A      MOV   R1,#0A                   y
3006  12 00 30   CALL  0030                      0
3009  22         RET                            "
300A  44 69      ORL   A,#69                    Di
300C  65 73      XRL   A,73                     es
300E  20 69 73   JB    69,3084     ;(2D.1)      is
3011  74 20      MOV   A,#20                    t
3013  65 69      XRL   A,69                     ei
3015  6E         XRL   A,R6                     n
3016  20 42 65   JB    42,307E     ;(28.2)      Be
3019  69         XRL   A,R1                     i
301A  73         JMP   @A+DPTR                  s
301B  70 69      JNZ   3086                     pi
301D  65 6C      XRL   A,6C                     el
301F  00         NOP
3020  00         NOP
3021  00         NOP
3022  00         NOP
3023  00         NOP

ASCII-Text:
t {0y   0"Dies ist ein Beispiel
```

Da der Disassembler zwischen Daten- und Programmbereich nicht unterscheiden kann, werden OpCodes auch als ASCII und ASCII-Daten (oder andere Daten) auch als OpCodes dargestellt. (siehe die ASCII-Daten: "Dies ist ein Beispiel" bei denen die danebenstehenden Befehle sinnlos sind. Genauso die neben dem Programm stehenden Buchstaben.)

5. Struktogramm



HEXDUMP



DISASSEMB

6. Programmbeschreibung

(F)...Funktion, (P)...Prozedur

6.1. HEX (F)

```
Aufruf:       String:=HEX(Integer)
Stringlänge: 2
Integer:      0 - 255
```

Eine ganze Zahl kleiner 256 wird in einen Hexstring umgewandelt. Die Zahl wird in High- und Lownibble (Division durch 16) zerlegt. Man erhält jeweils eine Zahl zwischen 0 und 15. Entsprechend dieser wird dann aus dem Konstantenfeld `hd` das zugehörige Zeichen ausgelesen und in `HEX` zurückübergeben.

6.2 HEX2 (F)

```
Aufruf:       String:=HEX2(Integer)
Stringlänge: 4
Integer:      0 - 65535 (wird als positive Zahl gesehen)
```

Eine ganze Zahl wird in einen Hexstring umgewandelt. Da in Pascal Integers nur bis +32767 gehen, und ein Erhöhen dieser zu negativen Zahlen führt (-32768), muß zuerst einmal festgestellt werden, ob die Zahl positiv oder negativ ist. Die Zahl wird dann in High- und Lowbyte aufgeteilt (Division durch 256). War die Zahl positiv, so werden High- und Lowbyte mit der Funktion `HEX` jeweils in einen Hexstring umgesetzt und dann zusammengefügt zurückgegeben. War die Zahl aber negativ, so muß von der Zahl das 2er-Komplement gebildet werden. Durch den begrenzten Wertebereich des Pascal, muß dies aber extra für das Highbyte (1er-Komplement) und das Lowbyte (2er-Komplement) erfolgen, wobei der Übertrag zu beachten ist.

6.3 GetLoHi (P)

```
Aufruf:       GetLoHi(String,Integer1,Integer2)
Stringlänge: >4          eingang
Integer:      0 - 255    ausgang
```

Der String wird in eine positive Zahl umgewandelt. Auf Grund des begrenzten Wertebereiches für Integers werden Highbyte und Lowbyte getrennt umgewandelt und zurückgegeben.

6.4 GetAdr (P)

```
Aufruf:  GetAdr
Var. Befehl:   String:  1 Steuerzeichen     (Adresse)
```

Diese Routine prüft, ob die Variable `Befehl` eine Adresse beinhaltet. Enthält sie keine, oder lautet der Befehl 'quit' so wird diese Routine beendet. Ansonsten wird mit `GetLoHi` die Adresse geholt und verglichen, ob sie kleiner als die Startadresse des eingelesenen Programms ist. Wenn ja, wird die Adresse gleich 0 gesetzt, da intern der Zähler von 0 an läuft. War die Adresse größer, so wird die Startadresse des eingelesenen Programms abgezogen. (Grund wie vorher -> Zähler von 0 weg).

6.5 GetBefehl (P)

```
Aufruf: GetBefehl
```

Diese Routine erzeugt einen blinkenden Pfeil und liest anschließend von der Tastatur in den String `Befehl` ein. Von diesem werden dann alle Spaces gelöscht und das erste Zeichen (Steuerzeichen) in die Variable `Taste` gespeichert und in einen Großbuchstaben umgewandelt.

6.6 Hexdump (P)

```
Aufruf: Hexdump
```

Diese Routine erzeugt einen Hexdump (24 Zeilen). Wurde als Hexdumpadresse eine größere angegeben, als das Programm lang ist, so wird die Adresse auf die letzte Adresse gesetzt, die noch einen vollen Bildschirm `Hexdump` ermöglicht. Diese Adresse ergibt sich aus der letzten (höchsten) Adresse - 12 Zeilen * 16Bytes. Es folgt die Ausgabe der Adresse und der 16 Hexbytes und dann die Ausgabe der 16 zugehörigen ASCII-Zeichen, die wieder auf druckbare Zeichen geprüft werden. Ist die 24.Zeile erreicht, so wird die Tastatur abgefragt. Erfolgt eine unzulässige Eingabe (z.B. Drücken von `PgUp` wenn man schon auf der erstmöglichen Adresse ist), so wird die Ausgabe gesperrt (`ausg:=false`), und es erfolgt eine neue Tastaturabfrage. Dies geschieht solange, bis mit `RETURN` die Routine abgebrochen wird.

6.7 Disassem (P)

```
Aufruf: Disassem
```

6.7.1 Zeile (F)

```
Aufruf:          String1:=Zeile(Integer,String2)
Stringlänge1,2:  3
Integer:         0 - 65535 (pos.)
```

Es wird ein String zurückgegeben, der das Hexbyte der Adresse (Integer) enthält. In `String2` wird der ASCII-Charakter dieses Hexbytes zu den Bestehenden in `String2` dazugefügt und zuerst auf druckbare Zeichen geprüft. Weiters wird der Adreßzähler um eins erhöht.

6.7.2 SFR8 (F)

```
Aufruf:       String:=SFR8(Integer)
Stringlänge: 6
Integer:      1,2
```

Diese Routine vergleicht den Hexwert mit den Special-Function-Registern SFR und ordnet dem String entweder ein SFR oder einen Leerstring zu. Einem Wert kleiner 128 wird sofort ein Leerstring zugewiesen, da die SFR erst bei 128 beginnen. Der Wert Integer wird von dem derzeitigen Stand des Adreßzählers abgezogen (Da der Adreßzähler schon auf den nächsten Befehl zeigt). Das Ergebnis weist dann auf das gewünschte Hexbyte. Integer gibt daher an, ob das letzte oder vorletzte Byte verglichen werden soll.

6.7.3 SFR1 (F)

```
Aufruf:       String:=SFR1(Integer)
Stringlänge:  6
Integer:      1,2
```

Diese Routine vergleicht den Hexwert mit den Bit-Symbolen und ordnet dem String entweder ein Bit-Symbol oder einen Leerstring zu. Bei einem Wert kleiner 128 wird dem String die Byteadresse zugeordnet, die aus dem Hexwert (dieser ist in diesem Fall eine Bitadresse) errechnet wird (Bitwert/8 + 20h). Der Wert Integer wird von dem derzeitigen Stand des Adreßzählers abgezogen, da der Adreßzähler schon auf den nächsten Befehl zeigt. Das Ergebnis weist dann auf das gewünschte Hexbyte. Integer gibt daher an, ob das letzte oder vorletzte Byte verglichen werden soll.

6.7.4 Register (F)

```
Aufruf:       String:=Register(Integer)
Stringlänge:  4
Integer:      0 - 15
```

Diese Routine liefert abhängig vom Integerwert entweder ein Register (8-15), Indirekt Register (6,7), immediate Data (4) oder Data (5) zurück.

6.7.5 SprAdr (F)

```
Aufruf:       String:=SprAdr(Integer)
Stringlänge:  4
Integer:      0 - 224  step 32  (-> nur Bits 5-7)
```

Diese Routine berechnet die absolute Sprungadresse für ACALL- und AJMP-Befehle. Die Adresse ergibt sich folgendermaßen:

```
(Bits 11-15 des Adresszählers)+(Bits 5-7 des OpCode)+Argument
Adr.: 15-11                    10-8                 7-0
```

Der OpCode wird mit Integer übergeben und das Argument aus dem Pufer+1 geholt. Durch Division des Opcodes durch 32 werden die Bits 5-7 auf 0-3 transferiert und durch Multiplikation mit 256 auf die Bits 10-8. Die Adresse wird mit 11111000 00000000B verundet, um nur die Adreßbits 11-15 zu haben.

6.7.6 RelAdr (F)

```
Aufruf: RelAdr
```

Diese Routine berechnet die absolute Sprungadresse für relative Sprünge. Je nach dem, ob der Hexwert (rel. Adr) >127 ist oder nicht, wird der Wert zum Adreßstand addiert oder sein 2er Komplement subtrahiert.

6.7.7 OneByte - TwoByte - ThreByte (P)

```
Aufruf: Onebyte(String), Twobyte(String), Threbyte(String)
Stringlänge: 10
```

Es wird die Funktion Zeile so oft, wie die Byteanzahl aufgerufen und der Rest mit Spaces aufgefüllt, sodaß alle 3 Prozeduren einen gleich langen String liefern, der die Hexbytes des Befehls enthält. Zusätzlich wird durch die Funktion Zeile ein String ASCII gebildet, der die ASCII-Zeichen dieser Bytes beinhaltet.

Hauptprogramm von Disassemb

Es wird zuerst der Opcode entsprechend der derzeitigen Adresse eingelesen. Der Programmteil wird aufgeteilt in ACALL- und AJMP-Befehle, in Registerbefehle und in die restlichen Befehle.

Die ACALL- und die AJMP-Befehle werden dadurch erkannt, daß die Bits 0-4 gleich 00001b=1 (->AJMP) oder gleich 10001b=17 (->ACALL) sind.

Die Registerbefehle werden dadurch erkannt, daß das Lownibble des OpBytes = 6-15 ist. Ist dies der Fall, so wird nur für jede Registergruppe mit Hilfe des Highnibble des OpBytes der Befehl decodiert.

Alle anderen OpBytes werden direkt verglichen und einzeln decodiert.

Ist die 21.Zeile erreicht, so wird der ASCII-Text aller Hexcodes am Bildschirm ausgegeben. Danach wird die Tastatur abgefragt. Wurde PgDn gedrückt, so wird ab der letzten Adresse weiter disassembliert und diese Adresse in einem Puffer gespeichert, um für die Steuertaste PgUp zur Verfügung zu stehen, welche eine 'Seite' zurückblättert. Erfolgt eine unzulässige Eingabe (z.B. Drücken von PgUp, wenn man schon auf der erstmöglichen Adresse ist), so wird die Ausgabe gesperrt (ausg:=false), und es erfolgt eine neue Tastaturabfrage. Dies geschieht solange, bis mit RETURN die Routine abgebrochen wird.

EDV...Elektronische Datenvernichtung

EDV...Ende der Vernunft

Schnittstelle PC <-> 8052

Paul KOSTAL, Peter ULLRICH: TGM, NT/EL, SLME86 CENTR52.PAS, TGM-DSK-169: CENT52.*

Der 8052-BASIC ist für die Bedienung mit einem Terminal ausgelegt. Da der 8052-BASIC keinen Anschluß für ein magnetisches Speichermedium hat, wurde als Terminal ein IBM-Computer gewählt, der mit Hilfe des Programms 'VTERM' betrieben wird und so die Möglichkeit bietet, Programme des 8052-BASIC abzuspeichern oder zu laden.

Die serielle Verbindung

Die Einstellung des IBM

Nach Laden des Programms VTERM kommt man durch Drücken von F5 ins SETUP - Menü. VTERM wurde folgendermaßen eingestellt:

```
S E T U P   1 - Terminal, Communications and Printer    Setup Name=VTERM

 TERMINAL     Type: VT100 (ANSI)   COMMUNICATIONS Rate: 19200
         New line: OFF                 Data bits/parity: 8 NONE
      Wrap around: ON                   Receive parity: IGNORE
           Scroll: NORMAL                    Stop bits: ONE
           Screen: NORMAL VIDEO          Auto xon/xoff: ON
      Cursor type: UNDERLINE                Local echo: OFF
      Margin bell: OFF                        Com port: 1
      DA response: ADVANCED VIDEO        On line/local: ON LINE
      Status line: ON
  Backspace sends: DELETE
 Shift-3 displays: #
    Screen format: 80 x 24          PRINTER   Output to: test
132 column method: SCROLLING                After PrtSc: NO FORMFEED
  Window movement: MANUAL                       Extent: ENTIRE SCREEN
Scrollback buffer: ALLOW OVERFLOW               Wide: OFF
```

Die Baudrate kann beliebig gewählt werden, da sich der 8052-BASIC auf alle Baudraten automatisch nach dem Einschalten einstellt. Dies erfolgt durch das Senden eines SPACE, anhand dessen der 8052 die Baudrate erkennt.

Inbetriebnehmen der Geräte

1. beide Geräte einschalten
2. IBM: VTERM
3. IBM: Space drücken
4. 8052 sendet Einschaltmeldung

1. Laden und Speichern von Programmen

1.1. Laden und Speichern von BASIC-Programmen

Das Laden und Speichern von Basic-Programmen erfolgt ebenfalls über die gleiche serielle Schnittstelle mittels der Funktionen Alt-S und Alt-R des VTERM. Leider funktionierte die serielle Schnittstelle zwischen IBM und 8052 nicht fehlerlos. Es muß daher im SETUP 2 Menü die Wartezeit nach jeder Zeile auf 3/10 sec eingestellt werden:

```
S E T U P    2 - File Transfer              Setup Name=VTERM

                          Protocol: ASCII TEXT

                  Maximum send rate: AT BAUD RATE
                   Wait after lines: FOR QUIET HOST
                   How many secs/10: 3
                        End of line: CR-LF
       Convert null lines to a space: NO
             Remove escape sequences: NO
                 Stop upon receiving:
```

1.1.1 Laden von BASIC-Programmen

1. Alt-S
2. Eingabe des Programmnamens
   (ASCII-TEXT)
3. Alt-S

Der 8052-Basicrechner bleibt im COMMAND-Mode. Dies ist der Zustand, in dem Programme eingegeben werden können. (Der zweite Mode ist der RUN-Mode).

Da der 8052 Basic einen sehr dürftig ausgefallenen Editor besitzt (Einzeileneditor), ist es günstiger, die BASIC-Programme mittels eines Textprogramms (z.B. Turbo oder WordStar) zu schreiben, abzuspeichern und dann mit VTERM in den 8052-BASIC-Rechner zu laden. Dazu muß aber nicht jedesmal VTERM neu geladen werden, sondern durch Drücken von 'SHIFT-SHIFT' kommt man ins System und wieder zurück zum VTERM.

```
1. VTERM
2. Shift-Shift  ->  System
3. TURBO
4. Programm editieren oder neues erstellen
5. abspeichern und quit
6. Shift-Shift  ->  VTERM
7. NEW eingeben um altes Programm im 8052 zu löschen
8. mit Alt-S Programm auf 8052 senden
9. mit RUN testen  (Basic-Befehl)
10. weiter mit Punkt 2
```

1.1.2 Speichern von BASIC-Programmen:

```
1. Alt-R
2. Programmnamen eingeben
3. Alt-R
4. LIST     (Basic-Befehl)
5. Alt-A    (Abort Transmission)
```

Da mit Alt-R alle ankommenden Zeichen unter den angegebenen Namen gespeichert werden, wird auch der Befehl LIST mitgespeichert. Dieser stört aber beim Laden des Programms nicht, kann aber mittels eines Texteditors gelöscht werden.

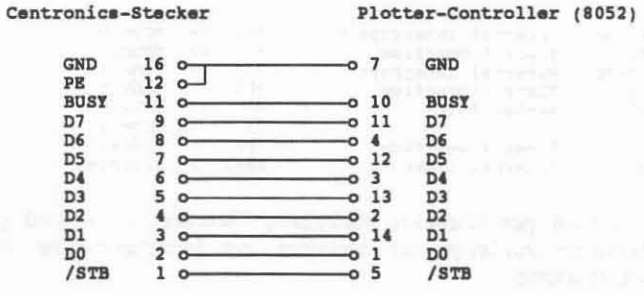
1.2. Laden von Assemblerprogrammen

Die einfachste Möglichkeit ist, die Hexcodes des Assemblerprogramms in DATA-Zeilen des BASIC-Textes zu verpacken und diese wie vorhergehend beschrieben zu laden und speichern. Es ergibt sich so aber der große Nachteil, daß das vom Assembler erzeugte Hexcode-File nicht direkt verwendet werden kann, sondern erst mühsam abgeschrieben und in DATA-Zeilen verpackt werden muß. Ein Beispiel ist im folgenden Kapitel gezeigt.

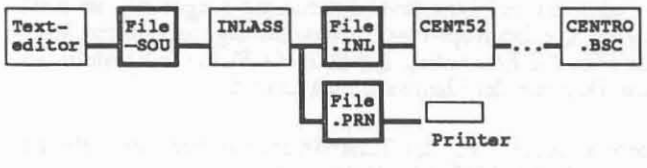
1.3. Laden von Assemblerprogrammen (Hexcodes) über Centronics

Da die serielle Schnittstelle nicht klaglos funktioniert, bei Assemblerprogrammen aber eine hohe Datensicherheit gefordert wird (ein Fehler führt zum Absturz des 8052-Basicrechners) wurde auf die Centronics-Schnittstelle übergegangen. Da die Centronics-Schnittstelle des Plotter Controller aber nur einlesen kann, konnte so nur das Laden von Assemblerprogrammen (assemblierte Hexcodes) realisiert werden.

Centronics-Schnittstelle:



Für das Überspielen von Programmen über Centronics sind nun 2 Hilfsprogramme notwendig. Das erste (CENT52) läuft am IBM und sendet über die Centronics, das zweite (CENTRO.BSC) läuft am 8052-Basicrechner und empfängt die Daten und speichert sie im RAM.



Der Überspielvorgang (vorausgesetzt VTERM ist geladen und CENTRO.BSC ist schon auf den 8052 überspielt):

1. RUN (starten von CENTRO.BSC am 8052-rechner) -> Einschaltmeldung
2. Eingabe der Anfangsadresse, auf die das Programm überspielt werden soll.
3. Shift-Shift -> System
4. CENT52 file (file ohne Extension = fix INL)
5. Eingabe der Adresse auf die relatives File verschoben werden soll. (ist gleich Adresse von Punkt 2)
6. Nach Beendigung der Überspielung :Shift-Shift -> VTERM
7. Ctrl-C zum Beenden von CENTRO.BSC
8. CALL adresse zum testen des Assemblerprogramms

1. CENT52.COM

Dieses Programm ist nur für das Senden von mit INLASS assemblierten Programmen geeignet (`File.INL`). Es ist gleich dem im Beitrag **INLASS-Inline-Assembler** beschriebenem INLOAD52 außer, daß statt der Erzeugung eines COM-Files hier die Daten über die Centronics-schnittstelle gesendet werden.

2. CENTRO.BSC

Dieses Programm liest interruptgesteuert die Centronics-Daten ein und speichert sie auf die angegebene Adresse. Es ist in BASIC geschrieben mit einer Interruptroutine in Maschinsprache die in DATA-Zeilen an das BASIC-Programm angehängt ist. CENTRO.BSC wird mit der vorher beschriebenen Methode (Alt-S) in den BASIC-Rechner geladen.

2. 8052-BASIC ASSEMBLERROUTINEN

```
Freier Speicherplatz im internen RAM:
     18h - 1Fh    =  R0 - R7 in Registerbank 3
     20h,21h      =  Bitadressabl
     ab 4Dh       =  Stack (Anfangsadresse durch 3Eh bestimmt;
                     ab 4Dh bis zum Wert in 3Eh ist freier
                     Speicherplatz auf Kosten des Stacks)
```

Aufruf der BASIC-Assembler - Routinen:

```
ANL  PSW,#11100111    ; Registerbank 0 !!!
MOV  A,#OPBYTE
CALL 30H
```

!!! Die BASIC-Assemblerroutinen verändern auch die Werte der Registerbänke (außer Bank 3) und DPTR, ... Diese müssen daher bei Bedarf zwischengespeichert werden. !!!

Jede Kommunikation mit den BASIC-Assemblerroutinen erfolgt über Registerbank 0 !!!

Bei jedem 'RUN' - Befehl wird der gesamte RAM-Speicher gelöscht und somit gehen alle darin befindlichen Assemblerprogramme verloren. Es muß daher zuerst die Systemvariable MTOP auf einen Adreßwert gesetzt werden, der unter dem der Assemblerprogrammen liegt, um dies zu verhindern. Es sollte daher nach dem Einschalten des Rechners immer sofort MTOP z.B. auf 2000h gesetzt werden, um darüber freien Raum für Maschinsprache zu haben. (darunter 200h - 2000h ist Platz für BASIC).

```
Rx,Ry      =  High,Low
(Rx,Ry)    =  Speicherstelle adressiert durch Rx,Ry
Stack      =  letzter Stackwert
Stack2     =  vorletzter Stackwert
```

2 Operanden Befehle

Ein `!` markiert, daß bei Fehler der Status im ACC übergeben wird, während sonst der Fehler vom BASIC gehandhabt wird -> Error Meldung am Bildschirm.

```
   Status:  ACC.0  -  Underflow
            ACC.1  -  Overflow
            ACC.2  -  Ergebnis 0 (kein Fehler)
            ACC.3  -  Division durch 0
Vergleiche:  true = 0ffffh
             false= 0
```

```
Code     Funktion                          {BASIC-Befehl}
-------------------------------------------------------
 00      Zurück zum COMMAND-Mode
 01      Stack    -> R3,R1                 {POP}
 02      (R2,R0)  -> Stack                 {PUSH}
 03      Stack    -> (R3,R1)               {POP}
 04      R3,R1 - Quelladresse              {PROG}
         R2,R0 - Zieladresse
         R7,R6 - Länge
 05      Eingabe eines Stirngs in den BASIC-Buffer
         (0007h). Ende mit Return.  {INPUT}
 06      Ausgabe eines Stirngs adressiert durch
         R3,R1                             {PRINT}
         BIT 52:  Set: Programm-Memory -> Ende mit
                       '"' = 22h
                  Cleard: Ext.Data-Memory -> Ende mit
                       <CR> = 0Dh
 07      Ausgabe: CR
 09      Stack2 ** Stack -> Stack          {EXP}
 0A      Stack2 * Stack -> Stack           {*}
 0B      Stack2 + Stack -> Stack           {+}
 0C      Stack2 / Stack -> Stack           {/}
 0D      Stack2 - Stack -> Stack           {-}
 0E      Stack2 XOR Stack -> Stack         {XOR}
 0F      Stack2 AND Stack -> Stack         {AND}
 10      Stack2 OR Stack -> Stack          {OR}
 12      Stack2 = Stack -> Stack           {IF =}
 13      Stack2 >= Stack -> Stack          {IF >=}
 14      Stack2 <= Stack -> Stack          {IF <=}
 15      Stack2 <> Stack -> Stack          {IF <>}
 16      Stack2 < Stack -> Stack           {IF <}
 17      Stack2 > Stack -> Stack           {IF >}
 18      Abselutbetrag des Stack           {ABS}
 19      Ganzzahlteil des Stack            {INT}
 1A      Stack=Vorzeichen des Stack        {SIGN}
 1B      Einerkomplement des Stack
 1C      Cosinus des Stack                 {COS}
 1D      Tangens des Stack                 {TAN}
 1E      Sinus des Stack                   {SIN}
 1F      Wurzel des Stack                  {SQR}
 20      CodeByte des Stack                {CBY}
 21      Exponentialfunktion d. Stack
 22      Arcustangens des Stack            {ATN}
 23      nat. Logarithmus des Stack        {LOG}
 24      DatenByte des Stack               {DBY}
 25      Ext.DatenByte des Stack           {XBY}
 26      Pi auf Stack                      {PI}
 27      Zufallszahl auf Stack             {RND}
 28      Spez. Var. GET auf Stack          {GET}
 29      Freie Speicherlänge auf Stack     {FREE}
 2A      Länge Basicprogr. auf Stack       {LEN}
 2B      Spez. Var. XTAL auf Stack         {XTAL}
 2C      Spez. Var. MTOP auf Stack         {MTOP}
 2D      Spez. Var. TIME auf Stack         {TIME}
 2E      Spez. Var. IE auf Stack           {IE}
 2F      Spez. Var. IP auf Stack           {IP}
 30      Spez. Var. TIMER0 auf Stack       {TIMER0}
 31      Spez. Var. TIMER1 auf Stack       {TIMER1}
 32      Spez. Var. TIMER2 auf Stack       {TIMER2}
 33      Spez. Var. T2CON auf Stack        {T2CON}
 34      Spez. Var. TCON auf Stack         {TCON}
 35      Spez. Var. TMOD auf Stack         {TMOD}
 36      Spez. Var. RCAP2 auf Stack        {RCAP2}
 37      Spez. Var. PORT1 auf Stack        {PORT1}
 38      Spez. Var. PCON auf Stack         {PCON}
 39      BASCI-Argument mit Variablen,
           Operatorn oder Konst. auf Stack
 3A      Berechnet Quarz-abhängige Werte
           mit Stack
 3F      Holt einen Buchst. aus dem Basictext
           Pointer auf Adr.  0Ah,08h  -> ACC
 40      wie 3F aber mit Increment des Pointes
 41      Holt einen Charakter von Konsole -> ACC
 42      Startet BASIC-Programm
           Startadresse in 13h,14h
 80      Ausgabe des Zeichens in R5        {CHR}
 81      Holt ASCII-Komma-Zahl von Ext.Ram -> Stack
           DPTR ist Begin-Adresse danach auf Ende
           keine Zahl -> ACC=0ffh
 82    ! Stack2 + Stack -> Stack           {+}
 84    ! Stack2 - Stack -> Stack           {-}
 88    ! Stack2 * Stack -> Stack           {*}
 8A    ! Stack2 / Stack -> Stack           {/}
 90      Ausgabe des Stacks                {POP}
 98      Ausgabe R3,R1 in HEX              {PHx.}
           Bit 36h = 0 -> 4 digits
           Bit 36h = 1 -> 2 digits
           vor HEX wird 1 Space ausgegeben
 9A      R2,R0  -> Stack                   {PUSH}
```

3. Interruptbehandlung

Alle Interruptadressen des 8052 wurden durch das BASIC auf die Adressen 4003H bis 402BH gespiegelt.

```
Belegung:
          4003h     External Interrupt 0
          400Bh     Timer 0 Overflow
          4013h     External Interrupt 1
          401Bh     Timer 1 Overflow
          4023h     Serial Port
          402Bh     Timer 2 Overflow / External Inter. 2
```

Der 8052 besitzt weiters 2 Register IE und IP, die für das Arbeiten mit Interrupts wichtig sind.

IE ist das Interrupt Enable Register, welches folgendermaßen definiert ist:

Bit	7	6	5	4	3	2	1	0
	EA	X	ET2	ES	ET1	EX1	ET0	EX0
	Enable All	Unde-fined	Timer2	Serial Port	Timer1	Ext1	Timer0	Ext0

1 = enable (freigabe) / 0 = disable (gesperrt)

IP ist das Interrupt Priority Register, welches folgendermaßen definiert ist:

Bit	7	6	5	4	3	2	1	0
	X	X	PT2	PS	PT1	PX1	PT0	PX0
	Unde-fined	Unde-fined	Timer2	Serial Port	Timer1	Ext1	Timer0	Ext0

1 = hoher Prioritätslevel / 0 = niederer Prioritätslevel

Priorität innerhalb eines Levels:

1.	External Interrupt 0	höchste
2.	Timer 0 Overflow	
3.	External Interrupt 1	
4.	Timer 1 Overflow	
5.	Serial Port	
6.	Timer 2 Overflow/Ext. Inter. 2	niederste

Wird ein Interrupt erkannt, so werden folgende Flags gesetzt und abhängig von IE zu den am Anfang genannten Adressen verzweigt:

HC/SC	External Interrupt 0	IE0 ->	TCON.0
HC	Timer 0 Overflow	TF0 ->	TCON.5
HC/SC	External Interrupt 1	IE1 ->	TCON.3
HC	Timer 1 Overflow	TF1 ->	TCON.7
SC	Serial Port	TI ->	SCON.0
		RI ->	SCON.1
SC	Timer 2 Overflow	TF2 ->	T2CON.7
SC	External Inter. 2	EXF2 ->	T2CON.6

SC - muß per Software rückgesetzt werden. HC - wird per Hardware zurückgesetzt nachdem zur Interruptroutine verzweigt wurde.

Spezialfall: IE0, IE1. Diese können flanken- oder zustandsgesteuert eingestellt werden:

```
IE0:     IT0 -> TCON.0          0 = Low Level
IE1:     IT1 -> TCON.1          1 = Fallende Flanke
```

Ist IE0 oder IE1 auf Low-level-Erkennung eingestellt, so muß das zugehörige Interrupt-Flag softwaremäßig rückgesetzt werden. Ist aber die Erkennung auf fallende Flanke eingestellt, so wird das Flag von der Hardware rückgesetzt.

Genauere Beschreibung der Timer-Interrupts und des seriellen Ports siehe INTEL MCS-51 - Datenbuch.

! ! ! ! ! ! ! ! ! ! W I C H T I G ! ! ! ! ! ! ! ! !

1. Bei jedem Interrupt wird ein LCALL auf die am Anfang stehenden Adressen (4000 - 402B) erzeugt, daher ist an die erwähnte Adresse nicht die Adresse des Interruptprogramms alleine zu schreiben, sondern ein JMP oder sonstiger Befehl. Davor erfolgt aber immer ein Speichern des PSW auf den Stack mit PUSH PSW. Daher muß vor Ende der Interruptroutine mit RETI, noch ein POP PSW Befehl gegeben werden, da sonst Unvorhersehbares passiert !!!

2. Falls der Interrupt während eines laufenden BASIC-Programms auftritt, so darf in dem Interruptprogramm keine BASIC-Assembler-Routine verwendet werden; da der Interrupt ja zu unvorhersehbaren Zeiten kommt, und daher das BASIC in einem 'Unwissenden' Zustand ist.

3. Vor dem Eintritt in eine Interruptroutine müssen alle Register (DPTR, ACC, ... nicht Regbank 3 !), die in dieser Routine verändert werden, gerettet werden. -> PUSH ... Nach Beendigung mit RETI müssen diese wieder (in richtiger Reihenfolge!; zuletzt das in Punkt 1 erwähnte PSW) mit POP zurückgeholt werden.

4. Einfache Beispielprogramme

4.1. Blink

Dieses Routine läßt zwei LED's abwechselnd blinken. Die LED's befinden sich auf der Tastaturprint des Plotters und werden über den 8255 (Port C) des Plotter - Controller angesprochen.

