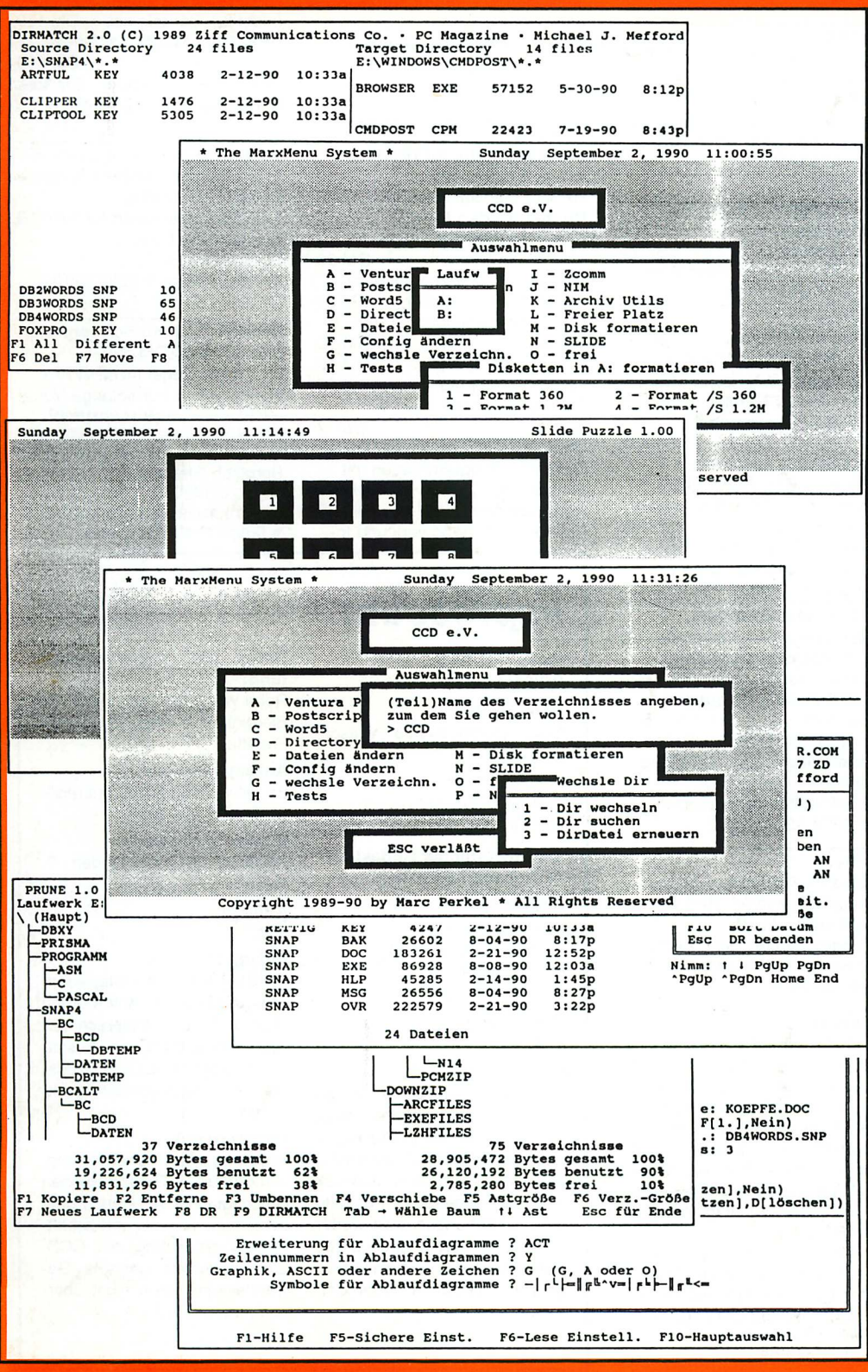
# PRISMA

Computerclub Deutschland e.V. · Postfach 11 04 11 · Schwalbacher Straße 50 · D-6000 Frankfurt am Main 1

Mai /Juni 1990 Nr. 3 D 2856 F




**Atari**
Shareware für den Atari-ST

**Clubnachrichten**
Ergebnisse der Fragebogen-Aktion

**Grundlagen**
C-Programme 4. Teil

**MS-DOS**
Inhaltsverzeichnis der Info's

**Serie 40**
Hyperbelfunktionen in M-Code

Schreibschrittlineal für den DeskJet

Input-Output Board Teil 1

Haushalt & CCD-Modul

**Serie 70**
Molmassen beliebiger Verbindungen

Funktionen zur Basisumwandlung im HP-71 B

**Taschenrechner**
Byteverbrauch in HP48SX-Programmen

Pas de deux - Programme für HP-28S und 48SX

Druck und Display-Utilities

Ergänzung zu "Baustatik mit HP 28S" - Einfeldträger

Objekt Handling
Utilities für den HP28S

Speed

---

Unser Titelbild zeigt diesmal Hardcopies von Bildschirmen aus Shareware- und Public Domain-Programmen der MS-DOS Info-Disketten

# Clubbörse

**Verkaufe** folgende **Bücher:**
HP 71 Basic leicht gemacht, J.H.Horn, 22 DM; Die Programmiersprache FORTH, Zech, 35 DM; In FORTH denken, Leo Brodie, 20 DM; 50 BASIC Programme, A.Romer, 10 DM; BASIC: Alles über PEEK und POKE, Requardt, 10 DM; BASIC: Dateien, Listen und Verzeichnisse, Busch, 10 DM; BASIC für Fortgeschrittene, Sacht, 5 DM.
☎ dienst 0231/188 22 52, privat 0231/20 21 29.

---

Verkaufe
HP-75 Zubehör
RAMerweiterung um 8k DM 190.-
HP-75 Programmsammlung FIGForth 1Kassette eigenes Betriebssystem DM 180.-
Alle Preise sind Endpreise incl. Porto und Verpackung
Peter Habicht, ☎ 06008/7235 ab 19.00 Uhr.

---

**MS-Flugsimulator 3.0 dt,** originalverpackt zu verkaufen.
Dazu Buch "Take Off".
Komplett DM 90.
**Wer hat Software für HP 110?**
A. Tobias, Weyerstraße 227, 5650 Solingen 19,
☎ 0212/33 02 48.

---

**Verkaufe HP-41CX**
HP 41CX (neuestes Modell)
+ Taktfrequenzumschaltung (2x)
+ 2 Extended Memory Module eingebaut
+ 2 Akkusätze mit ext. Ladegerät (W&W) DM 450.-
Magnetkartenleser 82104A
+ 160 Magnetkarten DM 200.-
CCD Modul Version A DM 100.-
IR-Modul 82242A DM 100.-
HP MATH I Module für HP 41CX DM 50.-

Literatur:
W.C. Wickes "Synthetische Programmierung..." + Gerhard Kruse "Optimales Programmieren..." DM 50.-
alles Zusammen: DM 850.-
Klaudius Chlebosch.
☎ 07032/820 23.

---

HP DeskJet plus: Speichererweiterung und Font-Kassetten gesucht. Wer weis, ob man den internen Ram-Speicher vergrößern kann? Thor Gehrmann,
☎ 02339/3963 (abends).

---

**HP 71B Rom-Module** zu verkaufen: Mathematik, Curve Fit, Statistik, Forth/Assembler, jeweils komplett mit Handbuch; Preis VS; Thor Gehrmann, ☎ 02339/3963 (abends).

---

Verkaufe billigst Centronics-Interface für HP-41CV/CX. Für Anschluss ist das HP-IL-Modul erforderlich. ☎ 0711/7412 85.

---

**HP-71B System:**
* HP-71B mit IL-Modul DM 680
* Kartenleser für HP-71B mit Magnetkarten DM 220
* Kassettenlaufwerk mit Netzteil DM 490
* ThinkJet (IL) mit Netzteil DM 480
* PAC Screen, einfach DM 120
* Advanced PAC Screen 2.0 mit Netzteil und Maus DM 390
* 12" Monitor bernstein passend für PAC Screen DM 150
* HP-IL/IB Interface mit Netzteil DM 350
* 2 Paar IL-Kabel 5m je DM 35
* HP41 Translator ROM DM 120
* AC-Circuit Anal. Pac DM 120

Alle IL-Geräte mit original Handbüchern und je 1 IL-Kabel.
Walter Becker, Buschkämpen 41, 2850 Bremerhaven,
☎ 0471/565 53 (abends.

---

**Suche** dringend für HP 41 **Magnetkartenleser**, Preis VS,
☎ 05141/835 13.

---

**HP 71B**
mit FORTH/ASSEMBLER-ROM, Magnetkartenleser mit 45 Magnetkarten, HP-IL-ROM und HP-IL ThinkJet-Drucker. Preis VHB. Telefon-Kontakte ab 18.00 Uhr unter 05439/2881.

---

**HP-41CX**
mit Drucker und Bobby Schenk
Navigationsmodul DM 850
Finanzmodul DM 75
Klaus Broll, Am Aisinger Moos 12, 8200 Rosenheim.

---

Verkaufe
HP-71, IL-ROM, Mathe-ROM, 32 kB RAM, 128 kB RAM, HP-9114A + W&W-Netzteil, Grabau GR7 + Philips-Monitor, Software (LEX etc.), nur komplett DM 4000;
Casio FP-200 Handheld, 32kRam, CETL-ROM, Vierfarbplotter FP-1011PL, Cassettenrecorder, Netzteil DM 350.
Dennis Föh, Hermann-Hanker-Straße 17, 3400 Göttingen,
☎ 0551/79 28 57.

---

**HP 2932 Matrixprinter**
(132 Colom): DM 1000.- VB
**HP 32 Pocket Calculator**
(Mathematisch): DM 50.-
**HP 27S Pocket Calculator**
(Scientific): DM 50.-
**HP 41CV** mit Akku: DM 150.-
Gysbert Hagemann, Alter Weg 1, D-6653 Blieskastel 2,
☎ 06842/2805, geschäftszeitlich: 06842/3041/3042.

---

**Suche:**
Petroleum Fluids Pac - Modul
Finanzmathematik - Modul
Real Estate - Modul
Reinhard Perner, Beim Andreasbrunnen 8, 2000 Hamburg 20,
☎ 040/48 29 85.

---

**Verkaufe:**
**HP-41CX** mit CCD- (Version A) und X-Memory-Modul, 128 kB-Speichererweiterung im Kartenlesergehäuse (mit DAVID-Assembler), Thermodrucker, Bücher: Emery/M-Code, Wickes, Jarett, Dearing, Assembler-Handbuch, alle Handbücher. Alles gut erhalten und funktionsfähig. Komplett DM 980.
**HP-41C** mit Quad-, X-Functions- und X-Memory-Modul, Magnetkartenleser, Anwendungshandbücher. Komplett DM 250.
Dr. Armin Schmidt,
☎ 02174/414 08.

---

HP-Plotter 7475 A mit IB-Interface zu verkaufen.
Detlef Mainx, 7317 Wendlingen a.N., ☎ 07024/3759.

---

Biete an:
HP-71B VB DM 300
HP82163B Video Interface DM 190
Bernd Podbielski,
☎ 0571/247 73.

---

**Verkaufe:**
HP-41CX DM 300.-
(kaum benutzt, da Zweitgerät)
W&W CCD-Modul DM 120.-
HP Mathe-Modul DM 30.-
Jede Menge Literatur zum HP-41.
Patrick Wolter, Von-Görschen-Straße 3, 5100 Aachen,
☎ 9241/53 26 44.,
Fax 0241/51 11 21.

---

Verkaufe
**Softwaremodul für HP-41:**
HP-41 Advantage Pac DM 40
HP-41 Financial Pac DM 40
**Bücher für den HP-41:**
Grafik mit dem HP-41 (NP 68 DM 30
Plotten und Drucken auf dem HP-41 Thermodrucker DM 20
Programmsammlung für den HP-41 DM 20
HP-41 in der Praxis DM 15
HP-41-Sammlung DM 15
Dienstprogramme (Tool-Kit) für den HP-41 DM 7
HP-41 Hilfen u. Anwend. DM 15
Optimales Programmieren mit dem HP-41 DM 20
Anwenderhandb. HP-41 DM 20
Der HP-41 in Handwerk und Industrie DM 15
Best of PRISMA DM 7
Henke Heinz, Burgauerstraße 30, 8900 Augsburg,
☎ 0821/40 29 71.

---

Verkaufe:
**HP-41CV** + X-Function DM 250.-
Kartenleser + Akku + Netzteil, evtl. mit Magnetkarten DM 150.-
Sharp PC-E500, 1/2 Jahr alt DM 400.-;
alle Geräte originalverpackt mit Handbüchern. F. Alex, ☎ 02374/120 03 bis 16.00 Uhr.

---

**Biete an für 41er:**
HP82143A Thermodrucker (nicht IL), Batterie neu DM 150
HP82104A Magnetkartenleser DM 150
HP82160A IL-Module DM 150.
B. Podbielski, ☎ 0571/247 73.

---

**Verkaufe** Kassettenlaufwerk HP-IL, 16 Bänder - davon 1 Software Development-Utility.
1 Tastatur Markquart für HP-71B incl. Beschreibung.
ThinkJet für HP-IL.
Jürgen Losem, ☎ 02223/213 82.

---

**Hewlett-Packard Sonderpreis!**
Z.B.:HP-48SXDM 599.-
HP-Laserjet IIIDM 4777.-
Sonderaktion nur solange Vorrat reicht, daher **sofort anrufen!**
W&W Software Products GmbH, Odenthaler Straße 214, 5060 Bergisch Gladbach 2,
☎ 02202/420 01, oder Ammerthalstraße 12, 8011 Kirchheim,
☎ 089/904 40 76.

---


## *Impressum*

**Titel:**
PRISMA

**Herausgeber:**
CCD - Computerclub Deutschland e.V.

Postfach 11 04 11
Schwalbacher Straße 50
6000 Frankfurt 1

**Verantwortlicher Redakteur:**
Alf-Norman Tietze (ant)

**Redaktion:**
Klaus Kaiser (kk)
Michael Krocker (mik)
Martin Meyer (mm)
Dieter Wolf (dw)

**Herstellung:**
CCD e.V.

**Manuskripte:**
Manuskripte werden gerne von der Redaktion angenommen. Honorare werden in der Regel nicht gezahlt. Die Zustimmung des Verfassers zum Abdruck wird vorausgesetzt. Für alle Veröffentlichungen wird weder durch den Verein noch durch seine Mitglieder eine irgendwie geartete Garantie übernommen.

**Druck und Weiterverarbeitung:**
Reha Werkstatt Rödelheim
Biedenkopfer Weg 40 a, 6000 Frankfurt

**Anzeigenpreise:**
Es gilt unsere Anzeigenpreisliste 3 vom Juni 1987

**Erscheinungsweise:**
PRISMA erscheint jeden 2. Monat

**Auflage:**
2500

**Bezug:**
PRISMA wird allen Mitgliedern des CCD ohne Anforderung übersandt. Ein Anspruch auf eine Mindestzahl von Ausgaben besteht nicht. Der Bezugspreis ist im Mitgliedsbeitrag enthalten.

# Inhalt

**Atari**
Shareware für den Atari-ST 4
**Clubnachrichten**
Ergebnisse der Fragebogen-Aktion 5
**Grundlagen**
C-Programme 4. Teil 7
**MS-DOS**
Inhaltsverzeichnis der Info's 21
**Serie 40**
Hyperbelfunktionen in M-Code 15
Schreibschrittlineal für den DeskJet 19
Input-Output Board Teil 1 29
ANUMDEL-Problem 31
Haushalt & CCD-Modul 40
**Serie 70**
Molmassen beliebiger Verbindungen 38
Hilferuf - spezielle HP-71 Tastenzuordnung 39
Funktionen zur Basisumwandlung im HP-71 B 39
**Taschenrechner**
Zusatz zur Byte Tabelle S. 27 20
Byteverbrauch in HP48SX-Programmen 27
Pas de deux - Programme für HP-28S und 48SX 25
Programmierkurs HP-28? 31
Druck und Display-Utilities 32
Ergänzung zu "Baustatik mit HP 28S" Einfeldträger 34
Anmerkungen zu ROOT und Vorgabe bei INPUT (HP48SX) 34
Objekt Handling Utilities für den HP28S 35
Speed 37
**Sonstiges**
Shareware 44
Clubbörse 2
Impressum 2
**Barcodes** 42
**Serviceleistungen** 43

# Sommerpause ?!

Das langersehnte und erwartete PRISMA ist da. Nur nicht von der Heftaufschrift "Mai/Juni 1990 Nr. 3" irritieren lassen. Diese Angabe dient hauptsächlich und korrekterweise dem Postzeitungsdienst. Wir wissen, daß bei Erscheinen dieser Ausgabe bereits September ist, und wir wissen, daß wir mit diesem Heft viel zu spät sind. Wir wissen auch, daß wir uns sehr beeilen müssen, damit wir in diesem Jahr noch unsere sechs Ausgaben erreichen. Und wir werden es schaffen.

Gerüchte sind zu dementieren, daß der Verzug ursächlich auf den Aufenthalt unseres Bundeskanzlers Helmut Kohl am Wolfgangsee oder auf die Golfkrise zurückzuführen sei. Die Gründe für diesen Verzug sind schnell genannt: Bei mehreren Mitarbeitern des Redaktionsteams waren zur gleichen Zeit, jedoch völlig unabhängig voneinander, persönliche Situationen eingetreten, die die nebenberuflichen Tätigkeiten für PRISMA massiv dämpften. Michael Krocker hat einen mehrmonatigen Auslandsaufenthalt zu Studienzwecken hinter sich. Martin Meyer hat sich verlobt und außerdem seine Wohnungssuche erfolgreich mit einem Umzug abgeschlossen. Und bei mir fielen in den betrachteten Zeitraum meine Diplomarbeit, Umzug, Hochzeit und anschließend meine Diplomprüfung. Nein, das war kein Streß, das war nur etwas viel auf einmal, aber sonst hätte man ja auch nichts zu erzählen. Nach dieser "verlängerten" Sommerpause sind jetzt alle wieder frisch bei der Arbeit.

Ein anderes Thema: DDR-Bibliotheken. Unsere Aktion, die Universitäts- und Fachhochschulbilbliotheken auf Wunsch in den PRISMA-Verteiler aufzunehmen, ist auf starke Resonanz gestoßen. Weit über 30 Bibliotheken haben positiv auf unser Angebot geantwortet. Das entspricht einem Rücklauf von über 75 % der angeschriebenen Einrichtungen. Im Gegensatz dazu sind die Anmeldungen von Einzelpersonen aus dem Gebiet der DDR eher als spärlich zu bezeichnen.

**ADCC - Konkurrenz am Horizont ?**

Neu im Getummel der Computer-Wissbegierigen ist der ADCC (Allgemeiner Deutscher Computerclub e.V.) in Berlin entstanden. Eine gewisse Ähnlichkeit zur Abkürzung ADAC des Allgemeinen Deutschen Automobilclubs ist wohl beabsichtigt. Schwerpunkt des neuen Computerclubs sind Taschencomputer und mobile Datensysteme. Alle zwei Monate erscheint die vom ADCC herausgegebene Zeitschrift "Pocket + Laptop Computer".

Der ADCC wird von einer EDV-Beratungsfirma in Berlin, der Fischel GmbH, getragen. Bereits auf den allerersten Blick ist "Pocket + Laptop Computer" von der Aufmachung her ein kommerzielles Reklameblättchen für die Fischel GmbH und andere Inserenten. Die Gestaltung des Heftes ist für die herausgebende EDV-Firma ein Armutszeugnis - oder ist es gerade typisch für solche, daß unbedingt 18 verschiedene Schriften von riesengroß bis klitzeklein im Quer- und Hochformat eng zusammengedrängt und unübersichtlich auf eine Seite gequetscht werden, um unbedingt vermeindlich alle "tollen" Computermöglichkeiten zu demonstrieren.

Auch inhaltlich kommt nicht viel "rüber". Bei fast allem, was irgendwie interessant ist, endet man bei einer deutlichen Bestellinformation für Bücher oder Produkte der Fischel GmbH, die das "Wesentliche" - sofern vorhanden - beinhalten sollen. Aus Anwenderkreisen von Sharp und Casio Taschencomputern verlautet, daß "Pocket + Laptop Computer" die direkte Fortführung der eingestellten Zeitschrift "Alles für Sharp Computer" aus selbem Hause sei. Dem geneigten Taschencomputeranwender sei ein Probelesen der genannten Zeitschrift dringend empfohlen, um sich eine eigene Meinung zu bilden. Für Anwender von HP Taschencomputern ist das besagte Reklameblatt indiskutabel.

Wir sind konkurrenzlos - und zugegebener Maßen auch konkurrenzlos unpünktlich. Happy Programming!

Alf-Norman Tietze(Chefredakteur)

# Shareware für den Atari-ST

## 3. Teil W. Müller

Diesmal erfolgt die Beschreibung des, seit Dezember 1989 als Shareware erhältlichen, Turbo-Assemblers (mit Debugger) zum Preis von 50.- DM. Dieser Assembler wurde bis Ende März 1990 von der Firma OMIKRON unter dem Namen OMIKRON-Assembler vertrieben. Nun ist dieser Assembler ausschließlich bei Sigma-Soft, Birkhahnkamp 38, 2000 Nordstedt 1 oder als Shareware auf Disketten erhältlich. Das Programm wird ausschließlich von Sigma-Soft weiter entwickelt. Die Firma OMIKRON hat sich von diesem Produkt zurückgezogen, liefert aber weiterhin Support für registrierte Benutzer ihres Assemblers.

Für den Betrag von 50.-

- wird man als Benutzer dieses Programms registriert
- erhält man die neueste Anleitung (zur Zeit 245 Seiten, sehr gut geschrieben, mit Inhaltsverzeichnis, Stichwortverzeichnis und vielen Abbildungen (leider fehlt ein Überblick über die Befehle des 68 000 Prozessors)
- bekommt man die neueste Version der Programme mit persönlicher Seriennummer
- das erste Update ist kostenlos
- Wenn man irgend einen Fehler findet und ihn genau beschrieben an Sigma-Soft schickt, so erhält man "gleich" eine fehlerfreie Version zurück

Zunächst etwas zu den inneren Werten des Assemblers:
Der Assembler verfügt über einen eingebauten Editor, der sehr komfortabel zu bedienen ist. Der Quellcode wird bei der Eingabe direkt in "Token" verwandelt, d.h. daß einzelne Maschinensprachen-Befehle als einzelne Bytes verschlüsselt gespeichert werden.

Für die Bildschirmdarstellung wird der Quellcode dann jedesmal entschlüsselt. Dieser Vorgang läuft jedoch so schnell ab, daß der Benutzer nichts davon merkt (es sei denn, man lädt große Quelldateien als ASCII Text ein). Das interne Format führt dazu, daß eine Assemblierung eines Quelltextes sehr schnell erfolgt. Im Mittel werden 1 Millionen Zeilen in einer Minute assembliert. In der Praxis führt diese Geschwindigkeit dazu, daß der Quellcode auf Knopfdruck fertig assembliert vorliegt und anschließend in den verschiedenen Formaten abgespeichert werden kann.

Zur Verfügung stehen Formate wie normale Programmdateien der Endungen PRG, TOS, ACC, auf Wunsch mit sogenanntem FAST-BIT für das neue Atari Betriebssystem TOS1.4 (bei gesetztem Bit wird vor dem Progammstart nicht mehr der gesamte RAM-Bereich gelöscht), DATA-Zeilen zum Einfügen des Programms in BASIC-Programmen, als Inlinecode für OMIKRON.BASIC oder GFA-BASIC, SMALL-DRI für das Linken des Programmes zu C Programmen und andere. Wer über das nötige Verständnis verfügt, kann mit Hilfe der in der Anleitung gegebenen Dokumentation eigene Formate definieren und zu dem Programm hinzufügen.

Der Assembler verfügt natürlich über eine Vielzahl von Fehlermeldungen und Warnungen, über einen eingebauten Taschenrechner, über eine eingebaute ASCII-Tabelle sowie über Druckroutinen, die es erlauben einen Block oder den gesamten Quelltext zu drucken.

Funktionen wie Suchen und Ersetzen können entweder gezielt in Symbolen oder global im Text ausgeführt werden und schließlich können alle möglichen Parameter im Editor eingestellt und abgespeichert werden.

Zusätzlich zu den Assemblerbefehlen des 68 000 Prozessors verfügt der Assembler über eine Vielzahl von Pseudooperatoren (kurz: Pseudoops). Quellcodes von anderen Assemblerprogrammen für den Atari-ST können, solange sie keine Makros verwenden, eingelesen und umgewandelt werden. Zeilen, die nicht umgewandelt werden können, werden als Kommentare ausgeklammert und können dann anschließend von Hand nachbearbeitet werden.

Zusätzlich zum Assembler gibt es den "Sourcelevel" Debugger. Dieses Programm kann vor dem Start des Assemblers gestartet und ständig im Speicher gehalten werden. Mit Hilfe dieses Debuggers ist es möglich, einzelne Befehle des Assemblerprogramms ablaufen zu lassen und per Knopfdruck zwischen Quellcode und ausführendem Programm hin und her zu springen. Zusätzlich zu diesem komfortablem Zusammenspiel zwischen Assembler und Debugger kann der Debugger auch allgemein für verschiedenste "Spionagedienste" im Atari verwendet werden.

Beliebige Programme können geladen und disassembliert, einzelne Speicherbereiche können untersucht oder Diskettensektoren geladen, modifiziert und wieder auf Diskette geschrieben werden. Es ist selbstverständlich, daß einige dieser Funktionen, ohne genaue Kenntnisse angewandt, das größte Unheil anrichten können!.

Ausgestattet mit der Möglichkeit alle Schritte z.B. auf dem Drucker oder in einer Datei zu protokollieren, erlaubt es dem Benutzer zumindest im nachhinein zu sehen, was er dort eigentlich angestellt hat.

Auf der Diskette zum Assembler/Debugger wird eine Vielzahl von Programmbeispielen mitgeliefert. So gibt es ein Spiel Namens THINK_-WORK, bei dem Kisten in einem Labyrinth von einer Stelle zur anderen verschoben werden müssen, oder eine resetfeste, autobootfähige RAMDISK, deren Inhalt in einer Datei abgelegt werden und beim Start automatisch in die RAMDISK gelegt werden kann und vieles mehr. Durch das Studieren dieser Quelltexte kann man schon eine Menge über den Umgang mit dem Assembler lernen. Trotz der umfangreichen Anleitung kommt man jedoch um den Kauf eines Lehrbuchs über die Funktion der 68 000er Befehle nicht herum, will man selbständig Programme erstellen.

Wer benötigt überhaupt einen Assembler auf dem Atari-ST ?

In der Tat können die meisten Anwendungen, insbesondere größere Programme auf dem Atari mit Hilfe einer Hochsprache wie C, PASCAL oder BASIC viel schneller geschrieben werden als in Assembler. Der Assembler bietet sich für den Anfänger für kleine Hilfsprogramme an, die z.B. nur einzelne Bits der Systemvariablen verändern. Ein BASIC Programm würde eine Länge von 3-5 kBytes benötigen, daß entsprechende Assembler-Programm würde mit 300 Bytes auskommen.

Auch für die Progammierung zeitkritischer Routinen, sollen z.B. Messwerte über eine Schnittstelle erfasst werden, bietet sich der Assembler an.

Meine ursprüngliche Idee, eine Kurzanleitung für beide Programme zu erstellen habe ich, angesichts der vielen eingebauten Befehle in Assembler und Debugger, fallen gelassen. Im Debugger kann durch die Eingabe des Wortes HELP eine Übersicht der Befehle angefordert werden. Zur Not könnt Ihr eine Referenzkarte über den OMIKRON-ASSEMBLER, erschienen in der Zeitschrift 68 000er im Heft 11 und 12/1989, zu Rate ziehen. Das Originalhandbuch, erhältlich bei der Firma SIGMA-SOFT (siehe oben), ist hierdurch natürlich nicht zu ersetzen.

Bevor man sich an das Abenteuer Assemblerprogrammierung heranwagt sollte man sich sehr genau überlegen, ob für die Lösung eines Problems eine Hochsprache geeigneter ist, zumal dann zu-

mindest der Hauch einer Chance besteht, daß das Produkt auf andere Rechnersysteme übertragbar bleibt.

Bei kommerziellen Softwareprodukten wird immer häufiger eine Mischung von Hochsprache und Assemblerprogrammierung verwendet. Häufig benutzte und zeitaufwendige Progammteile werden in Assembler, der Rest in einer Hochsprache geschrieben. Es macht wenig Sinn, eine Routine, die vielleicht ein oder zweimal während eines Programmlaufs verwendet wird, mit einem enormen Zeitaufwand in Assembler zu optimieren.

Das Sharewareprogammpaket Turbo-Assembler/Debugger ist eine hervorragende Programmierleistung auf dem Atari-ST Rechner. Dieses Programm sollte man sich, unabhängig von einer späteren Nutzung, einfach mal ansehen. Um zu sehen, wie schnell ein Assembler assemblieren kann, oder wie schnell in Assembler programmierte Programme auf dem Atari-ST ablaufen können , oder wie praktisch die autobootfähige Ramdisk ist, oder wie .....,

Ihr findet das Programm zum Anschauen auf der CCD-Diskette Nr. 25.

Obwohl das Programm eine spezielle Option für Programmtester, die für einschlägige Computerzeitschriften arbeiten, eingebaut hat (das Abspeichern des Bildschirminhalts auf Knopfdruck in eine Graphikdatei, macht schon die Hälfte eines Artikels... ) habe ich diesmal auf eine solche graphische Aufwertung dieses Artikels verzichtet (wäre auch viel zu einfach).

Viele Grüße aus Köln
Werner Müller,
E-Mail MBK1:W.MUELLER

# Ergebnisse der CCD Fragebogen-Aktion

Die Mitgliederversammlung 1989 hatte auf Vorschlag des Vorstandes und Beirates beschlossen, die Mitglieder nach dem Einsatz von, sowie dem Interesse für Hard- und Software zu befragen.

Von Beiratsmitglied Peter Kemmerling wurden insgesamt 166 Fragebögen ausgewertet. Die Ergebnisse dieser Befragung liegen nun vor und sind in der umseitigen Tabelle in einer dem Fragebogen ähnlichen Form abgedruckt.

Bei allen Zahlenangaben (in Prozent, aufgerundet) ist grundsätzlich zu beachten, daß sich diese auf insgesamt 166 Rücksendungen beziehen - von allen denjenigen jedenfalls, die daran interessiert sind, daß ihre Angaben von Vorstand und PRISMA-Redaktion berücksichtigt werden sollen. Eine höhere Anzahl wäre von der Aussagekraft für die Aufteilung der Anwendungen und Interessen in unseren Club sicher von Vorteil gewesen. Naja, vielleicht wird die Beteiligung an einer zukünftigen Befragung ansteigen.

Bei den Fragen 1 - 3 ist zu beachten, daß sowohl "Beruflich" als auch "Privat" gleichzeitig angegeben werden durfte, während "Interesse" nur als Einzelangabe zulässig war. Daher gibt für die tatsächliche Nutzung der jeweilige Maximalwert Aufschluß. Auch Hardware, Programmiersprachen und Software konnten mehrfach angegeben werden.

Bei der Frage zur Nutzung von Hardware ergibt sich folgende Rangfolge:

54 % HP-41C/CX/CV
41 % PC's (MS-DOS)
25 % HP-71
15 % Sonstige HP-Taschenrechner
10 % Atari
7 % CP/M Rechner
4 % HP-75
4 % HP Serie 80
4 % Sonstige Computer
2 % Amiga
1 % Macintosh

Das Interesse an Hardware von denjenigen, die diese zur Zeit weder beruflich noch privat anwenden, verteilt sich wie folgt:

11 % PC's (MS-DOS)
9 % Sonstige HP-Taschenrechner
8 % Atari
4 % HP-41C/CX/CV
4 % HP-71
4 % Macintosh
2 % Amiga
2 % Sonstige Computer
1 % HP-75
1 % HP Serie 80
1 % CP/M Rechner

Unter "Sonstige HP-Taschenrechner" sind die Modelle 10, 11, 12 und 16 sowie 17, 18, 19, 21, 22, 27, und 28 zu verstehen. Den HP48SX gab es zur Zeit der Fragebogenaktion noch nicht. In der jüngsten Zeit ist allerdings ein reges Interesse am 48'er deutlich geworden, was sich auch im PRISMA wiederspiegelt.

Bei den Programmiersprachen im Bereich der HP-Taschencomputer fällt auf, daß nicht alle HP-41 Anwender (54 %) den 41'er auch programmieren (45 %), während alle HP-71 Anwender dies zumindest in BASIC tun. Das Ergebnis bei den PC's zeigt ebenfalls, daß ein Teil der Anwender auch ohne eigene Programmierung mit den Geräten arbeitet. Jedoch ist der Anteil der Programmierer deutlich größer als es dem üblichen Erfahrungswert aus der PC Praxis entspricht.

Das Interesse an Programmiersprachen fällt größtenteils auf diejenigen Sprachen, die tatsächlich nur wenig genutzt werden. Bei den HP-Taschencomputern sind das HP-41 M-Code sowie HP-70 Forth und Assembler (Lexfiles). Eine Ausnahme stellt HP-41 mit Synthetik dar, weil diese sogar von 30 % als genutzt angegeben wurde. Bei den PC's sind es die Programmiersprachen C, Lisp, Modula und Prolog, die durch mehr Interesse als mit tatsächlicher Anwendung auffallen. Aber auch Assembler, BASIC und FORTRAN erzeugen die Neugier der PC-Anwender. Die Programmierfreudigkeit ist ein Hinweis darauf, daß CCD-Mitglieder über tiefergehende Kenntnisse als der durchschnittliche Computeranwender verfügen.

Die Anwendungssoftware verteilt sich wie folgt:

53 % Textverarbeitung
45 % Datenverwaltung/Datenbanken
27 % Planung/Kalkulation
18 % Datenkommunikation/-fernübertragung
18 % Sonstiges
17 % Spiele
15 % CAD und CIM
13 % Steuerung und Meßtechnik
10 % Lernsoftware
7 % Buchhaltung, Rechnungs- u. Lagerwesen

Steuerung und Meßtechnik, Datenfernübertragung (Telex, Mailbox, externe Datenbanken), CAD und CIM sowie Datenbanken weisen das mit Abstand deutlichste Interesse bei Nichtanwendern auf.

Die Struktur der Berufsgruppen innerhalb der Einsender stellt sich wie folgt dar:

50 % Angestellte
20 % Studenten
13 % Selbständige/Freiberufler
10 % Beamte
8 % Rentner/Pensionäre
1 % Sonstige

Fast 78 % der Einsender sind zwischen 26 und 55 Jahre alt, wobei die Verteilung allerdings zweigipfelig ist. Die stärkste Altersgruppe ist "26-35 Jahre" (35,5 %) gefolgt von "46-55 Jahre" (24,7 %) und "36-45 Jahre" (17,5 %).

Ganz nebenbei hat unsere Umfrage die Verteilung der Anwendergruppen beantwortet:

66,9 % HP- und PC-Anwender
22,9 % Nur HP-Anwender
10,2 % Nur-PC-Anwender

Voila - wir sind auf dem richtigen Kurs.

Alf-Norman Tietze
(2. Vorsitzender)

Kürzel:

B = Beruflich
P = Privat und Fortbildung
I = Interesse / Informationbedarf

In den Rubriken 1 - 3 waren mehrfache Nennungen möglich. Wenn die Spalte I angekreuzt wurde, sollte kein Kreuz in den Spalten B und/oder P stehen. Die *kursiven* Zahlenangaben sind die aufgerundeten Prozentanteile von insgesamt 166 Fragebögen.

**1. Welche Harware nutzen bzw. interessiert Sie?**

| B | P | I | |
|---|---|---|---|
| *42* | *54* | *4* | HP-41C/CV/CX |
| *16* | *25* | *4* | HP-71 |
| *5* | *4* | *1* | HP-75 |
| *5* | *4* | *1* | HP-85/86/87 |
| *15* | *15* | *9* | Sonstige HP-Taschenrechner |
| *4* | *7* | *1* | CP/M Rechner |
| *47* | *41* | *11* | MS-DOS Rechner (PC) |
| *5* | *10* | *8* | Atari |
| - | *2* | *2* | Amiga |
| *2* | *1* | *4* | Macintosh |
| *9* | *4* | *2* | Sonstige |

**2. Welche Programmiersprachen nutzen bzw. interessieren Sie?**

| B | P | I | |
|---|---|---|---|
| *36* | *45* | *4* | HP-41 Standard |
| *10* | *30* | *11* | HP-41 mit Synthetik |
| *1* | *5* | *15* | HP-41 M-Code |
| *16* | *25* | *4* | HP-70 Basic |
| *2* | *7* | *10* | HP-70 Forth |
| *1* | *7* | *10* | HP-70 Assembler |
| *4* | *3* | *1* | Sonstige HP-Sprachen |
| *10* | *15* | *8* | Assembler |
| *28* | *33* | *7* | Basic |
| *11* | *15* | *19* | C |
| *4* | *2* | *1* | Cobol |
| *11* | *7* | *7* | Fortran |
| *1* | *1* | *6* | Lisp |
| *1* | *1* | *6* | Modula |
| *12* | *26* | *9* | Pascal |
| *1* | *2* | *7* | Prolog |
| *8* | *11* | *4* | Sonstige |

**5. Welcher Altersgruppe gehören Sie an?**

| | |
|---|---|
| - | unter 18 |
| *6,0* | 18 bis 25 |
| *35,5* | 26 bis 35 |
| *17,5* | 36 bis 45 |
| *24,7* | 46 bis 55 |
| *9,6* | 56 bis 65 |
| *6,6* | über 65 |

**3. Welche Software nutzen bzw. interessiert Sie?**

| B | P | I | |
|---|---|---|---|
| *48* | *53* | *8* | Textverarbeitung |
| *42* | *45* | *11* | Datenverwaltung/Datenbanken |
| *7* | *6* | *5* | Buchhaltung, Rechnungs- und Lagerwesen |
| *27* | *14* | *5* | Planung, Kalkulation |
| *11* | *18* | *13* | Datenkommunikation, Datenfernübertragung |
| *13* | *7* | *18* | Steuerung und Messtechnik |
| *15* | *8* | *13* | CAD und CIM |
| *3* | *10* | *8* | Lernsoftware |
| - | *17* | *5* | Spiele |
| *18* | *14* | *7* | Sonstiges |

**4. Welcher Berufsgruppe gehören Sie an?**

Kürzel

T = technisch/wissenschaftlich
K = kaufmännisch/Dienstleistung
F = Fabrikation/Industrie

| T | K | F | |
|---|---|---|---|
| - | - | - | Schüler |
| *19* | *1* | - | Studenten |
| - | - | - | Auszubildende |
| *39* | *8* | *3* | Angestellte |
| *7* | *1* | *2* | Beamte (bitte sinngemäß ankreuzen: T=Planung, K=Verwaltung; F=Versorgung [z.b. Stadtwerke]) |
| *11* | *2* | - | Selbstständige/Freiberufler |
| *7* | *1* | - | Rentner/Pensionär |
| - | - | *1* | Sonstige |

**Verteilung der Anwendergruppen**

| | |
|---|---|
| *22,9* | Nur-HP-Anweder |
| *10,2* | Nur-PC-Anwender |
| *66,9* | HP- und PC-Anwender |

**Nächstes**

# Ortstreffen in Köln

## 22. September 1990,

ab 13 Uhr im Vereinshaus der Humboldt-Siedlung, Frankfurter Straße 855, 5000 Köln 91 (Ostheim)

Verantwortlich und Ansprechpartner:

Dr. Werner Müller,
Schallstraße 6,
5000 Köln 41,
Tel. (0221) 40 23 55

und

Dipl.-Ing. Jürgen Schramm,
Frankfurter Straße 853,
5000 Köln 91,
Tel (0221) 8 90 23 54.