```
Control EQU     78f7h       ;Adresse des Controlports des 8255
DatenC  EQU     78f6h       ;Adresse des Datenports C des 8255
                            ;beide befinden sich im ausge-
                            ;blendetem externen RAM
;
LED1    EQU     00000010B   ;LED 1 ist Bit 1 des C-Port
LED2    EQU     00000100B   ;LED 2 ist Bit 2 des C-Port
                            ;Bit=1 ist LED on
;
        ORG     3000H       ;Programmbegin auf Adresse 3000h
;
        MOV     DPTR,#Control ;Schreibt den Wert 90h in das
        MOV     A,#90H        ;Control-Register des 8255.
        MOVX    @DPTR,A       ;-> C Port = Ausgang
;
        MOV     DPTR,#DatenC  ;DPTR zeigt auf das C-Port
;
START:  MOV     A,#LED1       ;Bit 1 des C-Ports setzen
        MOVX    @DPTR,A       ;LED1 on, LED2 off
        CALL    LOOP          ;Warteschleife aufrufen
        MOV     A,#LED2       ;Bit 2 des C-Ports setzen
        MOVX    @DPTR,A       ;LED2 on, LED1 off
        CALL    LOOP          ;Warteschleife aufrufen
        JMP     START         ;dauerndes blinken
;
LOOP:    MOV    R1,#00        ;Warteschleife: Blinken langsam
LOOP1:   MOV    R0,#00        ;Schleifenzähler = 0
LOOP2:   DJNZ   R0,LOOP2      ;Schleife wird 256 mal
         DJNZ   R1,LOOP1      ;durchlaufen R0-1.
         RET                  ;äußere Schleife 256 mal
END                           ;durchlaufen
                              ;mit innerer Schleife
                              ;Es wird daher
                              ;R0 256mal von 0-FFh bis auf
                              ;0 verringert und R1 ein mal
                              ;bis zum Schleifenende.
```

4.2. Centro (Interruptprogramm)

Dieses Programm liest die Centronics-Schnittstelle des Plotter-Controlers ein. Dazu muß am Anfang eine Adresse eingegeben werden, auf die die Daten geschrieben werden sollen. Das Programm besteht aus einem BASIC-Teil und einem Maschinsprachteil der in DATA-Zeilen angehängt ist.

Der Maschinsprachteil wird durch den STROBE der Centronics gestartet, der auf INT0 geht. In dem Assembler-Programm wird dann nur das Busy-Signal der Centronics auf High (aktiv) gesetzt(->Centronics ist nicht frei für neue Daten) und die Daten der Centronics zwischengespeichert. Den Rest macht der BASIC-Teil.

Der BASIC-Teil fragt in einer Schleife dauernd das Busy-Signal ab, bis es High ist. Ist dies der Fall, so muß ein Interrupt erfolgt sein, da nur die Interruptroutine das Busy High setzt. Es werden nun die Daten aus einem Zwischenspeicher eingelesen und auf die gewünschte Adresse geschrieben. Danach wird das Busy wieder Low gesetzt (-> Centronics frei für neue Daten).

Das ganze Programm hätte auch nur in BASIC geschrieben werden können, da mit den Funktion ONEX1 und IDLE die Interruptsteuerung in Basic unterstützt wird. Dies ist aber hardwaremäßig nicht möglich, da der Strobe der Centronics mit INT0 verbunden ist und dieser vom Basic für DMA Prozesse vorgesehen ist (mit INT1 würde es gehen). Der zweite Grund für das verwenden einer Maschinsprachroutine war, daß das BASIC möglicherweise das Busy-Signal zu langsam setzt, und daher Daten verloren gehen könnten.

```
          ;Einschaltmeldung
20 ?" LIEST DIE CENTRONICS-SCHNITTSTELLE EIN"
30 ?" ----------------------------------------":?
32 ?" Abbrechen mit <Ctrl> C"
          ;Einlesen der DATA-Zeilen und schreiben dieser ab
;Adresse 2000h bis der Wert -1 eingelesen wird.
34 st=2000h : read hex
36 do
38 xby(st)=hex
40 read hex : st=st+1
42 until hex<0
          ;8255: Centronics sperren, Startadresse
          ;abfragen und die Interruptvektoren einstellen
45 PORTC=78F6H
47 CONTROL=78F7H
50 xby(CONTROL)=90H : rem Port C ist Ausgang
60 xby(PORTC)=01H : rem Busy High
70 input"Startadresse: ",st
80 call 2000H : rem Interrupt Routine installieren
          ;Centronics freigeben und auf Daten warten
90 xby(PORTC)=00h : rem Busy Low
100 if xby(PORTC) = 1 then 150 else goto 100
          ;Daten aus Zwischenspeicher holen
          ;Adresse um eins erhöhen
150 xby(st)=dby(18h) : st=st+1
160 goto 90

;Zuerst wird der INT0 gesperrt, damit während dem
;Einstellen der Interruptadressen keine Interrupts
;akzeptiert werden. Dann wird auf Adresse 4003h der
;Befehl 'JMP CENTRO' geschrieben und danach der INT0
;wieder freigegeben.

200 rem     CENT      EQU 78F8H
210 rem     PORTC     EQU 78F6H
215 data 0D2h,0AFh    :rem SETB EA          ;Enable Interrupts
220 data 0C2h,0A8h    :rem CLR EX0          ;Disable INT0
230 data 0D2h,88h     :rem SETB IT0         ;INT0
                                            ; flankengesteuert
240 data 90h,40h,03   :rem MOV DPTR,#4003h ;INT0 Adresse
241 data 74h,02       :rem MOV A,#JMP       ;OPCODE für JMP
242 data 0F0h         :rem MOVX @DPTR,A     ;auf Adresse 4003h
243 data 0A3h         :rem INC DPTR         ;nächste Adresse
250 data 74h,20h      :remMOVA,#HI(CENTRO) ;HiByte der Interrupt-
260 data 0F0h         :rem MOVX @DPTR,A     ;adresse auf 4004h
270 data 0A3h         :rem INC DPTR         ;nächste Adresse
280 data 74h,17h      :rem MOV A,#LO(CENTRO);LoByte
290 data 0F0h         :rem MOVX @DPTR,A      ;auf 4005h
300 data 0D2h,0A8h    :rem SETB EX0          ;Enable INT0
310 data 22h          :rem RET               ;zurück zum Basic
    ;Hier folgt die Interruptroutine
    ;zuerst werden DPTR und ACC gerettet um das durch diese
    ;Routine unterbrochene Basicprogramm nicht zum Absturz
    ;zu bringen.
322 data 0C0h,83h       :rem CENTRO   PUSH DPH
324 data 0C0h,82h       :rem   PUSH DPL
326 data 0C0h,0E0h      :rem   PUSH ACC
    ;High setzen des Busy-Signals damit Centronics gesperrt
330 data 90h,78h,0F6h  :rem   MOV DPTR,#PORTC
340 data 74h,01        :rem   MOV A,#01
350 data 0F0h          :rem   MOVX @DPTR,A
    ;Einlesesn der Daten von der Centronics in Accumulator
360 data 90h,78h,0F8h  :rem   MOV DPTR,#CENT
370 data 0E0h          :rem   MOVX A,@DPTR
    ;Daten in Zwischenspeicher bringen für Basicprogramm
380 data 0F5h,18h      :rem   MOV 18H,A
    ;DPTR und ACC zurückholen für Basic
381 data 0D0h,0E0h     :rem   POP ACC
382 data 0D0h,82h      :rem   POP DPL
383 data 0D0h,83h      :rem   POP DPH
    ;PSW zurückholen das bei Interrupts
    ;automatisch vom ;8052 gerettet wird (PUSH)
385 data 0D0h,0D0h     :rem   POP PSW
    ;Rückkehr vom Interrupt
390 data 32h           :rem   RETI
    ;Endmakierung für das Einlesen der DATA-Zeilen
400 data -1            :rem  END.
```

Einsatz eines 8031-Experimetiersystems im Unterricht

Andreas HEINBACH, HTL Wien-I TGM-DSK-169: ADAUTO.ASM, TGM-LIT-020

An der Abendschule der HTL Wien1, Abteilung Elektrotechnik findet ein zweisemestriger Lehrgang für Berufstätige für Informatik und Mikroelektronik statt. Teil des Unterrichts ist ein vierstündiges Laboratorium, das geblockt in zwei mal sechs Abenden zu je fünf Stunden gehalten wird. Erstmals im Sommersemester 1991 wurde es den Teilnehmern freigestellt, an zwei der zwölf angebotenen teilzunehmen. Das heißt, das Thema 8031-Prozessoren wurde an sechs Abenden mit ein und derselben Gruppe intensiv behandelt.

Durch dieses System sollte einerseits den Teilnehmern berufsspezifische Weiterbildung ermöglicht werden, andererseits konnte der Vortragende erwarten, mit seinem Wissensangebot auf großes Interesse zu stoßen. Als Problem erwies sich jedoch - wie wahrscheinlich in jedem Abendlehrgang - die Aufnahmefähigkeit, die nach einem achtstündigen Arbeitstag natürlich nicht mehr uneingeschränkt vorhanden ist. Auch auf divergierende Grundkenntnisse im Elektronikbereich mußte Rücksicht genommen werden.

Für die Durchführung der Übungen stand ein PC mit zwei Laufwerken zur Verfügung. Aus Komfort- und Datensicherheitsgründen wären 3½" HD-Laufwerke von Vorteil. Die Übungen fanden im Elektrotechnik Labor statt, Elektronik-Kleinmaterial war vorhanden. Eine Experimentierplatine MCB515 der Firma KEIL mit einem Monitorpogramm in einem Eprom wurde zusammen mit einer einfachen Peripherieplatine mit zweistelligem LED-Display, Optokoppler, Potentiometer an einem Analogeingang und einem Taster zum Austesten der Programme verwendet.

Die Mikrocontrollerprogramme konnten sowohl am PC simuliert werden, als auch mit der selben Benutzeroberfläche mit der Hardware ausgetestet werden. Wir verwendeten dafür die Programme DScope51 und MonScope51, ebenfalls von der Firma KEIL. Die Programme stellen zwei Einzelschrittmodi, Breakpoints und Watchpoints mit programmierbaren Prozeduren zur Verfügung. Die Bedienung der Programme ist auch für Ungeübte relativ rasch erlernbar, jedoch wäre eine ständige Anzeige des RAM-Bereiches (bzw. nur der Special Function Register) sicherlich vorteilhaft.

Wir versuchten, pro Laborabend ein Thema, bzw. eine Applikation zu erarbeiten. Am Anfang stand eine kurze theoretische Einführung, danach wurde eine Aufgabe formuliert, die bis zum Ende des Abends gelöst sein sollte.

Als Beispiel für so eine Aufgabe sei ein einfaches Spannungsmeßgerät mit automatischer Meßbereichsumschaltung zu programmieren. Am Analogeingang liegt eine Spannung zwischen 0V und 5V. Der Gesamtmeßbereich soll aufgrund der tatsächlich anliegenden Spannung folgendermaßen aufgeteilt werden:

```
0.....1.25 V  Meßbereich0
1.25..2.5  V  Meßbereich1
2.5...3.75 V  Meßbereich2
3.75..5    V  Meßbereich3
```

Im Programm wird bei der ersten Messung aufgrund der beiden höchstwertigen Bits des Meßergebnisses der Meßbereich für eine zweite Messung eingestellt und sodann zuerst Bereich und dann - nach einer Pause - der zweite Meßwert über eine Tabelle auskodiert und auf Port1 gelegt. Port1 liegt an beiden 7-Segment-Anzeigen, mit Port3.3 und Port3.4 können diese einzeln geschaltet werden. Durch die Meßbereichsumschaltung wird die Auflösung der Messung von 8 auf 10bit (5mV) erhöht, wobei nicht vergessen werden darf, daß die Genauigkeit des Vergleichs nur 10mV beträgt.

ADAUTO.ASM

```
        $ NOMOD51
        $ INCLUDE (c:REG515.inc)
        org     0000h
reset:          Dptr,#Tabelle     ; Zeichentabelle 7 Segment
start:  mov     R0, #00           ; 0 V bis 5 V
        call    wandle
        call    bereich
        call    wandle
        call    ausgabeB
        mov     a,r6
        call    ausgabeN
        call    paus1
        jmp     start
wandle: ; einlesen Kanal 0
        ; R0    Wert für DAPR ( Bereich ), Ergebnis in r6
loop:   clr     mx0           ; wähle Kanal 0
        clr     mx1
        clr     mx2
        mov     DAPR, R0      ; starte Wandlung
        jb      BSY, $        ; warte bis fertig umgewandelt
        mov     r6, ADDAT
        ret
bereich:      ;Setzt r0 in Abhängigkeit des Wandlereerg
        ;nr des messber ==> r7, Wert f. DAPR ==> r0
        mov     a,r6
        jb      acc.7,s1
        jb      acc.6,s2
        mov     r0,#40h
        mov     r7,#00
        jmp     aus
s2:     mov     r0,#84h
        mov     r7,#01
        jmp     aus
s1:     jb      acc.6,s3
        mov     r0,#0c8h
        mov     r7,#02
        jmp     aus
s3:     mov     r0,#0ch
        mov     r7,#04
aus:    ret
ausgabeN: ; Ausgabe Akku als HexZahl
        mov     b, #10h
        div     ab
        cjne    a,#00,zehner
        mov     r1,#0    ; 1. Stelle 0, Anzeige dunkel
        jmp     einer
zehner: movc    A,@A+Dptr
        mov     R1,A
einer:          A,B
        movc    A,@A+Dptr
        mov     R2,A
        mov r3,#07fh
Anzeige:        P1,R1    ; 1. Stelle
        clr     P3.4
        setb    P3.4
        mov     P1,R2    ; 2. Stelle
        clr     P3.3
        setb    P3.3
        djnz r5,Anzeige
        djnz r3,Anzeige ; Anzeige-/ Samplezeit
        ret
ausgabeB:     mov       a,r7
        movc    a,@a+dptr
        mov     p1,a
        clr     p3.3
        call    paus1
        setb    p3.3
        ret
Pause:          R3,#10h
Pause1: djnz    R3,Pause1
        ret
paus1:  mov r3,#0ffh
        mov r4,#04
13:     djnz r5,13
        djnz r3,13
        djnz r4,13
        ret
        ; Segment   GFEDCBA
Tabelle: DB 10111111b;   0
        DB 10000110b;   1
        DB 11011011b;   2
        DB 11001111b;   3
        DB 11100110b;   4
        DB 11101101b;   5
        DB 11111101b;   6
        DB 10000111b;   7
        DB 11111111b;   8
        DB 11100111b;   9
        DB 11110111b;   A
        DB 11111100b;   B
        DB 10111001b;   C
        DB 11011110b;   D
        DB 11111001b;   E
        DB 11110001b;   F
        end
```

Klein, aber oho !

80C32-Minimodul von nds als universeller Problemlöser

Robert NOWOTNY, nds TGM-DSK-169: MSB32*.PLT

Seit einiger Zeit sind mehrere Mikrocontroller-Minimodule von verschiedenen Herstellern im Handel erhältlich, die bis zur Scheckkartengröße (in SMT, versteht sich) geschrumpft und mehr oder weniger preisgünstig sind. Das hier vorgestellte Minimodul MSB32 ist etwas größer (weil in herkömmlicher Technologie (Dual Inline Packages) hergestellt), und oberflächlich gesehen bedeutet das einen technologischen Rückschritt. Der Einsatz von DIP's bringt jedoch auch Vorteile, z.B. können kritische IC's ohne zusätzlichen Platzaufwand gesockelt werden, was die Servicefreundlichkeit erhöht; abgesehen davon, daß einige SMD-Gehäuse überhaupt nicht sockelbar sind, ist bei Nicht-Direktlötung von SMDs der Hauptvorteil von geringerem Platzbedarf zumindest teilweise zunichte gemacht. Die meisten gängigen EPROM-Typen (2764, 27128, ..) sind seit einiger Zeit im PLCC32-Gehäuse (Plastic Leaded Chip Carrier) lieferbar, welche jedoch kein Löschfenster besitzen und daher als OTP-ROMs (One-Time Programmable ROMs) bezeichnet werden. Das bedeutet, einmal falsch programmiert -> ab in den Abfalleimer - bei Einzelanwendungen oder Kleinserien (die Hauptanwendung von Minimodulen wie dem MSB32) kein Einzelschicksal. Außerdem - die wenigsten EPROM-Programmer besitzen einen PLCC32-Nullkraftsockel, die meisten fertigen Umsetzer sind sündhaft teuer, und selbst die PLCC-Nullkraftsockel alleine (z.B. Textool oder Yamaichi) sind nicht billig und in Österreich (zumindest bis vor kurzem) kaum zu kriegen. Doch zurück zur Sache.

Funktionsbeschreibung

Das Herz des Modules bildet der 80C32 (s. Abb.1), ein großer Bruder des schon fast legendären 8031, welcher mit einer Taktfrequenz von 12.0 MHz eine ganz passable Performance (abhängig von der verwendeten Programmiersprache) erreicht. Es gibt zwar noch höher integrierte 8051-Familienmitglieder (z.B. 80515, 80535) die über einen integrierten ADC mit vorgeschaltenem 8-fach MUX verfügen. Wenn bei der einen oder anderen Anwendung ein ADC erforderlich ist (wobei die Frage offen bleibt, ob die Auflösung und Wandlungsqualität der internen ADC's genügt), sind meist Sensorverstärker, Pufferverstärker, Filter, etc. notwendig, sodaß ein kostengünstiger Einchip-ADC (z.B.MAX158) auf dem ohnehin zusätzlich erforderlichen Print den fehlenden On-Chip-ADC ersetzt.

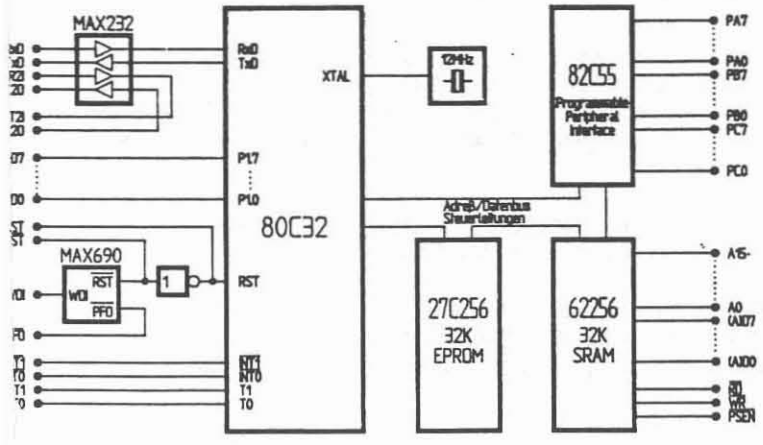


Abb.1) Funktionsblockschaltbild

Die Hälfte des gesamten Programm- und Datenspeicherbereiches (s. Abb.2) wird mit einem 32K-EPROM (27C256) und einem 32K-SRAM (62256) abgedeckt, die jeweils andere Hälfte steht dem Anwender für externen Speicher oder I/O zur Verfügung.

Weiters findet ein universeller Peripheriebaustein, der 82C55 (eine neue und schnellere CMOS-Version des seit Jahren im Einsatz befindlichen 8255) Anwendung, der als einfacher Porterweiterungsbaustein (3 bidirektionale 8-Bit Ports, die 2.5 mA in beiden Richtungen liefern können) bis hin zum bidirektionalen Busbaustein mit automatischer Hardwarehandshake-Generierung verwendet werden kann.

Zusätzlich zu diesen 3 Ports ist Port 1 des 80C32 herausgeführt, der zwar nicht so hoch belastbar ist, wie die Ports des 82C55, dafür aber der schnellste ist und sich für die Generierung von Handshakeleitungen für den seriellen Port, zur Triggerung des Watchdoges o.Ä. eignet. Als Watchdog kommt der MAX690 zum Einsatz, der neben des Power-On-Resets einen Power-Fail-Ausgang besitzt, der z.B. bei Akkubetrieb des Moduls mit einem der Interrupteingänge eine Unterspannung erkennen und nötige Maßnahmen einleiten kann. Der serielle Port ist standardmäßig über einen RS232-Pegelwandlerbaustein (MAX232) nach außen geführt (andere Standards z.B. RS422, RS485 bzw. kein Treiber auf Anfrage), wobei die beiden freien Treiber frei beschaltet und somit z.B. als Handshakeleitungen verwendet werden können (nur RS232).

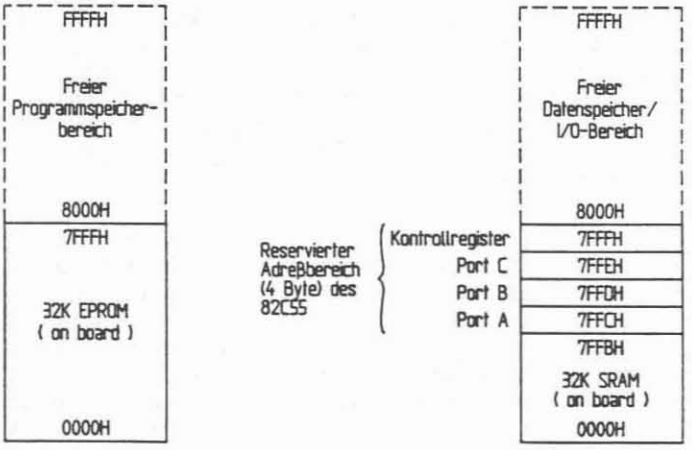


Abb.2) Speicherbereichsaufteilung

Prinzipiell stehen dem Anwender alle Konfigurationsmöglichkeiten zur Verfügung und lassen sich durch ausgeklügelte Anordnung der Portpins elegant realisieren. (Beim Einsatz als Stand-Alone-Gerät durch kleine Drahtbrücken, bei Huckepack-Anwendung des Moduls auf einer Mutterplatine durch kurze Leiterbahnen.)

Die gesamte Schaltung ist aus Standardbausteinen in Highspeed-CMOS-Technik aufgebaut, durch Vermeidung von PALs ist es auch nach Jahren selbst dem Anwender möglich, einen eventuell aufgetretenen Defekt kostengünstig durch den Tausch eines handelsüblichen ICs zu beheben.

Anwendungsmöglichkeiten

Und so sieht das MSB32 aus! Mit einer Länge von 94mm und einer Breite von 74mm ist es etwas länger und etwas breiter als eine Scheckkarte, jedoch noch immer sehr kompakt. Für die mechanischen Befestigungen sind Bohrungen mit Durchmesser 2.7 mm (für Schrauben M2.5) vorhanden, die Anschlußpadreihen sind mit 1.0 mm gebohrt und für gängige Stift- oder Buchsenleisten oder für direktes Einlöten von Drähten geeignet.

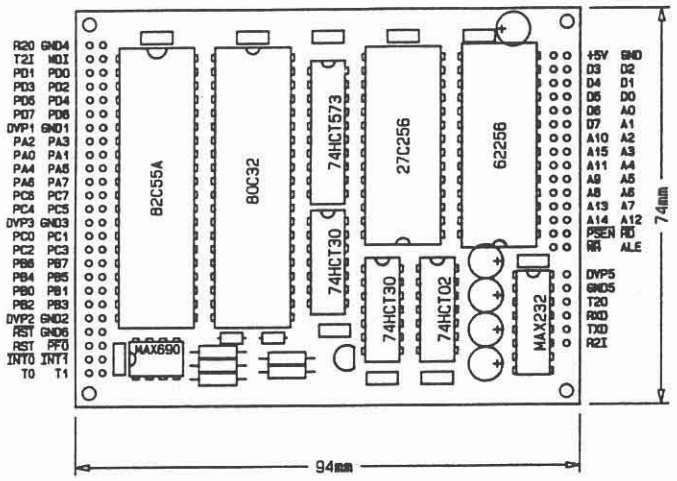


Abb. 3) Ansicht MSB32 (Maßstab M1:1)

Die Anwendungsmöglichkeiten sind so vielfältig, daß nur einzelne Hinweise sinnvoll erscheinen. Grundsätzlich ist der Einsaz überall dort rentabel, wo die Entwicklungskosten eines gleichwertigen Schaltungsteiles, umgelegt auf die Stückzahl, größer sind als das fertige und qualitätsgeprüfte Minimodul. Der Schwerpunkt eines Projektes, bei dem ein Minimodul wie das MSB32 angewendet wird, verlagert sich dann auf Softwareentwicklung und die Entwicklung des projektspezifischen Hardware-Teils.

Ein Beispiel ist die Huckepack-Montage auf einer Karte im Europakartenformat, wobei der Platz unter dem Modul bereits für anwendungsspezifische Schaltung genützt werden kann. Bei der Anwendung als Stand-Alone-Version sei auf die mehrfach vorhandenen Versorgungspins zwischen den Ports hingewiesen, die den Verkabelungsaufwand auf ein Minimum reduzieren und die Mitführung der Spannungsversorgung (z.B. mittels Flachbandkabel auf einen Tasten/Anzeigenprint) ermöglichen. Mittels RS422-Option (in der Rundfunktechnik verbreitete Steuerschnittstelle) oder RS485-Option können einzelne MSB32 zu einem Netz mit Entfernungen bis zu mehreren hundert Metern verbunden werden, ein PC mit entsprechender Schnittstelle übernimmt beispielsweise die Rolle der zentralen Steuerung und Datenauswertung.

Hinweise zur Bestellung

Folgende Varianten sind lieferbar:

MSB32	Standardausführung mit RS232-Schnittstelle
MSB32A	Ausführung ohne Schnittstellentreiber (Serieller Port ist direkt herausgeführt)
MSB32B	Ausführung mit RS422-Schnittstelle
MSB32C	Ausführung mit RS485-Schnittstelle
MSB32xN	wie oben, jedoch mit 32K Non-volatile RAM

Lieferfirma: nds (Nowotny Datensysteme)
1150 Wien, Märzstraße 116
Tel.: (0222) 98 21 005
Fax.: (0222) 98 25 033

Die Standardausführung (MSB32) kostet als Einzelstück 1950,- incl.20%Mwst.

Im Lieferumfang enthalten: Schaltungsunterlagen, Umfangreiche Unterlagen 82C55, Anwendungshinweise, 1 EPROM 27C256 (leer).

Preise übriger Versionen bzw. Staffelpreise und Händlerkonditionen auf Anfrage.

Anm.d.Red.: Unter der Bezeichnung 8052-BASIC ist ein dem 80C32 baugleicher Mikrocontroller erhältlich, der einen BASIC-Interpreter enthält und mit einem PC mit Terminalemulationsprogramm programmiert werden kann. Diese Möglichkeit beschreibt auch der Beitrag Schnittstelle PC <-> 8052. Dazu erreichte uns folgender Hinweis von nds*:*

Der **BASIC-Betrieb** mit dem MSB32 ist (ohne Zusatz) eingeschränkt möglich; die Einschränkung besteht darin, daß nur der On-Line Betrieb möglich ist, und zwar auf folgende Weise:

a) Entweder wird anstatt eines 80C32 ein 8052AH-BASIC eingesetzt, oder ein 80C32 wird mit einem EPROM mit BASIC-Interpreter betrieben (interner oder externer Programmspeicher mittels Jumper auswählbar).

b) Mittels Terminalprogramm wird On-Line ein BASIC-Programm erstellt oder ein bestehendes Programm in das RAM des MSB32 geladen. Mit dem BASIC-Befehl RUN wird dieses ausgeführt.

Die Option, ein EPROM (oder NV-RAM) zu programmieren bzw. den Inhalt automatisch nach Anlegen der Versorgung ausführen zu lassen, ist nur mit zusätzlicher Hardware möglich; da aber alle Adreß- und Datenleitungen, sowie die Steuer- und Portleitungen zur Verfügung stehen, ist diese Option z.B. durch einen kleinen Subprint relativ einfach nachrüstbar.

Auf die Möglichkeit, alle BASIC-Features auf dem MSB32 ausnützen zu können, wurde aus 2 Gründen verzichtet:

*) Es wäre zusätzliche Logik und Jumper nötig gewesen, das Platzangebot wurde jedoch mit der jetzigen Version ziemlich ausgeschöpft; ein Ausweichen auf SMT, größere Abmessungen oder der Einsatz von PALs wäre die Folge gewesen.

*) Durch die langsame Verarbeitungsgeschwindigkeit sind kaum ernstzunehmende Applikationen realisierbar.

Bei der Qualitätskontrolle allerdings ist der Einsatz von BASIC äußerst praktisch, die Module können relativ kostengünstig vor der Auslieferung auf Herz und Nieren getestet werden.

Am Anfang war das Chaos.

Hoffmann,TGM/NT/3ANK90

Zuerst haben wir einen Zeitplan erstellt, obwohl wir nur in sehr groben Zügen erfahren haben, was überhaupt zu realisieren war: eine Erweiterungsplatine für den µPROFI (ein 8051-Entwicklungssystem), dessen Funktion es sein sollte, nach acht Schalterfunktionen acht LEDs als Lauflicht oder blinkend zu schalten.

Obwohl der 8051 als Microcontroller I/O-Einheiten zur Verfügung stellt, bauten wir unsere doch selber auf (der µPROFI unterstützt den Port-betrieb nicht) und betrieben die CPU extern als normalen Prozessor. Die I/O-Einheit für Schalter und LEDs war unsere Schaltung, bestehend aus einem Gatterfriedhof zur Adreßdecodierung und diversen Latches und Puffer für Lese- und Schreibfunktionen auf dem Datenbus.

Die Schaltung wurde mit PCAD gezeichnet und nach einer Wirelist (Verdrahtungsliste) gefädelt. Das Fädeln, freilich, entpuppte sich nicht nur als eine mühsame, sondern auch als fehlerträchtige Arbeit, zumal wir diese Art des Aufbaues erst neu kennenlernten. Auf das Fädeln folgte die Fehlersuche, denn anfangs pflegte unser Drahtverhau über den µPROFI den PC (dieser bietet dem User einen FullscreenDebugger als Oberfläche zwecks Bedienung und Programmierung) regelmäßig zum Absturz zu bringen.

Auch PCAD bereitete uns Kopfzerbrechen, denn aufgrund unserer mangelnden Kenntnisse war es oft mühsam unsere Vorstellungen einer technischen Zeichnung durchzusetzen. Auch Computer haben ein Eigenleben! Tätigkeiten wie Routen oder Erstellen einer Stückliste entwickelte sich zu einer elektronischen Weltreise und wir kamen erst spät hinter alle Kniffe.

Angesichts der vielen Einzelprogramme von PCAD, die man nach Möglichkeit in der richtigen Reihenfolge aufrufen und auch bedienen können sollte, kam ich mir vor, wie Robin Hood im Datenwald. Letztendlich ist eine noch so schöne softwaregesteuerte Schaltung ohne Software völlig sinnlos. So bestand unsere letzte Hürde darin, auch programmtechnisch unserer Aufgabenstellung gerecht zu werden, wenn eine funktionstüchtige Schaltung vorausgesetzt wird.

Da Blinken oder Lauflicht nicht mit Lichtgeschwindigkeit ablaufen sollten, (man sieht ja nichts!) war es nötig, den internen Timer des 8051 für Verzögerungszwecke heranzuziehen.

Am Ende: Eine nette Bastelarbeit, die den Schülern lehren sollte, die oft schwierige Aufgabe zu meistern, Hard- und Software unter einen Hut zu bringen, natürlich auch bei Fehlfunktionen des Systems bei der Fehlersuche taktisch klug vorzugehen und mit Hilfe der Laborausrüstung und vieler Kniffe, die wir kennenlernten, am Ende Sieger zu bleiben.
Hoffmann 3ANK

HARDWARE-ERWEITERUNG zum µPROFI-51

Hoffmann/Scheiber/Vozdecky, TGM/NT/3ANK90 TGM-DSK-175, TGM-LIT-013

1.Programm FTKL im Schuljahr 1990/91:

SYSTEMZIELE

Die im Wesentlichen geforderten Leistungen des 1. FTKL-Programms liegen im Bereich der Mikroprozessor-Steuerungen. Es soll der Schüler dadurch lernen, einen Mikroprozessor zu verstehen, zu programmieren und eine entsprechende Hardware zu erstellen. Vor Beginn der Arbeit ist ein Projektplan über Gruppeneinteilung, Arbeitszeit und Abgabetermin zu erstellen. Dadurch steckt sich der Schüler selbst ein Ziel und lernt, eine Arbeit zu organisieren. Bei der Entwicklung wird das Nachschlagen in Datenbüchern und Prospekten notwendig, um für spezielle Anwendungen entsprechende Bauteile zu finden. Es wird dadurch die Kreativität und Eigenständigkeit jedes Einzelnen gefördert. Es werden weiters die Grundfunktionen der maschinennahen Programmierung (Assembler) und, in Zusammenhang damit, eine Verbindung zur praktischen Anwendung demonstriert. Zusätzlich wird auch auf die praktische Ausarbeitung Wert gelegt. Das heißt, der Schüler muß sich über verschiedene Fertigungskriterien Gedanken machen (geringe Kosten, einfache Bedienung, Möglichkeiten der Fertigung). Es wird eine vollständige Entwicklung eines Produktes durchgemacht (von Entwicklung über Fertigung zur Wartung und Austestung).

ANWENDERANFORDERUNGEN

Das komplette System besteht im wesentlichen aus drei Teilen:

- der PC (Personal Computer): er wird verwendet, den µPROFI zu initialisieren und diesen mit den für seine Funktion notwendigen Daten zu versorgen.
- der µPROFI: der µPROFI ist ein kleines, im Prinzip unabhängiges Mikroprozessor-System. In Kombination mit einer entsprechenden Zusatzplatine können kleine Demonstrationen vollzogen werden.
- die Hardware-Erweiterungsplatine: sie stellt für den µPROFI ein geeignetes Ein-/Ausgabegerät dar und kann nur in Zusammenhang damit funktionieren.

Kurzbeschreibung

In Abhängigkeit der Schalterstellung (immer nur 1 Schalter geschlossen) soll eine entsprechende Zahl auf der 7-Segmentanzeige ausgegeben, und verschiedene Muster an den 8 LED's dargestellt werden:

Schalter 1 geschlossen: es sollten bei richtiger Bedienung alle 8 LED's leuchten und eine "1" an der 7-Segmentanzeige anliegen.

Schalter 2 geschlossen: es entsteht ein langsames Lauflicht und eine "2" am Display.

Schalter 3 geschlossen: diese Schalterkombination ergibt ein schnelles Lauflicht und eine "3" auf der Anzeige.

Schalter 4 geschlossen: es entsteht ein Muster, wo abwechselnd, einmal die "geraden", einmal die "ungeraden" LED's leuchten. Am 7-Segment erscheint eine "4".

Schalter 5 geschlossen: es ergibt sich dabei ein Lauflicht, daß einmal von links nach rechts, und dann von rechts nach links läuft. Die 7-Segmentanzeige bringt die Zahl "5".

Schalter 6 geschlossen: es entsteht ein Muster, wo die Lichter von außen nach innen laufenn. Weiters wird eine "6" gezeigt.

Schalter 7 geschlossen: es kommmt dabei zu einem Lauflicht, bei dem von rechts beginnend immer eine LED hinzukommt, und einer "7" an der Segment-Anzeige.

Schalter 8 geschlossen: An der 7-Segmentanzeige wird eine "8" ausgegeben. An den LED's ergibt sich ein inverses Lauflicht.

Fehlbedienung: Der µPROFI ist allerdings so programmiert, daß er nur bei Betätigung eines Schalters die entsprechende Ausgabe tätigt. Bei Fehlbetätigung - d.h. bei Betätigen von zwei oder mehreren Schaltern - wird diese durch einen "F" (Fehler) am 7-Segment angezeigt. Dieser Zustand hält solange an, bis wieder eine Taste bedienungsrichtig betätigt wird. Die LED-Reihe zeigt dabei in der Mitte 2 leuchtende Punkte.

SYSTEMKURZBESCHREIBUNG

Das Projekt stellte die Funktion, Arbeitsweise, Entwicklung und Herstellung eines µP-Systems zur Aufgabe. Es wurde eine Platine, die als Ein-/Ausgabebaustein eines Mikrocontrollers (hier: µPROFI-51) dienen soll, dimensioniert und aufgebaut. Zur Erstellung des Schaltplanes, der Bauteilliste und des Layouts wurde ein Anwenderprogramm (PCAD-Personal Computer Aided Design) zur Verfügung gestellt. Es soll eine achtstellige Schalterkombination (8 Schalter am Print realisiert) vom µPROFI eingelesen werden, und abhängig davon ein Leuchtmuster auf 8 LED's bzw. ein Zeichen an einer 7-Segmentanzeige ausgegeben werden. Dabei wurde vorher über einen PC der µPROFI in Assembler programmiert (unter Verwendung eines Timers zur Verzögerung; in strukturierter Form programmiert, d.h einzelne, zusammengehörende Teile, z.B. einzelne Schalterstellungen, in einem Unterprogramm realisieren). Die Durchführung des Projekts wurde nach einem festgelegten Zeitplan vollzogen, welcher sich über einen Zeitraum von mehreren Wochen erstreckte.

Kurzbeschreibung der Arbeitsschritte bei der Fertigung:

- Funktion und Arbeitsweise des µPROFI51 anhand von Datenblättern erfassen.
- Problem der Ansteuer- und Durchschaltlogik der zu entwickelten Platine durchdenken.
- mittels PCAD (= Anwenderprogramm) Schaltplan, Layout und sämtliche zur Fertigung wichtigen Listen (cross-referenz-list, net-list) erstellen. Dabei muß allerdings auf vorhandene und zur Verfügung gestellte Bauteile Rücksicht genommen werden.
- Softwareprogramme entwickeln und programmieren. Als Programmiersprache wird ASSEMBLER verwendet.
- Fertigung der entworfenen Platine und anschließende Kopplung mit dem µPROFI.
- Tests, Optimierung, Wartung

HARDWAREKONFIGURATION

```
a) digitale IC's:
     1 x 74LS32              (= 4 OR-Gatter)
     1 x 74LS02              (= 4 NOR-Gatter)
     1 x 74LS00              (= 4 NAND-Gatter)
     2 x 74LS273             (= Latch)
     3 x 74LS244             (= LED-Treiber,Puffer)

b) diskrete Bauteile:
    16 x Widerstände à 330E   (= Vorwiderstände)
     8 x Widerstände à  47k   (= Pull-Up-Widerstände)
     8 x Tantal-Kondensatoren à 1µF (= Stützkondensatoren)

c) Elektromechanisches:
          1 x 64-polige Steckerleiste    (= Schnittstelle )
          8 x Umschalter
          1 x 7-Segmentanzeige
          8 x LED's

d) Zusätzliches Material:
          8 x Sockel für IC's
          1 x Lochraster-Platine (100x160mm)
          dünner, lackisolierter Draht
          zur Verdrahtung des Aufbaues
```

FUNKTIONALE BESCHREIBUNG

Anm.d.Red.: Aus Platzgründen wird hier auf die Wiedergabe der Schaltpläne, Listings und Struktogramme verzichtet. Interessenten können diese über TGM-LIT-013 anfordern. (Bestellkarte am Ende dieses Heftes).

Software

Bei Entwurf der Software waren einige Gesichtspunkte zu berücksichtigen. So war es zum Beispiel vorgeschrieben, die Software strukturiert zu gestalten, d.h es sollten einzelne Programmblöcke zu erkennen sein. Weiters soll ein Timer zur Verzögerung der Sendeimpulse verwendet werden. (Eine Verzögerung ist unbedingt notwendig, da z.B. ein Lauflicht nie als solches zu sehen wäre. Der interne Takt, mit dem normalerweise geschrieben wird, beträgt 12 MHz. Es würde also alle 12 µs eine LED leuchten).

Zu Beginn muß der Datenzeiger auf einen bestimmten Anfangswert gelegt werden. Es wurde hier der Wert A000h gewählt und mit dem Befehl ORG fix definiert. Das Hauptprogramm besteht im Wesentlichen aus 3 Teilen:

EINLESEN

Dieser Programmteil löscht zuerst den Akkumulator-Inhalt, setzt den Datenzeiger auf die gewählte Leseadresse (C004h), um die anliegenden Bitmuster lesen zu können, und gibt dieses in binärer Form in den Akkumulator.

EINSZÄHL

Hier wird die Anzahl der im eingeholten Bitmuster vorhandenen Einser gezählt. Dies ist insofern wichtig, um mögliche Fehlbedienungen, wie zum Beispiel die Betätigung mehrerer Schalter gleichzeitig, zu erkennen und entsprechende Maßnahmen zu setzen. Am Ende dieses Teiles steht im Register R4 die Anzahl der "1" im Bitmuster.

VERZWEIG

Die Aufgabe dieses Programmteils liegt darin, entsprechend der anliegenden Bitmuster Verzweigungen zu kleinen Unterprogrammen (Jobs -> strukturierte Programmierung) durchzuführen und Fehler zu erkennen und ebenfalls entsprechend zu reagieren. Eine Fehlbedienung liegt immer dann vor, wenn die Anzahl der Einsen ungleich 1 ist. Dies ist also hier in 2 Fällen möglich. Entweder es wird überhaupt kein Schalter betätigt (R4=0) oder mehr als einer (R4 > 1). Mit einer doppelten Abfrage wird unterschieden:

Bei richtiger Bedienung kommt es zur bitweisen Abfrage (JNB-Befehl: es wird nur dann auf die angegebene Adresse gesprungen, wenn das angewählte Bit nicht gesetzt ist. LJMP-Befehl: es wird ein unbedingter Sprung über 64k durchgeführt) und entsrechender Verzweigung zu den Jobs. (kleine Arbeitsschritte)

Fehlersuche

Obwohl zuvor alle Leitungen mit dem Ohmmeter auf ihre elektrische Verbindung überprüft wurden, funtionierte die Schaltung nicht. Der Computer stürzte immer wieder ab. Nach langwierigem Durchmessen der Leitungen der Steckerleiste gegeneinander, wurde festgestellt, daß 4 Stütz-Kondensatoren falsch gepolt eingebaut worden sind. Weiters war zu erkennen, daß der Datenbus nicht vollständig funktionsfähig war (Datenleitungen, Querverbindungen). Es lagen nur 2,5V an, was auf offene Leitungen schließen läßt.Nach Behebung dieser Fehler funktionierte bereits die Ausgabe. Jedoch war das Einlesen nicht möglich. Als Ursache ergab sich, das ein ODER-Gatter-IC defekt war (undefinierte Zustände am Ausgang; Ständig 1,8V). Da der Fehler dadurch nicht behoben werden konnte, versuchten wir, die Schaltung nach eventuell falschen Verbindungen abzusuchen (nach Verdrahtungsliste).

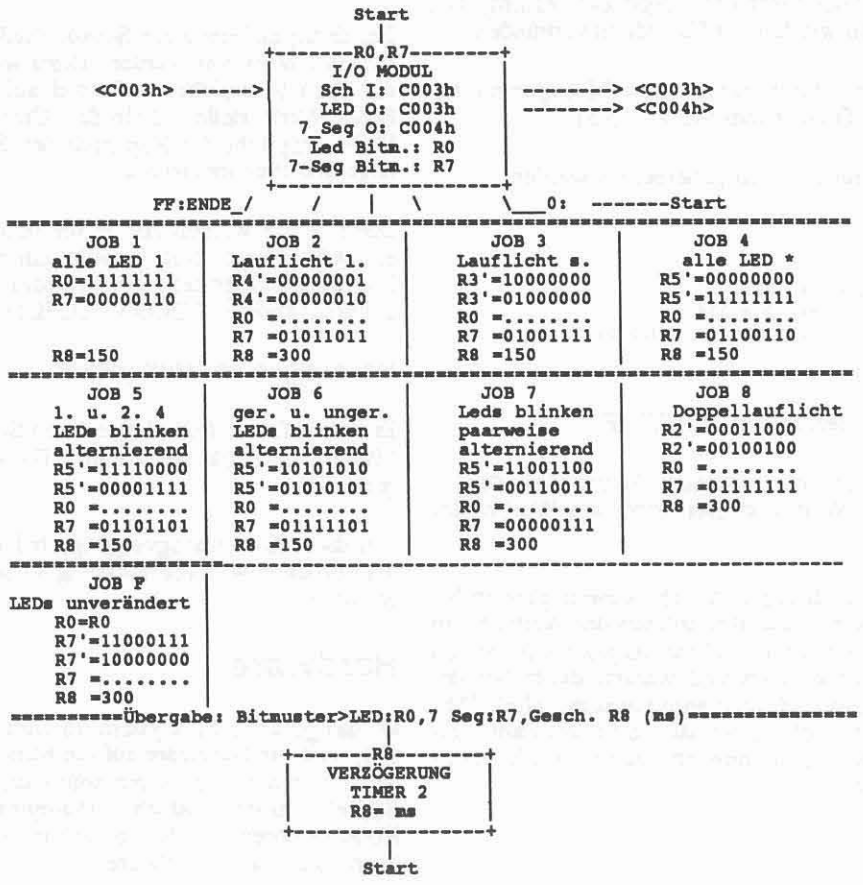
Es muß ein Fehler bei der PCAD-Programmierung unterlaufen sein. Statt einem ODER-Glied wurde versehentlich ein NAND-Glied eingelötet. Das Testprogramm wurde gestartet. Diesmal funktionierte die Ausgabe auf die 7-Segmentanzeige nicht. Der Grund dafür lag in der Software. Es wurden im Programm die beiden Adressen C003h und C004h verwechselt.

Nun war das Projekt voll funktionstüchtig.

ZEITPLAN

Schaltungsentwurf:	12 Std.
Schaltungsdimensionierung (Bauteilwahl):	6 Std.
PCAD	
- Schaltungsentwurf, Verbesserung, Optimierung:	30 Std.
- Zuweisung (Cross-Reference-List)	4 Std.
- Erstellung sonstiger Listen	4 Std.
- Plazieren der Bauteile (PCPLACE)	10 Std.
- Vorbereitung, Entflechtung in PCROUTE	3 Std.
- Ausdruck der Pläne,Listen	2 Std.
Printerstellung	
- Plazieren der Bauteile	4 Std.
- Verdrahtung nach Verdrahtungsliste	15 Std.
- Inbetriebnahme,Fehlersuche,Optimierung	15 Std.
Software	
- Wissensaneignung (FSD51,ASM51,8051..)	40 Std.
- Überlegung,Entwicklung,Erstellung	30 Std.
- Tests in Verbindung mit Hardware	10 Std.
Dokumentation,Wartung,Tests,Verbesserungen	30 Std.
	215 Std.

Bei diesen Stundenangaben sind nicht nur die verbrachten FTKL-Unterichtsstunden, sondern auch die im aktuellen Fachgebiet GLP und die privat aufgewendete Arbeitszeit miteinberechnet.



Drucksensor mit 80C522

Markus Andlinger, TGM/NT/0SL90 TGM-DSK-166, TGM-LIT-008

Anm.d.Red.: Die gesamte Dokumentation, insbesondere alle erklärenden Zeichnungen, Konstruktionsunterlagen können getrennt beim Clubbüro bestellt werden. (TGM-LIT-014: Layouts, Schaltplan, Bestückungsplan, Ausdruck des Assemblerprogrammes und des Pascalprogramms(zum Empfang der Daten am PC), Zusammenstellungszeichnung, Drehplan für Distanzbolzen, Verschiedene Datenblätter des 80C552 und DG509. Die PCAD-Zeichnungen, sowie die Source-Codes des TURBO-PASCAL-, sowie des ASSEMBLER-Programms sind auf Diskette TGM-DSK-166.

Aufgabenstellung

Für einen Robotergreifarm soll ein Drucksensor entwickelt werden. Dieser soll mittels kapazitiver Druckmessung arbeiten und aus acht Feldern bestehen. Das Meßsystem soll mit einem Mikrokontroller arbeiten, wobei der 80C552 empfohlen wurde. Die Genauigkeit der Messung soll im Promillbereich liegen.

Problemlösung

Aufbau des Sensors

Der Sensor besteht aus acht einzelnen Sensorflächen, die auf einer Fläche von 88 mm x 38mm untergebracht sind. Bei einem Abstand von ca. 3mm der einzelnen Flächen voneinander und von Rand ergibt sich eine Fläche von ca. 250 mm (14 x 18) .

Um eine möglichst große Kapazität zu erreichen wir der Sensor als Schichtkondensator mit zwei Lagen aufgebaut. Die beiden aussenliegenden Flächen werden dabei auf GND gelegt, wodurch die Störeinstrahlung verringert wird. (Die obenliegende GND-Fläche ist unterbrochen ausgeführt um die gegenseitige Belastung der Flächen gering zu halten).

Der Sensor wird aus acht einzelnen Flächen aufgebaut und mit einer druchgehenden Gummischicht abgedeckt. (1mm) Die Einzelnen Kupferflächen wurden mit Fädeldraht verbunden

Der Plattenabstand wird durch einen 2mm Moosgummi bestimmt. (angenommene Dielektrizitätsk. $\varepsilon r = 3,5$)

Aus diesen Angaben kann C wie folgt berechnet werden:

$$C = \frac{\varepsilon_0 \cdot \varepsilon r \cdot A}{d}$$

d ... Abstand der Platten ca. 2mm
A ... Sensorfläche ca. 2 x 250 mm
Dielektrizitätskonst. ca. 8,8E-12 x 3,5 As/Vm

Es ergibt sich so eine Kapazität von C = 7,7 pF

Es wird angenommen, daß die maximale Kapazitätsänderung ca. 50% des errechneten Wertes ist. Der errechnete Wert ist der Minimalwert.

Durch die geringe Kapazität ergeben sich Schwierigkeiten bei der Messung, da sich bereits die Kapazitäten des Aufbaues in dieser Größenordnung befinden. Diese Kapazitäten werden jedoch als konstant angenommen und werden daher bei der Eichung des Systems mitgeeicht und somit ausgeschaltet. Dies gilt auch für alle anderen Fehler, die kurzzeitig konstant bleiben. Langzeitige Änderungen müssen durch wiederholtes Eichen korrigiert werden.

Meßsystem

Die zu messenden Kapazitäten liegen im Bereich von 10 pF !

Zur Messung wird die Kapazität auf eine Referenzspannung (2.5V) aufgeladen und über einen Widerstand (10M) entladen. Nach einer bestimmten Entladedauer (ca. 300 µs) wird die Spannung Ux am Kondensator gemessen.

Berechnung der Kapazität aus der Spannung Ux

Die Kapazität kann aus den nun bekannten Größen : Ut=0 Ux=Ut=300 µs t berechnet werden.

Da der Sensor geeicht wird, ist eine Umrechnung nicht nötig. Es wird nur die Spannung Ux einem gewissen Druck zugeordnet, da die Referenzspannung und die Periodendauer t gleichbleiben.

Um Störungen durch einen 50 Hz Brumm auszuschließen soll über 20 ms gemessen werde. Dies ist jedoch nicht möglich, da für einen sinnvolle Endspannung Ux, C mit einem Widerstand von ca. 1,5 GOhm !! entladen werden muß. Es wird daher t mit ca 300 µs angenommen und über eine Periode von 20 ms wiederholt gemessen. Die erhaltenen Werte werden dann gemittelt.

Maximalwert der SPANNUNGSÄNDERUNG

Die maximale Spannungsänderung ist bei bekannten Cmin und Cmax von t abhängig.

Bei fertig aufgebautem Sensor muß die Leerlaufkapazität des Systems bestimmt werden. Dazu werden die ADC-Werte mit der oben beschriebenen Formel auf Kapazitäten umgerechnet. Diese Werte stellen Cmin dar. Cmax ist Cmin +7 pF . Dieser Wert entspricht der Kapazität des Sensors die im Absatz 2.2 ungefähr errechnet wurde.

Diese Werte werden nun in die obige Formel eingesetzt und t ermittelt. Diese Zeit t sollte dann auch im Programm als Entladezeit eingestellt werden. (das ist nur in 100 µs-Schritten möglich; ändern: ADCON, ZAHL1)

Messungen am Multiplexer

Es wurde festgestellt, daß es beim Schalten des Multiplexers zu Einschwingvorgängen kommt. Diese wurden am Oszilloskop gemessen.

Um diese Einschwingvorgänge bei der Messung möglichst zu unterdrücken wird die Messung 11 µs nach dem Schaltvorgang gestartet.

Hardware

Da das gesamte Meßsystem im Greifarm untergebracht werden soll, wird die Hardware auf ein Minimum reduziert. Sie besteht daher nur aus dem Mikrokontoller, einem ROM mit PLCC-Sockel, einem 8fach Multiplexer, Widerständen und Kondensatoren. Der Sensor soll mittels I C - Bus oder der V24-Schnittstelle kommunizieren.

Software

Die Software muß für die richtigen Meßintervalle, Einlesen der Meßwerte, eventuelle Fehlerkontrolle und die Datenübertragung sorgen. Zur Zeit dieser Dokumentation wurde die Software gerade mittels eines 80552 - Emulators getestet. Es wurde die Meßwerteermittlung, die Mittelwertbildung, Differenzwertbildung und die Übertragung über eine RS232 realisiert und getestet.

Das Programm ist in Assemblercode geschrieben. Das Programm DRSENS.ROM ist funktionsfähig und kann ins ROM gebrannt werden.

Softwarebeschreibung

Prinzipieller Aufbau:

Das Programm ist in vier Abschnitte gegliedert:
- Initialisierung
- Messung
- Bearbeitung der Meßwerte
- Übertragung der Meßwerte

Meßroutine

Die Meßroutine ist in ein 100 µs-Raster unterteilt, in dem die Messungen vorgenommen werden. Dieser Takt wird mit dem Timer 0 erzeugt, der regelmäßig Interrupts erzeugt. Während eines solchen Taktes wird zuerst der Multiplexer weitergeschaltet, ca. 20 µs (erzeugt mit Timer 1) später der ADC gestartet (um Einschwingvorg. des MUX nicht mitzumessen) und nach weiteren 54 µs (Wandelzeit des ADC) der eben gemessene Wert ausgelesen und teilweise bearbeitet (Addition zu den vorherigen Werten). Es muß dabei beachtet werden, daß der jeweilige Sensor erst 3 Perioden (320 µs) nach der erfolgten Aufladung gemessen wird.

Beispiel der Messung an C1

Da vor der Messung zuerst alle Kondensatoren aufgeladen werden müssen, sind während des ersten Meßzyklusses (ersten 4 ADC-Interrupts) die ADC-Werte ungültig.

Nach der Messung von 8x25 Werten ist die 20 mS Messung vollständig, und die Meßroutine wird beendet. (es besteht jedoch die Möglichkeit vielfache von 20 mS zu messen [Wert von MPER]). Diese 20 mS Messungen sollen eine Unterdrückung des 50 Hz-Brumms bewirken.

Zeitlicher Ablauf der Interrupts während der Messung

Da in einem fixen Zeitraster gearbeitet wird, muß darauf geachtet werden, daß die Abarbeitung der Routinen in dieses Raster paßt.

Die ADC-Interruptroutine (einlesen und Addition der Werte) muß abgeschlossens sein wenn ein neuer ADC-Interrupt auftritt (alle 100 µs).

Verwendete Funktionen des µC während der Meßroutine

Timer 0 : erzeugt 100 µs Raster HIGH Interruptpriority
Timer 1 : startet nach 20 µs den ADC HIGH Interruptpriority

(wird für die V24 benötigt und muß umprogrammiert werden wenn er hier verwendet wird. Es kann der ADC-Start jedoch auch mit NOP's verzögert werden. Diese 20 µs fehlen jedoch als Rechenzeit und verkürzen somit die maximale Länge der ADC-Introutine)

ADC : braucht 50 µs/Wandlung LOW Interruptpriority

Bearbeitung der Meßwerte

Die durch die Messung erhaltenen Werte sind Summen von 25 Einzelwerten. Es wird nun dividiert (mit DIVI-Routine) und so der Mittelwert gebildet.

Divisionsroutine

Die Divisionsroutine dividiert eine 16-Bitzahl durch eine 8-Bitzahl, wobei das Komma um zwei Stellen verschoben wird und so eine 12-Bitzahl das Ergebnis ist. (durch Oversampling gewonnene 2 Bit).

Formel für Bitgewinn :
Oversampling = $2^{2 \cdot n}$ n ... gewünschter Bitgewinn

Die Divisionsmethode beruht auf dem gleichen Prinzip wie die Division am "Papier" .

Da der Divisor beliebig gewählt werden kann, können Fehler ausgeschieden werden und dann durch den entspr. verringerten Divisor dividiert werden. (Es müssen jedoch mindestens 16 Werte für das Oversampling vorhanden sein)

Differenzwertbildung

Die Ergebnisse der ersten 20 ms Messung (nach starten des Programmes; Sensor muß unbelastet sein) werden als 'Nullabgleich' verwendet und die darauf folgenden Werte darauf bezogen. Diese Differenzbildung soll bewirken, daß nur die Änderung angezeigt wird die durch einen Druck auf den Sensor verursacht wird. Durch einen EXTINT1 wird ein neuerlicher Nullabgleich durchgeführt.

Übertragung der Meßwerte

Die Übertragung der Werte erfolgt über die V24-Schnittstelle. Es wird dazu der Timer 1 benötigt, der neu programmiert werden muß, wenn er in der Meßroutine verwendet wird.

Die Übertragung erfolgt mit 4800 Baud, wobei der Timer auf den Wert #0FAh programmiert werden muß. (Reloadmode; kein Interrupt) Der Start der Übertragung der Meßwerte wird mit dem zweimaligen Senden von #AAh angezeigt.

Wird die SIO nur zum Senden benutzt, so ist eine unterschiedliche Verwendung von Timer 1 im Programm möglich. Soll jedoch auch empfangen werden können, so muß dieser fix programmiert werden. (ausser wenn der PC als 'SLAVE' verwendet wird)

Verwendete Funktionen des µC

RS232-Schnittstelle, Timer 1

SOFTWARESTAND

Die Software erfüllt folgende Aufgaben:

- Messung der Kapazitäten der einzelnen Meßflächen
- Mittelung der erhaltenen Meßwerte
- Differenzbildung der gemittelten Werte zu einem Referenzwert (Nullabgleich)
- Übertragung der Differenzen bzw. der Absolutwerte über die RS232

Um die übertragenen Werte am PC einzulesen wurde das Pascalprogramm DRSENS.PAS geschrieben.

Verzeichnisse detailliert

In dieser Rubrik werden ab der jetzigen Ausgabe der PCNEWS die neu hinzugekommenen Literaturstellen, Arbeitsbehelfe und Disketten beschrieben. Danach werden in den **Verzeichnissen tabellarisch** alle bisher erschienenen abgedruckt, jeweils mit Preisangabe und einem Hinweis auf die Nummer der PCNEWS in denen die detaillierte Beschreibung zu finden ist.

Tabellen

TGM-TAB-001: PC-Zeichensatz PC-8 (bereits versendet) Grafische Symbole und Alpha-Zeichen systematisch angeordnet, Alle Zeichen in Dezimal-, Hexadezimal- und Binärschreibweise.

TGM-TAB-002: MSDOS-Version 5.0e: Für Anfänger: jeder Befehl wird erklärt. Für Fortgeschrittene: Jeder Befehl wird mit Aufrufsyntax und Liste aller Optionen angeführt.

TGM-TAB-003: WINDOWS 3.0d: Alle Tastaturkommandos: Die Mausbedienung in WINDOWS ist praktisch selbsterklärend, dafür ist oft unklar, welche Taste man zu drücken hat: System, Textmarkierung, Bearbeitung, Fenster, Hilfe, Menus, Cursorbewegung, Dialogfelder, Programm-Manager, Datei-Manager, Gruppenfenster, Paintbrush, Write.

TGM-TAB-004: WORD 5.0d: Tastaturbelegung: Deutscher Text mit englischer Tastenbezeichnung: Funktionstasten, Zeichenformate, Absatzformate, Erstellen, Markieren, Cursorbewegung, Löschen/Einfügen, Arbeiten mit Befehlen, Formulare, Datei-Manager, Suchen, Makroanweisungen, Makrofunktionen, Makrooperatoren, Reservierte Makro-Variable, Checkliste Serienbrief, Gliederung, Gliederung markieren, Bezeichnungen.

TGM-TAB-005: Zeichensätze am Laserdrucker: ROMAN-8, PC-8, ANSI-100, SYMBOL (erscheint im Herbst.)

TGM-TAB-006: COREL-DRAW: alle Schriftarten und Tastaturbelegung (erscheint im Herbst.)

TGM-TAB-007: POST-SCRIPT-ZEICHENSATZ: **Sonderzeichensätze**: Line-Draw, Symbol, Zapf-Dingbats im Vergleich mit PC-8, PostScript-Zeichensatz und Postscript-PC (Was druckt WORD). Alle Schriftarten in normal-fett-kursiv-fett+kursiv sowie in den Schriftgrößen 6/9/11/17/21/34/43 pt(A5) und 8/12/16/24/30/48/60 pt(A4) (AvantGarde, Bookman, Courier, Helvetica, HelvNarrow, LineDraw, NewCentSchlbk, Palatino, Symbol, TimesRoman, ZapfChancery, ZapfDingbats).

TGM-TAB-008: MSDOS-5.0d: Für Anfänger: jeder Befehl wird erklärt. Für Fortgeschrittene: Jeder Befehl wird mit Aufrufsyntax und Liste aller Optionen angeführt.

TGM-TAB-009: WORD-5.5d Tastaturbelegung: Deutscher Text mit deutschen Tastenbezeichnungen: Funktionstasten, Tastaturstatus, Bearbeiten, Bewegen Text, Bewegen Layout, Bewegen Formulare/Serientext, Dialogfelder, Verwenden von Befehlen, Makrofunktionen, Makroanweisungen, reservierte Variable für Makros, Makro-Operatoren, Tips für das Schreiben und Bearbeiten von Makros, Druckformate, Formatierung Zeichen, Formatierung Absatz, Berechnungen, Gliedern, Löschen und Einfügen, Markieren, Markierungen ändern, Maus: Bewegung /Fenster, /Bildlauf, /Textmarkierung, /Menüs, /Dialogfelder, /Lineal, /Zeichenleiste, Übersicht: von WORD-5.0 zu WORD-5.5.

TGM-TAB-010: WORD-5.5d Tastaturbelegung: Deutscher Text mit englischen Tastenbezeichnungen; Inhalt wie TGM-TAB-011.

TGM-TAB-011: BORLAND-C-2.0d: Tastaturbelegung, Compileroptionen, Englische Tastaturbezeichnung: Menü, Editor, Fenster, Hilfe, Programm, Blockbefehle, Cursor, Einfügen, Löschen, weitere Editierbefehle, Debugger-Tastenkürzel, Jokerzeichen, BC-Kommandozeilenoptionen, BCC-Kommandozeilenoptionen

Schreiben Sie uns, welche Kurzfassung für Ihre Arbeit nützlich wäre, wir erweitern dann unsere Reihe, da wir annehmen können, daß auch andere Klubmitglieder Bedarf haben werden.

Literatur

TGM-LIT-001: Dokumentation zu WOOD-WORKS, Diskette TGM-153, Vertrieb durch DR.BIT SOFTWARE, Jaquingasse 15, 1030 Wien, (0222)-78-65-60

TGM-LIT-002: NEAT-CHIPSÄTZE, Clubabend am 3.4.1991, Alle Overheadfolien und grundlegende Literaturauszüge.

TGM-LIT-003: Grundlagen über den 8051-Prozessor in 2 Teilen (Hardware, Software, Befehlssatz)

TGM-LIT-004: Technisches Referenz-Handbuch PC, BIOS, MSDOS bis Version 4.0. Ausdruck der TGM-Diskette 140 Stand April 1991. Die TGM-Diskette 140 wird laufend ergänzt.

TGM-LIT-005: COSMIC, Internationaler Software Katalog, Ausgabe 1991.

TGM-LIT-006: Single-Chip 8-bit Microcontrollers PCB83C552, Philips.

TGM-LIT-007: SuperBase 4 unter Windows, Trial-Pack, passend zu TGM-DSK 164+165.

TGM-LIT-008: Drucksensor für den Robotergreifarm, Markus Andlinger, 0SN90; Komplette Programmdokumentation für den Microcontroller 80C552, Dimensionierung, Schaltung, Layout mit PCAD, Struktogramm, Sourcelisting. Alle zugehörigen Dateien auf Diskette TGM-DSK-166.

TGM-LIT-009: Zusammenfassung der wichtigsten Tabellen für Turbo-C++, Version 1.0, entnommen aus den Handbüchern. Kann gemeinsam mit TGM-DSK 163 verwendet werden. Eine Übersicht über diese Blätter ist im Beitrag **Materialien für den C-Unterricht** enthalten.

TGM-LIT-010: Bedienungsanleitung zum Lexikon der EDV, TGM-DSK-168.

TGM-LIT-011: Dokumentation zu OOPS (Object-Oriented-Program-Support), TGM-DSK-170-172.

TGM-LIT-012: PSPICE-Kurzdokumentation für den Unterricht, TGM-DSK-139

TGM-LIT-013: Erweiterung zum µPROFI-51, Scheiber/Vozdecky/Hoffmann, Schriftliche

Konstruktionsunterlagen, auszugsweise abgedruckt in PC-NEWS-24. Dazugehörige Diskette: TGM-DSK-175.

TGM-LIT-014: EMS-Software-Utilities, vertrieben durch Witrans, 1060 Wien, Mariahilferstraße 47/5, (0222)-587-46-00, Hr. Farhad Akavanrezayat gewährt allen Clubmitgliedern einen 20%igen Nachlass auf die Produkte: EMS-123-lib (692,-), EMS-Assembler-lib (692,-), EMS-AutoCAD-lib (1384,-), EMS-C-lib (692,-), EMS-dBase-lib (1384,-), EMD-DOS-lib (692,-), EMS-Ventura-lib (1134,-), EMS-PD-Games (692,-), EMS-Novell-lib (1059,-), EMS-TURBO-PASCAL-lib (692,-), EMS-Windows-lib (692,-), EMS-Word-Perfect-lib (692,-). Der Preis in Klammern ist der gültige Listenpreis excl. MWST, Stand September 1991. Über TGM-LIT-014 kann die Kurzbeschreibung dieser Utilities angefordert werden. Es handelt sich größtenteils um Zusammenfassungen von Public-Domain-Programmen, teils aus bestehenden Sammlungen, teils auch aus BBS-Systemen downgeloaded.

TGM-LIT-015: µPROFI-51-Bauanleitung, mit Bestückungsplan, Schaltbild, Stückliste.

TGM-LIT-016: INLASS, Assembler im TURBO-PASCAL-Source-Code, Auszug aus CHIP-Spezial 2. Der Assembler übersetzt die Mnemonics der Prozessoren 8080, Z80, 8086, 6502. Das Ergebnis wird in TURBO-PASCAL-INLINE-Anweisungen übersetzt. Eine Erweiterung wurde im Beitrag **8052-Inline-Assembler** für den 8052-Prozessor beschrieben.

TGM-LIT-017: MCS BASIC-52 Users Manual: Beschreibung der BASIC-Variante für den 8052-Prozessor (INTEL).

TGM-LIT-018: Datenblatt 8052AH-BASIC

TGM-LIT-019: Konstruktionsunterlagen zum A3-Flachbett-Plotter: Mechanischer Aufbau, Blockschaltbild, Plottercontroller, Bedienungsfeld, Schrittmotoransteuerung, umschaltbares Netzteil, 8052-Inliner-Assembler, Plotter-Software, Schaltpläne, Timings, Layouts, Bestückungspläne.

TGM-LIT-020: Datenblatt zu SAB 80515/80535-Microcontroller, siehe Beitrag **Einsatz eines 8031-Experimentiersystems im Unterricht** (PC-NEWS-24).

TGM-LIT-021: µPROFI-Handbuch

Disketten

TGM-DSK-169: NEWS24, Texte