# C-Programme - ein Überblick

## 4. Teil

### Vorgehen bei größeren Programmierprojekten

Ehe wir auf das eigentliche Thema dieser Folge eingehen, seien zuvor noch ganz kurz einige C-Spezialitäten behandelt, deren Kenntnis hilfreich, aber im ersten Anlauf nicht erforderlich ist:

**4.1 einige Spezialitäten**

- Die bedingte Bewertung:

Statt der Anweisungsfolge, die das Maximum zweier Zahlen liefern soll

```
if(i > j)
  k= i;
else
  k= j;
```

kann mit Hilfe des Operatorpaars ?: auch geschrieben werden:

```
k= (i > j) ? i : j;
```

oder allgemein gesprochen: exp1 ? exp2 : exp3 . Zunächst wird exp1 bewertet. Ist er logisch wahr, also ungleich 0, wird exp2 bewertet und dessen Wert ist dann das Ergebnis der bedingten Bewertung. Andernfalls ist exp3 das Resultat.

Derartige kompakte Formulierungen trifft man oft in professionellen Bibliotheken. Ob man der orthodoxen Formulierung mit if-else oder ?: den Vorzug gibt, ist eine Frage der Übersichtlichkeit.

- Die bitmanipulierenden Operatoren

Damit bewegen wir uns praktisch auf Assembler- Ebene. Folgende Operatoren sind naturgemäß nicht für die Datentypen float und double geeignet:

| | |
|---|---|
| & | die "und" Verknüpfung von Bits |
| \| | die "oder" Verknüpfung von Bits |
| ^ | die "exklusive oder" Verknüpfung von Bits |
| << | Bit-Shift links |
| >> | Bit- Shift rechts |
| ~ | Bit Komplement |

Ohne Kenntnisse der Assembler-Programmierung sind diese Operatoren wenig griffig. Daher zuvor eine m.E. verständliche Andeutung anhand des Shift-links <<, sodann ein Programm-Beispiel.

Bit-Shiften links heißt: alle Bits eines Datums um eine definierte Anzahl nach links schieben, von rechts mit 0 auffüllen. Wozu braucht man das ? Die Zahl 5 als int-Wert hat die Bit-Darstellung

```
0000 0000 0000 0101  =
1*2**2 + 1*2**0 = 5
```

Mit

```
int i= 5;
i << 2;
```

entsteht folgendes Bitmuster:

```
0000 0000 0001 0100  =
1*2**4 + 1*2**2 = 20
```

Wir haben also unserer Zahl auf kürzestem Weg mit 4 multipliziert, denn

```
i << 1  ist i*2
i << 2  ist i*4
i << 3  ist i*8
```

Jeder Prozessor kann so etwas von Haus aus. Eine derartige Multiplikation (oder Division mit >>) mit Zweierpotenzen ist unendlich schneller gegenüber echten Prozessor-Multiplikationsroutinen (Beispiel b48.c / Listing 1).

Um den Effekt echt zu sehen, ist ein C-Compiler hilfreich, der ein Assembler-Listing erzeugen kann, z.B. MS-C mit Switch /Fa. Die Assembler-Source zeigt in der Tat, daß für k= i <<j der Intel 80X86-Befehl SHL (Shift Left) eingesetzt wird. Aber auch für den Ausdruck ip *= 2 wird der SHL-Befehl genutzt.

Dagegen wird nach Bestimmung von ipow der gegenüber SHL nun zwangsläufig derviel langsamere IMUL-Befehl in der Zeile m= i * ipow(j) angezogen. Beim 8086 benötigt SHL 8+(4*j), also für j=3 dann 20 Prozessortakte, für IMUL aber bis zu 154 Takten. Die Integer-Divison des 8086 IDIV braucht bis zu 184 Takten. Also, wenn es auf höchste Speed ankommt und wenn ohne Krampf möglich: Statt Division Multiplikation und statt Multiplikation Shift-Operation.

Der Bit-OR-Operator arbeitet wie folgt: Wenn Bit in erster Zahl gesetzt oder Bit in zweiter Zahl gesetzt, dann Bit im Resultat setzen:

Sei i= 5 und j= 10. Dann ist i | j = k= 15, denn

```
i =  0000 0000 0000 0101
j =  0000 0000 0000 1010
k =  0000 0000 0000 1111
```

Diese Eigenschaft wird benutzt, um bestimmte Bits in einer Variablen zu setzen. Typisch für Presentation Manager Programme:

```
static ULONG flFrameFlags=
FCF_TITLEBAR | FCF_SIZEBORDER
| FCF_MINMAX | FCF_SYSMENU
| FCF_MENU ;
```

Die Werte (Bitmuster) der FCF-Konstanten und der Datentyp ULONG sind in os2.h definiert. Die OR-Verknüpfung hier stellt sicher, daß das Frame-Window eine Überschriftsleiste, einen Rahmen zum Kantenverschieben, ein Min-Max-Symbol usw. hat. Mit anderen Worten: Setze das TITLEBAR-Bit,das SIZEBORDER-Bit, das MINMAX-Bit usw. in der Konstanten flFrameFlags.

Die anderen Operatoren können ähnlichen Zwecken dienen.

**4.2 Vorgehen bei größeren Projekten**

Die folgenden Erläuterungen gelten sinngemäß für alle Programmiersprachen. Ein größeres Programmierprojekt ist dadurch gekennzeichnet, daß es mehrere Sourcen umfaßt, die später gelinkt werden. Rein theoretisch ist der Fall schon gegeben, wenn das fertige Programm aus dem obligatorischen Hauptprogramm main und mindestens einer eigenen Function besteht. Wir wählen ein Beispiel aus einer früheren Folge:

**4.2.1 Eine einzige Compilerunit**
(Beispiel b49.c / Listing 2)

Hier haben wir den Fall einer sog. Compiler-Unit vorliegen, denn die Source umfaßt mehrere Functions (main ist auch eine Function). Prinzipiell kann eine Compiler-Unit ansich beliebig viele Function beinhalten, die Grenzen werden durch die Leistungsfähigkeit des Compilers gezogen. Es sind ohne weiteres Sourcen von Tausenden von Zeilen denkbar, wobei viele C-Compiler allerdings dann aussteigen aufgrund interner Compiler-Limits. Es gibt allerdings auch Compiler wie z.B. Lahey-FORTRAN, die problemlos Sourcen von 30.000 Zeilen in einem Zug bearbeiten (das ist bei älteren Sourcen, als strukturierte Programmierung noch nicht in Mode war, oft gegeben).

Heute strebt man an, daß eine C-Function, eine Pascal-Procedure, ein FORTRAN-FUNCTION-Unterprogramm oder eine FORTRAN-SUBROUTINE normalerweise nicht mehr als ca. 200 Zeilen aufweisen sollen. Dies als Anhaltswert.

Aber auch Programmeinheiten (vornehmlich Hauptprogramme) können mitunter über 1000 Zeilen aufweisen, ohne unübersichtlich zu werden. Gerade bei sehr modernen Programmen, wie z.B. Presentation-Manager-Programmen, ist es sinnvoll, die eigentliche Steuereinheit, die alle Messages interpretiert, nicht auseinanderzureißen.

Wie schon praktiziert, können wir die Compilereinheit b49.c in einem Zug übersetzen und linken: 1. Möglichkeit.

**4.2.2 getrennte Compilation von Hand**

Als zweite Möglichkeit werden wir b49.c in das Hauptprogramm b50.c und die Function b51.c zerlegen, womit sofort ersichtlich ist, daß Compilerunits völlig andere Namen als ihre eigentlichen internen Function-Namen haben können: (Beispiel b50.c / Listing 3, Beispiel b51.c / Listing 4).

Listing 1: Beispiel b48.c

```
#include <stdio.h>

int ipow(int j);

int main(void)
{
int i,j,k,m;
printf("\Gib i und j:");
scanf("%d %d",&i,&j);

k= i << j;
m= i * ipow(j);

printf("\nShift k= %d, Multip k= %d\n",k,m);
return (0);
}

int ipow(int j)
{
int i,ip;
ip= 1;

for(i= 0; i  j; i++)
   ip *= 2;

return (ip);
}
```

Listing 2: Beispiel b49.c

```
/*****************************************
* Eine Compiler- Unit, bestehend aus
* - main
* - fmul2
*****************************************/

/*---------------------------------------
*  include & define
*---------------------------------------*/
#include <stdio.h>

#define MAXELE 16000L

/*---------------------------------------
*  externe Deklarationen
*---------------------------------------*/
long nektor[MAXELE];
long max= MAXELE;

/*---------------------------------------
*  Function- Deklarationen
*---------------------------------------*/
void fmul2(void);

/*****************************************
*  Hauptprogramm main
*****************************************/
int main(void)
{
extern long nektor[];
extern long max;

long i;

for(i= 0L; i < max; i++) nektor[i]= i;

fmul2();

printf("\ndas groesste Element nektor[%ld]=
        %ld", max-1L,nektor[max-1L]);
return 0;
}

/*****************************************
*  Function fmul2
*****************************************/
void fmul2(void)
{
extern long nektor[];
extern long max;
long i;

for(i= 1L; i < max; i++) nektor[i] *= 2L;

return;
}
```

Listing 3: Beispiel b50.c

```
/*****************************************
* Compilerunit b50.c, enthaelt:
* - main
*****************************************/

/*---------------------------------------
*  include & define
*---------------------------------------*/

#include <stdio.h>

#define MAXELE 16000L

/*---------------------------------------
*  externe Deklarationen
*---------------------------------------*/
long nektor[MAXELE];
long max= MAXELE;

/*---------------------------------------
*  Function- Deklarationen
*---------------------------------------*/
void fmul2(void);

/*****************************************
*  Hauptprogramm main
*****************************************/
int main(void)
{
extern long nektor[];
extern long max;

long i;

for(i= 0L; i < max; i++) nektor[i]= i;

fmul2();

printf("\ndas groesste Element nektor[%ld]=
%ld",
      max-1L,nektor[max-1L]);
return 0;
}
```

Listing 4: Beispiel b51.c

```
/*****************************************
*  Compilerunit b51.c, enthaelt:
*  - fmul2
*****************************************/
void fmul2(void)
{
extern long nektor[];
extern long max;

long i;

for(i= 1L; i < max; i++) nektor[i] *= 2L;

return;
}
```

Es werden die Sourcen b50.c und b51.c getrennt übersetzt:

cl /c /Od /AL /FPi87 /W3 b50.c
cl /c /Od /AL /FPi87 /W3 b51.c

und dann gelinkt:

link b50 + b51;

Dies wären die Aufrufe für den MS-C Compiler und den MS-Linker. Bei anderen Compilern sieht das sinngemäß ähnlich aus. Die Compiler- Switches besagen:

/c :nur compilieren, aber nicht linken (das muß hier sein)

/Od :Optimizer ausschalten (kann entfallen)

/AL :Large Memory Modell (kann entfallen, wenn bei Installation das Small-Memory-Modell angeben wurde)

/FPi87:InLine-Code für 80x87 erzeugen (kann entfallen)

/W3 :höchster Warning-Level (sollte nicht entfallen)

Es kann also prinzipiell auch nur mit cl /c b50.c bzw. cl /c b51.c übersetzt werden.

Beim Linken ist es so, daß alle Objektcodes mit + verbunden werden, aber eine Trennung nur durch Leerzeichen geht auch:

link b50 b51;

Das Semikolon am Ende unterdrückt die Abfragen des Linkers nach Namen von EXE-File, MAP-File, den Libraries und dem DEF-File. So erhält das EXE-File automatisch den Namen des ersten Objektfiles, d.h. B50.EXE. Alternativ kann man aufrufen:

link b50 b51 (ohne Semikolon)

und antwortet mit CR oder setzt bei der Abfrage "Run File" z.B. HARRY ein.

Dann heißt der Lademodul, bestehend aus b50 und b51 dann HARRY.EXE.

Man kann aber gemischt vorgehen: Angenommen, b51 wäre bereits übersetzt, b50.c hätte eine Änderung erlebt. Dann können mehrere Compiler folgendes tun: cl b50.c b51.obj bzw. cl /AL /FPi87 b50.c b51.obj

Damit wird zunächst b50.c übersetzt und dann sofort mit b51.obj gelinkt. Für UNIX System V wäre der äquivalente Aufruf: cc b50.c b51.c, würde Lademodul a.out erzeugen bzw.
cc /ob50 b50.c b51.o, würde Lademodul b50 erzeugen (bei UNIX gibt es keine Extension wie .EXE)

### 4.2.3 Compilieren mit Include-Files

Es liegen mehrere getrennte Sourcen vor, die aber als eine Einheit compiliert werden sollen: Listing 5.
Hier fügt der C-Präprozessor die Source b51.c hinter dem Hauptprogramm ein, wodurch die Source b52.c sozusagen temporär beide Sourcen enthält. Diese Möglichkeit ist zunächst mit 4.2.1 vergleichbar, ist aber durch die getrennten Sourcen flexibler. Das Vorgehen ist bei manchen Compilern für Testzwecke in der Programmentwicklung sehr sinnvoll, da so der Compiler alle Sourcen quasi gleichzeitig sieht und feststellen kann, ob z.B. Datentypen konsistent verwendet werden und Parameterübergaben korrekt erfolgen. Rein theoretisch können beliebig viele Sourcen mit #include eingehängt werden, aber bei größeren Projekten steigen so die Compilierzeiten stark an und etliche Compiler laufen aus ihren Grenzen.

Das einzig sinnvolle Vorgehen bei größeren Projekten (ab 1000 Zeilen und 4 Functions aufwärts) ist

### 4.2.4 Das Compilieren mit Make

Größere Projekte haben es an sich, daß bei Änderungen in der einen Source oft gleichzeitig Änderungen in anderen Sourcen erfolgen müssen. Gerade Anfänger wollen es nicht glauben, aber dies alles von Hand nachzuziehen, wie in 4.2.2 erläutert, klappt in der Praxis kaum. Man hat z.B. im Hauptprogramm einen Übergabeparameter verändert, dies auch in der angezogenen Function editiert, aber (totsicher irgendwann) vergessen, die Function neu zu übersetzen. Chaos perfekt. Also muß man diesen Vorgang automatisieren. Und genau das kann man mit Make. Dies ist ein UNIX-Tool, das inzwischen bei allen guten Compilern, egal welcher Sprache, Eingang gefunden hat. Die Idee ist folgende: Make vergleicht Zeit und Datum von Files und leitet dann Aktionen ein. Für unser Beispiel sähe ein Makefile wie folgt aus (wir gehen von b50 und b51 aus):
Listing 6

Wir starten Make in diesem Fall mit
make b50.mak
wobei man dieses Kommando auch in ein .BAT oder .CMD-File stellen kann. Was passiert ? Links stehen die Outfiles b50.obj, b51.obj und b50.exe. Sie hängen von den rechts stehenden Infiles ab. Hat nun z.B. b50.c ein neueres Datum oder eine neuere Zeit wie b50.obj, dann wurden Änderungen in b50.c vorgenommen und b50.obj hat einen älteren Stand. Dann greift die Abhängigkeit und b50 wird neu übersetzt. Es können beliebig viele Aktionen nach einer Abhängigkeitszeile ausgeführt werden, Listing 7.

Wie man an der dritten Abhängigkeitszeile erkennt, kann ein Outfile von mehreren anderen Infiles abhängen, also b50.exe von b50.obj und b51.obj.

Es ist natürlich bei einem so einfachen Beispiel noch nicht nötig, mit Make zu arbeiten. Hier soll nur das Prinzip gezeigt werden. Unersetzlich wird Make in der Tat, wenn ein Projekt aus vielen Files gebildet wird. So können Make-Files, die man gern .mak nennt (aber nicht muß), bei Programmen für GEM, Windows oder gar den Presentation- Manager ohne weiteres mehrere Seiten Umfang annehmen. Ein Presentation-Manager Programm für OS/2 wird nicht nur aus den eigentlichen C-Sourcen gebildet, sondern zusätzlich aus .H, .RC, .RES, .DEF, .ICO und .PTR-Files. Es ist unmöglich, hier sicher ohne Make zu arbeiten.

Make ist tatsächlich eine der wichtigsten Utilities und man sollte sich angewöhnen, generell mit getrennten Sourcen zu arbeiten und diese von Make automatisch auf dem neusten Stand halten zu lassen. Es ist nun nicht nötig, für jede einzelne Function eine Extra-Quelle vorzuhalten. Bei großen Projekten geht man z.B. so vor: Listing 8.
Dazu mögen das File P100.H (für eigene Variable), P100.RC (Resource-File), P100.RES (compiliertes Resource-File) und P100.DEF (Definition-File für Linker) definiert sein. Dann könnte P100.MAK in seiner einfachsten Form wie folgt aussehen, wobei cp.cmd eine Batch-Datei für C-Compiler ist (z.B cl /c /Od /AL /FPi87 /W3 %1.c) : Listing 9.

Hier haben wir den Fall, daß zwei Abhängigkeitszeilen für das gleiche File, hier P100.EXE, vorliegen. Im ersten Fall wird neu gelinkt und das Resource-File angehängt, im zweiten Fall wird nur das Resource-File angebunden. Die Möglichkeiten von Make gehen weit über das Gesagte hinaus und C-Programmierer, aber auch die Fans von FORTRAN 77, BASIC oder Pascal, sollten Make kennen und praktisch immer ohne Ausnahme einsetzen.

Make wird von UNIX selbst genutzt: Der Betriebssystem-Kern von UNIX, Kernel genannt, ist als Lademodul zwischen 500 und 800 KByte groß und enthält auch intern maßgebliche Informationen z.B. für Anzahl gleichzeitig zu mountender Partitions, Speicherlimits und dgl. Werden nun solche grundlegenden Informationen verändert, was in bestimmten Definitionsdateien wie /etc/master erfolgt, dann muß der Kernel "rebuilded", d.h. neu compiliert werden. Dazu wechselt man in das Directory /usr/sys und sagt einfach: make. Das Make der UNIX-Systeme sucht nach einem Makefile namens Makefile bzw. makefile, wenn nicht explizit ein Name wie oben (P100.MAK) angegeben ist. In dem Directory /usr/sys ist ein Makefile gegeben, der Kernel wird neu übersetzt. Auch daher ist jedem UNIX-System der C-Compiler cc beigefügt. Sodann benennt man den nun alten Kernel unix in z.B. unix.old um und kopiert den eben generierten Kernel unix.std in unix um. Nach einem Warmstart greift dann der neue Kernel. Achtung Hacker und Cracker: Das alles darf nur der Superuser ausführen.