```
NEWS24P   DFV      4096 02.09.91   21:35
NEWS24    TXT    409600 02.09.91   21:31  Text
NEWS24T   PS        383 02.09.91   17:19  Titelseite
NEWS24L   PS        142 02.09.91   21:32  Letzte Seite
POSTSCR   INI     12608 25.08.91   18:44
POSTSCR   DBS     28295 25.08.91   21:11
ADAUTO    ASM      1848 26.05.91    8:49
MSB32ANS  PLT     35017 18.08.91   12:23
MSB32MEM  PLT     29061 15.08.91   20:46
MSB32BLO  PLT     47448 21.08.91   12:13
TAB         <DIR>     01.09.91   10:25  Tabellenblätter
AVSIM       <DIR>     01.09.91   10:27  805x-Simulator
MAP2SYM     <DIR>     01.09.91   10:29  Symdateien-Anpassung
EXTMOD      <DIR>     01.09.91   10:29  Externe Module
C           <DIR>     01.09.91   10:30  C-Compiler f.805x
\TAB
BC20D     DFV      1536 20.08.91   15:15
BC20D     TXT     21504 20.08.91   13:35
DOS50D    BAT      2920 17.06.91   19:29
LAS_PS    DFV      2560 25.08.91   21:04
LAS_PS    TXT     23040 30.08.91    7:53
MSDOS50   DFV      1024 18.08.91   12:53
MSDOS50D  TXT     15360 18.08.91   12:42
MSDOS50E  DFV      1536 05.06.91   21:40
MSDOS50E  TXT     14336 30.08.91    6:54
WIN30D    DFV      1024 18.08.91   12:41
WIN30D    TXT     12800 18.08.91   13:29
WORD50D   DFV      1024 18.08.91   13:28
WORD50D   TXT     17920 18.08.91   13:28
WORD55D   DFV      1536 19.08.91   20:05
WORD55D   TXT     68608 30.08.91    9:14
\AVSIM
8051      BAT       110 12.12.90   10:07
ADD       ASM       193 19.09.90   11:26
ADD       HEX        44 17.10.90   11:48
ADD       LST       940 17.10.90   11:48
ADD       SYM        29 17.10.90   11:48
ASM       BAT        39 19.09.90   11:27
AT1       ASM      2371 12.09.90   11:07
BASIS     ASM      3456 22.01.87   18:29
BINBCD    ASM       243 20.03.90    7:46
DEMO      CMD       118 29.08.86   19:57
DEMO      MSG       313 29.08.86   19:57
DEMO1     ASM      4334 29.08.86   19:57
DEMO1     HEX       372 29.08.86   19:57
DEMO1     SYM       333 29.08.86   19:57
DEMO2     ASM      4639 16.10.90   22:12
DEMO2     HEX       424 16.10.90   22:12
DEMO2     LST     11188 16.10.90   22:12
DEMO2     OBJ       981 16.10.90   22:12
DEMO2     SYM         1 16.10.90   22:12
DOK       DFV      1024 26.09.90    0:00
DOK       TXT     20480 25.09.90   23:52
PAR       DAT         5 24.10.90    0:25
README                343 17.10.90   12:33
RUN_DEMO  BAT       239 29.08.86   19:57
T1        ASM      1280 25.09.90   20:33
T2        ASM       428 26.09.90    0:37
T3        ASM       426 04.04.90   15:55
T4        ASM       640 22.01.87   17:01
T5        ASM       217 17.10.90    0:03
T6        ASM       233 12.12.90   10:05
T6        HEX        38 12.12.90   10:08
T6        LST      1331 12.12.90   10:07
T6        OBJ       689 12.12.90   10:07
T6        SYM        29 12.12.90   10:08
T7        ASM       250 12.12.90   10:59
T7        HEX        48 12.12.90   11:00
T7        LST      1348 12.12.90   11:00
T7        OBJ       950 12.12.90   11:00
T7        SYM        71 12.12.90   11:00
T7A       SYM        43 12.12.90   10:17
T7M       SYM        37 12.12.90   10:23
T7N       SYM        40 12.12.90   10:24
T7S       SYM       110 12.12.90   10:19
T7X       SYM        55 12.12.90   10:23
TEST      ASM      2371 12.09.90   11:07
TEST      HEX       366 19.09.90   10:54
TEST      IO          2 07.11.90   10:58
TEST1     ASM       553 14.12.90   16:18
TEST1     HEX        44 14.12.90   16:24
TEST1     M51       267 14.12.90   16:11
TEST1     OBJ       580 14.12.90   16:19
ASS       BAT       728 30.01.89    0:01
RUNME     BAT       852 30.01.89    0:01
\MAP2SYM
MAP2SYM   C       16260 15.05.91   13:38
MAP2SYM   EXE     13939 14.05.91   17:12
\EXTMOD
EXTMOD1               115 02.07.91    9:03
EXTMOD1   ASM      2098 02.07.91    9:03
EXTMOD1   HEX       210 02.07.91    9:29
EXTMOD1   LST      5964 24.08.91    8:39
EXTMOD1   M51      1042 02.07.91    9:03
EXTMOD1   OBJ       604 24.08.91    8:40
EXTMOD2   ASM      1369 27.06.91   17:52
EXTMOD2   LST      3378 27.06.91   17:57
EXTMOD2   OBJ       127 27.06.91   17:57
EXTMOD3   ASM      1478 28.06.91    7:22
EXTMOD3   LST      3555 24.08.91   10:48
EXTMOD3   OBJ       602 24.08.91   10:48
EXTMOD1A  ASM      2427 25.08.91   12:01
EXTMOD1A  HEX       223 25.08.91   12:22
EXTMOD1A  LST      4901 25.08.91   12:03
EXTMOD1A  MAP      1587 25.08.91   12:22
EXTMOD1A  OBJ       920 25.08.91   12:03
EXTMOD2A  ASM      1614 25.08.91   12:06
EXTMOD2A  LST      3451 25.08.91   12:06
EXTMOD2A  OBJ       782 25.08.91   12:06
EXTMOD3A  ASM      2069 25.08.91   12:08
EXTMOD3A  LST      4387 25.08.91   12:08
EXTMOD3A  OBJ       792 25.08.91   12:08
\C
INIT      BAT       148 20.06.91   10:52
SETPATH   BAT        86 20.06.91   10:56
C51       BAT        96 18.06.91   15:54
TEST      C         142 13.06.91   13:00
TEST      HEX      1132 21.06.91   10:15
C552SR    H       14572 17.02.90   17:37
C552SRP   H       14269 24.05.91   15:42
C552SR    DEF      4383 17.02.90   17:37
DATH      LNK       112 18.06.91   15:59
DATX      LNK       112 18.06.91   16:00
RT51      SRC     12898 06.06.91   13:06
SIM       BAT        30 31.05.91   14:31
SIM       CMD        39 18.06.91   15:58
SIM       SYM        45 26.04.91   18:03
DIAGRAMM  DOC      2155 20.06.91   16:48
ENTWICKL  WP      13015 21.06.91   11:40
FSD51     WP       5173 21.06.91   11:58
FSD51     TXT      3332 28.08.91   19:03
ENTWICKL  ASC     58368 28.08.91   20:57
ENTWICKL  TXT     13102 28.08.91   19:18
```

TGM-DSK-170 - 172: OOPS 2.2

Dazugehörige Dokumentation TGM-LIT-11.

The Object-Oriented Program Support (OOPS) class library is a portable collection of classes similar to those of Smalltalk-80 that has been developed using the C++ programming language under the UNIX operating system. The OOPS library includes generally useful data types such as String, Date, and Time, and most of the Smalltalk-80 collection classes such as OrderedCltn

(indexed arrays), LinkedList (singly-linked lists), Set (hash tables), and Dictionary (associative arrays). Arbitrarily complex data structures comprised of OOPS and user-defined objects can be stored on disk files or moved between UNIX processes by means of an Object I/O facility. Classes Process, Scheduler, Semaphore, and SharedQueue provide multiprogramming with coroutines.

Here's the hierarchy of the OOPS classes implemented in Version 2, Release 2:

Object -- Root of the OOPS Class Inheritance Tree
- **Bitset** -- Set of Small Integers (something like Pascal's type SET)
- **Class** -- Class Descriptor
- **Collection** -- Abstract Class for Collection Functions
  - **ArrayOb** -- Array of Object Pointers
  - **Arraychar** -- Byte Array
  - **Bag** -- Unordered Collection of Objects
  - **Set** -- Unordered Collection of Non-Duplicate Objects
    - **Dictionary** -- Set of Associations
      - **IdentDict** -- Dictionary Keyed by Object ID
  - **SeqCltn** -- Abstract Class for Ordered, Indexed Collections
    - **LinkedList** -- Singly-Linked List
    - **OrderedCltn** -- Ordered Collection of Object Pointers
      - **SortedCltn** -- Sorted Collection of Objects
    - **Stack** -- Stack of Object Pointers
- **Date** -- Calendar Date
- **Float** -- Floating Point Number Object
- **Fraction** -- (Toy) Rational Arithmetic
- **Link** -- Abstract Class for LinkedList Links
  - **LinkOb** -- Link Containing Object Pointer
  - **Process** -- Co-routine Process Object
- **LookupKey** -- Abstract Class for Dictionary Associations
  - **Assoc** -- Association of Object Pointers
  - **AssocInt** -- Association of Object Pointer with Integer
- **Integer** -- Integer Number Object
- **Iterator** -- Collection Iterator
- **Nil** -- The Nil Object
- **Point** -- X-Y Coordinate Pair
- **Random** -- Uniform Random Number Generator
- **Range** -- Integer Range
- **Rectangle** -- Rectangle
- **Scheduler** -- Co-routine Process Scheduler
- **Semaphore** -- Process Synchronization Class
- **SharedQueue** -- Shared Queue of Objects
- **String** -- Character String
  - **Regex** -- Regular Expression Pattern
- **Time** -- Time of Day
- **Vector** -- Abstract Class for Math Vector Functions
  - **BitVec**
  - **ByteVec**
  - **ShortVec**
  - **IntVec**
  - **LongVec**
  - **FloatVec**
  - **DoubleVec**
  - **ComplexVec**

Anmerkung: Diese Bibliothek wurde für UNIX geschrieben. Sollte sie für MSDOS etwa mit TURBO-C oder ZORTECH-C kompiliert werden, sind Änderungen durchzuführen. Clubmitglieder, die diese Bibliothek verwenden wollen und die MSDOS-Erweiterung durchführen, werden gebeten, uns die modifizierte Version zur Verfügung zu stellen.

TGM-DSK-170: OOPS 2.2 1(3)

```
README          5996 11-23-88   9:31a
BUGS121         7914 11-23-88   9:32a
INSTALL         9028 11-23-88   9:33a
MAKEFILE        2915 11-23-88   9:33a
COPYRIGH         718 11-23-88   9:33a
PKUNPAK EXE    15112 08-02-88  12:00a
SRC     <DIR>        11-22-59   7:18p
TEST    <DIR>        11-22-59   7:20p
EX      <DIR>        11-22-59   7:20p
PAPER   <DIR>        11-22-59   7:22p
\SRC
SRC     ARC   215997 11-22-59   6:56p
\EX
EX      ARC     8359 11-22-59   6:59p
\TEST
TEST    ARC    68834 11-22-59   7:02p
\PAPER
PAPER   ARC     7267 11-22-59   7:00p
```

TGM-DSK-171: OOPS 2.2 2(3)

```
ERRFAC          <DIR>        11-22-59   7:24p
VECTOR          <DIR>        11-22-59   7:24p
VECTEST         <DIR>        11-22-59   7:24p
SRC             <DIR>        11-22-59   7:26p
EX              <DIR>        11-22-59   7:29p
PKUNPAK EXE    15112 08-02-88  12:00a
PAPER           <DIR>        11-22-59   7:31p
\ERRFAC
ERRFAC  ARC     16137 11-22-59   6:57p
\VECTOR
VECTOR  ARC     40940 11-22-59   7:04p
\VECTEST
VECTEST ARC     12597 11-22-59   7:03p
\SRC
SRC     ARC   215997 11-22-59   6:56p
\EX
\PAPER
PAPER   ARC     7267 11-22-59   7:00p
```

TGM-DSK-172: OOPS 2.2 3(3)

```
LIBVAX          <DIR>        01-01-80  12:55a
LIBSUN3         <DIR>        01-01-80  12:56a
LIBRT-AI        <DIR>        01-01-80  12:56a
LIBRT-AC        <DIR>        01-01-80  12:56a
LIBMASSC        <DIR>        01-01-80  12:56a
LIBHP320        <DIR>        01-01-80  12:56a
\LIBHP320
MAKEFILE 1          159 01-06-89   2:29p
MAKEFILE 2          592 01-06-89   2:29p
EXCHJ    S         2316 01-06-89   2:29p
MAKE     ALL         33 01-06-89   2:30p
\LIBMASSC
MAKEFILE 1          159 01-06-89   2:28p
MAKEFILE 2          159 01-06-89   2:28p
EXCHJ    C         2392 01-06-89   2:28p
EXCHJ    S        13452 01-06-89   2:28p
EXCHJJ   S         2996 01-06-89   2:28p
\LIBRT-AC
README              689 01-06-89   2:25p
CSTACK   C          636 01-06-89   2:25p
MAKEFILE 1          164 01-06-89   2:26p
MAKEFILE 2          164 01-06-89   2:26p
EXCHJ    C         2392 01-06-89   2:26p
EXCHJ    S        18190 01-06-89   2:26p
EXCHJJ   S         3236 01-06-89   2:27p
\LIBRT-AI
MAKEFILE 1          164 01-06-89   2:23p
MAKEFILE 2          164 01-06-89   2:23p
README               44 01-06-89   2:24p
\LIBSUN3
MAKEFILE 1          159 01-06-89   2:22p
MAKEFILE 2          159 01-06-89   2:22p
EXCHJ    S         2989 01-06-89   2:22p
\LIBVAX
MAKEFILE 1          159 01-06-89   2:20p
MAKEFILE 2          159 01-06-89   2:20p
EXCHJ    S         2511 01-06-89   2:21p
```

TGM-DSK-173: PC-Professionell-Utilities, Teil IV

Diese Diskette enthält die PC-Professionell-Utilities (Teil IV) und liegt der Ausgabe 9/91 der Computerzeitschrift PC-Professionell bei. Der Inhalt dieser Diskette setzt sich aus zwei außergewöhnlichen Utilities, Demo-Programmen der Rubrik Knowhow sowie den Programmen und Stapeldateien der Tips&Tricks-Rubrik zusammen:

CDX: ein Utility zum raschen Finden von Dateien und Verzeichnissen

ENCORE: ein Utility zur Automatisierung von Tastatureingaben

KNOWHOW6: Demo-Programme für EGA- und VGA-Karten (32 Textseiten)

KNOWHOW8: Demo-Programme für VGA-Karten

PCPSAVER: ein Screensaver für Windows 3.0 (Version 1.2)
TTPROG: eine Sammlung kleiner Programme und Stapeldateien

Alle Programme außer dem Screensaver liegen auch im Quellcode vor und lassen sich von dem Microsoft-Assembler MASM, von Borlands Turbo Pascal sowie den aktuellen C-Compilern von Borland und Microsoft in lauffähigen Programmcode übersetzen. Eine ausführliche Anleitung zum Utility ENCORE finden Sie in der Ausgabe 9/91 der PC Professionell. Das Utility CDX, die Programme des Verzeichnisses KNOWHOW8 sowie die Programme und Stapeldateien des Verzeichnisses TTPROG sind in der Ausgabe 8/91 genau beschrieben. Über Einsatz und Zweck der Programme des Verzeichnisses KNOWHOW6 erhalten Sie ausführliche Informationen in der Ausgabe 6/91 der PC Professionell. Für den PCP-Saver finden Sie die Installationshinweise auf dieser Diskette und weitere Bedienungshinweise in der umfangreichen integrierten Hilfsfunktion.

```
Dateien der PC-Professionell-Utilities  (Teil IV)

\CDX                    Pfadfinder für Dateien/Verzeichnisse
CDX      .C             Quelltext-Datei des Pfadfinders
CDX      .COM           Programmcode des Utilites
COMPILE  .BAT           Batch-Datei zum Compilieren
\ENCORE                 Tastatur-Recorder
ENCORE   .ASM           Quelltext-Datei des Recorders
ENCORE   .COM           Programmcode des Recorders
\KNOWHOW6               32 Textseiten bei EGA- und VGA-Karten
VGA32C   .C             Quelltext-Datei in C
VGA32C   .EXE           Programmcode für das C-Programm
VGA32P   .EXE           Programmcode für das Pascal-Programm
VGA32P   .PAS           Quelltext-Datei in Pascal
\KNOWHOW8               2 Grafikseiten im 320x200-Modus
256-2C   .C             Quelltext-Datei in C
256-2C   .EXE           Programmcode für das C-Programm
256-2P   .EXE           Programmcode für das Pascal-Programm
256-2P   .PAS           Quelltext-Datei in Pascal
\PCPSAVER               PCP-Screensaver für Windows 3.0
INSTALL  .TXT           Installationshinweise zum Screensaver
PCPSAVER.EXE            Programmcode des PCP-Saver
PCPSAVER.HLP            Hilfstexte zum PCP-Saver
PCPSHOOK.DLL            Funktionsbibliothek zum PCP-Saver
\TTPROG                 Tips & Tricks für Programmierer
DOSVER   .COM           Programmcode des DOSVER-Utilities
DOSVER   .DEB           Eingabe-Datei für den MS-DOS-Debugger
NORT1    .BAT           Batch-Datei für die NC-Menüauswahl
NORT2    .BAT           Batch-Datei für die NC-Menüauswahl
NUMOFF   .COM           Programmcode des NUMOFF-Utilities
NUMOFF   .DEB           Eingabe-Datei für den MS-DOS-Debugger
NUMON    .COM           Programmcode des NUMON-Utilities
NUMON    .DEB           Eingabe-Datei für den MS-DOS-Debugger
PRTOFF   .COM           Programmcode des PRTOFF-Utilities
PRTOFF   .DEB           Eingabe-Datei für den MS-DOS-Debugger
PRTON    .COM           Programmcode des PRTON-Utilties
PRTON    .DEB           Eingabe-Datei für den MS-DOS-Debugger
RENDIR   .EXE           Programmcode des RENDIR-Utilities
RENDIR   .PAS           Quelltext-Datei des RENDIR-Utilities
```

TGM-DSK-174: Texte für den Unterricht, Elektronik, EDV

Folgende Beispielsammlungen habe ich für den Unterricht auf Diskette zusammengestellt und stelle sie gerne allen Kollegen zur Verfügung:

Elektronik 1 (ELEKTR1.TXT, 42kB, Word)
Diode, Transistor, Kleinsignalverstärker

Digitaltechnik 1 (DIGITAL1.TXT, 13kB, Word)
Grundgatter, Boole'sche Rechengesetze, binäres Rechnen, Karnaugh-Diagramm

Mechanische Technologie (FTKL1., 15kB, Wordstar)
Metalle, Nichtmetalle, spanende und spanlose Formgebung, etc.

MS-DOS (EDV1., 11kB, Wordstar)

Turbo-Pascal (EDV2., 50kB, Wordstar)
Theoriefragen und Programmierbeispiele, Struktogramme, etc.
incl. Dateien, ohne Zeiger

MS-DOS-Befehlsübersicht (DOSUBERS., 5kB, Wordstar)
Checkliste für den Unterricht ohne Beschreibung

Bei Interesse bitte:
- eine leere, formatierte Diskette mit Name und Dateiwunsch an
  TGM - Assistenten Nachrichtentechnik, 12. Stock
  oder
  TGM - Melchart, Zimmer 1428

bzw.
- Diskette und frankiertes Rückkuvert an
  Josef Melchart
  Baumgartenstraße 61
  1140 Wien

bzw.
- Bestellformular am Ende dieses Heftes verwenden.

```
ELEKTR1  TXT     41984 21.05.91   21:35
DIGITAL1 TXT     16384 02.06.91   22:14
FTKL1            15360 08.03.91   23:34
EDV1             11648 20.05.91   22:44
EDV2             50048 20.05.91   22:55
DOSUBERS          4992 07.04.91   21:55
```

TGM-DSK-175: FTKL-Programm Hardwareerweiterung zum µPROFI 51

Zur Bearbeitung der Dateien sind folgende Programme erforderlich: \TXT: SideKick (oder gleichwertiger ASCII-Editor), \PCAD: Leiterplattenentflechtungssystem PCAD, \SOFT: X8051, LINK, FSD-51. Beschreibung des Konstruktionsprogramms im Beitrag Hardware-Erweiterung zum µPROFI 51.

```
PCAD            <DIR>      06.06.91   20:57
TXT             <DIR>      06.06.91   20:58
SOFT            <DIR>      06.06.91   20:59
\TXT
SKFTKL   TXT     23735 16.01.91    7:22
WPFTKL   TXT      7583 03.08.00   22:55
\PCAD
BAUTEIL  PLT     13836 11.01.91   17:37
BESTUCK  PLT     15016 11.01.91   19:25
EURO     PCB      2914 16.04.91   12:34
FTKL1    BNL     11829 11.12.90   15:33
FTKL1    CMD      3650 11.12.90   15:33
FTKL1    CMP      9585 09.01.91   18:36
FTKL1    CTL       980 11.01.91   18:52
FTKL1    FIL      1330 11.12.90   14:50
FTKL1    HIS      1453 11.12.90   17:37
FTKL1    HP      61815 08.01.91   14:29
FTKL1    LIB      1747 11.12.90   14:53
FTKL1    MAT      2742 09.01.91   18:36
FTKL1    NDE     17986 09.01.91   18:36
FTKL1    NLC      7700 08.01.91   15:07
FTKL1    NLT     11823 09.01.91   18:33
FTKL1    PCB     35880 11.01.91   19:00
FTKL1    PKG     28004 11.01.91   18:20
FTKL1    PKL      7207 09.01.91   18:36
FTKL1    PLC     28004 11.01.91   18:40
FTKL1    PLT     26888 08.01.91   14:27
FTKL1    PNL     10037 08.01.91   15:03
FTKL1    REP      6604 11.01.91   19:00
FTKL1    RTE     30611 11.01.91   18:52
FTKL1    RTS     25600 11.01.91   18:59
FTKL1    SCH     53778 11.12.90   13:09
FTKL1    WRL      7386 09.01.91   18:36
HORILAY  PLT     18524 11.01.91   17:36
LAYOUT1  PLT     11496 11.01.91   19:21
LAYOUT2  PLT     12304 11.01.91   19:22
PCAD1    CTL       903 11.01.91   18:48
PCCARDS  CMD      5057 11.01.91   19:25
PCCARDS  DBG       117 11.01.91   19:25
ROUTCTFL CTF      1073 11.01.91   19:00
VERTLAY  PLT     18276 11.01.91   17:34
\SOFT
FTKL     OBJ      1764 09.01.91   17:31
FTKL     ASM     11284 09.01.91   17:30
FTKL     LST     20025 09.01.91   17:31
FTKL     HEX       997 09.01.91   17:31
FTKL     SYM         1 09.01.91   17:31
ASM      BAT        39 19.09.90   11:27
8051     BAT       110 12.12.90   10:07
```

TGM-DSK-176-188: Programmdisketten zum Microsoft System Journal, deutsche Ausgabe 9/89-10/91

Der PCC-TGM hat das MSJ, deutsche Ausgabe abonniert und stellt den Mitgliedern die Inhalte im Rahmen der folgenden Aufstellung vor. Wenn Sie Interesse an einem Artikel haben, werden Ihnen die Damen und Herren des Clubbüros gegen Ersatz der Unkosten eine Kopie des Textes anfertigen; bitte am Diskettenbestellschein vermerken.

TGM-DSK-176: Microsoft System Journal Ausgabe Sep./Okt. 1989, "MSJDSK35"

\8259 L1.LST, L2.LST, L3.LST: Listings aus dem Artikel "Steuerung von Ein-/Ausgabegeräten", S.166

\BASIC GETCOLOR.BAS, WORDWRAP.BAS: Listings aus dem Artikel "Basic als professionelle Programmiersprache", S.59

\OOP OOPDEMO.PAS: Listing aus dem Artikel "Das neue QuickPascal", S.68

\SAA5 DLG.C, DLGAB.C, DLGDEMO.C, DLGEDIT.C, DLG.H: Listings aus dem Artikel "Erstellung und verwaltung von SAA-Dialogboxen", S.116

\VEC VECTFONT, VECTFONT.C, VF00.C, VF01.C, VF02.C, VF03.C, VF04.C, VF05.C, VF06.C, VF07.C, VF08.C, VF09.C, VF10.C, VF11.C, VF12.C, VF13.C, VF14.C, VF15.C, VECTFONT.DEF, VECTFONT.H, VECTFONT.LNK, VECTFONT.RC, VECTFONT.SYM: Listings aus dem Artikel "Vektorfonts unter dem Presentation-Manager", S.6

\VMM VM.C, VM.H: Listings aus dem Artikel "Eine virtuelle Speicherverwaltung in C", S.104

\WINHELP HELPMGR.C, SHAPE.C, SPLIT.C, SHAPE.DEF, DEFS.H, HELPMGR.H, SHAPE.MD, SHAPE.RC, HELP.TXT: Listings aus dem Artikel "Hilfe-Verwaltung für Windows", S.30

TGM-DSK-177: Microsoft System Journal Ausgabe Nov./Dez. 1989, "MSJDSK36"

\8087 MATHMOD.ASM, UMFANG.ASM, WURZEL.ASM, GRAPH.C Listings aus "Auf, auf und davon", S. 187

\DDE CLIENT.C, CLIENTMN.C, CLINIT.C, SERVER.C, SRVRINI.C, SRVRMAIN.C, CLIENT.H,SERVER.H,ST.H Listings aus "Dynamischer Datenaustausch unter dem Presentation Manager", S. 6

\DEBUG DEBUGSRC.ASM Listings aus "Die Debugregister des Intel 386", S. 178

\DISKS FREE.C, LIST1.C, LIST2.C, LIST3.C, LIST4.C, LIST6.C Listings aus "Diskettenstruktur unter DOS", S. 150

\DUMP DUMP.C, FILE.C, PANE.C, SPLITH.CUR, SPLITV.CUR, SPLITX.CUR, DUMP.DFD, DUMP.DFP, DUMP.DLG, DEFS.H, DUMP.H, DUMP.ICO, DUMP.MD, DUMP.MP, DUMP.RC Listings aus "Eine Hex-Dump-Applikation für Windows", S. 35

\OVER OV, OVASM, SMERGE, OV.ASM, OVASM.ASM, OVRMGR.ASM, P1ASM.ASM, P2ASM.ASM, RDOVLY.ASM, SUPPORT.ASM, OV.C, P1.C, P2.C, RDOVLY.C, SMERGE.C, EXEHDR.H Listings aus "Ein Overlay-Manager für DOS", S. 161

\QP DISKPARA.PAS Listing aus "Systemnahe Programmierung mit QuickPascal", S. 80

\SAA6 DLGDEMO.C, DLGPB.C, DLGRB.C, DLG.H Listings aus "Noch mehr Objekte für Ihre Dialogboxen", S. 115

\THREADS HELLO0, HELLO0.C Listings aus "Multithread-Programme unter OS/2", S. 19

TGM-DSK-178: Microsoft System Journal Ausgabe Jan./Feb. 1990, "MSJDSK41"

VIO.LZH "Ansprechende Benutzeroberflächen unter OS/2", S. 17.

BIT.LZH "Drehen von Zeichnungen und Rasterbildern", S. 47.

CAL.LZH "Datumsarithmetik in QuickBasic", S. 101.

SAA.LZH "Weitere Dialogobjekte und ein Blick zurück", S. 107 (enthält alle bisher veröffentlichten Module).

IMR.LZH "Kombinieren von Word-Indexdateien", S. 125.

CKR.LZH "Programmieren in C: Rund um Dateioperationen", S. 133.

TGM-DSK-179: Microsoft System Journal Ausgabe März/April 1990 "MSJDSK42"

JIGSAW.LZH "Schnellere Grafik durch Offscreen-Methoden", S. 6

KM.LZH "Die Schlüsselfunktionen der Tastatur- und Maus-Subsysteme", S. 16

BAS.LZH "So nutzen Sie OS/2 von Basic aus", S. 30

FINDER.LZH "Gewußt wo...", S. 40

DIA.LZH "Neue Kontrollklassen für Windows-Dialogboxen", S. 59

SAA8.LZH "Interdependente Dialogobjekte", S. 104

PTR.LZH "Einfachere Zeigersyntax für zuverlässigere Programme", S. 120

CKURS.LZH "C-Kurs: Funktionen und dynamische Speicherverwaltung", S. 131

QA.LZH "Assembler-Programmierung in einer integrierten Umgebung", S. 148

EXEC.LZH "Alles über die EXEC-Funktion von MS-DOS", S. 153

TGM-DSK-180: Microsoft System Journal Ausgabe Mai/Juni 1990 "MSJDSK43"

WINMEM.LZH "Erweiterte Speicherverwaltung für Windows, Teil II", S. 22

SPYGLASS.LZH "SpyGlass, eine Lupe für Windows", S. 42

PMBASIC.LZH "Presentation Manager-Interface für Basic-Programme", S. 53

SAA9.LZH "Eine Dateiauswahl-Box", S. 100

CKURS.LZH "Programmieren in C", S. 110

SCRSAVE.LZH "Der Einsatz von Extended Memory", S. 125

HIMEM.LZH Weitere Informationen über HIMEM und XMS mit Source und Doc

SQLPOP.LZH "SQL-Server-Anwendungen mit der DB-Bibliothek entwickeln", S. 140

COBOL.LZH "Cobol, der aktuell Oldie", S. 156

TGM-DSK-181: Microsoft System Journal Ausgabe Juli/August 1990 "MSJDSK44"

DAME2.LZH "Die Programmierung des Dame-Spielbretts"

EMS.LZH "Der Umgang mit Expanded Memory"

MAC2WIN.LZH "So portiert man Mac-Programme nach Windows"

SLEUTH.LZH "Gemeinsame Quelltexte für PM, Windows und Macintosh"

SAA10.LZH "Alles geregelt" (SAA-Serie)

CAPT.LZH "Bildschirm-Fotos unter Windows 3"

TGM-DSK-182: Microsoft System Journal Ausgabe Sept./Okt. 1990 "MSJDSK45"

CHECKER3.LZH Listings zum Artikel "Checkers: Die Bewegung der Steine"

FMT.LZH Listings zum Artikel "Formatierte Eingabefelder für den Presentation Manager"

HELP.LZH Listings zum Artikel "Die Hilfe-Verwaltung von Windows 3"

IPC.LZH Listings zum Artikel "Interprozeßkommunikation und Gerätemonitore"

NLS.LZH Listings zum Artikel "Programmierung internationaler Programme"

QC25.LZH Listing zum Artikel "QuickC/Quick-Assembler 2.5 und Based-Pointer"

SAA11.LZH Listings zum Artikel "Ein SAA-Texteditor"

TRACER.LZH Listings zum Artikel "Tracer: Ein Debug-Werkzeug für den PM"

TGM-DSK-183: Microsoft System Journal Ausgabe Nov./Dez. 1990 "MSJDSK46"

AUFDLL.LZH Listings zu "Aufbau und Einsatz von DLLs unter OS/2"

ENTWDLL.LZH Listings zu "Entwicklung von OS/2 DLL-Bibliotheken"

PRTSC.LZH Listings zu "DLL unter Windows am Beispiel..."

CLIPDLL.LZH Listings zu "Erweiterte Zwischenablage mit DLLs"

DIPOP.LZH Listings zu "Der Einsatz von Subsystemen und IPC unter OS/2"

PMACCESS.LZH Listings zu "PM-Hilfsmittel für Kernel-Programme"

RANDOM.LZH Listings zu "Grafische Segmente im Presentation Manager"

OS2DEV.LZH Listings zu "Gerätetreiber unter OS/2"

GLF.LZH Listings zu "Der Golem Baumeditor"

EMS.LZH Listings zu "Der Umgang mit Expanded Memory"

TGM-DSK-184: Microsoft System Journal Ausgabe Jan./Feb. 1991 "MSJDSK51"

PAGE.LZH Listings zu "Erweitern Sie Windows durch installierbare Kontrollelemente"

MDI.LZH Listings zu "Das neue Windows-MDI-API"

IMON.LZH Listings zu "Ein Interrupt-Monitor unter Windows"

ISAM.LZH Listings zu "Windows-Entwicklung durch Client-Server-Architektur vereinfachen"

GLF2.LZH Listings zu "Der Golem-Baumeditor (II)"

TGM-DSK-185: Microsoft System Journal Ausgabe März/April 1991 "MSJDSK52"

FONTGEN.LZH Listings zu "Softfonts für LaserJets und Kompatible", S. 26

FORM.LZH Listings zu "Dateneingabemasken effektiv programmieren", S. 96

STOCK.LZH Listings zu "Ressourcen und Menüs", S. 107

POLY.LZH Listings zu "Unterfenster und metrische Mapping-Modi", S. 123

CHECKERS.LZH Listings zu "Die Spielstrategie des Dame-Spiels", S. 71

BASIC71.LZH Listings zu "Microsoft Basic 7.1 - Des Kaisers...", S. 133

GOLEM.LZH Listings zu "Ein objektorientiertes Wörterbuch", S. 149

PARCOM.LZH Listings zu "Bidirektionale Kommunikation parallel", S. 141

TGM-DSK-186: Microsoft System Journal Ausgabe Mai/Juni 1991 "MSJDSK53"

MACDLL.LZH Listings zu "Erweitern Sie Standardprogramme...", S. 37

FORMS.LZH Listings zu "Dateneingabemasken effektiv programmieren", S. 46

POLY.LZH Listings zu "Schriften und WYSIWYG-Darstellung", S. 74

WINPROG3.LZH Listings zu "Steuerfenster und MDI-Unterstützung", S. 105

SPELL1.LZH Listings zu "Die effektive Speicherung von Wörterlisten", S. 141

PSBUCH.LZH Listings zu "PostScript, Werkzeuge und Techniken", S. 153

TGM-DSK-187: Microsoft System Journal Ausgabe Juli/August 1991 "MSJDSK54"

VB.LZH Listings zu "Visual Basic, die Standard-Programmiersprache für Windows", S. 6

W32.LZH Listings zu "32-Bit-Programme unter Windows", S. 16

VMON.LZH Listings zu "So programmiert man virtuelle Geräte für Windows 3.0", S. 25

WMEM2.LZH Listings zu "Die Speicherverwaltung von Windows 3.0", S. 39

STOCK.LZH Listings zu "Kontrollelemente und Dialogfenster", S. 61

POLY3.LZH Listings zu "Grafikexport mit Windows-Metafiles", S. 83

OPT.LZH Listings zu "Analyse der Optimierung durch C-Compiler", S. 111

BASED.LZH Listings zu "Basiszeiger, die Vorteile von Far- und Near-Zeigern in einem", S. 130

SPELL2.LZH Listings zu "Die Suche in komprimierten Wörterlisten", S. 143

WOCBASE.LZH Demo-Programm zu "Programmier-Know-how nachschlagen mit WOCBase", S. 157

TGM-DSK-188: Microsoft System Journal Ausgabe September/Oktober 1991 "MSJDSK55"

KERMIT.LZH Dateien zu "Kermit für Windows", S. 35

POSTMAN.LZH Dateien zu "Der Postmann bringt Licht ins Dunkel", S. 45

GDI.LZH Dateien zu "Das GDI, die Grafikschnittstelle von Windows", S. 60

MATH.LZH Dateien zu "Ein mathematischer Parser als DLL für Windows", S. 100

TEXT.LZH Dateien zu "Exakte Textdarstellung unter Windows", S. 114

EXEC30.LZH Dateien zu "Kindprogramme ohne Speicheraufwand starten", S. 126

VIO.LZH Dateien zu "Aufbau einer geräteunabhängigen Video-I/O-Bibliothek in Microsoft C", S. 141

PDQ.LZH Dateien zu "P.D.Q.", S. 152

TGM-DSK-189: µPROFI-51, FSD51, MON_51, V2.0

```
FSD51          <DIR>      26.06.90   21:31
MONITOR        <DIR>      26.06.90   21:31
README              2968  28.06.90    7:42
PROFI    LZH       89743  02.09.91   21:41
\FDS51
FSD51    EXE       96320  26.06.90   21:28
\MONITOR
INSTALL  BAT         651  09.02.90   10:45
HARDWARE LST        5091  26.06.90   21:41
HARDWARE OBJ         435  26.06.90   21:41
HARDWARE SRC        1952  26.06.90   21:35
MONITOR  LST       85316  26.06.90   21:41
MONITOR  OBJ        4018  26.06.90   21:41
MONITOR  SRC       46715  09.02.90   10:57
MON_51   BIN        3846  26.06.90   21:41
MON_51   HEX        5653  26.06.90   21:41
MON_51   M51        6390  26.06.90   21:41
```

TGM-DSK-189: INLASS-Turbo-Assembler, modifiziert für 8052