Wenn also UNIX selbst sich Make bedient, kann es nicht übel sein. Immer einsetzen! Übrigens ist das Make des Compilerherstellers A nicht an den Com-

Listing 5: Beispiel b52.c

```
/*****************************************
* Compilerunit b52.c, enthaelt:
* - main als Source b52.c
* - fmul2 als Source b51.c, mit include
*****************************************/

/*---------------------------------------
*  include & define
*---------------------------------------*/
#include <stdio.h>

#define MAXELE 16000L

/*---------------------------------------
*  externe Deklarationen
*---------------------------------------*/
long nektor[MAXELE];
long max= MAXELE;

/*---------------------------------------
*  Function- Deklarationen
*---------------------------------------*/
void fmul2(void);

/*****************************************
*  Hauptprogramm main
*****************************************/
int main(void)
{
extern long nektor[];
extern long max;

long i;

for(i= 0L; i < max; i++) nektor[i]= i;

fmul2();

printf("\ndas groesste Element nektor[%ld]=
%ld",
       max-1L,nektor[max-1L]);
return 0;
}

#include "b51.c"
```

Listing 6 : b50.mak

```
# erste Zeile Abhängigkeit
b50.obj:    b50.c
            c1 /c /AL /FPi87 /W3 b50.c

# zweite Zeile Abhängigkeit

b51.obl:    b51.c
            c1 /c /AL /FPi87 /W3 b51.c

# dritte Zeile Anhängigkeit
b50.exe:    b50.obj  b51.obj
            link b50 + b51;
```

Listing 7:

```
# erste Zeile Abhängigkeit
b50.obj:    b50.c
            c1 /c /AL /FPi87 /W3 b50.c
            copy b51.c  a:

# zweite Zeile Abhängigkeit
b51.obj:    b51.c
            c1 /c /AL /FPi87 /W3 b51.c
            copy b51.c  a:

# dritte Zeile Abhängigkeit
b50.exe:    b50.obj  b51.obj
            link b50 + b51;
```

Listing 8:

```
/*****************************************
*  Compilerunit 1 namens P100.C, enthaelt:
*  - main
*****************************************/

/*****************************************
*  Compilerunit 2 namens P100ST.C, enthaelt:
*  - SteuerDiaProc   (PM- Steuerprogramm)
*  - quit            (Ende- Routine)
*****************************************/

/*****************************************
*  Compilerunit 3 namens P100DY, enthaelt:
*  - dynamic         (Speicher anfordern)
*  - dynfree         (Speicher freigeben)
*****************************************/

/*****************************************
*  Compilerunit 4 namens P100DIA.C, enthaelt:
*  - FileDiaProc     (Window fuer Files)
*  - ColorDiaProc    (Window fuer Farben)
*  - LabelDiaProc    (Window fuer Labels)
*  - QuitDiaProc     (Window fuer Ende)
*  - AboutDiaProc    (Window Programmstart)
*****************************************/

/*****************************************
*  Compilerunit 5 namens P100M1.C, enthaelt:
*  mathematische Routinen
*  - array           (mathematische Funct.)
*  - scaliere        (mathematische Funct.)
*  - cholesky        (mathematische Funct.)
*  - gauss           (mathematische Funct.)
*****************************************/
```

Listing 9:

P100.MAK

```
P100.RES      P100.H P100.RC
              rc -r P100

P100.OBJ:     P100.H P100.C
              cp  P100

P100ST.OBJ:   P100.H P100ST.C
              cp  P100ST

P100DY.OBJ:   P100.H P100DY.C
              cp  P100DY

P100DIA.OBJ:  P100.H P100DIY.C
              cp  P100DIA

P100M1.OBJ:   P100.H P100M1.c
              cp  P100M1

P100.EXE:     P100.OBJ  P100ST.OBJ P100DY.OBJ P100DIA.OBJ\
              P100M1.OBJ  P100.DEF
              link P100 + P100ST + P100DY + P100DIA +P100M1,\
               /align:16, NUL, os2, P100
              rc  P100.RES

P100.EXE:     P100.RES
              rc P100.RES
```

Listing 10: Beispiel b53.c

```
/*********************************
*  function fbegin
*********************************/
#include <stdio.h>

void fbegin(void)
{
printf("\nStart des Programms ..\n");
}
```

piler A gebunden. Man kann ohne weiteres Make von Microsoft (das bei MS-C und MS-FORTRAN mitgeliefert wird) auch für Compiler anderer Hersteller, wenn diese kein Make mitliefern, einsetzen.

Make vergleicht rigoros Zeit- und Datumsstempel. Mitunter wurden Änderungen an einem File vorgenommen, das ein Outfile einer Makedatei ist. Diese Änderungen brauchen aber keine Neuübersetzung, weil z.B. nur Kommentare verändert wurden. Make würde nun trotzdem Aktionen einleiten, obwohl dies von der Sache nicht nötig wäre. Dazu müßte der Zeit- und Datumsstempel der betreffenden Datei verändert werden und zwar derart, daß der Stempel älter ist als der des Objektfiles. Das macht man mit Touch, auch ein UNIX-Tool.

Touch ändert Zeit- und Datumeintrag eines oder mehrerer Files:
Touch [-cfv] [-d mm-dd-yy] [-t hh:mm:ss] [-i file] file file ..

Die Switches c,f,v,i brauchen Erläuterung:

-c: File nicht erzeugen, wenn nicht vorhanden

-f: Schreiben auch bei Schreibschutz erzwingen

-v: Die von Touch bearbeiteten Files anzeigen

-i: Den Stempel dieses Files für die anderen nehmen

Wir wollen ein File namens HARRY.C, das den Eintrag 8.5.90, 8:00:00 Uhr hat, auf den 24.12.90, 12:00:00 setzen:
touch -v -d 12-24-90 -t 12:00:00 HARRY.C

Mit Touch kann man also Make überlisten oder seinem Chef (oder sich selbst) vortäuschen, daß man zu den unüblichsten Zeiten gearbeitet hat..

Touch läßt sich auf alle Filetypen anwenden, also .EXE, .CMD, .BAT, .C, .FOR usw. Der Fileinhalt wird nicht berührt und man kann ganze Gruppen auf einen neuen Zeit/Datumseintrag setzen:
touch -v -d 12-24-90 -t 12:00:00 *.C

Nicht alle Compilerhersteller liefern Touch mit (wäre sonst viel zu einfach), aber auch hier kann selbstverständlich Touch anderer Hersteller genutzt werden. Ob es wirklich immer sinnvoll ist, Make mit Touch nach anscheinend unbedeutenden Änderungen (die oftmals aufgrund von Schreibfehlern mehr kaputt als gut machen) zu überlisten, sei dahingestellt. Touch ist in dieser Beziehung ein typisches UNIX-Tool: UNIX unterstellt immer, daß der Systemnutzer zu jedem Zeitpunkt genau weiß, was er tut !

### 4.3 Weitere Features für größere Projekte

#### 4.3.1 Libraries

Man kann mehrere eigenen Functions in compilierter Form in eine selbstgeschaffenen Bibliothek geben, anstelle sie wie in 4.2 gezeigt, in Compilerunits zu stellen. Wann arbeitet man mit Compilerunits und wann mit eigenen Bibliotheken ? Compilerunits sind leichter zu handhaben, d.h. zu verändern, Bibliotheken haben eher statischen Charakter. Bibliotheken legt man für eigene Funktions an, die ständig in allen möglichen Programmen eingesetzt werden. Vor wenigen Jahren, als die C-Compiler und Compiler aller anderen Sprachen nur die eigentlichen Sprachstandards abdeckten, war es für fortschrittlichere Programmierprojekte erforderlich, sich eigene Bibliotheken anzulegten. Diese enthielten selbstgeschriebene Funktionen für Bildschirmmanipulationen wie Windowmanagement und Grafikfunktionen. Heute werden bei allen namhaften C- und FORTRAN-Compilern derartige Leistungen, die über den Sprachstandard hinausgehen, mitgeliefert oder sie können von Drittanbietern zugekauft werden. Damit haben zumindest im PC-Bereich eigene Bibliotheken nicht mehr den Stellenwert wie früher.

Das Anlegen einer eigenen Bibliothek ist einfach: Man braucht dazu ein Utility, das Bibliotheken anlegen, verwalten, verändern und löschen kann.

Dies ist bei UNIX System V die Utility "ar" und für DOS und OS/2-Systeme gibt es reihenweise solche Librarymanager. Den Microsoft C- und FORTRAN-Compilern wird LIB.EXE beigegeben, bei anderen Compilern ähnlich.

Ein Beispiel mit MS-LIB: Wir möchten eine Bibliothek anlegen, welche die Objektmodule b53, b54 und b55 enthält; die Bibliothek möge OWN.LIB heißen.

Schaffen wir uns dazu die drei primitiven Functions b53, b54 und b55: Listing 10.
Normalerweise würde man b53, b54 und b55 separat an b56 anlinken:
link b56 + b55 + b54 + b53;

Nun geben wir die drei Functions in die Library OWN.LIB, voher müssen sie getrennt compiliert werden, sodaß die Objekts b53.obj, b54.obj und b55.obj vorliegen:
LIB OWN +B53 +B54 +B55;

Fertig. Um nun in einem zweiten Schritt zu sehen, was die Libraray OWN.LIB enthält:
LIB OWN (keine weiteren Angaben, kein Semikolon !)

LIB arbeitet nunmehr interaktiv: Auf die Frage "operations" einfach mit (Enter-Taste) antworten. Bei der nächsten Frage "list file" gibt man einen Dateinamen an, z.B. OWN.LST. Diese Datei OWN.LST enthält dann alle Modulnamen und die Public Symbols.

Wollen wir nun unser Hauptprogramm direkt mit der eigenen Bibliothek linken, dann ist wie folgt vorzugehen:
1) interaktiv:
LINK B56 (ohne Semikolon!)
Antwort auf "Run file" : <CR>
Antwort auf "List file" : <CR>
Antwort auf "Libraries" : OWN <CR>
Antwort a. "Definitions file" : <CR>

2) als Command:
LINK B56,,,OWN;

Beachte die drei Kommata zwischen B56 und OWN ! Stehen für: Run File als Defaultnamen und Listfile als Defaultnamen, mit 1) vergleichen..

Statt mehrerer Objekts haben wir an das Hauptprogramm b56 direkt eine Bibliothek angelinkt, welche die drei Functions enthält.

Wie wir an der Zeile LIB OWN +B53 +B54 +B55; erkennen, bedeutet "+objekt": Dieses Objekt der Library hinzufügen.
Weitere Funktionen:
-objekt : Objekt löschen

-+objekt: Updaten

*objekt : Kopiert ein Objekt aus der Library in ein File namens objekt.OBJ

-*objekt: Löscht ein Objekt aus der Library und stellt es in ein File namens objekt.OBJ

Mit LIB kann man also auch fertige Bibliotheken untersuchen und auseinandernehmen.

#### 4.3.2 Lohnt sich der Debugger-Einsatz?

In einen Debugger lädt man fertige Lademodule, um sie schrittweise abzuarbeiten und dabei die diversen Programmvariablen und ggf. Maschinenregister zu beobachten. Wie der Name Debugger sagt, wird er eingesetzt, um in fehlerhaften Programmen die faulen Stellen aufzuspüren. Manche Compiler werden mit Debugger ausgeliefert, bei anderen wird in Normalpakete und "Profipakete", die natürlich preislich differieren, unterschieden und bei anderen Anbietern wird ein Debugger erst im Lauf der Zeit hinzugefügt.

Manche Autoren, die in Computerzeitschriften Compiler- Reviews vornehmen, feiern regelrecht, daß Compiler XYZ in der Version 4.7.11 nun auch einen Debugger hat. Muß also ungeheuer wichtig sein. Ich werde nun sofort wütendste Kritik der Programmier-Profis (und solcher, die sich dafür halten), einfangen: Wer ein selbstgeschriebenes Programm nur dadurch auf die Reihe bekommt, indem er seine Fehler mit hartem Debugger-Einsatz aufspürt, verdient es nicht besser. Ein gut strukturiertes Programm braucht keinen Debugger ! Die Uralt-Methode, an ganz gezielten, fraglichen Stellen ein printf einzubauen, das dubiose Variable ausgibt, hat einen ganz entscheidenden Vorteil: Sie zwingt dazu, ernsthaft über das Fehlverhalten des Programms nachzudenken (was bei Debugger-Einsatz meist nicht gemacht wird) und wirklich gezielt vorzugehen. Damit habe ich selbst bei Presentation-Manager-Programmen in C oder vieltausendzeiligen

Listing 10: Beispiel b54.c

```
/***********************************
*  function fbell
***********************************/
#include <stdio.h>

void fbell(void)
{
printf("\a");
}
```

b55.c

```
/***********************************
*  function fend
***********************************/
#include <stdio.h>

void fend(void)
{
printf("\nEnde des Programms ..\n");
}
```

Dazu ein kleines Hauptprogramm:
b56.c

```
/***********************************
*  main, ruft
*   - fbegin
*   - fbell
*   - fend
***********************************/
#include <stdio.h>

void fbegin(void);
void fbell(void);
void fend(void);

int main(void)
{
printf("\nEin Test fuer Libraries ..\n");

fbegin();
fbell();
fend();

printf("\nAlles klar ..stop\n");
return (0);
}
```

Listing 11: Beispiel b57.c

```
/*****************************************
*  C- Programm B57.C ruft ein
*  Assembler- Unterprogramm
*  MS-C V 5.1
*****************************************/
#include <stdio.h>

int aygetc(void); /* die Assembler- Routine */

int main(void)
{
int c;

printf("\nGib eine Taste: ");
c= aygetc();

printf("\nDas war die Taste %c\n",c);
return (0);
}
```

Das externe Assemblerprogramm:

```
AYGETC.ASM
;***************************************
; aygetc liefert Tastendruck
; MS- Macro- Assembler V 5.1
;***************************************
.MODEL LARGE
.CODE

PUBLIC _aygetc

_aygetc PROC

  push bp     ; Base- Pointer sichern
  mov bp,sp   ; Stack- Pointer auf Base-Pointer

  mov ah,0    ; Routine fuer Holen Tastendruck
  int 22      ; ROM-BIOS Tastatur- Interrupt

  pop bp      ; Base- Pointer zurueck
  ret         ; Ruecksprung ins C-Programm

_aygetc ENDP
END
```

Listing 12: Beispiel b58.c

```
/*****************************************
*  C- Programm B58.C ruft ein
*  Assembler- Unterprogramm: Cursor setzen
*  Assembler- Unterprogramm: Taste holen
*****************************************/
#include <stdio.h>

/*---------------------------------------
*  Deklarationen
*---------------------------------------*/
int ayscpos(int izeile, int ispalte);
int aygetc(void);

/*---------------------------------------
*  C- Programm
*---------------------------------------*/
int main(void)
{
int c,izeile,ispalte;

printf("\nGib Zeile und Spalte: ");
scanf("%d %d",&izeile,&ispalte);

ayscpos(izeile,ispalte);
c= aygetc(); /* auf Taste warten */

return (0);
}
```

und neu dazu:

```
ayscpos.asm
;***************************************
; ayscpos setzt Cursor auf izeile,ispalte
; MS- Macro- Assembler V 5.1
;***************************************
.MODEL LARGE
.CODE

PUBLIC _ayscpos

_ayscpos PROC

  push bp     ; Base-Pointer sichern
  mov bp,sp   ; Stack-Pointer auf Base-Pointer
  push di     ; zur Sicherheit
  push si     ; zur Sicherheit

  mov dh, [bp+6] ; izeile vom Stack, in dh
  mov dl, [bp+8] ; ispalte vom Stack, in dl
  mov bh,0       ; Bildschirmseite 0

  mov ah,2    ; Routine fuer Setzen Cursor
  int 16      ; ROM-BIOS Bildschirm- Interrupt

  pop si      ; si restaurieren
  pop di      ; di restaurieren
  pop bp      ; Base- Pointer zurueck
  ret         ; Ruecksprung ins C-Programm

_ayscpos ENDP
END
```

F77-Programmen nie einen Debugger gebraucht.

Bei den Debuggern sind eigentlich drei Typen zu unterscheiden:

- Primitiv- Debugger, die nur auf Objektebene arbeiten, wie DEBUG, der Teil von MS-DOS ist. Nur für Assembler-Programmierer und Cracker sinnvoll.
- Debugger auf Quellcodeebene. Es wird am Bildschirm schrittweise das C-Programm (oder andere Sprache) angezeigt, mit einfachen Tastenbefehlen werden Breakpoints gesetzt, Variable verändert und angezeigt usw. Meist relativ kompakt und einfach zu lernen. Typische Beispiele: PROBE der Propero-Compiler (Pascal, F77 und C) und SOLD für Lahey-F77-Compiler.
- Hochkomfortable Debugger auf Window-Basis. Der bekannteste Vertreter: MS-CodeView, der Teil des Lieferumfangs der MS-Compiler ist. Brauchen meist große Teile des Kernspeichers für sich selbst (was bei großen zu debuggenden Programmen empfindlich stören kann), die Bedienung ist durch die sehr vielfältigen Möglichkeiten regelrecht zu erlernen.

Als C-Beginner sollte man zunächst auf Debugger-Einsatz verzichten, sonst kämpft man mit dem Debugger und nicht mit dem eigentlichen Problem. Später sind, wenn überhaupt, einfache Debugger auf Quellcodebasis ausreichend. Nur wer Assembler-Kenntnisse hat und parallel zum Sourcecode den Maschinencode betrachten möchte, kann hochkomfortable Window-Debugger gekonnt einsetzen.

### 4.3.3 Mischen verschiedener Sprachen

Es ist ohne weiteres möglich, Objekts, die aus verschiedenen Programmiersprachen hervorgingen, zusammenzulinken. Das funktioniert immer dann, wenn die Objektformate im gleichen Schema aufgebaut sind. Beispiele:
Mischbar sind untereinander: MS-C, MS-FORTRAN, MS-Pascal, MS-BASIC, MASM (MS-Macro-Assembler). Andere Mischformen: Lahey- FORTRAN, MASM, Turbo-C, MS-C. Auf Atari- Seite: Prospero-FORTRAN, Prospero-C, Prospero-Pascal.

Wer mit wem verträglich ist, ist in den jeweiligen Compilerhandbüchern aufgelistet. Wozu ist das gut ? Es ist rein theoretisch sinnvoll (und wird in der Literatur gern als Beispiel gebracht), C mit seinen Stärken auf der Low-Level-Seite und FORTRAN mit seiner Rechenpower zu einem Lademodul zu vereinen, der die Stärken beider Sprachen nutzt. Andere Ansätze sind z.B. bereits vorhandene Pascal-Module, die nun in einem C-Hauptprogramm gerufen werden sollen. Die Beispiele und Ansätze sind Legion. Von der Sache her kein Problem: Dem Linker ist es völlig egal, ob Objekt A aus C entstanden ist, Modul B aus FORTRAN und Modul C aus MASM. Was theoretisch so ideal klingt, kann in der Praxis sehr schwierig werden. Zunächst ist eines ganz klar:
Wer z.B. C und FORTRAN mischen möchte, muß ein exzellenter Kenner beider Sprachen sein, übliche Durchschnittskenntnisse reichen keinesfalls.

Das hängt damit zusammen, daß manche Datentypen nicht kompatibel sind, die Parameterübergaben anders gehandled werden, Speichermodelle unterschiedlich sind und mehr. Kurz: Wer sich auf so etwas einläßt, der muß sich warm anziehen.

Es gibt eine aus meiner Sicht sinnvolle Kombination: Hochsprache ruft Assembler-Unterprogramm. Es gibt eine Reihe von Aktionen, die von DOS und dem eine Ebene tiefer liegenden BIOS sehr elegant ausgeführt werden können, die direkt in der Hochsprache schwierig zu handhaben sind. Man nutzt sozusagen die Hochsprache als Träger für die eigentlich interessanten Assemblerroutinen. Das ist unendlich leichter, als ein Programm vollständig in Assembler zu schreiben. Auch die Assemblerkenntnisse müssen dann nicht tiefgehend sein. Auf Seite der Hochsprache ist nur zu wissen, wie sie die Parameter auf den Stack schiebt und wie der Rücksprung gehandled wird.

Wir demonstrieren dies an einem sehr einfachen Beispiel: Ein C-Programm B57.c möge eine Assembler-Unterroutine rufen, die den Wert einer Taste zurückliefert (das geht natürlich auch direkt in C mit getch). Das Assemblerprogramm heiße aygetc.asm: Listing 11.
Es wurde angenommen, daß das C-Programm als Large- Memory- Modell vorliegt; es wurde mit
cl /c /AL /W3 b57.c
compiliert. Das Assembler- Unterprogramm wurde mit
masm /MX aygetc;
assembliert. Das Linken geht wie üblich vor sich: link b57 + aygetc;

Obwohl das Assemblerprogramm extrem einfach ist, zeigt es den generellen Aufbau un die Vorgehensweise: Die Deklaration .MODEL LARGE besagt, daß das rufende C-Programm Large- Memory ist (sonst stünde da .MODEL SMALL oder ein anderes Speichermodell). Das ist wichtig für die Größe der Rücksprungadresse, hier 4 Bytes. Weiter haben immer C-Functionen intern für den Linker einen führenden Unterstrich, daher PUBLIC _aygetc. Bei _aygetc PROC (PROC für Procedure) beginnt das Programm. Die Sequenz

```
push bp      ; Base-Pointer sichern
mov bp,sp ; Stack-Pointer auf Base-Pointer
...
...
pop bp    ; Base- Pointer zurueck
ret       ; Ruecksprung ins C-Programm
```

ist generell vorzusehen, wenn sie auch hier nicht einsichtig ist. Der Basepointer, auch Framepointer genannt, wird benutzt, um auf den Stack zuzugreifen. Auch das rufende C-Programm wird ihn benutzen. Also ist er beim Eintritt in eine Routine zu sichern, weil er möglicherweise in der Routine verändert wird (z.B. durch irgendwelche DOS- und ROM-BIOS-Interrupts, was man auf den ersten Blick nicht sieht). Außerdem sehen wir gleich, daß wir im Assembler-Unterprogramm ebenfalls einen Basepointer brauchen, wenn wir Werte vom Stack holen wollen. Wie wird der erkannte Tastendruck zurückgeliefert ? Der ROM-BIOS-Interrupt 22 schiebt diesen Wert in das Register AX. Da C-Functions beim Typ int immer den Function-Rückgabewert im AX-Register erwarten, ist hier die Wert-Lieferung quasi vollautomatisch erfolgt.

Als zweites Beispiel werden wir den Cursor an eine definierte Bildschirmstelle setzen: Listing 12.
Hier ist zunächst als Sicherheitsmaßnahme hinzugekommen, daß die Register SI und DI gesichert werden, da diese beiden Register vom Optimizer des C-Compiler oft genutzt werden. Sichern und späteres Restaurieren kann nicht schaden ! Mit [bp + offset] werden Werte vom Stack geholt, sodaß hier ein eigener Basepointer erforderlich ist, der zu Beginn auf die Position des Stackpointers gesetzt wird. Der ROM-BIOS-Interrupt 16 des IBM-PC ist der Video-Interrupt allgemein. Steht bei seinem Start im AH- Register der Wert 2, dann bedeutet das Cursor setzen, wobei die Cursor-Position im Register DX (genauer in den beiden Halbregistern DH und DL) erwartet wird und BH enthält die Bildschimrseite. Um die Maßnahme [bp + offset] zu verstehen, sei der Stack betrachtet:

```
-------------------------------
Argument 2 (2 Byte)        <---- BP + 8
-------------------------------
Argument 1 (2 Byte)        <---- BP + 6
-------------------------------
Rücksprung- Adresse
  (4 Byte)                 <---- BP + 4
-------------------------------
gesicherter BP (2 Byte)    <---- BP
-------------------------------
```

Dieser Stackaufbau gilt für MS-C mit Modell Large (Rücksprung 4 Bytes). Beim Small- Model wäre der Rücksprung 2 Bytes, mithin stünden die Werte bei [bp + 4] und [bp + 6]. Bei anderen Sprachen sieht das ganz anders aus: Bei MS-FORTRAN stünde das erste Argument bei [bp + 8], das zweite Argument bei [bp + 6], also vertauscht. Lahey-FORTRAN dagegen hat den gleichen Aufbau wie MS-C. Eine Assembler-Unterroutine, die von einer Hochsprache gerufen wird, muß also genau an den jeweiligen Hochsprachen-Compiler angepaßt sein.

Dazu sind die entsprechenden Compiler-

handbücher zu konsultieren. Der Linker dagegen ist unkritischer: Bei PCs funktioniert fast immer MS-LINK für fast alle Compiler-Hersteller. Manche Linker, besonders extrem schnelle Linker (keine Namen!), können dagegen Probleme beim Mischen verschiedener Sprachen bereiten. Handbücher! Für den Atari ST gilt Sinngemäßes.

Wie ersichtlich, sind solche Assembler-Unterprogramme kurz und gar nicht schwierig. Sie können aber die Leistung eines Programms ungemein steigern.

Mit obigem Vorgehen ist angedeutet, wie man auf DOS- Interrupts und ROM-BIOS-Interrupts des PCs zugreift. Damit hat der C-Programmierer Zugang zum Innersten eines PC oder Atari. Eine gute Übersicht für die Interrupts des IBM-PC gibt Peter Norton in "Programmierhandbuch für den IBM-PC",Vieweg 1986.

Mit dieser Folge ist unsere vierteilige Serie über C abgeschlossen. Die für den Einsteiger maßgeblichen Features wurden angesprochen. Die Voraussetzungen für einen guten C-Start sind ein guter C-Compiler, komplett mit allen Handbüchern (Achtung Raubkopierer !) und ein Computer mit Festplattenlaufwerk.

Wer als Betriebssystem UNIX oder OS/2 hat, tut sich besonders leicht (wegen der Abstürze am Anfang ..). Eine ganz gute Alternative können Pocket-Computer wie der CASIO PB 2000 C sein, die einen eingebauten C-Interpreter haben und preiswert sind. Natürlich ist das Spektrum der C-Funktionen hier beschränkt.

Um C richtig zu lernen, braucht man drei Dinge: Übung, Übung und abermals Übung.

Wir wünschen einen guten Start !

Dr.- Ing. Frank Rieg
2002
Darmstadt

# Hyperbelfunktionen in MCode

von Klaus Huppertz

**Die Funktionen sind:**

SINH/COSH;

TANH: benutzt COSH/SINH;

SINHC: benutzt ADCHK, SINH/COSH, MODW;

COSHC: benutzt ADCHK, SINH/COSH, SINHC, MODW;

TANHC/COTHC: benutzt RADL, SINH/COSH, MODW;

ARSINH/ARCOSH: benutzt ARCOTH;

ARCOTH;

ARIANHC: benutzt NEN, RAD2, ARIANH, MODW

weil die Gemeinsamkeiten groß sind noch die komplexen trigonometrischen Funktionen:

SINC: benutzt ADCHK, SINH, COSH, MODW;

COSC: benutzt ADCHK, SINH, SINC;

TANC: benutzt ADCHK, SINH, COSH, MODW.

Die reellen Funktionen benutzen die Register X und L; in L steht nach dem Aufruf das Argument. Die komplexen Funktionen benutzen die Register X, Y und L: in L steht nach dem Aufruf ein Zwischenergebnis der Rechnung. Die übrigen Stackregister werden nicht verändert. Alle Funktionen sind unabhängig vom eingestellten Winkelmodus. Wo es nötig ist stellt das Programm auf Radiant um und nach der Rechnung den vorherigen Winkelmodus wieder ein. Die meisten benutzten Formeln stehen im Bartsch Seite 190. Die komplexen trigonometrischen Funktionen habe ich dem MATH1B-ROM abgeschaut.

Da die Mathefunktionen im Betriebssystemlisting praktisch nicht dokumentiert sind, gebe ich noch folgende Hinweise:

Die Funktionen "AD2-10" (1807), "MP2-10" (184D) und "DV2-10" (1898) erwarten den ersten Operanden (Dividend) in CPU-A und den zweiten (Divisor) in CPU-C. Bei OVERFLOW wird das Exponentenvorzeichen (Nybble2) auf 1 gesetzt; es erfolgt keine Fehlermeldung.

Die Exponentialfunktion "EXP10" (1A0A) erwartet wie alle Funktionen mit einem Argument den Wert in CPU-C. Gleichzeitig müssen die CPU-Flags entsprechend gesetzt werden. Die sicherste Methode ist mit ST=0(3C4) alle Flags zurücksetzen. Aus der Funktion "LN20" (1B45) läßt sich über CPU-Flag 5 die Basis einstellen: Flag 5 gesetzt = LOG, Flag 5 gelöscht = LN. "SQR10" (18BE) erfordert nur das Argument in CPU-C ebenso wie "ON/X10" (188B;1/X).

Alle trigonometrischen Funktionen erwarten das Argument auch in CPU-N, das sind: "SIN" (1288), "COS" (127C), "TAN" (1282), "ASIN" (1098), "ACOS" (107D) und "ATAN" (10AA). "CHK$S" (14D8) prüft das C-Register auf ALPHA DATA. "CHK$S1" (14D4) prüft C- und A-Register auf ALPHA DATA. Beide Routinen stellen auch den Dezimalmodus ein, der für alle Mathefunktionen notwendig ist.

Das Unterprogramm "RAD1/2" (919E/F) macht die Funktionen unabhängig vom eingestellten Winkelmodus. Es stellt Radiant ein und speichert den vorher eingestellten. "MODW" (91AE) stellt den alten Zustand wieder her. Zu beachten ist, daß das Programm lageabhängig ist aufgrund der "PORT DEP: XQ xxxx"'s.

Die wichtigen Adressen sind im Programm unterstrichen und die entsprechenden Aufrufe markiert. Sie müssen entsprechend der neuen Lage des Programms geändert werden!

```
9155 088 "H"
9156 00E "N"
9157 009 "I"
9158 013 "S"
9159 248 SETF 9
915A 033 JNC 9160 +06
915B 088 "H"
915C 013 "S"
915D 00F "O"
915E 003 "C"
915F 244 CLRF 9
9160 104 CLRF 8
9161 3C4 ST=0
9162 0F8 READ 3(X)
9163 361
9164 050 ?NCXQ 14D8 CHK$S
9165 029
9166 068 ?NCXQ 1A0A EXP10
9167 070 N=C ALL
9168 2EE ?C#0 ALL
9169 22D
916A 061 ?CXQ 188B ON/X10
916B 24C ?FSET 9
916C 013 JNC 916E +02
916D 2BE C=-C-1 MS
916E 0AE A<>C ALL
916F 0B0 C=N ALL
9170 01D
9171 060 ?NCXQ 1807 AD2-10
9172 0AE A<>C ALL
9173 04E C=0 ALL
9174 35C R= 12
9175 090 LD@R 2
9176 261
9177 060 ?NCXQ 1898 DV2-10
9178 10E A=C ALL
9179 276 C=C-1 XS
917A 276 C=C-1 XS
917B 289
917C 003 ?CGO 00A2 ERROF
917D 10C ?FSET 8
917E 360 ?C RTN
917F 0F8 READ 3(X)
9180 128 WRIT 4(L)
9181 0AE A<>C ALL
9182 0E8 WRIT 3(X)
9183 3E0 RTN
9184 088 "H"
9185 00E "N"
9186 001 "A"
9187 014 "T"
9188 260 SETHEX
9189 379
918A 03C PORT DEP:
918B 15F XQ 915F COSH
918C 0F8 READ 3(X)
918D 2BE C=-C-1 MS
918E 268 WRIT 9(Q)
918F 138 READ 4(L)
91CB 161 XQ 9161 COSH(2)
91CC 0B8 READ 2(Y)
91CD 135
91CE 060 ?NCXQ 184D MP2-10
91CF 0A8 WRIT 2(Y)
91D0 248 SETF 9
91D1 260 SETHEX
91D2 379
91D3 03C PORT DEP:
91D4 161 XQ 9161 SINH(2)
91D5 138 READ 4(L)
91D6 135
91D7 060 ?NCXQ 184D MP2-10
91D8 0E8 WRIT 3(X)
91D9 2AB JNC 91AE -2B
91DA 083 "C"
91DB 008 "H"
91DC 013 "S"
91DD 00F "O"
91DE 003 "C"
91DF 260 SETHEX
91E0 379
91E1 03C PORT DEP:
91E2 199 XQ 9199 ADCHK
91E3 1F1
91E4 048 ?NCXQ 127C COS
91E5 128 WRIT 4(L)
91E6 0B8 READ 2(Y)
```

```
91E7 070 N=C ALL
91E8 221
91E9 048 ?NCXQ 1288 SIN
91EA 0A8 WRIT 2(Y)
91EB 248 SETF 9
91EC 260 SETHEX
91ED 379
91EE 03C|PORT DEP:
91EF 161|XQ 9161        SINH(2)
91F0 0B8 READ 2(Y)
91F1 135
91F2 060 ?NCXQ 184D MP2-10
91F3 0A8 WRIT 2(Y)
91F4 244 CLRF 9
91F5 2E3 JNC 91D1 -24
91F6 083 "C"
91F7 008 "H"
91F8 00E "N"
91F9 001 "A"
91FA 014 "T"
91FB 238 READ 8(P)
91FC 05E C=0 MS
91FD 228 WRIT 8(P)
91FE 053 JNC 9208 +0A
91FF 083 "C"
9200 008 "H"
9201 014 "T"
9202 00F "O"
9203 003 "C"
9204 238 READ 8(P)
9205 05E C=0 MS
9190 029
9191 068 ?NCXQ 1A0A EXP10
9192 0AE A<>C ALL
9193 278 READ 9(Q)
9194 01D
9195 060 ?NCXQ 1807 AD2-10
9196 0AE A<>C ALL
9197 0F8 READ 3(X)
9198 2F3 JNC 9176 -22
ADCHK 9199 0B8 READ 2(Y)
919A 0AE A<>C ALL
919B 0F8 READ 3(X)
919C 355
919D 050 ?NCXQ 14D5 %
                        (AD-check)
RAD1 919E 108 SETF 8
RAD2 919F 238 READ 8(P)
91A0 0AE A<>C ALL
91A1 3B8 READ 14(d)
91A2 11A A=C M
91A3 0AE A<>C ALL
91A4 228 WRIT 8(P)
91A5 0AE A<>C ALL
91A6 00E A=0 ALL
91A7 17A A=A+1 M
91A8 370 C=C OR A
91A9 3A8 WRIT 14(d)
91AA 0B8 READ 2(Y)
91AB 070 N=C ALL
91AC 2A0 SETDEC
91AD 3E0 RTN
MODW 91AE 238 READ 8(P)
91AF 0AE A<>C ALL
91B0 3B8 READ 14(d)
91B1 0BA A<>C M
91B2 3A8 WRIT 14(d)
91B3 04E C=0 ALL
91B4 228 WRIT 8(P)
91B5 3E0 RTN
91B6 083 "C"
91B7 008 "H"
91B8 00E "N"
91B9 009 "I"
91BA 013 "S"
91BB 260 SETHEX
91BC 379
91BD 03C|PORT DEP:      ADCHK
91BE 199|XQ 9199
91BF 1F1
91C0 048 ?NCXQ 127C COS
91C1 128 WRIT 4(L)
91C2 0B8 READ 2(Y)
91C3 070 N=C ALL
91C4 221
91C5 048 ?NCXQ 1288 SIN
91C6 0A8 WRIT 2(Y)
91C7 244 CLRF 9
91C8 260 SETHEX
91C9 379
91CA 03C|PORT DEP:
9206 23E C=C+1 MS
9207 228 WRIT 8(P)
9208 0F8 READ 3(X)
9209 0AE A<>C ALL
920A 0B8 READ 2(Y)
920B 355
920C 050 ?NCXQ 14D5 %
                        (AD-check)
920D 10E A=C ALL
920E 01D
920F 060 ?NCXQ 1807 AD2-10
9210 0E8 WRIT 3(X)
9211 0B8 READ 2(Y)
9212 10E A=C ALL
9213 01D
9214 060 ?NCXQ 1807 AD2-10
9215 0A8 WRIT 2(Y)
9216 260 SETHEX
9217 379
9218 03C|PORT DEP:      RAD1
9219 19E|XQ 919E
921A 1F1
921B 048 ?NCXQ 127C COS
921C 128 WRIT 4(L)
921D 0B8 READ 2(Y)
921E 070 N=C ALL
921F 221
9220 048 ?NCXQ 1288 SIN
9221 0A8 WRIT 2(Y)
9222 244 CLRF 9
9223 260 SETHEX
9224 379
9225 03C|PORT DEP:      COSH(2)
9226 161|XQ 9161
9227 138 READ 4(L)
9228 0EE C<>B ALL
9229 238 READ 8(P)
922A 0EE C<>B ALL
922B 2DE ?B#0 MS
922C 013 JNC 922E +02
922D 2BE C=-C-1 MS
922E 000 NOP
922F 01D
9230 060 ?NCXQ 1807 AD2-10
9231 128 WRIT 4(L)
9232 0AE A<>C ALL
9233 0B8 READ 2(Y)
9234 0AE A<>C ALL
9235 261
9236 060 ?NCXQ 1898 DV2-10
9237 0EE C<>B ALL
9238 238 READ 8(P)
9239 0EE C<>B ALL
923A 2DE ?B#0 MS
923B 013 JNC 923D +02
923C 2BE C=-C-1 MS
923D 0A8 WRIT 2(Y)
923E 260 SETHEX
923F 248 SETF 9
9240 379
9241 03C|PORT DEP:      SINH(2)
9242 161|XQ 9161
9243 138 READ 4(L)
9244 261
9245 060 ?NCXQ 1898 DV2-10
9246 0E8 WRIT 3(X)
9247 260 SETHEX
9248 369
9249 03C|PORT DEP:      MODW
924A 1AE|GO 91AE
924B 088 "H"
924C 00E "N"
924D 009 "I"
924E 013 "S"
924F 012 "R"
9250 001 "A"
9251 248 SETF 9
9252 043 JNC 925A +08
9253 088 "H"
9254 013 "S"
9255 00F "O"
9256 003 "C"
9257 012 "R"
9258 001 "A"
9259 244 CLRF 9
925A 0F8 READ 3(X)
925B 361
925C 050 ?NCXQ 14D8 CHK$S
925D 10E A=C ALL
925E 135
925F 060 ?NCXQ 184D MP2-10
9260 10E A=C ALL
9261 276 C=C-1 XS
9262 276 C=C-1 XS
9263 289
9264 003 ?CGO 00A2 ERROF
9265 04E C=0 ALL
```

```
9266 35C R= 12
9267 050 LD@R 1
9268 24C ?FSET 9
9269 01F JC 926C +03
926A 2BE C=-C-1 MS
926B 000 NOP
926C 01D
926D 060 ?NCXQ 1807 AD2-1ø
926E 2F9
926F 060 ?NCXQ 18BE SQR1ø
9270 0AE A<>C ALL
9271 0F8 READ 3(X)
9272 01D
9273 060 ?NCXQ 1807 AD2-1ø
9274 084 CLRF 5
9275 115
9276 06C ?NCXQ 1B45
9277 1AB JNC 92AC +35
9278 088 "H"
9279 00E "N"
927A 001 "A"
927B 014 "T"
927C 012 "R"
927D 001 "A"
927E 248 SETF 9
927F 043 JNC 9287 +08
9280 088 "H"
9281 014 "T"
9282 00F "O"
9283 003 "C"
9284 012 "R"
9285 001 "A"
9286 244 CLRF 9
9287 0F8 READ 3(X)
9288 361
9289 050 ?NCXQ 14D8 CHK$S
928A 24C ?FSET 9
928B 013 JNC 928D +02
928C 2BE C=-C-1 MS
928D 0AE A<>C ALL
928E 04E C=0 ALL
928F 35C R= 12
9290 050 LD@R 1
9291 24C ?FSET 9
9292 017 JC 9294 +02
9293 2BE C=-C-1 MS
9294 000 NOP
9295 01D
9296 060 ?NCXQ 1807 AD2-1ø
9297 070 N=C ALL
9298 0F8 READ 3(X)
9299 0AE A<>C ALL
929A 04E C=0 ALL
929B 35C R= 12
929C 050 LD@R 1
929D 01D
929E 060 ?NCXQ 1807 AD2-1ø
929F 0AE A<>C ALL
92A0 0B0 C=N ALL
92A1 261
92A2 060 ?NCXQ 1898 DV2-1ø
92A3 3C4 ST=0
92A4 115
92A5 06C ?NCXQ 1B45 LN1ø
92A6 0AE A<>C ALL
92A7 04E C=0 ALL
92A8 35C R= 12
92A9 090 LD@R 2
92AA 261
92AB 060 ?NCXQ 1898 DV2-1ø
92AC 0AE A<>C ALL
92AD 0F8 READ 3(X)
92AE 128 WRIT 4(L)
92AF 0AE A<>C ALL
92B0 0E8 WRIT 3(X)
92B1 3E0 RTN
NEN 92B2 2A0 SETDEC
92B3 0F8 READ 3(X)
92B4 10E A=C ALL
92B5 135
92B6 060 ?NCXQ 184D MP2-1ø
92B7 24C ?FSET 9
92B8 017 JC 92BA +02
92B9 2BE C=-C-1 MS
92BA 268 WRIT 9(Q)
92BB 0B8 READ 2(Y)
92BC 10E A=C ALL
92BD 135
92BE 060 ?NCXQ 184D MP2-1ø
92BF 24C ?FSET 9
92C0 017 JC 92C2 +02
92C1 2BE C=-C-1 MS
92C2 0AE A<>C ALL
92C3 278 READ 9(Q)
92C4 01D
92C5 060 ?NCXQ 1807 AD2-1ø
92C6 0AE A<>C ALL
92C7 04E C=0 ALL
92C8 35C R= 12
92C9 050 LD@R 1
92CA 01D
92CB 060 ?NCXQ 1807 AD2-1ø
92CC 0AE A<>C ALL
92CD 3E0 RTN
92CE 083 "C"
92CF 008 "H"
92D0 00E "N"
92D1 001 "A"
92D2 014 "T"
92D3 012 "R"
92D4 001 "A"
92D5 0F8 READ 3(X)
92D6 0AE A<>C ALL
92D7 0B8 READ 2(Y)
92D8 355
92D9 050 ?NCXQ 14D5 %
92DA 260 SETHEX (AD-Check)
92DB 244 CLRF 9
92DC 379
92DD 03C PORT DEP:
92DE 2B2 XQ 92B2   NEN
92DF 35E ?A#0 MS
92E0 0B5
92E1 0A3 ?CGO 282D ERRDE
92E2 0AE A<>C ALL
92E3 22D
92E4 060 ?NCXQ 188B ON/X1ø
92E5 260 SETHEX
92E6 379
92E7 03C PORT DEP:
92E8 19F XQ 919F   RAD 2
92E9 260 SETHEX
92EA 248 SETF 9
92EB 379
92EC 03C PORT DEP:
92ED 2B2 XQ 92B2   NEN
92EE 0F8 READ 3(X)
92EF 268 WRIT 9(Q)
92F0 0AE A<>C ALL
92F1 261
92F2 060 ?NCXQ 1898 DV2-1ø
92F3 10E A=C ALL
92F4 01D
92F5 060 ?NCXQ 1807 AD2-1ø
92F6 0E8 WRIT 3(X)
92F7 260 SETHEX
92F8 379
92F9 03C PORT DEP:
92FA 27E XQ 927E   ARTAN
92FB 0F8 READ 3(X)
92FC 128 WRIT 4(L)
92FD 278 READ 9(Q)
92FE 0E8 WRIT 3(X)
92FF 260 SETHEX
9300 244 CLRF 9
9301 379
9302 03C PORT DEP:
9303 2B2 XQ 92B2   NEN
9304 0B8 READ 2(Y)
9305 0AE A<>C ALL
9306 261
9307 060 ?NCXQ 1898 DV2-1ø
9308 10E A=C ALL
9309 01D
930A 060 ?NCXQ 1807 AD2-1ø
930B 070 N=C ALL
930C 2A9
930D 040 ?NCXQ 10AA ATAN
930E 0AE A<>C ALL
930F 04E C=0 ALL
9310 35C R= 12
9311 090 LD@R 2
9312 268 WRIT 9(Q)
9313 261
9314 060 ?NCXQ 1898 DV2-1ø
9315 0A8 WRIT 2(Y)
9316 138 READ 4(L)
9317 0AE A<>C ALL
9318 278 READ 9(Q)
9319 261
931A 060 ?NCXQ 1898 DV2-1ø
931B 0E8 WRIT 3(X)
931C 260 SETHEX
931D 369
931E 03C PORT DEP:
931F 1AE GO 91AE   MODW
```

```
9320 083 "C"
9321 00E "N"
9322 009 "I"
9323 013 "S"
9324 379
9325 03C|PORT DEP:
9326 199|XQ 9199      ADCHK
9327 0F8 READ 3(X)
9328 070 N=C ALL
9329 1F1
932A 048 ?NCXQ 127C   COS
932B 268 WRIT 9(Q)
932C 0F8 READ 3(X)
932D 0AE A<>C ALL
932E 0B8 READ 2(Y)
932F 0E8 WRIT 3(X)
9330 0AE A<>C ALL
9331 0A8 WRIT 2(Y)
9332 260 SETHEX
9333 379
9334 03C|PORT DEP:
9335 159|XQ 9159      SINH
9336 0AE A<>C ALL
9337 278 READ 9(Q)
9338 135
9339 060 ?NCXQ 184D   MP2-10
933A 0E8 WRIT 3(X)
933B 0B8 READ 2(Y)
933C 070 N=C ALL
933D 221
933E 048 ?NCXQ 1288   SIN
933F 0A8 WRIT 2(Y)
9340 0F8 READ 3(X)
9341 268 WRIT 9(Q)
9342 138 READ 4(L)
9343 0E8 WRIT 3(X)
9344 260 SETHEX
9345 379
9346 03C|PORT DEP:
9347 15F|XQ 915F      COSH
9348 0AE A<>C ALL
9349 0B8 READ 2(Y)
934A 135
934B 060 ?NCXQ 184D   MP2-10
934C 0E8 WRIT 3(X)
934D 278 READ 9(Q)
934E 0A8 WRIT 2(Y)
934F 260 SETHEX
9350 369
9351 03C|PORT DEP:
9352 1AE|GO 91AE      MODW
9353 083 "C"
9354 013 "S"
9355 00F "O"
9356 003 "C"
9357 379
9358 03C|PORT DEP:
9359 199|XQ 9199      ADCHK
935A 0F8 READ 3(X)
935B 070 N=C ALL
935C 221
935D 048 ?NCXQ 1288   SIN
935E 268 WRIT 9(Q)
935F 0F8 READ 3(X)
9360 0AE A<>C ALL
9361 0B8 READ 2(Y)
9362 0E8 WRIT 3(X)
9363 0AE A<>C ALL
9364 0A8 WRIT 2(Y)
9365 260 SETHEX
9366 379
9367 03C|PORT DEP:
9368 159|XQ 9159      SINH
9369 0AE A<>C ALL
936A 278 READ 9(Q)
936B 135
936C 060 ?NCXQ 184D   MP2-10
936D 2BE C=-C-1 MS
936E 0E8 WRIT 3(X)
936F 0B8 READ 2(Y)
9370 070 N=C ALL
9371 1F1
9372 048 ?NCXQ 127C   COS
9373 263 JNC 933F -34
9374 083 "C"
9375 00E "N"
9376 001 "A"
9377 014 "T"
9378 379
9379 03C|PORT DEP:
937A 199|XQ 9199      ADCHK
937B 0F8 READ 3(X)
937C 10E A=C ALL
937D 01D
937E 060 ?NCXQ 1807   AD2-10
937F 0E8 WRIT 3(X)
9380 070 N=C ALL
9381 1F1
9382 048 ?NCXQ 127C   COS
9383 268 WRIT 9(Q)
9384 0B8 READ 2(Y)
9385 10E A=C ALL
9386 01D
9387 060 ?NCXQ 1807   AD2-10
9388 0AE A<>C ALL
9389 0F8 READ 3(X)
938A 0A8 WRIT 2(Y)
938B 0AE A<>C ALL
938C 0E8 WRIT 3(X)
938D 260 SETHEX
938E 379
938F 03C|PORT DEP:
9390 15F|XQ 915F      COSH
9391 0AE A<>C ALL
9392 278 READ 9(Q)
9393 01D
9394 060 ?NCXQ 1807   AD2-10
9395 268 WRIT 9(Q)
9396 138 READ 4(L)
9397 0E8 WRIT 3(X)
9398 260 SETHEX
9399 379
939A 03C|PORT DEP:
939B 159|XQ 9159      SINH
939C 0AE A<>C ALL
939D 278 READ 9(Q)
939E 261
939F 060 ?NCXQ 1893   DV2-10
93A0 0E8 WRIT 3(X)
93A1 278 READ 9(Q)
93A2 128 WRIT 4(L)
93A3 0B8 READ 2(Y)
93A4 070 N=C ALL
93A5 221
93A6 048 ?NCXQ 1288   SIN
93A7 0AE A<>C ALL
93A8 138 READ 4(L)
93A9 261
93AA 060 ?NCXQ 1898   PV2-10
93AB 0AE A<>C ALL
93AC 0F8 READ 3(X)
93AD 0A8 WRIT 2(Y)
93AE 0AE A<>C ALL
93AF 0E8 WRIT 3(X)
93B0 260 SETHEX
93B1 369
93B2 03C|PORT DEP:
93B3 1AE|GO 91AE      MODW
```