```
BERECHNE INC        7975  08.04.86   15:24
CENT52   COM       13053  19.12.86   13:47
CENT52   PAS        3646  19.12.86   13:47
DISAS52  COM       27912  19.12.86    3:11
DISAS52  PAS        5578  19.12.86    3:10
DISASSEM INC       12532  23.01.87   21:22
DOPASS   INC       12532  11.11.86    0:34
EMU8031  ASM       45952  03.09.86   22:15
ERROR    INC        1280  08.04.86   15:24
HL1-Z80  INC       10752  08.04.86   15:23
HL2-6502 INC        6656  08.04.86   15:24
HL3-8080 INC        6528  08.04.86   15:23
HL4A8086 INC       12640  08.04.86   15:24
HL4B8086 INC       15749  08.04.86   15:23
HL5-8052 INC        8972  11.11.86    1:25
INL      INC        3699  11.11.86   14:24
INLASS   000       36096  11.11.86    0:35
INLASS   COM       33896  11.11.86    0:36
INLASS   PAS        2924  11.11.86   14:35
INLOAD   ORI        3229  08.04.86   15:25
INLOAD52 COM       12962  11.11.86   14:58
INLOAD52 PAS        3370  11.11.86   14:58
PARAMSTR BIB        2688  27.03.85   11:19
READ     ME         2560  10.04.86   16:42
SCANLINE INC        5196  08.04.86   15:25
SUCHBAUM INC        6144  08.04.86   15:24
SWAP     INC          85  19.08.86   13:21
SWAP     INL        1513  19.08.86   13:38
SWAP     PRN        1449  19.08.86   13:38
SWAP     SOU         579  19.08.86   13:37
DOPASS   ORI       12681  08.04.86   15:24
INL      ORI        3712  08.04.86   15:24
INLASS   ORI        2944  08.04.86   15:25
```

Verzeichnisse tabellarisch

PC-NEWS

Bestellnummer	Text auf TGM-DSK-xxx	Seitenzahl	Beschriebene TGM-Disketten, TGM-Literatur, TGM-Tabellen
PC-NEWS-01	001	40	DSK-1-9
PC-NEWS-02	010	50	DSK-10,11
PC-NEWS-03	024	90	DSK-12-26
PC-NEWS-04	027	72	DSK-27-29
PC-NEWS-05	030	53	DSK-30-46
PC-NEWS-06	047	52	DSK-47-51
PC-NEWS-07	052	64	DSK-52-57
PC-NEWS-08	058	41	DSK-58-64
PC-NEWS-09	065	61	DSK-65-77
PC-NEWS-10	078	56	DSK-78-79
PC-NEWS-11	095	44	DSK-80-95
PC-NEWS-12	096	30	DSK-96
PC-NEWS-13	096	30	DSK-97
PC-NEWS-14	098	70	DSK-98-101
PC-NEWS-15	134	43	DSK-102-134
PC-NEWS-16	135	60	DSK-135
PC-NEWS-17	BTX		
PC-NEWS-18	136	56	DSK-136-139
PC-NEWS-19	141	44	DSK-140-141
PC-NEWS-20	151	66	DSK-142-149,151
PC-NEWS-21	150	44	DSK-150
PC-NEWS-22	152	86	DSK-152-157
PC-NEWS-23	158	96	DSK-158-168; LIT-001,002,004,005,007,009,010; TAB-001,002
PC-NEWS-24	169		DSK-169-190; LIT-003,006,011-021; TAB-003,004,007-011

Tabellen

Bestellnummer	Beschrieben in PC-NEWS	Sprache(d=deutsch, e=englisch)	Blätter	Text auf TGM-DSK-xxx	
TGM-TAB-001	23	d	1		PC-Zeichensatz PC-8
TGM-TAB-002	23	e	1	169	MSDOS-5.0e
TGM-TAB-003	24	d	1	169	WINDOWS-3.0d
TGM-TAB-004	24	d	1	169	WORD 5.0d
TGM-TAB-005	24	d	1		ROMAN-8,PC-8,ANSI-100,SYMBOL
TGM-TAB-006	24	e	1		COREL-DRAW
TGM-TAB-007	24	d	1	169	POST-SCRIPT-ZEICHENSATZ
TGM-TAB-008	24	d	1	169	MSDOS-5.0d
TGM-TAB-009	24	d	2	169	WORD-5.5d Tastaturbelegung (deutsche Tastenbezeichnungen)
TGM-TAB-010	24	d	2	169	WORD-5.5d Tastaturbelegung (englische Tastenbezeichnungen)
TGM-TAB-011	24	d	1	169	BORLAND-C-2.0d

Literatur

Bestellnummer	Beschrieben in PC-NEWS	Sprache	Seiten	Zugehörige Diskette TGM-DSK-xxx	Titel
TGM-LIT-001	23	d	17	153	WOOD-WORKS-Kurzdokumentation
TGM-LIT-002	23	d	22		NEAT-Chipsätze
TGM-LIT-003	24	d	92		8051-Grundlagen, Hard- und Software
TGM-LIT-004	23	e	155	140	Referenz-Handbuch IBM-PC
TGM-LIT-005	23	e	8	159	COSMIC-Software-Katalog
TGM-LIT-006	24	e	77		Mikro-Controller 83C552
TGM-LIT-007	23	d	54	164 165	Super-Base 4 unter Windows, Demo
TGM-LIT-008	25	d	58	166	FTKL-Progr.Drucksensor mit 80C522
TGM-LIT-009	23	d	41		Tabellen aus dem TURBO-C++-Handbuch
TGM-LIT-010	23	d	20	168	Lexikon der EDV, Bedienung
TGM-LIT-011	24	e	96	170 171 172	OOPS
TGM-LIT-012	24	d	47	139	PSPICE-Kurzdokumentation
TGM-LIT-013	24	d	27	175	Erweiterung zum µPROFI-51
TGM-LIT-014	24	e	40		Verzeichnis der EMS-Software-Bibliothek
TGM-LIT-015	24	d	5		µPROFI-51-Bauanleitung
TGM-LIT-016	24	d	35	169	INLASS, Assembler(TURBO)
TGM-LIT-017	24	e	104		MCS BASIC-52 Users Manual
TGM-LIT-018	24	e	5		Datenblatt 8052AH-BASIC
TGM-LIT-019	24	d	138	169	Konstruktionsunterlagen zum A3-Flachbett-Plotter
TGM-LIT-020	24	e	16		Datenblatt zu SAB 80515/80535-Microcontroller
TGM-LIT-021	24	d	30	189	µPROFI-Handbuch

Disketten

Bestellnummer	Beschrieben in PC-NEWS	Diskettentype (1=360k 2=1,2MB 3=720k 4=1,44MB)	zugehörige Dokumentation (TGM-LIT)	Titel
TGM-DSK-001	01	1		PCNEWS-1
TGM-DSK-002	01	1		Basic-Programme für Unterrichtszwecke
TGM-DSK-003	01	1		Basic-Programme für Unterrichtszwecke
TGM-DSK-004	01	1		Datenbanksystem (BASIC)
TGM-DSK-005	01	1		Datenübertragungssoftware (in C)
TGM-DSK-006	01	1		Kermit-1
TGM-DSK-007	01	1		Kermit-2
TGM-DSK-008	01	1		Anpassung an ASCII-Drucker
TGM-DSK-009	01	1		PC-SIG Nummer 1-200 Programmverzeichnis
TGM-DSK-010	02	1		PCNEWS-2
TGM-DSK-011	02	1		Basic-Programme: Eliza, Kurvendiskussion
TGM-DSK-012	03	1		PC-SIG Nummer 201-400 Programmverzeichnis
TGM-DSK-013	03	1		PC-SIG Nummer 393-514 Programmverzeichnis
TGM-DSK-014	03	1		PC-BLUE Nummer 1-200 Programmverzeichnis
TGM-DSK-015	03	1		Chi-Writer
TGM-DSK-016	03	1		Z80 und CP/M 2.2 Emulator unter PC-DOS mit Dokumentation
TGM-DSK-017	03	1		PC-Talk III + Dokumentation
TGM-DSK-018	03	1		FIDO Bulletin Board System v10.0 Nr.1(2)
TGM-DSK-019	03	1		FIDO Bulletin Board System v10.0 Nr.2(2)
TGM-DSK-020	03	1		Remote Bulletin Board System v12.5A Nr.1(2)
TGM-DSK-021	03	1		Remote Bulletin Board System v12.5A Nr.2(2)
TGM-DSK-022	03	1		RBBS-PC v12.5b Nr.1(2)
TGM-DSK-023	03	1		RBBS-PC v12.5b Nr.2(2) documentation/files/utilities
TGM-DSK-024	03	1		PC-NEWS-3 Texte
TGM-DSK-025	03	1		PC-NEWS-3 Programme-1
TGM-DSK-026	03	1		PC-NEWS-3 Programme-2
TGM-DSK-027	04	1		PC-NEWS-4 Texte
TGM-DSK-028	04	1		PC-NEWS-4 Texte,Programme
TGM-DSK-029	04	1		PC-NEWS-4 Programme
TGM-DSK-030	05	1		PC-NEWS-1/87 : TEXTE
TGM-DSK-031	05	1		PC-NEWS-1/87 : FASTBOX
TGM-DSK-032	05	1		PC-NEWS-1/87 : PROGRAMME
TGM-DSK-033	05	1		PC-NEWS-1/87 : MANDELBROT
TGM-DSK-034	05	1		KERMIT - TURBO
TGM-DSK-035	05	1		KERMIT - C64 1(3)
TGM-DSK-036	05	1		KERMIT - C64 2(3)
TGM-DSK-037	05	1		KERMIT - C64 2(3)
TGM-DSK-038	05	1		DISKETTENVERZEICHNISSE
TGM-DSK-039	05	1		KERMIT - CPM-80
TGM-DSK-040	05	1		KERMIT - CPM-86
TGM-DSK-041	05	1		KERMIT - UNIX-C
TGM-DSK-042	05	1		KERMIT -
TGM-DSK-043	05	1		KERMIT - Apple-DOS
TGM-DSK-044	05	1		KERMIT - Apple-DOS
TGM-DSK-045	05	1		KERMIT -
TGM-DSK-046	05	1		KERMIT -
TGM-DSK-047	06	1		PC-NEWS-2/87 : TEXTE
TGM-DSK-048	06	1		PC-NEWS-2/87 : PROGRAMME
TGM-DSK-049	06	1		GENEALOGY
TGM-DSK-050	06	1		LOG-ON/OFF
TGM-DSK-051	06	1		QDMG/PLOT/GRAPHICS
TGM-DSK-052	07	1		PC-NEWS 87/3
TGM-DSK-053	07	1		Sprüche aus den PC-NEWS 2/86-3/87
TGM-DSK-054	07	1		TURBO-PASCAL-Utilities
TGM-DSK-055	07	1		Clubmappe
TGM-DSK-056	07	1		Personal-Informations-System
TGM-DSK-057	07	1		Utilities selbst gemacht, Krebs
TGM-DSK-058	08	1		PC-NEWS 4/87, Lfd.Nr.: 8 : Texte
TGM-DSK-059	08	1		PC-NEWS 4/87, Lfd.Nr.: 8 : Programme
TGM-DSK-060	08	1		PC-NEWS 4/87, Lfd.Nr.: 8 : Beitrag TEACH-ME
TGM-DSK-061	08	1		Druckertreiber NEC
TGM-DSK-062	08	1		Druckertreiber NEC
TGM-DSK-063	08	1		Druckertreiber NEC
TGM-DSK-064	08	1		Druckertreiber NEC, STAR, PANASONIC
TGM-DSK-065	09	1		PC-NEWS 1/88, Lfd.Nr.: 9 Texte
TGM-DSK-066	09	1		PC-NEWS 1/88, Lfd.Nr.: 9 Programme
TGM-DSK-067	09	1		Kommunikationsprogramm, Archivierungsprogramm
TGM-DSK-068	09	1		PROCOMM : Terminalprogramm
TGM-DSK-069	09	1		OPUS 1(4)
TGM-DSK-070	09	1		OPUS 2(4)
TGM-DSK-071	09	1		OPUS 3(4)
TGM-DSK-072	09	1		OPUS 4(4)
TGM-DSK-073	09	1		PROCOMM-PLUS
TGM-DSK-074	09	1		SEQUENTIAL-PASCAL-COMPILER (SOURCE-CODE) 1(2)
TGM-DSK-074	09	1		SEQUENTIAL-PASCAL-COMPILER 2(2), PASCAL-INTERPRETER (PLM), PASCAL-COMPILER IN BASIC

Disketten

Bestellnummer	Beschrieben in PC-NEWS	Diskettentype (1=360k 2=1,2MB 3=720k 4=1,44MB)	zugehörige Dokumentation (TGM-LIT)	Titel
TGM-DSK-076	09	1		C-TREE, Datenbank in C (Source)
TGM-DSK-077	09	1		POWER-MOS (PHILIPS)
TGM-DSK-078	10	1		PC-NEWS 2/88, Lfd.Nr.10 : Texte
TGM-DSK-079	10	1		PC-NEWS 2/88, Lfd.Nr.10 Programme
TGM-DSK-080	11	1		Binkley 1(4)
TGM-DSK-081	11	1		Binkley 2(4)
TGM-DSK-082	11	1		Binkley 3(4)
TGM-DSK-083	11	1		Binkley 4(4)
TGM-DSK-084	11	1		Literatursuchdienst-Übungsdiskette
TGM-DSK-085	11	1		Dokumentation des FIDO-Mailboxsystems
TGM-DSK-086	11	1		Demo- und Lernprogramme Elektronik 1-4
TGM-DSK-087	11	1		RW-File, Dateiprogramm
TGM-DSK-088	11	1		STN-Overview
TGM-DSK-089	11	1		Calendar Maker
TGM-DSK-090	11	1		PD 1: S1-511
TGM-DSK-091	11	1		PD 2: S501-711, B1-411
TGM-DSK-092	11	1		PD 3: S701-1110, B411-440
TGM-DSK-093	11	1		PD 4: E1-119, X1-153
TGM-DSK-094	11	1		PD 5: TURBOSIX, TURBO-UG, CAPITAL
TGM-DSK-095	11	1		PC-NEWS 3/88, Lfd.Nr.11 : Texte
TGM-DSK-096	12	1		PC-NEWS 12 und 13, 4/88 und 1/89 Texte
TGM-DSK-097	13	1		PC-NEWS 12 und 13, 4/88 und 1/89 Programme
TGM-DSK-098	14	1		PC-NEWS 2/89 Texte/Programme
TGM-DSK-099	14	1		EPSON-Druckertreiber-1
TGM-DSK-100	14	1		EPSON-Druckertreiber-2
TGM-DSK-101	14	1		EPSON-Hardcopy
TGM-DSK-102	15	1		PD-Verzeichnis PC-BLUE
TGM-DSK-103	15	1		PD-Verzeichnis PCSIG
TGM-DSK-104	15	1		PD-Verzeichnis PCSIG
TGM-DSK-105	15	1		PD-Verzeichnis PCSIG
TGM-DSK-106	15	1		PD-Verzeichnis PCSIG
TGM-DSK-107	15	1		PD-Verzeichnis E, X
TGM-DSK-108	15	1		Utilities
TGM-DSK-109	15	1		Utilities
TGM-DSK-110	15	1		Utilities
TGM-DSK-111	15	1		Utilities
TGM-DSK-112	15	1		Utilities
TGM-DSK-113	15	1		Utilities
TGM-DSK-114	15	1		Utilities
TGM-DSK-115	15	1		Demoprogramme Elektronik
TGM-DSK-116	15	1		Demoprogramme Elektronik
TGM-DSK-117	15	1		Demoprogramme Elektronik
TGM-DSK-118	15	1		Demoprogramme Elektronik
TGM-DSK-119	15	1		DOS-POWER-TOOLS PC-MAGAZINE
TGM-DSK-120	15	1		Chemie 15 Kernspinresonanz NMR
TGM-DSK-121	15	1		Chemie 15 HPLC
TGM-DSK-122	15	1		STATISTIK in TURBO-PASCAL
TGM-DSK-123	15	1		CT
TGM-DSK-124	15	1		CT
TGM-DSK-125	15	1		ROBOTER
TGM-DSK-126	15	1		PC-DESKMATES
TGM-DSK-127	15	1		Utilities
TGM-DSK-128	15	1		Utilities
TGM-DSK-129	15	1		Assembler-Utilities
TGM-DSK-130	15	1		BASIC-Tools
TGM-DSK-131	15	1		Utilities
TGM-DSK-132	15	1		Utilities
TGM-DSK-133	15	1		Utilities
TGM-DSK-134	15	1		PC-NEWS 15, 3/89
TGM-DSK-135	16	1		PC-NEWS 16, 4/89
TGM-DSK-136	18	1		PC-NEWS 18, 2/90
TGM-DSK-137	18	1		Lohnsteuerberechnungsprogramm
TGM-DSK-138	18	2		DEMO-DISK, LAB-WINDOWS, National Instruments
TGM-DSK-139	18	2	012	PSPICE (TGM-LIT-12)
TGM-DSK-140	19	2	004	PC-XT/AT-Technisches Handbuch, englisch, TGM-LIT-004
TGM-DSK-141	19	1		PC-NEWS 19, Texte, Programme
TGM-DSK-142	20	2		BTX-Decoder auf HD-Diskette, entpacken bei Aufruf selbständig
TGM-DSK-143	20	1		DECODIX 1.4
TGM-DSK-144	20	1		Schnuki
TGM-DSK-145	20	1		XBTX
TGM-DSK-146	20	1		JANUS-BTX
TGM-DSK-147	20	1		AMARIS-Demo-Version
TGM-DSK-148	20	1		AMARIS lauffähige Version, verringerter Leistungsumfang
TGM-DSK-149	20	1		Festplatten, VGA-Test, Spice
TGM-DSK-150	21	1		NEWS21 Texte+Programme
TGM-DSK-151	20	1		NEWS20 Texte+Programme
TGM-DSK-152	22	1		NEWS22, Texte, Programme
TGM-DSK-153	22	2	001	WOOD-WORKS - Demoversion, TGM-LIT-001
TGM-DSK-154	22	1		ORCAD Schematic Design Tools, SDT III, DEMO-Disk
TGM-DSK-155	22	1		ORCAD Printed Circuit Board Layout Tools, PCB II, DEMO-Disk
TGM-DSK-156	22	1		ORCAD Programmable Logic Design Tools, PLD, DEMO-Disk
TGM-DSK-157	22	1		ORCAD Verification & Simulation Tools, VST, DEMO-Disk
TGM-DSK-158	23	3		NEWS-23, Texte, Programme
TGM-DSK-159	23	4	005	Programmverzeichnis COSMIC, TGM-LIT-005
TGM-DSK-160	23	2		TextMaker 3.0, DataMaker 1.0 Demoversion
TGM-DSK-161	23	1		PC-Professionell 4/91 Utilities
TGM-DSK-162	23	1		PC-Professionell 5/91 Utilities
TGM-DSK-163	23	1		Materialien für den C-Unterricht
TGM-DSK-164	23	3	007	Superbase 4, Teil1, TGM-LIT-007
TGM-DSK-165	23	3	007	Superbase 4, Teil2, TGM-LIT-007
TGM-DSK-166	23	2	008	Drucksensor für den Robotergreifarm, TGM-LIT-008
TGM-DSK-167	23	1		Depotmaster
TGM-DSK-168	23	1	010	Lexikon der EDV, TGM-LIT-010
TGM-DSK-169	24	2		NEWS-24, Texte, Programme
TGM-DSK-170	24	1	011	OOPS 2.2 1(3), TGM-LIT-011
TGM-DSK-171	24	1	011	OOPS 2.2 2(3), TGM-LIT-011
TGM-DSK-172	24	1	011	OOPS 2.2 3(3), TGM-LIT-011
TGM-DSK-173	24	1		PC-Professionell-Utilities, Teil-IV
TGM-DSK-174	24	1		Texte für den Unterricht, EDV, Elektronik
TGM-DSK-175	24	4		FTKL-Programm Hardwareerweiterung zum µPROFI-51
TGM-DSK-176	24	1		Microsoft System Journal Ausgabe Sep./Okt. 1989, "MSJDSK35"
TGM-DSK-177	24	1		Microsoft System Journal Ausgabe Nov./Dez. 1989, "MSJDSK36"
TGM-DSK-178	24	1		Microsoft System Journal Ausgabe Jan./Feb. 1990, "MSJDSK41"
TGM-DSK-179	24	1		Microsoft System Journal Ausgabe März/April 1990 "MSJDSK42"
TGM-DSK-180	24	1		Microsoft System Journal Ausgabe Mai/Juni 1990 "MSJDSK43"
TGM-DSK-181	24	1		Microsoft System Journal Ausgabe Juli/August 1990 "MSJDSK44"
TGM-DSK-182	24	1		Microsoft System Journal Ausgabe Sept./Okt. 1990 "MSJDSK45"
TGM-DSK-183	24	1		Microsoft System Journal Ausgabe Nov./Dez. 1990 "MSJDSK46"
TGM-DSK-184	24	1		Microsoft System Journal Ausgabe Jan./Feb. 1991 "MSJDSK51"
TGM-DSK-185	24	1		Microsoft System Journal Ausgabe März/April 1991 "MSJDSK52"
TGM-DSK-186	24	1		Microsoft System Journal Ausgabe Mai/Juni 1991 "MSJDSK53"
TGM-DSK-187	24	1		Microsoft System Journal Ausgabe Juli/August 1991 "MSJDSK54"
TGM-DSK-188	24	1		Microsoft System Journal Ausgabe September/Oktober 1991 "MSJDSK55"
TGM-DSK-189	24	1	021	µPROFI-51, FSD51, MON_51, V2.0, Text-Handbuch
TGM-DSK-190	24	1	016	INLASS-Turbo-Assembler f.versch.Prozessoren

Preise, Stand September 1991

Literatur	ÖS	-,80	/Seite
Tabellen A5	ÖS	15,-	/Blatt
Tabellen A4	ÖS	25,-	/Blatt
Disketten 5-1/4",360k	ÖS	40,-	/Disk
Disketten 5-1/4",1.2M	ÖS	50,-	/Disk
Disketten 3-1/2",720k	ÖS	50,-	/Disk
Disketten 3-1/2",1.44M	ÖS	60,-	/Disk

Je nach Verpackungsart tritt zum **Gesamtpreis** ein Verpackungskostenanteil von S 20,- oder S 26,- hinzu.

Sonderpreisliste nds für PCCTGM

nds, Nowotny Datensysteme (früher DATACRAFT), Märzstr. 116, 1150 Wien Tel. 98-21-005, Fax. 98-25-033.
Wir bitten um Vorbestellung per Fax/Tel. PREISE INCL. MWST.

KOMPLETTGERÄTE

NDS-286 H 40 . . . . . ÖS 9.240,-
CPU 80286, Taktfrequenz 12 Mhz, 1 Mbyte Ram,erweiterbar on Board auf 4 Mbyte, 1 Laufwerk 5.25 / 1.2 Mb, IDE-40 Mbyte Festplatte mit 28 ms Zugriffszeit, 2 serielle, 1 parallele Schnittstelle, Herculesgrafikkarte, Desktop Gehäuse.

NDS-286 V 40 . . . . . ÖS 9.960,-
wie vorhergehender, jedoch mit VGA-Grafikkarte 800x600

NDS-386SX/16 V 40 . . . . . ÖS 14.160,-
CPU 80386SX, Taktfrequenz 16 Mhz,2 Mbyte Ram, erweiterbar on Board auf 8 Mbyte, 1 Laufwerk 5.25 / 1.2 Mb,IDE-40 Mbyte Festplatte mit 28 ms Zugriffszeit, 2 serielle, 1 parallele Schnittstelle, VGA-Grafikkarte 800x600/256kB, Baby Tower Gehäuse

DS-386/25/C32 V 40 . . . . . ÖS 21.600,-
CPU 80386, Taktfrequenz 25 Mhz, 32 kB Cache, 4 Mbyte Ram, erweiterbar on Board auf 8 Mbyte, 1 Laufwerk 5.25 / 1.2 Mb, IDE-40 Mbyte Festplatte mit 28 ms Zugriffszeit, 2 serielle, 1 parallele Schnittstelle, VGA-Grafikkarte 1024x768 / 512 k, Tower Gehäuse

NDS-386/33/C64 V . . . . . ÖS 23.400,-
CPU 80386, Taktfrequenz 33 Mhz, 64 k Cache-Speicher, sonst wie vorhergehender

NDS-486/25/C128 V 40 . . . . . ÖS 33.000,-
CPU , Taktfrequenz 25 MHz, 128 k Cache-Speicher, sonst wie vorhergehender.

NDS-486/33/C128 V 40 . . . . . ÖS 40.920,-
CPU 80486, Taktfrequenz 33 MHz, 128 k Cache-Speicher, sonst wie vorhergehender.

Aufpreise:
2. Laufwerk 3.5 Zoll / 1.44 Mbyte . . . . . ÖS 948,-
124 Mbyte Festplatte . . . . . ÖS 3.492,-
210 Mbyte Festplatte . . . . . ÖS 7.800,-

MONITORE

Hercules Monochrom 14" DTK . . . . . ÖS 1.524,-
VGA - Mono 640x480 14" DTK . . . . . ÖS 1.788,-
VGA - Color 1024x768, 0.28 dot 14" DTK . . . ÖS 5.016,-
VGA - Color 1024x768, 0.26 dot 14" TAXAN . ÖS 10.428,-
non Interlaced, Sony Black Trinitron
VGA - Color 1280x1024, 0.31 dot 17" Sampo . . ÖS 15.000,-
VGA - Color 1280x1024, 0.31 dot 20" Sampo . . ÖS 26.388,-
strahlungsarme Schirme nach schwedischer Norm auf Anfrage

COPROZESSOREN

80287 / 12 Mhz . . . . . ÖS 1.224,-
80387SX / 16 Mhz . . . . . ÖS 3.146,-
80387SX / 20 Mhz . . . . . ÖS 3.420,-
80387 / 16 Mhz . . . . . ÖS 3.516,-
80387 / 20 Mhz . . . . . ÖS 3.948,-
80387 / 25 Mhz . . . . . ÖS 5.100,-
80387 / 33 Mhz . . . . . ÖS 5.544,-

LAUFWERKE & FESTPLATTEN

IDE - Seagate ST-157A . . . . . ÖS 3.108,-
40 MB, 28 ms, 3.5 Zoll

IDE - Seagate ST-1144A . . . . . ÖS 6.600,-
124 MB, 19 ms, 3.5 Zoll

IDE - Seagate ST-1239A . . . . . ÖS 10.908,-
210 MB, 19 ms, 3.5 Zoll

IDE - NEC D3755 . . . . . ÖS 6.240,-
100 MB, 25 ms, 3.5 Zoll. Halbhohe Bauweise, ideal für Laptops

Epson/NEC 5.25 Zoll / 1.2 Mbyte . . . . . ÖS 990,-
Citizen/Sony 3.5 Zoll / 1.44 Mbyte . . . . . ÖS 924,-
Einbaurahmen für 3.5 Zoll . . . . . ÖS 190,-

GRAFIKKARTEN

Hercules . . . . . ÖS 237,-
VGA OAK 16 bit 800x600 256k . . . . . ÖS 960,-
VGA Paradise 1024x768 512k . . . . . ÖS 1.668,-
VGA Tseng ET3000 1024x768 512k . . . . . ÖS 1.788,-
VGA Tseng ET4000 1024x768 1MB . . . . . ÖS 2.490,-
70/72 Hz Option für ET 4000 . . . . . ÖS 228,-
Programmers Reference Manual für ET 3000 . . ÖS 588,-

GEHÄUSE

Baby Tischgehäuse 3 Einschübe 5.25 Zoll . . . ÖS 1.224,-
Baby Tower Gehäuse 2 / 5.25 Zoll 3 / 3.5 Zoll . ÖS 1.716,-
Tower Gehäuse 6 Einschübe 5.25 Zoll . . . . . ÖS 2.364,-
mit Mhz-Display

Slim Line Mini Gehäuse . . . . . ÖS 2.280,-
max. 4 Slots, 1 x 5.25 Zoll, 2 x 3.5 Zoll Slim

MOTHERBOARDS
XT / 12 Mhz . . . . . ÖS 948,-
AT 286 / 12 Mhz . . . . . ÖS 1.584,-
AT 286 / 16 Mhz . . . . . ÖS 1.788,-
386 SX / 16 Mhz . . . . . ÖS 4.548,-
80386 / 25 Mhz / 32 k - Cache . . . . . ÖS 9.348,-
80386 / 33 Mhz / 64 k - Cache . . . . . ÖS 10.788,-
80486 / 25 Mhz / 128k - Cache . . . . . ÖS 19.800,-
80486 / 33 Mhz / 128k - Cache . . . . . ÖS 28.680,-

SPEICHERCHIPS
41256 . . . . . ÖS 27,-
414256 . . . . . ÖS 79,-
411000 . . . . . ÖS 79,-
Simm 256 KB . . . . . ÖS 211,-
Simm 1 MB . . . . . ÖS 708,-
Simm 4 MB . . . . . ÖS 3.348,-
Aufpreis SIM -> SIP . . . . . ÖS 24,-
Alle Chips mit 80 ns / Preis per Stück

CONTROLLER

XT Harddiskcontroller . . . . . ÖS 624,-
MFM FDC / HDC 1:1 / bis 4 Laufwerke . . . . ÖS 744,-
IDE FDC / HDC . . . . . ÖS 353,-
IDE FDC / HDC / 2 x serielle / 1 x parallele . . . ÖS 570,-
IDE CAT P60 . . . . . ÖS 1.428,-
mit diesem Controller ist es möglich IDE & MFM Harddisks zu mischen

ZUBEHÖR

Sound Blaster / Musik- Synth. Karte . . . . . ÖS 2.904,-
SICOS Handy Scanner 256 GS 400 dpi . . . . . ÖS 3.708,-
Nahtlose Bildzusammensetzung durch Überscannen bis A3
Modemkarte 2400 Baud . . . . . ÖS 1.560,-
FAX - Modemkarte 9600/2400 Baud . . . . . ÖS 3.960,-
Modem extern 2400 Baud . . . . . ÖS 2.508,-
Streamer DJ10 60/120 Mb . . . . . ÖS 4.872,-
Adaptersteckkarte für DJ10 . . . . . ÖS 1.188,-
Tape f. DJ10 60 Mb . . . . . ÖS 516,-
Keyboard Keytronic 102 Tasten dt. . . . . ÖS 1.044,-
Keyboard Noname 102 Tasten dt. . . . . ÖS 624,-

FÜR DEN TECHNIKER

Universalprogrammer & Tester Expro 60 . . . . ÖS 14.988,-
40 poliger Testsockel, Eprom/EEprom bis 8 Megabit, bip. prom, pal, cmos pal, gal, peel, epld, fpl, microcomputer 8748, 8751, z8 ic-test für ttl 74/54 & cmos 40/45 memory test dram/sram intel/motorola/tektronics/jedec files können verwendet werden.
Logic Analyzer Karte CLK 27100 . . . . . ÖS 27.588,-
100 MHz 6 Kanäle, 50 MHz 24 Kanäle, incl. Klemmen, Software
diverse andere Epromprogrammer, Meßkarten usw., auch low cost auf Anfrage.

NETZWERK

Netzwerksteckkarte
NE 1000 Ethernet komp. 8 bit . . . ÖS 2.139,-
Netzwerksteckkarte
NE 2000 Ethernet komp. 16 bit . . ÖS 2.376,-
Netzwerksteckkarte
NE 3200 orig. Novell 32 bit EISA 23.988,-
NOVELL Netware 2.2 / 5 Benutzer . . . . . ÖS 13.536,-
NOVELL Netware 2.2 / 10 Benutzer . . . . . ÖS 30.096,-
NOVELL Netware 2.2 / 50 Benutzer . . . . . ÖS 51.480,-
NOVELL Netware 2.2 / 100 Benutzer . . . . . ÖS 81.720,-
NOVELL Netware 3.11 / 20 Benutzer . . . . . ÖS 51.480,-
NOVELL Netware 3.11 / 100 Benutzer . . . . . ÖS 103.492,-
NOVELL Netware 3.11 / 250 Benutzer . . . . . ÖS 182.880,-
UPS (unterbrechungsfreie Stromversorgung) . . ÖS 9.900,-
incl Monitoring Board
NDS-NOVELL-NETZWERK
SONDERANGEBOT . . . . . ÖS 118.800,-
bestehend aus:
* Server 386-33Mhz / 64k Cache, 8 MB RAM, 210 MB Festplatte, 1 LW 1.2 MB, Herculesgrafikkarte, 14" Monochrome Monitor, 2 paralelle Schnittstellen, Tower Gehäuse.
*3 Arbeitsstationen: AT 286 /12 Mhz, 2 MB RAM, 1 Laufwerk 5.25 /1.2 MB, VGA-Grafikkarte + VGA-Monochrome Monitor, Baby Tischgehäuse.
* NOVELL Netware 2.2 / 5 User
* Incl. aller Kabel, Adapter und Netzwerksteckkarten
* Incl. Zustellung und Installation (Raum Wien)

DRUCKER TINTENSTRAHL

Canon BJ-10e Schubtraktor 142 Zeichen / sec . . ÖS 6.660,-
incl. Einzelblatteinzug
Canon BJ-300 Schubtraktor 300 Zeichen / sec . . ÖS 11.460,-
Canon BJ-330 Schubtraktor
300 Zeichen / sec A3 . . . . . ÖS 13.488,-
Tintenpatrone für BJ-10e . . . . . ÖS 432,-
Tintenpatrone für BJ-300/330 . . . . . ÖS 300,-
Einzelblatteinzug für BJ-300 . . . . . ÖS 1.548,-
Einzelblatteinzug für BJ-330 . . . . . ÖS 1.764,-

DRUCKER LASER

Texas Instruments . . . . . ÖS 17.388,-
6 Seiten pro Minute, 512k, HP-Laserjet komp.
1 Mbyte Speichererweiterung . . . . . ÖS 1.560,-
Toner Cartridge f. ca 3000 Blatt . . . . . ÖS 756,-