Klaus Hupertz
Nievelsteinstr. 30
4050 Mönchengladbach 3

# Schreibschrittlineal für den DESKJET

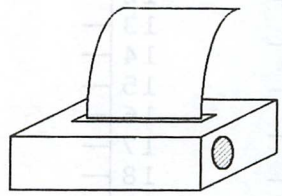
von Dr. Martin Hochenegger

**HP 41, CCD, HP-IL, Deskjet, SIZE 001**

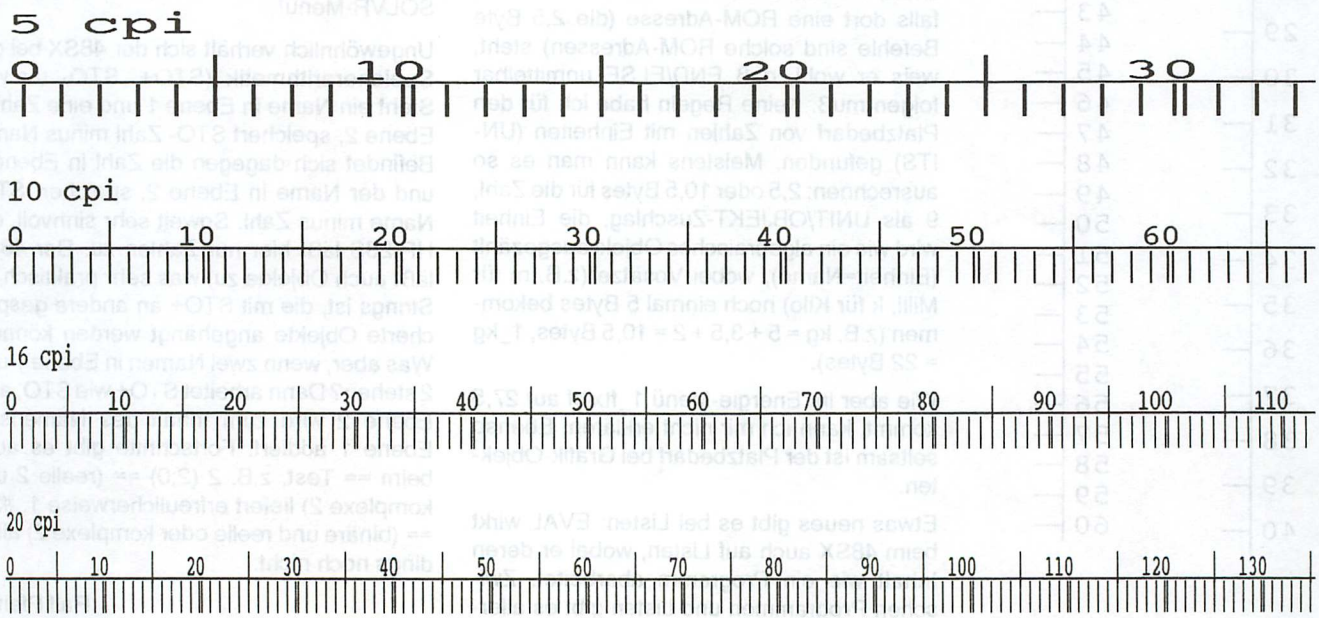
Nach meinen Programmen für den ThinkJet-Drucker, erschienen in Prisma 5/88, S. 39 und 6/88, S. 35, hier ähnliche Programme für den DeskJet. Die dazu erforderliche Schnittstelle liegt auf Adresse #1 in der Schleife. Man muß diese ja extra anwählen, da nur echte Drucker mit IL-Schnittstelle als Ziele für Druckbefehle wie PRA oder ACA automatisch angewählt werden. Ein RS232-Interface oder ein Parallel-Interface dagegen müssen ja kein Drucker sein, deshalb ist hier eine "manuelle" Anwahl der primären Zieladresse mittels des Befehls SELECT in Zeile 5 erforderlich, in der Zeile davor steht die gewünschte Adresse, hier 1 (= E), es kann jede beliebige andere sein.

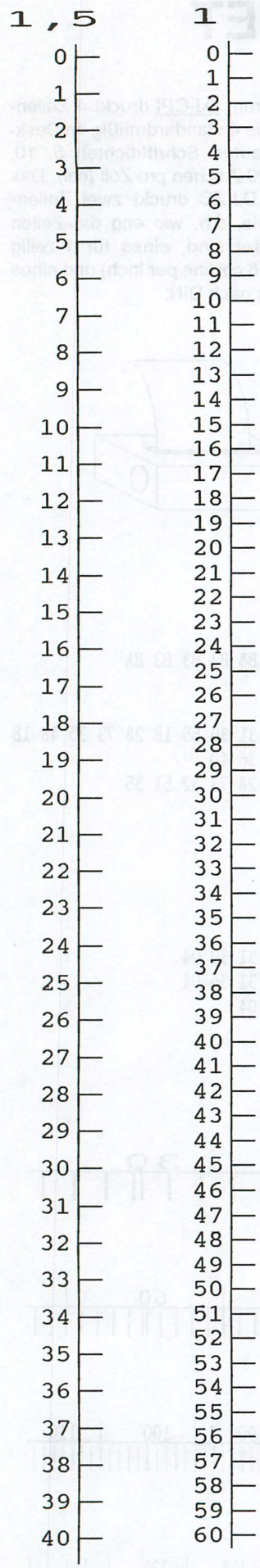
Das Programm DJ-CPI druckt 4 Zeilenlineale für die 4 standardmäßig im DeskJet eingebauten Schriftdichten 5, 10, 16,67 und 20 Zeichen pro Zoll (cpi). Das Programm DJ-ZC druckt zwei Zeilendichtenlineale, d.h. wie eng die Zeilen untereinander sind, eines für 1-zeilig nach DIN (=6 cpi line per Inch) und eines für 1,5-zeilig nach DIN.

```
01*LBL "DJ-CPI"
02 FIX 0
03 CF 29
04  E
05 SELECT
06 "⊕&d1D"
07 ATOBUFX
08 "⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕"
09 2
10 ATOBUFX
11 ATOXL
12 3
13 ATOBUFX
14 " cpi⊕="
15 4
16 ATOBUFX
17 "⊕(s"
18 5
19 ATOBUFX
20 "⊕1⊕!⊕(10U⊕(s5H⊕"
21 "}&a3136M"
22 ACA
23 "⊕&k0W⊕(s2Q5"
24 ACA
25 10,0401
26 STO 00
27 "⊕"
28 SACA
29 PRBUF
30*LBL 00
31 SF 17
32 "⊕0"
33 SACA
34*LBL 01
35 4,032
36 ACLX
37 "⊕"
38 OUTA
39 CLA
40 5
41 RCL 00
42 ACAXY
43 ISG 00
44 GTO 01
45 PRBUF
46 "⊕"
47 SACA
48 FS? 00
49 RTN
50 3
51*LBL 02
52 "⊕"
53 SACA
54 DSE X
55 GTO 02
56 "⊕⊕"
57 OUTA
58 PRBUF
59 CF 17
60 FS? 00
61 RTN
62 "⊕(s10H10⊕"
63 SACA
64 PRBUF
65 10,0701
66 STO 00
67 SF 00
68 XEQ 00
69 6
70 XEQ 02
71 CF 17
72 "⊕(s16H16⊕"
73 SACA
74 PRBUF
75 10,1201
76 STO 00
77 XEQ 00
78 11
79 XEQ 02
80 "⊕20H20⊕"
81 SACA
82 PRBUF
83 10,1401
84 STO 00
85 XEQ 00
86 13
87 XEQ 02
88 CF 00
89 "⊕"
90 ACA
91 .END.
```



```
006: F5 1B 26 64 31 44
008: FB B3 B3 B3 B3 B3 B3 B3 B3 B3 B3 BA
014: F6 20 63 70 69 1B 3D
017: F3 1B 28 73
020: FF 1B 31 1B 21 1B 28 31 30 55 1B 28 73 35 48 1B
021: F8 7F 26 61 33 6C 33 36 4D
023: FB 1B 26 6B 30 57 1B 28 73 32 51 35
027: F1 04
032: F2 01 30
037: F1 B3
046: F1 02
052: F1 03
056: F2 0A 0A
062: F9 1B 28 73 31 30 48 31 30 04
072: F9 1B 28 73 31 36 48 31 36 04
080: F7 05 32 30 48 32 30 04
089: F1 0C
```









```
01*LBL "DJ-ZL"
02  E
03 SELECT
04 "@1@!@(10U@(s2Q"
05 "}@&a3L@&kOW"
06 ACA
07 FIX 0
08 CF 28
09 CF 29
10*LBL 00
11 "@(s5H"
12 FC? 00
13 "} 1"
14 FS? 00
15 "}1,5"
16 "}@(s10H@="
17 ACA
18 PRBUF
19 FC? 00
20 6 E-2
21 FS? 00
22 4 E-2
23 STO 00
24 SF 17
25*LBL 01
26 CLA
27 4
28 RCL 00
29 INT
30 ACAXY
31 "@@"
32 FS? 00
33 XEQ 02
34 OUTA
35 PRBUF
36 ISG 00
37 GTO 01
38 "@"
39 ACA
40 FS? 00
41 RTN
42 SF 00
43 XEQ 00
44 CF 00
45 CF 17
46 STOP
47*LBL 02
48 "}@=@@@"
49 .END.
```

```
004: FE 1B 31 1B 21 1B 28 31 30 55 1B 28 73 32 51
005: FB 7F 1B 26 61 33 4C 1B 26 6B 30 57
011: F5 1B 28 73 35 48
016: F9 7F 1B 28 73 31 30 48 1B 3D
031: F2 C3 C4
038: F1 0C
048: F6 7F 1B 3D 08 08 B3
```

Dr. Martin Hochenegger
Heidelberger Landstr. 97
6100 Darmstadt 13

# Zusatz zur Byte Tabelle (Seite 27)

Die angegebene Byte-Tabelle weicht nur in wenigen Punkten von der des HP-28S ab. Algebraische Objekte zählt man wie Programme aus, Klammern brauchen dann keinen Platz. Beispiel: 'SIN(X+Y)-3' = 24 Bytes, X und Y sind Namen (je 4,5 Bytes), SIN, + und - Befehle (je 2,5 Bytes), die 3 braucht auch nur 2,5 Bytes und dann kommen noch die " mit 5 Bytes hinzu. Ganze Zahlen (-9 ß +9) erkennt der HP-48SX auch als Ergebnis mathematischer Operationen und verbraucht nur 2,5 Bytes.