2. Einzugsschacht . . . . . ÖS 5.940,-
DRUCKER NADEL
NEC P20, 24 Nadeldrucker A4 . . . . . ÖS 6.144,-
NEC P30, 24 Nadeldrucker A3 . . . . . ÖS 7.854,-
Einzelblatteinzug P20 . . . . . ÖS 1.788,-
Einzelblatteinzug P30 . . . . . ÖS 2.388,-

MULTI MEDIA

Screen Machine . . . . . ÖS 25.788,-
(Einblenden, scannen und bearbeiten von Videobildern in Echtzeit, Software für Windows und Programmierinterface für alle Hochsprachen.
Als besonderes Zuckerl ist es uns gelungen einen über serielle Schnittstelle ansteuerbaren Videorecorder ausfindig zu machen. Die Kombination dieser zwei Geräte ermöglicht Ihnen die Realisierung von rechnergestützten Auskunftssystemen mit Einblendung von Videoclips z.B. für Messen, Fremdeninformation udgl. In Verbindung mit einer computergesteuerten Videokreuzschiene eröffnen sich ungeahnte Möglichkeiten, z.B. auf dem Gebiet Videoüberwachung:
Videorecorder SVHS Panasonic . . . . . ÖS 51.588,-
incl. RS232 Interface
Audio Option für Screen Machine . . . . . ÖS 4.680,-

SOFTWARE

MS Windows 3.0 . . . . . ÖS 2.837,-
MS Word f. Windows . . . . . ÖS 9.922,-
MS Word 5.5 . . . . . ÖS 9.100,-
MS Excel 3.0 . . . . . ÖS 9.200,-
Aldus Pagemaker 4.0 . . . . . ÖS 16.416,-
Lotus 123 3.1 + Ami Pro . . . . . ÖS 15.120,-
Ventura Publisher Gold f. Windows . . . . . ÖS 19.195,-
Corel Draw 2.0 . . . . . ÖS 9.576,-
Designer 3.01 . . . . . ÖS 12.902,-
Harvard Graphics 2.3 . . . . . ÖS 8.957,-
Geo Works Ensemble . . . . . ÖS 3.327,-
Norton Commander 3.0 . . . . . ÖS 2.275,-
Norton Advanced Utilities 5.0 . . . . . ÖS 2.491,-
PC - Tools 7.0 . . . . . ÖS 2.462,-
Turbo Pascal 6.0 Professional . . . . . ÖS 4.003,-
MS Quick Basic 4.5 . . . . . ÖS 2.232,-
F & A 4.0 . . . . . ÖS 8.626,-
dBase IV 1.1 . . . . . ÖS 14.328,-
LapLink III 3.0 . . . . . ÖS 2.246,-
Auto Sketch 3.0 . . . . . ÖS 3.441,-
Alle Programme in deutsch; über 500 weitere Programme lieferbar. Preis auf Anfrage. Alle Preise incl. 20 % Mwst.

Sonderpreisliste excon für PCCTGM gültig ab 1.9.91

Zahlungskonditionen: Barzahlung
Preise: 9/91, incl. 20% MWSt.
Lieferung: ab Lager Wien
Mindestbestellwert: ÖS 1.000,-
Garantie: 12 Monate auf Komplett-Geräte 6 Monate auf Einzel- und Ersatzteile
Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
Anschrift: EXCON Ing.Günther Hanisch, 1090 Wien, Rögergasse 6-8

Mit dieser Preisliste sind alle vorangegangenen Preislisten ebenso ungültig, wie eventuell in Zusammenhang mit diesen Listen gemachte Sonderkonditionen. Irrtümer und Änderungen jederzeit vorbehalten. Im übrigen gelten die allgemeinen Geschäftsbedingungen der Elektroindustrie Österreichs.

ALLE COMPUTER WERDEN SPEZIELL NACH IHREN WÜNSCHEN KONFIGURIERT!!!

SAMMELBESTELLUNGEN AUF GETRENNTEM BLATT AM ENDE DES HEFTES

Personalcomputer AT 286

A286ST23 AT 286 STANDARD /12Mhz . . ÖS 10.620,-
* Baby AT-Gehäuse mit LED-SPEED Anzeige
* 200W Netzteil
* SPEED: Landmark 15.9, Norton SI 13.4
* Einbauplätze f. Floppy- u.Harddisks: 3x5¼, 1x3½
* CPU 80286-12, 6/12MHz, 0 Wait State
* (HAEDLAND CHIP SET)
* 1 MB RAM, erweiterbar auf 2/4MB
* 1.2MB Diskettenlaufwerk (TEAC)
* 40 MB/28ms Festplatte IDE-AT-BUS (SEAGATE ST157A)
* FDD/HDD-Controller IDE-AT-BUS, Int. 1:1
* 2 seriell/ 1 parallel/ 1 Game Port
* Mono Graphik Printer-Karte (Herc. kompatibel)
* erweiterte Tastatur - 102 Keys, deutsch od. US
* EMS Treiber 4.0

A286DL23 AT 286 DeLUXE /16Mhz . . . . ÖS 11.052,-
* SPEED: Landmark 21.0, Norton SI 18.7
* CPU 80286-16, 8/16MHz, 0 Wait State
* (HAEDLAND CHIP SET)

Alle anderen Daten gleich wie AT 286 Standard

AUFPREISE für AT-286 BASIS-KONFIGURATIONEN

FESTPLATTEN IDE-AT-BUS

80MB/19ms (ST1102A)	ÖS	2.100,-
124MB/19ms (ST1144A)	ÖS	3.840,-
200MB/15ms (ST1239A)	ÖS	8.340,-
320MB/16ms (ST2383A)	ÖS	20.100,-
400MB/15ms (ST1480A)	ÖS	21.780,-
120MB/9-15ms Impulse	ÖS	9.060,-
170MB/9-15ms Impulse	ÖS	9.720,-
210MB/9-15ms Impulse	ÖS	11.160,-

AUFPREISE SCSI+ESDI FESTPLATTEN AUF ANFRAGE

VGA 800x600/ 16Bit/ 256kB OAK	ÖS	660,-
VGA 1024x768/ 16Bit /512kB ET-3000	ÖS	1.500,-
VGA 1024x768/ 16Bit /1MB TSENG ET-4000	ÖS	2.160,-
VGA 1024x768/ 16Bit /1MB TRIDENT-8900	ÖS	1.560,-
Speichererweiterungen		
von 1MB auf 2MB	ÖS	948,-
von 1MB auf 4MB	ÖS	2.784,-

Personalcomputer 386SX

A386CL23 386SX LowCOST /20Mhz . . . .ÖS 14.220,-
* Baby TOWER Gehäuse mit LED-SPEED Anzeige
* 200W Netzteil
* SPEED: Landmark 26.7, Norton SI 23.0
* Einbauplätze f. Floppy- u.Harddisks: 4x5¼, 1x3½
* CPU 80386SX-20, 10/20MHz, 0 Wait State
* (INTEL CHIP SET)
* 1 MB RAM, erweiterbar auf 2/4/8/16MB
* 1.2MB Diskettenlaufwerk (TEAC)
* 40 MB/28ms Festplatte IDE-AT-BUS (SEAGATE ST157A)
* FDD/HDD-Controller IDE-AT-BUS, Int. 1:1
* 2 seriell/ 1 parallel / 1 Game Port
* Mono Graphik Printer-Karte (Herc. kompatibel)
* erweiterte Tastatur - 102 Keys, deutsch od. US
* EMS Treiber 4.0

Personalcomputer 386

A386ST23 386 STANDARD /25MHz . . . .ÖS 16.380,-
* SPEED: Landmark 33.4, Norton SI 28.2
* CPU 80386-25, 20/25MHz, 0 Wait State
* CHIPS Technology CHIP SET, aufrüstbar auf 2/4/8MB

Alle anderen Daten gleich wie AT 386 SX 20MHz

AUFPREISE für 386 u.SX BASIS-KONFIGURATIONEN

FESTPLATTEN IDE-AT-BUS

80MB/19ms (ST1102A)	ÖS	2.100,-
124MB/19ms (ST1144A)	ÖS	3.840,-
200MB/15ms (ST1239A)	ÖS	8.340,-
320MB/16ms (ST2383A)	ÖS	20.100,-
400MB/15ms (ST1480A)	ÖS	21.780,-
120MB/9-15ms Impulse	ÖS	9.060,-
170MB/9-15ms Impulse	ÖS	9.720,-
210MB/9-15ms Impulse	ÖS	11.160,-

AUFPREISE SCSI + ESDI FESTPLATTEN AUF ANFRAGE

VGA 800x600/ 16Bit/ 256kB OAK	ÖS	660,-
VGA 1024x768/ 16Bit /512kB ET-3000	ÖS	1.500,-
VGA 1024x768/ 16Bit /1MB TSENG ET-4000	ÖS	2.160,-
VGA 1024x768/ 16Bit /1MB TRIDENT-8900	ÖS	1.560,-
Speichererweiterung von 1MB auf 2MB (SX20)	ÖS	696,-
von 1MB auf 2MB (386/25)	ÖS	864,-
von 1MB auf 4MB	ÖS	2.256,-
von 1MB auf 8MB (SX20)	ÖS	6.576,-
von 1MB auf 8MB (386/25)	ÖS	5.376,-

Personalcomputer 386

A386SD23 **386 DeLUXE/33 MHz/64k CACHE**
. . . . . ÖS **19.140,-**
* Big TOWER Gehäuse mit LED-SPEED Anzeige
* SPEED: Landmark 41.9, Norton SI 31.6
* 220 W Netzteil, CHIPS Technology CHIP SET
* Einbauplätze f. Floppy- u.Harddisks: 6x5¼,
* CPU 80386-33, 16/33MHz, 0 Waitst., 64KB TTL-CACHE
* 1 MB RAM, erweiterbar auf 2/4/8/16/32MB
* 1.2MB Diskettenlaufwerk (TEAC)
* 40 MB/28ms Festplatte IDE-AT-BUS (SEAGATE ST157A)
* FDD/HDD-Controller IDE-AT-BUS, Int. 1:1
* 2 seriell/ 1 parallel / 1 Game Port
* Mono Graphik Printer-Karte (Herc. kompatibel)
* erweiterte Tastatur - 102 Keys, deutsch od. US

A3864023 **386 DeLUXE/40 MHz/64k CACHE**
. . . . . ÖS **21.420,--**
* CPU 80386-40, 10/40MHz, 0 Wait State
* 64KB TTL-CACHE
* alle anderen Daten wie A386SD23

AUFPREISE für AT-386 BASIS-KONFIGURATIONEN
FESTPLATTE IDE-AT-BUS

80MB/19ms (ST1102A)	ÖS	2.100,-
124MB/19ms (ST1144A)	ÖS	3.840,-
200MB/15ms (ST1239A)	ÖS	8.340,-
320MB/16ms (ST2383A)	ÖS	20.100,-
400MB/15ms (ST1480A)	ÖS	21.780,-
120MB/9-15ms Impulse	ÖS	9.060,-
170MB/9-15ms Impulse	ÖS	9.720,-
210MB/9-15ms Impulse	ÖS	11.160,-

AUFPREISE SCSI+ESDI FESTPLATTEN AUF ANFRAGE

VGA 800x600/ 16Bit/ 256kB OAK	ÖS	660,-
VGA 1024x768/ 16Bit /512kB ET-3000	ÖS	1.500,-
VGA 1024x768/ 16Bit /1MB TSENG ET-4000 .	ÖS	2.160,-
VGA 1024x768/ 16Bit /1MB TRIDENT-8900 .	ÖS	1.560,-
Speichererweiterung von 1MB auf 2MB	ÖS	864,-
Speichererweiterung von 1MB auf 4MB	ÖS	2.256,-
Speichererweiterung von 1MB auf 8MB	ÖS	5.376,-

Personalcomputer 486 ISA-BUS

A486ST23 **486 STANDARD/25MHz/128k CACHE**
. . . . . ÖS **26.280,-**
* Big TOWER Gehäuse mit LED-SPEED Anzeige
* SPEED: Landmark 113.6
* 220 W Netzteil
* Einbauplätze f. Floppy- u.Harddisks: 6x5¼,
* CPU 80486-25, 0 Wait State, 128KB TTL-CACHE
* 1 MB RAM, erweiterbar auf 2/3/4/7/8/12/16MB
* 1.2MB Diskettenlaufwerk (TEAC)
* 40 MB/28ms Festplatte IDE-AT-BUS (SEAGATE ST157A)
* FDD/HDD-Controller IDE-AT-BUS, Int. 1:1
* 2 seriell/ 1 parallel / 1 Game Port
* Mono Graphik Printer-Karte (Herc. kompatibel)
* erweiterte Tastatur - 102 Keys, deutsch od. US

A486DL23 **486 STANDARD/33MHz/256k CACHE**
. . . . . ÖS **29.760,-**
* CPU 80486-33, 0 Wait State, 256kB TTL-CACHE

Alle anderen Daten wie A486ST23

AUFPREISE für AT-486 BASIS-KONFIGURATIONEN
FESTPLATTEN IDE-AT-BUS

80MB/19ms (ST1102A)	ÖS	2.100,-
124MB/19ms (ST1144A)	ÖS	3.840,-
200MB/15ms (ST1239A)	ÖS	8.340,-
320MB/16ms (ST2383A)	ÖS	20.100,-
400MB/15ms (ST1480A)	ÖS	21.780,-
120MB/9-15ms Impulse	ÖS	9.060,-
170MB/9-15ms Impulse	ÖS	9.720,-
210MB/9-15ms Impulse	ÖS	11.160,-

AUFPREISE SCSI+ESDI FESTPLATTEN AUF ANFRAGE

VGA 800x600/ 16Bit/ 256kB OAK	ÖS	660,-
VGA 1024x768/ 16Bit /512kB ET-3000	ÖS	1.500,-
VGA 1024x768/ 16Bit /1MB TSENG ET-4000 .	ÖS	2.160,-
VGA 1024x768/ 16Bit /1MB TRIDENT-8900 .	ÖS	1.560,-
Speichererweiterung von 1MB auf 2MB	ÖS	864,-
Speichererweiterung von 1MB auf 4MB	ÖS	2.256,-
Speichererweiterung von 1MB auf 8MB	ÖS	5.376,-

LAP TOP Personalcomputer

LT360023 **CHICONY LT-3600 / 20Mhz . . ÖS 34.200,-**
* CPU 80286-16, 8/20Mhz, 0 Wait Sate (NEAT-CHIP-SET)
* SPEED: Landmark 26.7, Norton SI 20.2 A B V E R K A U F
* 1 MB RAM, erweiterbar auf 2 oder 4MB
* VGA Karte mit LCD Display (640x480) 8 Graustufen
* Anschluß für Analog VGA Monitor
* Keyboard 82 Tasten + externer Zahlenblock
* eingebauter AKKU für netzunabhängigen Betrieb
* 40 MB/28ms Festplatte IDE-AT BUS
* 1,44 MB/3½" Diskettenlaufwerk

NOTEBOOK Personalcomputer

NB560223 **CHICONY NB-5600/20Mhz/20MB**
. . . . . ÖS **39.000,-**
* CPU 80386SX-20, 10/20Mhz, 0 Wait Sate
* 1 MB RAM, erweiterbar auf 5MB
* 1.44MB/3½" Diskettenlaufwerk
* 20 MB/23ms Festplatte IDE-AT-BUS
* VGA Karte mit LCD Schirm (640x400) 16 Graustufen
* 2 seriell/ 1 parallel Interface
* Anschluß für ext.Floppy 5¼",1.2MB
* Anschluß für externen Analog VGA Monitor
* Keyboard 82 Tasten + externer Zahlenblock
* eingebauter AKKU für netzunabhängigen Betrieb
* Sockel für 80387SX-20 Math-Co-prozessor

NB560423 **CHICONY NB-5600/20Mhz/40MB**
. . . . . ÖS **45.000,-**
* 40 MB/23ms Festplatte IDE-AT-BUS

alle anderen Daten wie NB560223

NB560523 **CHICONY NB-5600/20Mhz/60MB**
. . . . . ÖS **47.880,-**
* 60 MB/23ms Festplatte IDE-AT-BUS AB OKTOBER 91

alle anderen Daten wie NB560223

MONOCHROM - MONITORE

4001MW24	14" Monochrom - Monitor . . . . ÖS * weiss (SAMTRON SM-460PWD)		1.750,-
4001MB24	14" Monochrom - Monitor . . . . ÖS * Bernstein (SAMTRON SM-430A)		1.750,-
4301MV24	14" Monochrom VGA Monitor SM-440 . . . . . .ÖS * SAMTRON SM-440 /weiss * 640 x 480 Bildpunkte		1.980,-
4301MV24	14" Monochrom VGA . . . . . auf Anfrage * 1024 x 768 Bildpunkte		

COLOR - MONITORE

4305MV24	14" VGA-Monitor . . . . . ÖS * SAMTRON SC-428VX * 1024x768, Analog Eingang * 0.28mm, 15.5 - 38Mhz		5.490,-
4301MM24	14" TVM- Supersync 3A . . . . . auf Anfrage * 1024x768 interlaced * Analog/TTL Eingang * 0.28mm, 15.5 - 38Mhz		
4302B024	14" N E C - Mult.-sync 3D . . . . ÖS * 1024x768 interlaced * Analog od.TTL Eingang		11.760,-
4302BS24	14" N E C - Multisync 3D SSI . . ÖS * 1024x768 interlaced, * Analog od.TTL Eingang		14.640,-
4517MM24	17" IDEC - MF-5117 Multisync . .ÖS * 1024x768 non-interlaced * 21.8 - 50kHz, 0.28mm * RGB Analog od. TTL Eingang		18.600,-
4513MM24	16" EIZO - FLEXSCAN 9070S . .ÖS * 1024x768 non-interlaced * 20 - 50kHz, 0.28mm * RGB Analog od. TTL Eingang		17.520,-
4522MM24	16" EIZO - FLEXSCAN 9080i . . ÖS * 1280x1024 non-interlaced * 20 - 64kHz, 0.28mm * RGB Analog od. TTL Eingang		20.640,-
4512MM24	20" EIZO - FLEXSCAN 9400i . . ÖS * 1280 x 1024 non-interlaced * 30 - 64kHz, 0.31mm * RGB Analog od. TTL Eingang		40.560,-

DTP - MONITORE

4518DT24	15" SIGMA PAGEVIEW . . . . .ÖS * A4 Monitor * 1024x768 non-interlaced * incl. Graphikkarte * incl. Treibersoftware		20.760,-
4519DT24	19" SIGMA LASERVIEW . . . . ÖS * A3 Monitor * 1664x1200 non-interlaced * incl. Graphikkarte * incl. Treibersoftware		31.320,-
4520DT24	19" SIGMA LASERVIEW . . . . ÖS * A3 Monitor * 4 Graustufen alle anderen Daten wie 4519DT24		34.500,-

MATRIX - DRUCKER

5006P024	Citizen LSP 120-D . . . . . ÖS * 9-Nadel-Matrix Drucker * 120 Z/Sek.EDV-Qualität (Pica)		2.940,-
5009P024	Citizen Swift 24 Nadel/A4 . . . . ÖS * 190 Z/Sek. EDV-Qualität * 4 Schriftarten		6.660,-
5010P024	Citizen Swift 24 Nadel/A3 . . . . ÖS * 190 Z/Sek. EDV-Qualität * 4 Schriftarten		9.720,-
5010CL24	Sheetfeeder für Citizen Swift A4 . ÖS		1.980,-
5014CL24	Sheetfeeder für Citizen Swift A3 . ÖS		3.780,-
5011CL24	Sheetfeeder für Citizen 120D . . . ÖS		1.980,-
5009CO24	Farbaufrüstsatz für Swift A4 . . . ÖS		930,-
5010CO24	Farbaufrüstsatz für Swift A3 . . . ÖS		1.170,-
5009IF24	Serielles Intergace f. Citizen 120D ÖS		1.170,-
5010FB24	Farbband für Swift 24 A4 / 120D . ÖS		90,-
5011FB24	Farbband für Swift 24 A3 . . . . .ÖS		192,-

TINTENSTRAHL - DRUCKER

BJ10E024	CANON Bubble Jet BJ10e . . . . ÖS * Tintenstrahldrucker * 360x360 dpi, 100 Zeichen/Sec * Parallel Interface		6.660,-
BJ10BA24	Batterie für BJ10e . . . . . .ÖS * für ca. 30 Blatt		840,-
5015CL24	Sheetfeeder für Canon BJ10e . . . ÖS * für 30 Blatt		1.290,-
BJ10DR24	Druckkopf - Tintenpatrone . . . . ÖS * für Canon BJ10e		480,-

LASER - DRUCKER

5013P024	HP Laserjet III P . . . . . ÖS * 4 Seiten/Minute, 300x300 dpi * 512k, Seriell + Parallel IF		22.680,-
5012P024	HP Laserjet III . . . . . ÖS * 8 Seiten/Minute, 300x300 dpi * 1MB, Seriell + Parallel IF		34.680,-
5016P024	HP Laserjet III D . . . . . .ÖS * 8 Seiten/Minute, 300x300 dpi * 1MB, Seriell + Parallel IF * Duplex Druck		54.600,-
5017P024	HP Laserjet III Si . . . . . .ÖS * 16 Seiten/Minute, 300x300 dpi * 1MB, Seriell + Parallel IF * 2 Papierladen á 500 Blatt * incl. DUPLEX Aufrüstsatz		91.200,-
5020P024	ORIGINAL ADOBE POSTSCRIPT CARTDRIGE . . . . . .ÖS		10.080,-
	Speichererweiterungen f.HPLJ (Pacific):		
5021P024	1MB . . . . . ÖS		2.340,-
5022P024	2MB . . . . . ÖS		3.360,-
5023P024	4MB . . . . . ÖS		6.120,-

FARBBÄNDER FÜR BELIEBIGE DRUCKERTYPEN AUF ANFRAGE !!

MOTHERBOARDS

121A0025 AT286 MOTHERBOARD /12 MHz (2/3-SIZE) . . . . . ÖS 1.188,-
* 6/12 MHz, 8 Slots, CPU 80286-12
AUSLAUFMODELL !
* Headland-CHIP-SET, AMI-BIOS
* 0 Wait State, XT 2/3-Size
* 5x16Bit, 2x8Bit SLOT
* Sockel für 80287 (10Mhz)
* 0k RAM , aufrüstbar wie folgt:
1 MB: 8*414256 + 4*41256
2/4 MB: 2/4*SIM Modul 1MB

122A0025 AT286 MOTHERBOARD /16 MHz (2/3-SIZE) . . . . . ÖS 1.620,-
* 8/16 MHz, CPU 80286-16 (AMD)
* Headland-CHIP-SET, AMI-BIOS
* 0 Wait State, XT 2/3-Size
* 5x16Bit, 1x8Bit SLOT
* Sockel für 80287 (12Mhz)
* 0k RAM , aufrüstbar wie folgt:
1 MB: 8*414256 + 4*41256
2/4 MB: 2/4*SIP Modul 1MB

191AM025 386SX MOTHERBOARD /20Mhz (2/3-Size) . . . . . .ÖS 4.080,-
* INTEL-CHIP-SET, incl. EMS Driver
* 2/3 XT-Size, AMI-BIOS mit Password
* 6x16Bit, 2x8Bit SLOT
* Sockel für 80387SX (20Mhz)
* 0k RAM, erweiterbar wie folgt:
1 MB : 4*SIMM Modul 256k
2/4 MB : 2/4*SIMM Modul 1MB
8/16 MB: 2/4*SIMM Modul 4MB

190AM025 386 MOTHERBOARD /25MHz (XT-Size) . . . . . .ÖS 6.240,-
* CPU 80386-25, AMI-BIOS
* XT-Size, CHIPS-TECHNOLOGY CHIP-SET
* 3x8, 4x16 und 1x32 Memory BITS Slots
* SOCKEL f.80287(8Mhz) u.80387(20 od.25Mhz)
* 0k RAM, erweiterbar wie folgt:
1/2 MB: 4/8*SIMM Modul 256k
4/8 MB: 4/8*SIMM Modul 1MB

193AM025 386/64k CACHE MOTHERBOARD /33MHz (BABY) . . . . . ÖS 8.340,-
* CPU 80386-33, 64kB CACHE,
* CHIPS-TECHNOLOGY-CHIP-SET
* AMI BIOS mit Password
* 2x8, 6x16 Slots, XT-Size
* SOCKEL für 80387 (33Mhz) od. Weitek
* 0k RAM, erweiterbar wie folgt:
1/2 MB: 4/8*SIMM Module 256k
4/8 MB: 4/8*SIMM Module 1MB
16/32 MB: 4/8*SIMM Module 4MB

201AM025 386/64k CACHE MOTHERBOARD /40MHz (BABY) . . . . . ÖS 10.620,-
* CPU 80386-40, 20/40 MHz
alle anderen Daten wie 193AM025

196AM025 486/128k CACHE MOTHERBOARD/25MHz ISA . . . . . .ÖS 15.480,-
* CPU 80486-25, 128kB Cache
* 8x16Bit Slots, Big AT Size
* SOCKEL für Weitek 4167 Coproz.
* 0k RAM, erweiterbar wie folgt:
1 MB: 4*SIMM Module 256k
2 MB: 8*SIMM Module 256k
4 MB: 4*SIMM Module 1MB oder 36*411000
8 MB: 8*SIMM Module 1MB
12 MB: 12*SIMM Module 1MB
16 MB: 12*SIMM Module 1MB + 36*411000

197AM025 486/64k CACHE MOTHERBOARD/33MHz ISA . . . . . .ÖS 17.880,-
* CPU 80486-33, 10/33Mhz
* 64 kB CACHE, 2x8Bit, 6x16Bit Slot
* 0k RAM, erweiterbar wie folgt:
1/2MB: 4/8*SIMM Module 256k
4/8MB: 4/8*SIMM Module 1MB
16/32MB: 4/8*SIMM Module 4MB
alle anderen Daten wie 196AM025

198AM025 486/256k CACHE MOTHERBOARD/33MHz ISA . . . . . .ÖS 18.960,-
* 256 kB CACHE
alle anderen Daten wie 197AM025

GEHÄUSE - STROMVERSORGUNG

3202C027 BABY-AT-GEHÄUSE . . . . . .ÖS 1.800,-
* inkl. 200 W Netzteil
* 3x5¼, 1x3½ Slim Einschubplätze
* LED - Speed Display
* für XT/Baby Size Motherboards

3204C027 BABY - TOWER . . . . . .ÖS 2.460,-
* incl. 200W Netzteil
* 4x5¼", 1x3½" Slim Einschubplätze
* LED - Speed Display
* für XT u. Big Size Motherboards

3205C027 BIG - TOWER . . . . . .ÖS 3.120,-
* incl. 220W Netzteil
* 6 Slim Einschubplätze 5¼"
* LED - Speed Display
* für XT u. Big Size Motherboards

3206C027 SLIM LINE GEHÄUSE . . . . . ÖS 1.800,-
* inkl. 200 W Netzteil, max 4 Slot
* 1x5¼, 2x3½ Slim Einschubplätze
* für XT/Baby Size Motherboards

3201C027 SUPER SLIM LINE GEHÄUSE . ÖS 1.560,-
* inkl. 60W Netzteil ohne Lüfter
* max. 2 Slots
* 1x3½ Slim Einschubplätze
* max. 2/3 Size-Boards

3208C027 GEHÄUSE F.EXT. FDD/HDD . . ÖS 1.320,-

1100S027 Thermo-Lüftersteuerung . . . . . ÖS 600,-
* für beliebige PC-Netzteile

NETZTEILE

1200S027 200W f. Baby-AT Gehäuse . . . .ÖS 1.380,-
1201S027 200W f. Baby-Tower . . . . . .ÖS 1.500,-
1202S027 220W f. Big-Tower . . . . . .ÖS 1.560,-
1203S027 200W f. Slim Line Gehäuse . . . .ÖS 1.500,-

FLOPPY - HARDDISK - CONTROLLER

411F0026	Floppy-Disk-Controller XT/AT . . . ÖS * 2x 360/720kB/1.2/1.44MB Floppy	360,-
450F0026	Floppy-Disk-Controller XT/AT . . . ÖS * 4x 360/1.2/1.44/2.8MB Floppy	1.176,-
431F/H26	MFM Floppy-Harddisk Controller . . ÖS * 2 x Floppy- u.2 x Hard Disk * MFM, Interleave 1:1 * incl. Kabel-Set	948,-
436F/H26	SUPER I/O Controller . . . . . ÖS * 2 x Floppy- u. 2 x Harddisk * IDE-AT-BUS, Interleave 1:1 * 2 Seriell/1 Parallel/1 Game Port * incl. Kabel-Set	480,-
440F/H26	IDE-AT-BUS Seagate ST08 . . . . . ÖS * 2 x Floppy- u. 2 x Harddisk * IDE-AT-BUS, Interleave 1:1 * incl. Kabel-Set	540,-
447F/H26	IDE-AT-BUS Contr. . . . . ÖS * 2 x Floppy- u. 2 x Harddisk * IDE-AT-BUS, Interleave 1:1 * mit BIOS (erkennt die Harddisk-Parameter automatisch) * incl. Kabel Set	660,-
448F/H26	IDE-AT-BUS HARDDISK Contr. . . ÖS * 2x intern/2x externe Harddisk * IDE-AT-BUS, Interleave 1:1 * mit BIOS, kann zusätzlich zu bereits vorhandenen Harddisk Controllern eingebaut werden !! * incl. Kabel Set	1.320,-
449F/H26	AT-BUS+SCSI FD/HD Contr. . . . . ÖS * 4 x Floppy Disk, 2x IDE-AT-BUS und SCSI Controller (Future-Domain kompatibel) * incl. Kabel Set	1.080,-
437F/H26	SCSI Harddisk-Controller . . . . . ÖS * Seagate ST01 * 2xSeagate SCSI Harddisk * incl. SCSI Kabel * ACHTUNG! nur für MS-DOS geeignet	480,-
438F/H26	SCSI Floppy-Harddisk Controller . . ÖS * Seagate ST02 * 2xSeagate SCSI Harddisk + 2xFloppy * incl. Kabel-Set * ACHTUNG! nur für MS-DOS geeignet	708,-
439F/H26	SCSI Floppy-Harddisk Controller . . ÖS * Future Domain kompatibel * 2 x Floppy u. 2 x Harddisk * incl. Treiber f.Novell u. Xenix * incl. Kabel Set	1.020,-
435F/H26	SCSI Floppy-Harddisk Controller . . ÖS * ADAPTEC 1542-B * 2 x Floppy u. 2 x Harddisk	5.400,-
433F/H26	SCSI FDC/HDC - ADAPTEC-KIT . . . . . ÖS * ADAPTEC 1542-B * incl. Treiber für OS/2 und NOVELL * incl. Kabel Set	6.600,-
443F/H26	PARALELL -> SCSI ADAPTER . . ÖS * zum Anschluß an die parallele Schnittstelle (extern) * incl. Software	2.880,-
432F/H26	ESDI Floppy-Harddisk Controller . . ÖS * Western Digital WD 1007	3.600,-

CACHE - HARDDISK - CONTROLLER

444F/H26	FAST IDE CACHE Contr. . . . . ÖS * 2x Floppy/2x Harddisk Controller * 512kB Cache	6.300,-
445F/H26	FAST SCSI CACHE Contr. . . . . ÖS * 2x Floppy/2x Harddisk Controller * 500kB Cache	10.440,-
446F/H26	FAST SCSI CACHE Contr. . . . . ÖS * 2x Floppy/2x Harddisk Controller * 2MB Cache	13.560,-

RAM

41256080	Dyn.RAM 41256-80 (256kx1) . . ÖS	24,--
41464080	Dyn.RAM 41464-08 (64kx4) . . . ÖS	24,--
41425680	Dyn.RAM 414256-08 (256kx4) . . ÖS	90,--
41100080	Dyn.RAM 411000-08 (1024kx1) . ÖS	90,--
SIM25608	SIMM MODULE 80nS (256kx9) . ÖS	216,-
SIM1MB08	SIMM MODULE 80nS (1024kx9) ÖS	780,-
SIM4MB08	SIMM MODULE 80ns (4096kx9) ÖS	3.720,-
SIP25608	SIP MODULE 80nS (256kx9) . . ÖS	300,-
SIP1MB08	SIP MODULE 80nS (1024kx9) . . ÖS	900,-

SCHNITTSTELLEN - KARTEN

630C0026	Multi I/O - Karte für AT . . . . . ÖS * 2x Ser/1x Par/1x Game	300,-
360D0026	Parallel - Printer - Karte . . . . . ÖS	180,-
601C0026	RS 232 (4 Port) incl. Treiber . . . ÖS	840,-
	INTELLIGENTE SCHNITTSTELLENKARTEN f. UNIX/XENIX AUF ANFRAGE !	

GRAPHIK - KARTEN

310D0026	Mono/Graphic/Printer - Karte . . . ÖS * Hercules-kompatible Karte	240,-
337D/B26	VGA - Karte /16Bit . . . . . ÖS * 256 k, 16 Farben * OAK Chip-Set (800x600) * aufrüstbar auf 512 kB (1024x768)	900,-
334D/B26	VGA - Karte /16-Bit ET-3000 . . . ÖS * 1024 x 768 Bildpunkte, 512kB * interlaced od non-interlaced * TSENG ET-3000 kompatibel	1.740,-
335D/B26	TSENG-LAB'S MegaVGA . . . . ÖS * ORIGINAL ET4000 * 1024 x 768 Bildpunkte, 1 MB * interlaced od non-interlaced * 1024x768/43,5/60/70 HZ * 800x600/56/60/72 HZ	2.400,-