Der Stern * bedeutet, daß hier die Struktur 5 Bytes kürzer ist, falls zwischen den bezeichneten Befehlen genau ein Befehl mit 2,5 Bytes Länge steht. Die Ursache ist wohl, daß z.B. nach einem THEN der Rechner speichert, wo das END bzw. das ELSE steckt; falls dort eine ROM-Adresse (die 2,5 Byte Befehle sind solche ROM-Adressen) steht, weis er wohl, daß END/ELSE unmittelbar folgen muß. Keine Regeln habe ich für den Platzbedarf von Zahlen mit Einheiten (UNITS) gefunden. Meistens kann man es so ausrechnen: 2,5 oder 10,5 Bytes für die Zahl, 9 als UNIT/OBJEKT-Zuschlag, die Einheit wird wie ein algebraisches Objekt ausgezählt (Einheit=Name), wobei Vosätze (z.B. m für Milli, k für Kilo) noch einmal 5 Bytes bekommen (z.B. kg = 5 + 3,5 + 2 = 10,5 Bytes, 1_kg = 22 Bytes).

Wie aber im Energie-Menü 1_ftxlbf auf 27,5 kommt, kann ich mir nicht erklären. Ebenso seltsam ist der Platzbedarf bei Grafik-Objekten.

Etwas neues gibt es bei Listen: EVAL wirkt beim 48SX auch auf Listen, wobei er deren Inhalt wie ein Programm abarbeitet. Zwischen Programmen und Listen gibt es allerdings noch kleine Unterschiede: Zwei Listen lassen sich mit + verketten, dafür starten Programme die man unter einem Namen speichert, von selbst. Die Funktionen des rechten MEMORY-Menüs (blaue Vortaste, dann MEMORY, enthält z.B. STO) funktionieren in Listen nur selten: Namen *können* in Listen keine " bekommen, STO funktioniert also nur, wenn dieser Name noch nicht existiert. Listen lassen sich auch so verwenden: Das SOLVR-Menü zeigt alle numerischen Variablen in dem unter EQ gespeicherten Programm an, es sei denn, sie stehen in einer Liste. Soll ein Programm also auf numerische Konstanten zurückgreifen, die irgendwo im Speicher stehen, packt man deren Namen in eine Liste, gefolgt von EVAL, und dieser Name erscheint garantiert nicht im SOLVR-Menü!

Ungewöhnlich verhält sich der 48SX bei der Speicherarithmetik (STO+, STO- u.s.w.): Steht ein Name in Ebene 1 und eine Zahl in Ebene 2, speichert STO- Zahl minus Name. Befindet sich dagegen die Zahl in Ebene 1 und der Name in Ebene 2, speichert STO- Name minus Zahl. Soweit sehr sinnvoll, der HP-28S läßt hier nur Zahlen zu. Der 48SX läßt auch Objekte zu, was sehr praktisch für Strings ist, die mit STO+ an andere gespeicherte Objekte angehängt werden können. Was aber, wenn zwei Namen in Ebene 1 und 2 stehen? Dann arbeitet STO+ wie STO, also Ebene 2 wird zum Inhalt des Namens in Ebene 1 addiert. Fortschritte gibt es auch beim == Test, z.B. 2 (2;0) == (reelle 2 und komplexe 2) liefert erfreulicherweise 1, #2 2 == (binäre und reelle oder komplexe 2) allerdings noch nicht.

Ralf Pfeiffer

# Inhaltsverzeichnis der MS-DOS INFO's

Bisher sind 54 INFOs erschienen: 1986 die Nummern 1-12, 1987 die Nummern 13-24, 1988 die Nummern 25-36, 1989 die Nummern 37-48, 1990 die Nummern 49-54.

Nach der Nummer des INFO's folgen die Unterverzeichnisse (z.B. \DFUE) und die darin befindlichen Programme. Steht hinter einer Beschreibung in Klammern der Name einer Programmiersprache (z.B. Assembler), ist der dazugehörige Sourcecode mit auf dem INFO.

**INFO 01:**

\DFUE

| | |
|---|---|
| KERMIT | MS-Kermit (Datenfernübertragungsprogramm) |
| PC-VT | VT-100 Terminal-Emulation |

\PASCAL

| | |
|---|---|
| THELP | Pull-Down Hilfe für Turbo-Pascal (mit Source) |

\UTILITY

| | |
|---|---|
| ALSEARCH | Sucht nach einer Datei in allen Unterverzeichnissen |
| ALTER | Utility zum Verändern der Dateiattribute |
| CWEEP1 | SWEEP für MS-DOS |
| UIBM | Umlautkonvertierungsprogramm (Assembler) |
| WSASCII | Konvertiert von WS 3.3 zu ASCII |

**INFO 02:**

\DBASE2

| | |
|---|---|
| A-DBA | Datenverwaltungsprogramm in dBASE II |

\PASCAL

| | |
|---|---|
| HERCUL2 | Routinen für die Herculeskarte |
| UTIL | Routinen für RS232 |
| WSCLEAN | Konvertiert WS 3.3 zu ASCII |

\PIANO

| | |
|---|---|
| PIANOMAN | Verwandelt den IBM-PC in eine Orgel |

\UTILITY

| | |
|---|---|
| ZSQ-ZUSQ | Squeezer/Unsqueezer |

**INFO 03:**

\ARC

| | |
|---|---|
| ARC | Dateiarchivierungssystem |

\PASCAL

| | |
|---|---|
| PC-DISK | Disketten-Katalogverwalter |
| SIDEWYTR | Druckt Dateien vertikal aus auf EPSON-Druckern |
| TURBO.PAT | Turbo-Pascal Patches ("Include Error Message ?") |
| TURBO-UT | nützliche Turbo-Pascal Routinen |
| XLIST | Programmlister mit Crossreferenz für Turbo-Pascal |

\UTILITY

| | |
|---|---|
| LABEL | Utility zum Erstellen (Ändern) des Volume Labels (C) |
| NO | Utility zum Ausschließen von Datein bei Befehlen (Assembler) |
| UHR | "Full-Screen"-Digitaluhr (Basic) |

\HUMOR

| | |
|---|---|
| BUGRES | Zwei schöne Programme in |
| DRAIN | Sachen Computerhumor |

**INFO 04:** PC-FILE

**INFO 05:**

\UTILITY

| | |
|---|---|
| BACKSCRL | Erlaubt Zurückscrollen von Texten (z.B. bei TYPE) |
| BROWSE | Verbessertes TYPE |
| D | Super-Directory 5.00 |
| ERA | Verbessertes Erase/Del |
| FILEDUMP | Hexdumper |
| GCOPY | Verbessertes COPY |
| GDEL | Verbessertes Erase/Del |
| MEMBRAIN | Ramdisk-Treiber |
| MOVE | Erlaubt Verschieben von Dateien zwischen Directories |
| NDOSEDIT | Editor |
| RDISK | Ramdisk-Treiber (Assembler) |
| SDIR26 | Super-Directory 2.6 |
| U-SDIR26 | ^ (MS-DOS) |
| VDEL | Verbessertes Erase/Del |
| WHEREIS | Sucht nach Dateien in allen Unterverzeichnissen |

**INFO 06:**

\PASCAL

| | |
|---|---|
| KANALYSE | Konfigurationsanalysator |
| PLIST40 | Programmlister |

\UTILITY

| | |
|---|---|
| DOSAMATC | Neue DOS-Benutzeroberfläche mit Program-switching |
| FPRINT | Neues Print (ersetzt das MS-DOS Utility PRINT) |
| FRED | Full-Screen Editor |
| KEY-FAKE | Übergibt Tastatureingaben an Programme |
| PUSHDIR | 2 Utilities zum Festhalten von |
| POPDIR | Pfaden (Assembler) |
| SQUEEZE | Squeezer/Unsqeezer |
| UNSQUEEZE | |
| STAT | Ändert Dateiattribute (Assembler) |
| UNCRASH | Fängt "System-Crashs" ab |

**INFO 07:**

\UTILITY

| | |
|---|---|
| HERCMODE | Grafiktreiber für die Hercules-Karte (Assembler) |
| SCHACH | Schach-Programm (nur IBM-Color-Grafik) |
| PACGIRLA | PACMAN |

\BASIC

| | |
|---|---|
| DIGGER | Spiel für den IBM-PC (Color-Grafik) |
| ROBOTER | Programm zum Steuern von Robotern |

\EXPERT

| | |
|---|---|
| EXPERT | Programm zum Entwickeln von Expertensystemen |

**INFO 08:**

\UTILITY

| | |
|---|---|
| CWEEP214 | SWEEP ähnliches Programm für MS-DOS |

\BASIC

| | |
|---|---|
| HOPPER | Spiel für den IBM-PC (Color-Grafik) mit Source |
| POSTER | Druckt große Schriftzüge quer (auf jedem Drucker) |

\PASCAL

| | |
|---|---|
| PIANO | Wie Piano-Man, aber in Pascal und mit Sourcecode ! |
| KEYDEMO | Testet die Tastaturkompatibilität |
| MAKAMOVI | Programm zum Erstellen von "Filmen", die aus einzelnen Textseiten zusammengesetzt sind |
| SHOWMOVI | Wie MAKAMOVI, allerdings nur zum Ansehen von Filmen |
| UTIL | Sammlung von Turbo-Utilities |

**INFO 09:**

Das ab diesem INFO erstmals erschienene NEWS-Directory mit Nachrichten und Informationen aus der Computerbranche wird nicht mehr aufgeführt, die behandelten Themen dürften inzwischen überholt sein.

\PASCAL

| | |
|---|---|
| PRIMFAK | Primfaktorenanalysator in Pascal und Basic |
| PRIMINT | Primzahlenprogramm (Integer) |
| PRIMREAL | ^ (Realzahlen) |

\UTILITY

| | |
|---|---|
| ASMGEN | Dissassembler (Assembler) |
| PP | Druckt Texfiles mit Rand (Batch) |
| SETUP | Drucker-Kontrollprogramm (Assembler) |

\DFUE

| | |
|---|---|
| PC-DIAL | Datenübertragungsprogramm |

**INFO 10:**

\UTILITY

| | |
|---|---|
| PC-TOUCH | Schreibmaschinenlehrer |
| COMMANDO | Neue DOS-Benutzeroberfläche |

\GAME

| | |
|---|---|
| STARGATE | Spiel, nur für IBM mit CGA (Farbgrafikkarte) |

**INFO 11:**

\KARTEN

Daten für Landkarten (BRD, Frankreich, Österreich, Schweiz), 1. Teil (BRD). Dazu ein Basic-Programm zur grafischen Darstellung der Karten auf dem Schirm.

\UTILITY

| | |
|---|---|
| DUMP | File-Dump-Utility (Assembler) |

\C

| | |
|---|---|
| ASCIISET | Beispielprogramm für Char-Funktionen |
| FDIR | Directory-Utility |

\PASCAL

| | |
|---|---|
| PATHS | Utility zum Verarbeiten von Pfaden in Turbo-Pascal |

**INFO 12:**

\KARTEN

Landkarten 2. Teil (BRD/Österreich/Schweiz).

\GAME

| | |
|---|---|
| BABY | Spiel für den IBM PC (CGA) |

\C

| | |
|---|---|
| SQ | Squeezer mit Sourcecode (C) |
| USQ | Unsqueezer |
| TYPESQ | Type für gesqueezte Dateien |

**INFO 13:**

| | |
|---|---|
| \C | |
| DSTAT | Disk-Statistik-Utility (C) |
| DSTATPAS | ^ (Turbo-Pascal) |
| \GAME | |
| MEMORY | Memory in BASIC |
| \INFO | |
| PATCH123 | Patch für Lotus 1-2-3 |
| \KARTEN | |
| Daten für Landkarten (BRD, Frankreich, Österreich, Schweiz), 3. Teil (Frankreich). | |
| \UTILITY | |
| CURSOR | Programm zur Manipulation des Cursors (Assembler) |

**INFO 14:**

| | |
|---|---|
| \CODEVIEW | CODEVIEW-Demo (Teil 1) |
| \DBASE | |
| CURSOR | Verändert die Form des Cursors in dBASE III+ (Beispiel-Programm von Ashton-Tate für die Benutzung der dBASE III+ Assembler-Schnittstelle) (Assembler) |
| SCRNSAVE | Sichert Bildschirmseiten im RAM und kann sie wieder auf den Schirm bringen (für dBASE III+) (Assembler) |
| KOPF | Dateistruktur von dBASE III |
| NEUKOPF | Kann zerstörte Datenbank-Header wiederherstellen (für dBASE III) (Basic) |
| \TEST | |
| XT-286 | Testbericht XT-286 Karte |
| TEST1 | |
| TEST2 | Benchmarks (Pascal) |
| \UTILITY | |
| EPSON | Residentes Umlautkonvertierutility für Epson-Drucker |

**INFO 15:**

| | |
|---|---|
| \CODEVIEW | Zweiter Teil des CODEVIEW-DEMOs (siehe INFO 14) |
| \BASIC | |
| STINPUT | Einige Basic-Routinen |
| \MOVIE | Erster Teil (von dreien) einer Spielfilm-Datenbank |

**INFO 16:**

| | |
|---|---|
| \DBASE | verschiedene dBASE III+ Routinen mit Sources |
| \PASCAL | |
| DBCONV | Konvertierungsutility (dBASE III --- ASCII) |
| \UTILITY | |
| RPN | Taschenrechner mit UPN |
| RPN7 | Version für den 8087 |

**INFO 17:**

| | |
|---|---|
| \CODEVIEW | |
| CVR | Der letzte Teil des CODEVIEW-Demos |
| \MOVIE | |
| MFIND | Das Verwaltungsprogramm für die Spielfilm-Datenbank |
| \GAME | |
| MEMORY | Memory |

**INFO 18:**

| | |
|---|---|
| \DBASE | |
| SCRNSAVE | Tool für dBASE III+: Kann Bildschirmseiten im RAM oder auf Disk sichern und blitzschnell wieder auf den Schirm bringen (Assembler). |
| \GAME | |
| GOBBLE | Spiel (Basic) |
| SUBMARIN | U-Boot Spiel (Basic) |
| \HUMOR | |
| FACE | Aufrufen ! |
| \MOVIE | Letzer Teil der Film-Datenbank |
| \PASCAL | |
| CONVERT1 | Konvertiert von WS -- WS 3.4 (Pascal) |
| MOUSE | Routinen für die PC-MOUSE und kompatible Mäuse (Pascal) |
| \UTILITY | |
| CHK4BOMB | Stellt fest, ob ein Programm ein Trojanisches Pferd, bzw. ein Virus ist. |

**INFO 19:**

| | |
|---|---|
| \GAME | |
| CAT | Spiel für IBM mit Farbgrafik (CGA) |
| PC-TREK | Spiel für IBM (Basic) |
| XO | Spiel für IBM mit Farbgrafik (CGA) |
| \UTILITY | |
| KEY2 | Neuer Keybgr (Assembler) |

**INFO 20:**

| | |
|---|---|
| \ASM | |
| DR | Full-Screen Directory Utility (Assembler) |
| HRT_COM | Timer für COM-Files (Assembler) |
| HRT_EXE | Timer für EXE-Files (Assembler) |
| TEST_COM | ^ Test für hrt_com (Assembler) |
| TEST_EXE | ^ Test für hrt_exe (Assembler) |
| KEY21 | Neue KEY-Version (Assembler) |
| KEY22 | ^ |
| REMINDER | Terminkalender (Assembler) |
| PCMAP | Drei Utilities zum Entfernen von residenten Programmen |
| INSTALL | ^ (Assembler) |
| REMOVE | ^ |
| DRIVPARM | Konfiguriert logisches Laufwerk als 80-Spur/720kb-Laufwerk (Assembler) |
| \GAME | |
| FOOTBALL | American Football (Basic) |
| \HUMOR | |
| COMMAND | Neue COMMAND.COM-Version mit Spezialeffekten (C) |
| \PASCAL | |
| BREAKOUT | Spiel für die CGA (Pascal) |
| BREAKOU2 | ^ Hercules-Version mit Erweiterungen |

**INFO 21:**

| | |
|---|---|
| \PKARC | |
| PKX35A35 | Neues Datei-Archivierungsutility |
| \ASM | |
| RN | Verzeichnisverwaltungsutility (Assembler) |
| KEY23 | Neue Key-Version (Assembler) |
| BRK | Utility um Ctrl-Break ständig abzufragen (Assembler) |
| \PASCAL | |
| REFORMAT | Utility um die Festplattendaten zu optimieren (wie DISK OPTIMIZER oder Norton SPEED DISK, Pascal) |

**INFO 22:**

| | |
|---|---|
| \ARC | Archive von A. Klingelnberg: BACKUP, COOKIE; CPM--DOS; HDU; MISC; PRINT. |
| \UTILITY | |
| POLYCOPY | Programm für Merfachkopien |
| CED | Utility zum Editieren der DOS-Kommandozeile (wie DOSEDIT - kann aber mehr) |
| \HERCULES | |
| SIMCGA | Simuliert eine CGA auf einem Rechner mit Herculeskarte |

**INFO 23:**

| | |
|---|---|
| \ASM | |
| CARDFILE | Residenter Karteikasten mit Telfonnummernwählfunktion (Assembler) |
| CTYPE | Utility zum Einstellen des Cursortyps (Assembler) |
| DIRNOTES | Utility zum Kommentieren von Diskettenverzeichnissen (Assembler) |
| \DOC | Dokumentationen aus dem PRISMA |
| CED | DOC |
| PKARC | DOC |
| PKXARC | DOC |
| \PASCAL | |
| HEXUTIL | Konvertiert zwei Typen von Hex-Dateien (.HEX und .BIN) (Pascal) |
| \UTILITY | |
| DISNDATA | Disassembler |
| KAT | Diskettenkatalog |

**INFO 24:**

| | |
|---|---|
| \ASM | |
| BROWSE | Zeigt Dateien seitenweise auf dem Bildschirm an (Assembler) |
| CO | Utility zum Manipulieren von Dateigruppen (Assembler) (ähnlich RN und DR) |
| DOSKEY | Bringt mehr Komfort in den DOS-Prompt (Assembler) |
| PRN2FILE | Leitet die Druckerausgabe in Dateien um (Assembler) |
| SAFARI | Verbessert DOS-Fehlermeldungen (Assembler) |
| SNIPPER | Verarbeitet Bildschirmausschnitte (Assembler) |
| \C | |
| INT10H | Interrupt 10h Funktionen unter C (C) |
| \C\80286 | ^ für 80286 compilierte Version |
| \DBASE | |
| CD | Verwaltung von Compact Disks (dBASE III) |
| \PASCAL | |
| PLIST41 | Programmlister in Turbo Pascal 4.0 |

**INFO 25:**

| | |
|---|---|
| \NARC | |
| NARC | Archive Extrakter (Menügesteuert) |
| \DFUE | Beitrag von Jürgen Schramm zu DFÜ |

**INFO 26:**

| | |
|---|---|
| \PASCAL | |
| ODOAWEG | Enfernt Leerzeilen aus Basic-Programmen (Pascal) |
| \SELFCOMP | |
| ERRORL | Ermöglicht Benutzereingaben in Batch-Dateien |
| SELFCOMP | Batch-Datei zu Erstellung von Selbstdekoprimierenden Dateien |
| \PROCOMM1 | 1. Teil von PROCOMM 2.4.2 (Kommunikationssoftware) |
| \HCAT | |
| HCAT | Diskettenkatalog-Utility |
| HFD | Dateikommentierungs-Utility |