335DOB26	70/72Hz OPT. für ET-4000 . . . . ÖS (Aufrüstung für ältere Versionen)	180,-
335DCB26	32 K Color OPTION für ET-4000 . ÖS * 800x600/56 Hz, 640x480/60/70/72 Hz	600,-
335CEG26	ESUN CEG-Option 32k Color . . ÖS * für TSENG ET3000, ET4000 MegaVGA und Trident TAVA 8900 VGA Karten * incl. Treiber f. Windows,Lotus 1-2-3, Autocad.	1.560,-
336D/B26	VGA - Karte /16-Bit TAVA 8900 . ÖS * 1024x768 , 1 MB, Trident 8900 komatibel, * interlaced od non-interlaced	1.800,-
338D/B26	HERCULES GB1024/1MB . . . . ÖS * 1024x768, 1MB, Ti Graphic Processor	12.480,-
341D/B26	HERCULES GB1024/3MB . . . . ÖS * wie 338D/B26 mit 3MB	17.880,-
339D/B26	Programmers Reference Manual . ÖS * für Tseng ET3000	468,-
340D/B26	Reference Manual für ET4000 . . ÖS	468,-

DISKETTEN-LAUFWERKE

810F/J27	1,2MB/5¼" FLOPPY TEAC . . . . .ÖS	1.116,-
811F/027	720kB/3½" FLOPPY TEAC . . . . . ÖS (ohne Rahmen)	948,-
812F/027	1,44MB/3½" FLOPPY TEAC . . . . ÖS (ohne Rahmen)	936,-
820F/J027	TEAC 5¼" . . . . . .ÖS * Einbaukit für 3½" Floppy * Rahmen, Adapterplatine u. Stromadapter graue Blende passend zu o.a.TEAC Floppy	180,-
821F/J027	5¼" ADAPTER für 3½" Floppy . . . ÖS * wie 820F/J027 aber mit beiger Blende	120,-

FESTPLATTEN MFM

920MFM27	20 MB MFM/40ms . . . . . .ÖS SEAGATE ST124, 3½"	3.840,-
940MFM27	40 MB MFM/28ms . . . . . .ÖS SEAGATE ST251-1, 5¼"HH	4.080,-
980MFM27	80 MB MFM/16.5ms . . . . . ÖS SEAGATE ST1100, 3½"	11.100,-

FESTPLATTEN AT-BUS

9040AT27	40 MB IDE/28ms . . . . . .ÖS SEAGATE ST157A,3½"	3.540,-
9080AT27	80 MB IDE/19ms . . . . . .ÖS SEAGATE ST1102A,3½"	5.640,-
9124AT27	124MB IDE/19ms . . . . . ÖS SEAGATE ST1144A,3½"	7.380,-
9200AT27	200MB IDE/15ms . . . . . ÖS SEAGATE ST1239A,3½"	11.880,-
9320AT29	320MB IDE/16ms . . . . . ÖS SEAGATE ST2383A,5¼"HH	23.640,-
9400AT29	400MB IDE/15ms . . . . . ÖS SEAGATE ST1480A.3½"	25.320,-
9120AT29	120MB IDE/9-15ms . . . . . ÖS IMPULSE,3½" +Adapterkit	12.420,-
9170AT29	170MB IDE/9-15ms . . . . . ÖS IMPULSE,3½" +Adapterkit	13.080,-
9210AT29	210MB IDE/9-15ms . . . . . ÖS IMPULSE,3½" +Adapterkit	14.520,-

FESTPLATTEN SCSI

9150SC27	150MB SCSI/15ms . . . . . .ÖS SEAGATE ST1186N,3½"	11.520,-
9200SC27	200MB SCSI/15ms . . . . . .ÖS SEAGATE ST1239N,3½"	12.120,-
9320SC27	320MB SCSI/10.7ms . . . . . .ÖS SEAGATE ST4385N,5¼"FH	26.880,-
9350SC27	350MB SCSI/14ms . . . . . .ÖS SEAGATE ST1400N,3½"	22.920,-
9400SC27	400MB SCSI/14ms . . . . . .ÖS SEAGETE ST1480N,3½"	25.320,-
9600SC27	600MB SCSI/16ms . . . . . .ÖS SEAGATE ST4702N,5¼"FH	28.800,-
9000SC27	1 GB SCSI/16ms . . . . . .ÖS SEAGATE ST41200N,5¼"FH	42.120,-
9120SC27	120MB SCSI/9-15ms . . . . . .ÖS IMPULSE 3½"	11.460,-
9170SC27	170MB SCSI/9-15ms . . . . . .ÖS IMPULSE 3½"	12.120,-
9210SC27	210MB SCSI/9-15ms . . . . . .ÖS IMPULSE 3½"	14.160,-

FESTPLATTEN ESDI

9100ES27	100MB ESDI/15ms . . . . . .ÖS SEAGATE ST1111E,3½"	11.400,-
9180ES27	180MB ESDI/15ms . . . . . .ÖS SEAGATE ST1201E,3½"	14.280,-
9320ES27	320MB ESDI/15ms . . . . . .ÖS SEAGATE ST2383E,5¼"HH	23.640,-
9680ES27	680MB ESDI/15ms . . . . . .ÖS SEAGATE ST4766E,5¼"FH	30.120,-
9600ES27	680MB ESDI/11,9ms . . . . . .ÖS SEAGATE ST4767E,5¼"FH	36.960,-

FESTPLATTEN MONTAGE-KIT

822F/J27	Universal Rahmen für 3½" FP . . ÖS * Bestehend aus Rahmen u. Schrauben	72,-
823F/J27	Universal Adapterkit für 3½" FP . ÖS * Bestehend aus Rahmen, Schrauben, Frontblende + LED	108,-

CD ROM

9201CD27	CD ROM intern CM 201 . . . . . ÖS * incl.Controller u.Treibersoftware f.DOS	5.616,-
9050CD27	CD ROM extern CM 50 . . . . . .ÖS * incl.Controller u.Treibersoftware f.DOS * Audioausgang	9.120,-

STREAMER TAPES

1940S027	TECMAR 40 MB int. . . . . ÖS * zum Anschluß an den FDD-Contr.	5.490,-
1941S027	TECMAR 120 MB intern . . . . .ÖS * zum Anschluß an den FDD-Contr.	8.640,-
1942S027	TECMAR Controller Karte . . . .ÖS	2.100,-
1943S027	40MB Cardridge für TECMAR . .ÖS	564,-
1907S027	MAYNARD 60 MB intern . . . . ÖS * incl. Controller Karte * für NOVELL Netzwerk geeignet	13.560,-
1908S027	MAYNARD 155 MB intern . . . .ÖS * incl. Controller Karte * für Novell Netzwerk geeignet	15.840,-
1909S027	KASSETTE 60 MB . . . . . .ÖS	600,-
1913S027	KASSETTE 155 MB . . . . . ÖS	720,-
1919S027	IRWIN 40/120MB intern . . . . .ÖS * zum Anschluß an den Floppy Controller oder IRWIN Interface oder Controller	8.520,-
1914S027	IRWIN 80/250MB intern . . . . .ÖS * zum Anschluß an den Floppy Controller oder IRWIN Interface oder Controller	9.600,-
1915S027	IRWIN 40/120MB extern . . . . .ÖS * benötigt IRWIN Interface oder Controller	9.960,-
1916S027	IRWIN 80/250MB extern . . . . .ÖS * benötigt IRWIN Interface oder Controller	11.975,-
1917S027	IRWIN INTERF 4251 int/ext . . .ÖS * Installationskit für XT,AT,386	1.560,-
1918S027	IRWIN CONTROLLER 4100AT .ÖS * Controller für int.u.externe Streamer * ermöglicht DOUBLE SPEED mit o.a. IRWIN Streamer !	3.840,-
1920S027	IRWIN DC2000-40 Cartr 40MB . ÖS	480,-
1921S027	IRWIN DC2000-80 Cartr 80MB . ÖS	516,-
1922S027	IRWIN DC2000XL-60 Cartr 60MB . . . . . .ÖS	648,-
1923S027	IRWIN DC2000XL-120 Cartr 120MB . . . . . .ÖS	648,-

DAT STREAMER TAPES

DAT1SI27	ARCHIVE PHYTON 1.3GB int . .ÖS * ohne Controller u. ohne Software	41.520,-
DAT1SE27	ARCHIVE PHYTON 1.3GB ext . ÖS * ohne Controller u. ohne Software	49.080,-
DAT2SI27	ARCHIVE PHYTON 2.0GB int . .ÖS * ohne Controller u. ohne Software	41.520,-
DAT2SE27	ARCHIVE PHYTON 2.0GB ext . ÖS * ohne Controller u. ohne Software	49.080,-
DAT13B27	STREAMER BAND 1.3GB 4mm .ÖS	468,-
DAT13B27	STREAMER BAND 2.0GB 4mm .ÖS	984,-
DATSYG27	SYTOS German 3.1 Streamer Software . . . . . ÖS	1.500,-
DATSYP27	SYTOS PLUS Streamer Software .ÖS	3.120,-
DATSYX27	SYTOS XENIX Streamer Softw . ÖS	2.100,-

TASTATUREN

2100K027	102 KEY XT/AT (Datacomp) GR	ÖS	720,-
2101K027	102 KEY XT/AT (Datacomp) US	ÖS	720,-
2102K027	102 KEY (CHERRY) GR	ÖS	1.440,-

MATH-CO-PROZESSOREN

701CP087	80S87 für 80286 bis 20 MHz (IIT)	ÖS	1.650,-
707CP087	83S87-20Mhz (Cyrix) f.386SX	ÖS	2.520,-
705CP087	83D87-25Mhz (Cyrix) f.386	ÖS	3.480,-
706CP087	83D87-33Mhz (Cyrix) f.386	ÖS	3.780,-
708CP087	83D87-40MHz (Cyrix) f.386	ÖS	4.320,-

BETRIEBSSYSTEME

7000D031	MS-DOS 3.3 (englisch)	ÖS	828,-
7002D031	MS-DOS 3.3 (deutsch)	ÖS	1.080,-
7001D031	MS-DOS 4.01 (englisch)	ÖS	1.080,-
7003D031	MS-DOS 4.01 (deutsch)	ÖS	1.080,-
7004D031	MS-DOS 5.0 (deutsch) Umsteigerpaket	ÖS	1.800,-

* Umsteigerpaket kann nur installiert werden, wenn bereits auf dem PC eine alte DOS Version vorhanden ist !!!!!

UNTERBRECHUNGSFREIE STROMVERSORGUNGEN

1901S027	UPS 600 VA (Stand-by)	ÖS	7.200,-
1902S027	UPS 1000 VA (Stand-by)	ÖS	9.000,-

MOUSE - SCANNER - GRAPHIK TABLETS

5103A028	MICROSOFT BUS- oder SERIELL-MOUSE	ÖS	1.590,-
5100A028	SERIELL GENIUS GM6000	ÖS	570,-

* MS-PC - kompatibel AUSLAUFMODELL

5110A028	SERIELL GENIUS F-302	ÖS	690,-

* MS-PC - kompatibel, PS/2 tauglich

5101A028	TRACBALL GENIUS GTK-320 SERIELL	ÖS	1.410,-

* MS-PC - kompatibel

5111A028	Adapterstecker GENIUS F-302	ÖS	126,-

* für PS/2

5150A028	GENIUS Handy Scanner 4500	ÖS	2.880,-

* 100-400 DPI, 32 Graustufen
* DrGenius, Scan Edit, OCR-Software

5149A028	GENIUS OCR Software für GENIUS 4500	ÖS	1.260,-
5161A028	EPSON GT-4000 Flachbettscanner	ÖS	31.800,-

* Farbscanner, 50-400 dpi, 256 Graustufen,
* 15-90 sec/200dpi, ZOOM Funktion 50-200%
* benötigt BI-DI Interface u. Software !

5162A028	EPSON BI-DI Interf. f. GT-4000	ÖS	1.440,-

* incl. Parallel Drucker Kabel
SOFTWARE für EPSON SCANNER auf Anfrage !!

MOUSE - SCANNER - GRAPHIK TABLETS

5160A028	GENITIZER 1212B GRAPHIKTABLET	ÖS	5.580,-

* 12"x12", für ACAD geeignet

5159A028	GENIUS GT-S01 Stylus Pen für GT1212B	ÖS	780,-
	5151A028 QUICK SHOT QS-113P JOYSTICK f. PC	ÖS	420,-

* incl. Karte mit Anschluß für 2 Joystick

5152A028	QUICK SHOT QS-113 JOYSTICK f. PC	ÖS	198,-

* zum Anschluß an den PC-GAME-PORT

5106A028	DISKETTENBOX 5¼" für 100Stk.	ÖS	117,60
5107A028	DISKETTENBOX 3½" für 100Stk	ÖS	117,60

DISKETTEN

5700A028	Noname 5¼" DS/DD	ÖS	4,80
5701A028	Maxell 5¼" DS/HD	ÖS	18,--
5710A028	Noname 3½" DS/DD	ÖS	13,80
5720A028	Maxell 3½" DS/HD	ÖS	27,60
5722A028	HOST 3½" DS/HD	ÖS	14,40

KABEL

5300A028	DRUCKERKABEL PARALLEL 1.8 m	ÖS	120,-
5306A028	DRUCKERKABEL PARALLEL 7 m	ÖS	276,-
5207A028	DRUCKERKABEL PARALLEL 10 m	ÖS	360,-
5301A028	CENTRONICS/CENTRONICS 3 m	ÖS	192,-
5303A028	TASTATURKABEL 2 m (Verlängerung)	ÖS	108,-
5304A028	MONITORKABEL 2 m RGB-TTL (Verlängerung)	ÖS	144,-
5309A028	MONITORKABEL 2 m VGA (Verlängerung)	ÖS	192,-
5403A028	FLOPPY KABEL	ÖS	60,-
5402A028	HARDDISK-KABELSET	ÖS	108,-

* Kabel f. MFM/ESDI Controller

5405A028	HARDDISK-KABELSET	ÖS	108,-

* Kabel f. SCSI Controller

5404A028	HARDDISK-KABELSET	ÖS	108,-

* Kabel f. IDE-AT-BUS Controller

5003Z028	ADAPTERSTECKER RS 232	ÖS	108,-

* 9/25 Pin Adapter

5310A028	Netzkabel	ÖS	108,-
5311A028	Netzkabel zum Anschluß am PC-Netzteil	ÖS	108,-

SONSTIGES ZUBEHÖR

5514MF28	MONITOR-COLOR-FILTER 14"	ÖS	150,-
5512MF28	MONITOR-COLOR-FILTER 12"	ÖS	132,-
5513MF28	MONITOR GLAS-FILTER	ÖS	1.908,-

* zum Schutz gegen Elektrostatik, UVA und UVB Strahlung 12-14"

5621A028	TASTATURLADE UNTERBAU	ÖS	660,-
5502A028	CPU Ständer für PC-Tischgehäuse	ÖS	180,-
5500A028	DRUCKERSTÄNDER A4	ÖS	228,-
5501A028	DRUCKERSTÄNDER A3	ÖS	228,-
DS101027	DATA SWITCH CENTRONICS 2 fach (AB)	ÖS	348,-
DS101127	DATA SWITCH CENTRONICS 4 fach (ABCD)	ÖS	528,-
5601A028	KONZEPTHALTER A4 mit Standkonsole	ÖS	360,-
5010SH24	SCHUTZHÜLLE F. A3 DRUCKER	ÖS	120,--

NETZWERK - KARTEN

ARC01026	ARC-NET (SMC-kompatibel) 8 Bit	ÖS	1.080,-
ARC01126	ARC-NET (SMC-kompatibel) 16 Bit	ÖS	1.440,-
ARC20026	AKTIVE HUB 4-PORT (intern) . * für max. 600 m Kabellänge	ÖS	1.080,-
ARC03026	AKTIVE HUB 8-PORT extern . . * für max. 600 m Kabellänge	ÖS	3.000,-
ARC02026	PASSIVE HUB 4-PORT . . . . . * für max. 10 m Kabellänge	ÖS	216,-
ARCK6226	ARCNET Kabel (Preis/Meter) . .	ÖS	13,2
ARCBNC26	ARCNET BNC-Connector	ÖS	43,2
ARCTER26	ARCNET Terminator	ÖS	90,-
ETH10026	ETHERNET CARD, 8-Bit Original NE-1000	ÖS	3.480,-
ETH20026	ETHERNET CARD, 16-Bit Original NE-2000	ÖS	4.320,-
ETHROM26	BOOTROM FÜR NE-1000/2000 .	ÖS	870,-
ETH01026	ETHERNET CARD, 8-Bit . . . . * 10MB/sec, NE-1000 kompatibel * incl. BOOT-ROM	ÖS	2.028,-
ETH01126	ETHERNET CARD, 16-Bit . . . * 10MB/sec, NE-2000 kompatibel * incl. BOOT-ROM	ÖS	2.280,-
ETHXIR26	XIRCOM POCKET ETHERNET Adapter BNC . . . . . * Für jeden PC mit Paralleler Schnittstelle geeignet	ÖS	8.640,-
ETHK5826	ETHERNET Kabel (Preis/Meter) .	ÖS	13.2
ETHBNC26	ETHERNET BNC-Connector . . .	ÖS	43,2
ETHTER26	ETHERNET Terminator	ÖS	90,-
ETHUP026	ETHERNET UNTERP. DOSE . .	ÖS	750,-
ETHAPO26	ETHERNET AUFPUTZ DOSE . .	ÖS	708,-
0KONF026	KONFEKTIONIEREN per Kabel .	ÖS	300,-

NOVELL-NETZWERK-SOFTWARE

NOV22026	NETWARE (2.2) - 5 USER	ÖS	13.620,-
NOV22126	NETWARE (2.2) - 10 USER . . .	ÖS	30.360,-
NOV22226	NETWARE (2.2) - 50 USER . . .	ÖS	53.160,-
NOV22326	NETWARE (2.2) - 100 USER . .	ÖS	83.760,-
NOV30226	NETWARE 386 (3.11) 20-USER .	ÖS	53.820,-
NOV31026	NETWARE 386 (3.11) 100-USER	ÖS	106.560,-
NOV32526	NETWARE 386 (3.11) 250-USER	ÖS	190.560,-
0INSTS20	Installation Server	ÖS	9.000,-
0INSTT30	Installation pro Workstation	ÖS	600,-

NOVELL NETZWERK VAP u. NLM SUBSYSTEME

NET25V26	PROLINE 250Q STREAMER TAPE extern . . . . * Komplett incl. QUIC Adapter, Kabel, und VAP für Novell Advanced 2.x * Anschluß direkt am Server	ÖS	32.040,-
NET25N26	PROLINE 250Q STREAMER TAPE extern . . . . * Komplett incl. QUIC Adapter, Kabel, und NLM für Novell 3.x * Anschluß direkt am Server	ÖS	36.000,-
NET50V26	PROLINE 525 SCSI-STREAMER TAPE extern . * Komplett incl. SCSI Adapter, Kabel, und VAP für Novell Advanced 2.x * Anschluß direkt am Server	ÖS	55.560,-
NET50N26	PROLINE 525 SCSI-STREAMER TAPE extern . * Komplett incl. SCSI Adapter, Kabel, und NLM für Novell 3.x * Anschluß direkt am Server	ÖS	59.280,-
NET1GV26	PROLINE D-VAULT 1.3GB STREAMER extern . . . . . * Komplett incl. SCSI Adapter, Kabel, und VAP für Novell Advanced 2.x * Anschluß direkt am Server	ÖS	74.160,-
NET1GN26	PROLINE D-VAULT 1.3GB STREAMER extern . . . . . * Komplett incl. SCSI Adapter, Kabel, und NLM für Novell 3.x * Anschluß direkt am Server	ÖS	77.760,-

DIENSTLEISTUNG

0TECHN20	Techniker - Stunde	ÖS	720,-
0ASSEM20	Assembling (Montage) u. Test . .	ÖS	720,-
0ZUSTE20	Zustellpauschale (gilt nur für WIEN)	ÖS	480,-

Sonderangebot für PCC - TGM (Gültig nur für Sammelbestellung bis 20.12.91)

EXCON AT286/16MHz,
Einsteigermodell . . . . . .ÖS 8.670,-
* BABY-AT-Gehäuse + 200 W Netzteil
* CPU 80286-16, 16 Bit, 8/16MHz, 0 Wait State
* 1MB RAM, erweiterbar auf 2/4MB
* 1.2 MB Diskettenlaufwerk (TEAC)
* Floppy + AT-Bus Harddisk-Controller
* 2 Seriell, 1 Parallel Interface
* 16Bit VGA-Karte OAK (800x600)
* erweiterte Tastatur - 102 Keys
* 14" VGA Monochrom Monitor (SAMTRON)

CITIZEN 120D+ . . . . . .ÖS 2.790,-
* 9 Nadel Matrix Drucker
* incl. Druckerkabel
* 120 Zeichen/sec

CITIZEN SWIFT 24 . . . . . .ÖS 6.360,-
* 24Nadel Matrix Drucker
* incl. Druckerkabel
* 190 Zeichen/sec
* Farbaufrüstung OPTIONAL
* Automatischer Einzelblatteinzug OPTIONAL

GENIUS GM6000 MOUSE . . . . . .ÖS 570,-
GENIUS F302 MOUSE . . . . . .ÖS 690,-

ERWEITERUNGEN/AUFPREISE für AT286/16

AUFPREIS 1MB ==> 2MB RAM . . . . . .ÖS 948,-
AUFPREIS 1MB ==> 4MB RAM . . . . . .ÖS 2.784,-
AUFPREIS
14" Farb VGA Monitor + 16Bit VGA ET3000 (1024x768/512k) . . . . . ÖS 3.900,-
AUFPREIS
14" Farb VGA Monitor + 16Bit VGA ET4000 (1024x768/1M) . . . . . ÖS 4.980,-
HARDDISK 40MB/28ms Seagate ST157A . . .ÖS 3.360,-
HARDDISK 80MB/19ms Seagate ST1102A . . ÖS 5.370,-
HARDDISK 124MB/19ms Seagate ST1144A . .ÖS 6.990,-
FLOPPY-DISK 1.44MB, 3½" (TEAC) incl. Einbauset . . . . . .ÖS 1.056,-
AUFPREIS
Baby-AT-Gehäuse -> Baby Tower Gehäuse . . .ÖS 630,-
AUFPREIS
Baby-AT-Gehäuse -> Big Tower Gehäuse . . . .ÖS 1.200,-

EXCON 386SX/20MHz
Einsteigermodell . . . . . .ÖS 7.980,-
* BABY-AT-Gehäuse + 200 W Netzteil
* CPU 386SX-20, 16 Bit, 10/20 MHz,0/1 Wait
* 1MB RAM, erweiterbar auf 2/4/8/16MB
* 1.2 MB Diskettenlaufwerk (TEAC)
* Floppy + AT-Bus Harddisk-Controller
* 2 Seriell, 1 Parallel Interface
* erweiterte Tastatur - 102 Keys
* OHNE GRAFIKKARTE, OHNE MONITOR

CITIZEN 120D+ . . . . . .ÖS 2.790,-
* 9 Nadel Matrix Drucker
* incl. Druckerkabel
* 120 Zeichen/sec

CITIZEN SWIFT 24 . . . . . .ÖS 6.360,-
* 24Nadel Matrix Drucker
* incl. Druckerkabel
* 190 Zeichen/sec
* Farbaufrüstung OPTIONAL
* Automatischer Einzelblatteinzug OPTIONAL

GENIUS GM6000 MOUSE . . . . . .ÖS 570,-
GENIUS F302 MOUSE . . . . . .ÖS 690,-

ERWEITERUNGEN/AUFPREISE für 386SX/20

AUFPREIS 1MB ==> 2MB RAM . . . . . .ÖS 696,-
AUFPREIS 1MB ==> 4MB RAM . . . . . .ÖS 2.256,-
14" Mono VGA Monitor + 16Bit VGA Karte OAK (800x600/256kb) . . . . . ÖS 2.700,-
14" Farb VGA Monitor + 16Bit VGA ET3000 (1024x768/512kB). . . . . ÖS 6.840,-
16" EIZO 9070S Monit + 16Bit VGA ET4000 (1024x768/1MB) . . . . . ÖS 18.900,-
HARDDISK 80MB/19ms Seagate ST1102A . . ÖS 5.370,-
HARDDISK 124MB/19ms Seagate ST1144A . .ÖS 6.990,-
HARDDISK 200MB/15ms Seagate ST1239A . .ÖS 11.200,-
FLOPPY-DISK 1.44MB, 3½" (TEAC) incl. Einbauset . . . . . .ÖS 1.056,-
AUFPREIS
Baby-AT-Gehäuse -> Baby Tower Gehäuse . . .ÖS 630,-
AUFPREIS
Baby-AT-Gehäuse -> Big Tower Gehäuse . . . .ÖS 1.200,-
AUFPREIS
386SX/20MHz->386/33 MHz Motherboard, 64kB Cache . . . . . .ÖS 3.990,-

Zahlungskonditionen	: Barzahlung od. Nachnahme
Lieferbedingungen	: Abholung oder kostenpflichtige Zustellung, solange der Vorrat reicht
Preise	: incl. 20% MWST
Garantie	: 12 Monate auf Komplettgeräte, ausgenommen Monitore; Geräte zusammengebaut und überprüft!!
Anschrift	: EXCON, Ing.Günther Hanisch, Rögergasse 6-8, 1090 Wien, 0222/310 99 74 0

✂----------

An den PCCTGM, Postfach 59, 1202 Wien

Ich bestelle lt. Angebot:

_____

_____

_____

Ich bin Schüler der _____(Abt./Jgg.) und nehme das Angebot der kostenlosen Probemitgliedschaft gemäß den Clubstatuten des PCC-TGM für ein Jahr an. Ich wünsche ❑ Abholung/❑ Zustellung und werde mittels Postkarte verständigt (Zutreffendes ankreuzen).

Name_____Plz_____Ort_____

Straße/Nr._____Telefon_____

Datum_____Unterschrift(Erziehungsberecht.)_____

Comp Delphin

Sammelbestellung PCC-TGM

EDV-Zentrale:
Berggasse 5, 1090 WIEN
Tel.: 310-53-56/18 od. 19
Fax.: 319-75-50

EDV-Shop:
Universität Wien, NIG
Universitätsstraße 7
1010 WIEN
Tel.: 40-103/2082

CMP Notebook 20 MB — 29.990,- öS

Prozessor 80386SX
Coprozessorsockel
Taktfrequenz 16 MHz
Hauptspeicher 2 MByte
erweiterbar auf 4 MB
Floppy 3,5", 1,44 MB
Festplatte 20 MB, 22ms
9 Zoll VGA-LCD Display
640x480, 16 Graustufen
Adapter f. ext. Monitor
serielle Schnittstelle
parallele Schnittstelle
PS-2 Adapter für Maus
Tastatur mit 82 Tasten
Akku, Tragtasche
Netzgerät, Handbuch
Maus, Gewicht: 2,8 kg
279x216x51 mm
DR-DOS 5.0

CMP Notebook 40 MB — 34.990,- öS

Technische Daten wie Notebook 20 MB jedoch Festplatte mit 40 MByte Speicher und 19ms Zugriffzeit

Optionen:

zusätzl. Akku	1.290,- öS
Speichererweiterung 2 MB	4.490,- öS

Canon LBP-4 — 14.990,- öS

Einschachtlaserdrucker
Druckgeschindigkeit: 4 Seiten/Min
Druckauflösung: 300x300 dpi
Seitenbeschreibungssprache: CaPSL-III
Druckspeicher: 512 KByte (max. 2,5 MB)
Papiergröße: A4, Letter, Legal
serielle-, parallele- und Video-Schnittstelle
Papierzufuhr: Mehrzweckfach für max. 50 Blatt
Briefumschläge, Folien
2 interne Bitmapped Schriften (Courier, Symbol)
2 interne skalierbare Schriften (Swiss, Dutch)
Softwareemulation von HP-LJ
Netzkabel, Handbuch, Tonerpatrone

Optionen:

Speichererweiterung: 1 MByte	1.990,- öS
Fontkarte SC-1: (7 skal. Schriften)	2.970,- öS

Barzahlung bei Abholung, Preise inkl. Mwst., 12 Monate Garantie, Irrtümer und Druckfehler vorbehalten!

✂----------

An den PCCTGM, Postfach 59, 1202 Wien

Ich bestelle lt. Angebot:

_____

_____

_____

Ich bin Schüler der _____(Abt./Jgg.) und nehme das Angebot der kostenlosen Probemitgliedschaft gemäß den Clubstatuten des PCC-TGM für ein Jahr an. Ich wünsche ☐ telefonisch/☐ per Postkarte verständigt zu werden (Zutreffendes ankreuzen).

Name_____ Plz_____ Ort_____

Straße/Nr._____ Telefon_____

Datum_____ Unterschrift(Erziehungsberecht.)_____

Notizen:

An den
PCCTGM

Postfach 59
1202 Wien

MNUM: _____
NAME: _____
❑ Abholung ❑ Postversand
Ich verfüge über folgende Diskettenlaufwerke:
❑ 5,25" DD ❑ 5,25" HD ❑ 3,5" DD ❑ 3,5" HD

Bitte graue Fläche nicht beschriften!

Nummer	TAB/A5	TAB/A4	DSK	LIT	5-DD	5-HD	3-DD	3HD	L	REK	DOK
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							
	❑	❑	❑	❑							

Datum:	Unterschrift:

STUDENTEN-SERVICE

Studentenkonto

Fürs Stipendium oder das monatliche Taschengeld, für laufende Zahlungen und Eingänge – und nach dem Studium als Privat- oder Gehaltskonto.

Ab sofort mit Versicherungsschutz über 1 Million für Personen- und Sachschäden.

Automatisch zahlen

Regelmäßige Zahlungen, wie Miete, Gas und Strom, Versicherungsbeiträge oder Raten, werden von uns zuverlässig und termingerecht erledigt.

Das erforderliche Startkapital zur Existenzgründung erreicht man am bequemsten mit einem automatischen Sparauftrag – Geld vom Konto aufs Sparbuch.

Karten

Für jeden Zweck die richtige Karte:

- die Kundenkarte zur Information über den Kontostand oder zur Behebung bei den Erste-Geldausgabe-automaten
- die eurocheque-Karte zum bargeldlosen Einkauf oder auf Wunsch mit Bankomatfunktion zur Bargeld-beschaffung rund um die Uhr.

ERSTE-Unisofortkredit

Wenn man einmal knapp bei Kasse ist, kann man ruhig überziehen:

- im ersten Studienabschnitt bis S 10.000,–
- im zweiten bis S 30.000,–

ERSTE-Unikredit

Zur Studienfinanzierung oder für diverse Anschaffungen während des Studiums: bis S 200.000,–

ERSTE-Startkredit

Damit man sich auf den Studienabschluß konzentrieren kann, bietet DIE ERSTE diesen Kredit zu besonders günstigen Zinsen:

- bis zu S 300.000,– mit Pauschalraten-Rückzahlung

oder

- monatliche Auszahlung zwischen S 1.000,– und S 4000,– für 12 oder 18 Monate.

Keine Bearbeitungsgebühr und über Wunsch auch 12 Monate rückzahlungsfrei!

Profitieren Sie von den Erfahrungen unserer Studentenbetreuer

Weiters informieren unsere Studentenbetreuer auch über:

- „Der Erste Österreichische Jugendpreis"
- das CLUB 1-Magazin mit Studentencorner und Clubprogramm
- Steuer- und Wirtschaftsinformationen
- Wertpapierinformationen
- Wirtschaftstelegramm
- Versicherungen, Reisen, Leasing
- Spezialberatung zur Existenzgründung (ob privat oder zum Einstieg ins Berufsleben)

Der Weg zum Selbständig-Sein.

Ein Besuch bei einem Studentenbetreuer in einer ERSTEN-Filiale lohnt sich in jedem Fall!

DIE ERSTE
österreichische Spar-Casse-Bank

Absender:

PCC-TGM
Wexstraße 21
Postfach 59
1202 Wien

P.b.b.
Verlagspostamt
1200 Wien

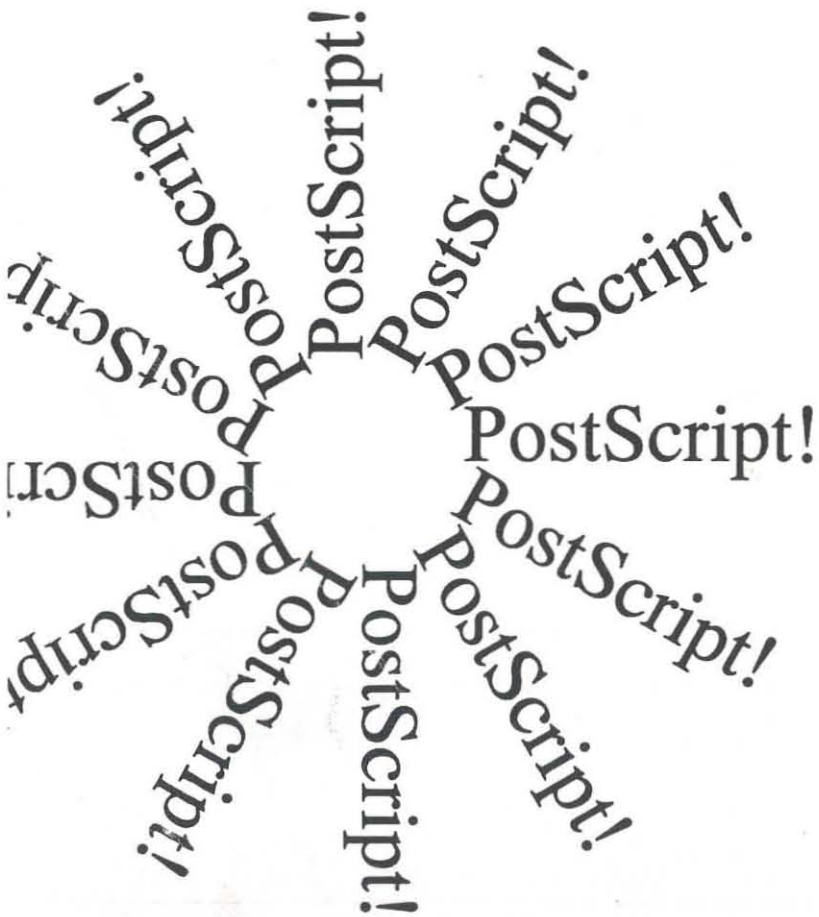
An:

230/DVR:0596299

Mnum: 77
Dipl.-Ing. Franz FIALA
Siccardsburggasse 4/1/22
1100 Wien

Wenn unzustellbar, bitte zurück an Absender.

NEWS24L.PS




```
% rotate-Demo

/Times-Roman-F findfont 30 scalefont setfont
150 190 translate
12 {
        0 0 moveto
        30 rotate
        (     PostScript!) show
} repeat

showpage
```

Inkludiert mit:

```
. Z .NEWS24L.PS;5 cm;5 cm;Postscript
```