**INFO 27:**

| | |
|---|---|
| \PROCOMM2 | 2. Teil von PROCOMM 2.4.2 |

**INFO 28:**

| | |
|---|---|
| \BACKUP | |
| BACKUP | Testbericht von 6 Backupprogrammen |
| FBP | Batch-Datei: Demo Fastback plus |
| TB | Batch-Datei: Demo Turbo-Bakkup |
| FB | Batch-Datei: Demo Fastback |
| PCB | Batch-Datei: Demo PC-Backup |
| BM | Batch-Datei: Demo Backup-Master |
| DSB | Batch-Datei: Demo DS-Bakkup+ |
| \ASM | |
| CAPTURE | Residentes Programm zum Erstellen von "Bildschirmfotos" auf Disk. (Assembler) |
| RUN | Ermöglicht jetzt auch unter DOS 2.xx Programme aus Verzeichnissen heraus einfach mit Pfadangabe aufzurufen (Assembler) |
| HELP | Residentes Programm zum Abrufen von Hilfebildschirmen (Assembler) |

**INFO 29:**

| | |
|---|---|
| \ASM | |
| ASPRN | Residentes Programm zum Erstellen von Druckmakros. (Assembler) |
| \C | |
| SHOW | Spielt Dateien mit "Bildschirmfotos" ab. (C) |
| \BAS | |
| HANGMAN | Spiel in Turbo-Basic (Basic) |
| TILGUNG | Programm zur Berechnung von Laufzeit und Zinssumme bei Darlehen (Basic) |
| \WINDOWS | Sammlung von Programmen für Microsoft WINDOWS |
| BOXES | ^ |
| CALPOP | ^ |
| CUBE | ^ |
| FISH | ^ |
| FUSE | ^ |
| GLOBE | ^ |
| MONDRIAN | ^ |
| PALETTE | ^ |
| \HERCULES | |
| SIMCGA | Version 4.00 des CGA-Emulators SIMCGA |
| \DBASE | |
| SCRNSAVE | Neue Version des dBASE III+ Screensavers (jetzt Seiten mit 4000 Bytes - kompatibel zu CAPTURE, PAINT, HELP und SHOW) (Assembler) |

**INFO 30:**

| | |
|---|---|
| \ASM | |
| PAINT | Utility zum Editieren von Bildschirmdateien (von CAPTURE, HELP, SHOW, SCRNSAVE, etc.) (Assembler) |
| \BASIC | |
| BRFEMPF | Programm zum Beschriften von Briefen (Basic) |
| \C | |
| SHOW | Korrigierte Fassung von SHOW (C) |
| \HCAT | |
| HCAT | Neue verbesserte Version des bekannten Katalog- |
| HFD | Programms HCAT/HFD |

**INFO 31:**

| | |
|---|---|
| \ASM | |
| P7GRAFIK | Treiber für Umsetzung von 8-Nadel auf 24-Nadel (Assembler) |
| TOUCH | Utility zum Ändern vom Dateidatum |
| \BASIC | |
| BARCODE | Programm zum Drucken von HP-41 Barcodes |
| \KALK | |
| KALK | Programm zum Vergleichen von Reiseangeboten |

**INFO 32:**

| | |
|---|---|
| \BENCH | |
| BENCH | PC-Magazine Benchmarks Version 4.00 |

**INFO 33:**

| | |
|---|---|
| \ASM | |
| COMPARE | Utility zum Vergleichen von Dateien (Assembler) |
| \PK361 | |
| PK361 | Archivutility von PKWARE Version 3.61 |
| \PASCAL | |
| INLINE | Inline-Assembler für Turbo-Pascal (Pascal) |
| UNINLINE | Inline-Assembler für Turbo-Pascal (Pascal) |

**INFO 34:**

| | |
|---|---|
| \GAME | |
| BOWL | Bowling Spiel für CGA |
| \ASM | |
| ALLKEYS | Utility zum Abschalten sämtlicher Hot-Keys (Assembler) |
| CALC | Residenter Taschenrechner (Assembler) |
| RECORDER | Residentes Programm, das alle Dateizugriffe auf-zeichnet (Assembler) |
| STAYDOWN | Residentes Programm, das die Eingabe von zwei-Tasten Ctrl-, Alt- und Shift-Kombinationen als zwei separate Anschläge erlaubt (Assembler) |
| \TEXT | |
| DBASE4A | Erfahrungsbericht dBASE IV, erster Teil |
| DOS4 | Erfahrungsbericht DOS 4.0 |

**INFO 35:**

| | |
|---|---|
| \ASM | |
| LOG | Utility, das ein Logbuch der Aktivitäten auf dem PC erstellt (Assembler) |
| TOGGLE | Utility zum softwaremäßigen Schalten von Num-, Caps- und Num-Lock (Assembler) |
| \BASIC | |
| BINDEZ | Prozeduren zum Umwandeln von Zahlenformaten (Dez, Hex, Bin) |
| DEZBIN | (Basic) |
| DEZHEX | ^ |
| HEXDEZ | ^ |
| UMSETZEN | ^ |
| \PASCAL | |
| GROSS | Programm zum Formatieren von Pascal-Programmen (Pascal) |
| \UTILITY | |
| EMACS | Editor, der auf vielen Rechnertypen verfügbar ist |
| LANDM | Benchmarkprogramm |

**INFO 36:**

| | |
|---|---|
| \ASM | |
| MAKEBAS | Utility zum Umwandeln einer Binärdatei in ein ASCII-Basic Programm zur Datenübertragung (Assembler) |
| ZCOPY | Utility zum Kopieren von Dateien zwischen zwei Rechnern über die serielle Schnittstelle (Assembler) |
| \GAME | |
| NYET | Spiel (für alle Grafikkarten geeignet) |
| TOPPGUN | Flipperspiel für CGA (Läuft mit SIMCGA) |
| \UTILITY | |
| CORETEST | Benchmarkprogramm für die Festplatte |

**INFO 37:**

| | |
|---|---|
| \GAME | |
| SKYRUNNER | Spiel für die CGA (läuft mit SIMCGA) |
| \PKZIP | |
| PKZIP092 | Nachfolger der Archivierungssoftware PKPAK |
| \ARC2ZIP | |
| A2Z | Utility zum Unwandeln von ARC/PAK-Archiven in ZIP-Archive (Pascal) |
| \WORD | Word-Makros aus PRISMA 1/89 |

**INFO 38:**

| | |
|---|---|
| \PIBCAT | |
| PIBCAT | Festplattenkatalogprogramm (Pascal) |

\ASM
ANSID — Verbesserter ANSI-Treiber (Assembler)

**INFO 39:**
\ASM
ATLAHM — Utility zum "Bremsen" von schnellen Rechnern (für Spiele, etc.) (Assembler)
BCOPY — Speicherresidentes Programm zum Kopieren von Dateien im Hintergrund (Assembler)
DOS4COMP — Utility, das eine DOS 4.0 Inkompatibilität beseitigt (Assembler)
LITES — Zeigt den Status der seriellen Schnitstelle an (Assembler)
MUL32TST — Testet 80386 Prozessoren auf korrektes Multiplizieren (Assembler)
SMOOTH — Browser für EGA/VGA (Assembler)
SPLIT — Teilt große Dateien auf mehrere Disketten auf (Assembler)
ZEIT — Stellt die DOS-Zeit auf die CMOS-Zeit eines AT (Assembler)
\DBASE
LITPROG — Ein Datenbankverwaltungsprogramm
\GAME
OFFROAD — Spiel für CGA (läuft auch mit SIMCGA 4.0)

**INFO 40, 41 und 42:**
\ASM
CMOS — Universelles Setup für AT-Rechner (Vorsicht nur für Fortgeschrittene !!! / Assembler)
DCACHE — Festplattencache (Assembler)
EMS40 — Expanded Memory Emulator für ATs (Assembler)
NEC — Residentes Druckerutility für NEC Drucker (Assembler)
\PASCAL
GROSS — Programm zum Formatieren von Pascalprogrammen (Neue Version) (Pascal)
\DBASE
ADRESSEN — Adressenverwaltungprogramm (dBASE III+)
\BENCH5 — PC-Magazine Benchmarks 5.0

**INFO 43 und 44:**
\CMDPOST
CMDPOST — Ersetzt das MS-DOS-Window von MicroSoft Windows
\PKZIP
PKZ101 — Der Archiver PKZIP in der neuen Version 1.01
\UTILITY
SHEZ45 — Utility zum Bearbeiten von Archiven
\WINDOWS — Sammlung von WINDOWS-Programmen
BACKGAM — gleichnamiges Spiel
BROWSER — Datei-Browser
HUNTER — Datei-Sucher
TETRIS — Spiel
GCP — Konvertierprogramm für Grafikdateien

**INFO 45:**
\WINDOWS
ALARM — Wecker
CLOSER — Utility zum Schließen von Programmen
TICTAC3D — Spiel
\INH42
INH42 — Programm zum Drucken des Diskketteninhaltes
\HP16
XACT16 — HP-16 Emulator
\HP11
XACT11 — HP-11 Emulator
\SPIELE
MLSHUT — Würfelspiel

**INFO 46:**
\ASM
DIRMATCH — Utility zum Abgleichen von Verzeichnissen (Assembler)
\WINDOWS
FIFTEEN — Spiel
\CONFMT
CONFMT — Diskettenformatierprogramm
\COPYQM
COPYQM — Diskettenkopierprogramm
\FORMATQM
FORMATQM — Diskettenformatierprogramm

**INFO 47:**
\FRIEDL
FRIEDL — Statistikprogramm
\WINDOWS
APPLETS — Utilitysammlung für MS-Windows

**NFO 48:**
\CCD2018 — Beitrag von Arndt Klingelnberg (2018)
COOKIE45 — Sprücheklopfer
KEY-FAKE — Übergibt Tastatureingaben an Programme
LETTER — Tastaturabfrage für Batchdateien
PUSHDIR — Sichert Verzeichnissnamen
POPDIR — Wechselt in mit PUSHDIR gesichertes Verzeichnis
READATTR — Gibt Dateiattribute in ERRORLEVEL-Variable aus
UT — Batch "Enhancer"
WAITN — Wartet eine angegebene Zeit
GLOBAL — Läßt Befehle auf alle Unterverzeichnisse wirken (VORSICHT !)
K3 — Tastaturtreiber
PR-TEST — Druckertester
HICARDAT — Programm zur Hauptspeichervergrößerung
SWAPCOM — Tauscht COM1: und COM2: außerdem zahlreiche Batchdateien

**INFO 49:**
\SCAN
SCAN — Virus-Suchprogramm
VIRLIST — Liste der bisher bekannten Viren
\UTILITY
FREE — Stellt die Belegung der Platte grafisch dar
PYRO — Feuerwerk auf dem Bildschirm
\ASM
CHKFRAG — Ermittelt die Zersplitterung der Dateien auf der Platte

**INFO 50:**
\PKZ110
PKZ110EX — Exportversion 1.10 des Archivers PKZIP
\ICE
ICE — Nachfolger des Archivers LHARC
\LHX
LHX — Utility zum Retten von beschädigten LZH/ICE-Archiven

**INFO 51:**
\ASM
PRUNE — Utility zum Manipulieren von Verzeichnisästen. Arbeitet wahlweise mit DIRMATCH und DR (INFO 20) zusammen. (Assembler)
DIRMATCH — Version 2.0 des Utilities zum Abgleichen von Verzeichnissen (Assembler)
\UTILITY
SYSID5 — Programm das diverse Systemparameter anzeigt

**INFO 52:**
\WINDOWS
KLOTZ — Spiel für WINDOWS
\SPIELE
MAHJONGG — Chinesisches Denkspiel (Eine Sammlung von Spielsteinen und ein Programm zum Erstellen von Spielsteinen auf einer der nächsten INFOs !)
\ZINSEN
ZINSEN — Programm zur Zinsberechnung

**Inhalt von INFO 53:**
\UTILITY
AWARD — Award System Benchmark
\SNAP4
SNAP4 — dBASE-Programmdokumentator

**Inhalt von INFO 54:**
\COPYQM22
COPYQM — Leistungfähiges Diskettenkopierprogramm (Version 2.2)
\BATCHMAN
BATCHMAN — "Batchenhancer" Utility (Assembler)
\PAS
MEMMAP — Zeigt die Speicherbelegung an (Pascal)
\SPIELE
TILES — Spielsteine für Mahjongg (INFO 52)

*****

Alle bisher erschienen INFO's sind vorrätig. Bezugsbedingungen siehe unter Serviceleistungen (vorletzte Seite) in diesem Heft.

# Pas de deux

## Programme für HP-28S und HP-48SX

Beide Rechner gleichen sich in der Art der Programmierung so, daß viele HP-28S Programme auch auf dem HP-48SX laufen oder optimiert werden können. Die folgende Programmsammlung enthält daher auch Entwicklungen für den HP-28S. Vor den Programmen steht in jedem Listing der Rechnertyp, für den es geschrieben wurde. Außerdem halten sich alle Programme an folgende Konventionen: Die Inhalte von Stackebenen, die *nicht* als Eingabe dienen, bleiben erhalten, die Eingaben im Stack für Programme werden durch die Ergebnisse ersetzt (also wie bei Standardfunktionen), keine Verwendung finden CLEAR oder KILL. Für lokale Variablen verwende ich nur Kleinbuchstaben außer e und i (die numerischen Konstanten), so daß bis auf die interaktiven Programme alle als Unterprogramme nutzbar sind.

Für die A-Modelle des HP-48SX weist HP auf einen Bug hin, bei dem die Matrixinversion mit 1/X bzw. INV fehlerhaft sein kann. Dazu wurde mit der Bedienungsanleitung das Programm MINV vertrieben. Mein Vorschlag enthält eine zusätzliche 1-Schritt-Residuumskorrektur, sodaß **MINV** (funktioniert wie INV) noch genauer ist als INV.

Das Beste am Infrarotdrucker ist wahrlich sein Preis. In PRISMA 88.5.56 befindet sich ein Bild und die Erklärung der Arbeitsweise des Druckers. Bei einem HP-19C habe ich die Druckmechanik (entspricht der des IR-Druckers) bereits zerlegt, und ordentlichen Verschleiß festgestellt. Außerdem habe ich schon Ausdrucke gesehen, bei denen senkrechte Linien versetzt wurden, ein Problem, das nach einer Schmierung verschwand. Hier deshalb etwas für Schmierfinken: Zuerst besorgt man sich Fett. Ich verwende Vaseilne (z.B. in der Drogerie als medizinische Vaseline erhältlich) da es nicht verharzt und zäh ist (je zäher das Fett, um so besser die hydrodynamische Tragfähigkeit, unter günstigen Bedingungen bedeutet das kein Verschleiß, weil das Fett die reibenden Teile vollständig trennt!). Jetzt schaltet man den Drucker kurz ein - der Druckkopf läuft sofort los - bis er ungefähr in der Mitte steht, und schaltet schnell wieder aus. Das braune Sichtfenster erst rechts, dann links mit einem dünnen (!) Schraubenzieher heraushebeln und nach hinten (= in Richtung des Papiervorschubs) herausschieben. Jetzt etwas Vaseline auf den Druckkopf geben (da wo die Achse durchläuft) und auf die darunterliegende Welle (sie dreht sich bei laufendem Motor, und sieht ziemlich zerfurcht aus) links und rechts vom Druckkopf je einen stecknadelkopfgroßen Vaselinehaufen. Drucker kurz einschalten, das Fetten ggf. wiederholen, dann Sichtfenster wieder aufschieben und dessen Haken mit Hilfe des Schraubenziehers einrasten lassen.

Die Möglichkeiten der Programmdokumentation auf dem HP-48SX sind etwas mager. Daher hier der Programmzeilendrucker für den HP-48SX. **PRP** druckt Programme mit Zeilennummerierung, Bytebedarf, Prüfsumme und einen Hinweis auf den HP-48SX (damit keine Verwechselungen mit dem HP-28S-Programmen entstehen) aus. Bytes und Prüfsumme entstehen, wenn das Programm in Ebene 1 steht und Bytes ausgeführt wird (man kann BYTES auch auf den Namen anwenden, was aber diese beiden Werte ändert). PRP erwartet in Ebene 1:

a) Ein Programm, welches in der Kopfzeile Datum und Uhrzeit des Ausdrukkes enthält.

b) Den Namen eines Programmes, welcher dann fettgedruckt erscheint.

c) Den Namen eines Directorys, dessen Programme fettgedruckt werden sollen (enthält es weitere Subdirectories, werden auch deren Programme gedruckt). Im Ausdruck erscheinen keine Hinweise auf den Namen des Directorys. Erkennt PRP ein Directory, ruft es sich selbst auf.

d) Eine Liste, die eine beliebige Mischung der unter a, b und c beschriebenen Objekte enthält.

Leider akzeptiert das Programm PRP weder die Directories HOME noch PORT0, PORT1 und PORT2, weil das eingebaute Funktionen sind. Programme in diesen Menues müssen also in eine Liste eingegeben werden.

Einige Hinweise zum "Umstricken" der Programme (die Klammern beziehen sich auf Hinweise für's 28S Programm).

Zeile 27 (16) enthält eine 2, die jede zweite Zeile numeriert. Diese Zahl kann natürlich beliebig geändert werden. Um den Ausdruck lesbarer zu gestalten, habe ich viele Zeichen mit der CHR-Funktion erzeugt. Diese kann man natürlich durch Strings ersetzen, so den Normal- und Fettdruck in Zeile 10-12 (7+8 an "28S" anhängen, 10, 11) oder CR = 10 CHR der sogar auf den Tastaturen in Zeile 24, 30, (19) zu finden ist. Wenn man den Namen der 48SX Version ändert, muß man den Neuen auch in Zeile 17 ändern, hier beginnt nämlich die Rekursion.

Die HP-28S-Version von PRP druckt keine Uhrzeit und keine Directorys und bricht Strings mit mehr als 18 Zeichen um, funktioniert aber sonst wie die 48SX Version.

**LNG** mißt auf dem 28S jetzt die Bytes wie BYTES im 48SX. **CHKSUM** berechnet eine Prüfsumme für das Objekt in Ebene 1 (steht dort ein 'Name', dann für das in ihm gespeicherte Objekt).

**PRDIR** (nur HP-48S) druckt eine Liste mit den Namen und Typen aller Objekten im gewählten Directory (einschließlich der Subdirectorys) aus und rückt sie entsprechend der Ordnung ein. Das besondere an PRDIR ist die rekursive Struktur, wobei der arbeitende Teil des Programms selbst in einer lokalen Variablen (p) von PRDIR steckt. Das ist notwendig, weil zum Programmstart noch ein leerer String auf den Stack geschoben und der Name des aktuellen Directorys gedruckt wird, was sich aber bei keinem rekursiven Aufruf wiederholen darf. Die 'FORWARD'- Definition von Pascal kennt der HP-48SX nicht. In Pascal muß (wie beim 48SX) immer zuerst die Variable erzeugt werden, dann kann man sie überhaupt erst verwenden - nachträgliches Erzeugen akzeptiert der Rechner nicht!

In einigen Fällen ist das sogar unmöglich: Dann, wenn zwei Prozeduren (Programmteile) sich gegenseitig aufrufen (haben wir hier zwar nicht, lässt sich aber wie später beschrieben lösen), sagt man dem Computer mit FORWARD "Moment, da kommt gleich dieser Name: ... , aber die Erklärung (=die Prozedur) kommt erst nach diesem Namen". Wie gesagt, dieses FORWARD kennt der 48SX nicht, daher in PRDIR folgenden Trick: Zuerst erzeugt man die Variable p (0 in p speichern, Zeile 3), dann kommt das Programm mit der Variablen p (Zeile 5-24), und schließlich speichert man das Programm in p (und überschreibt so die 0, Zeile 25). Jetzt kann sich p selbst aufrufen (Zeile 26, p wurde in Zeile 25 mit DUP kopiert). Noch einmal: Hätte ich in PRDIR zuerst das Unterprogramm (Zeile 5-24) geschrieben, hätte der 48SX p für eine globale Variable gehalten, und seine Meinung auch nach dem Erzeugen von p nicht mehr geändert. Der HP-48SX hätte fest an die Existenz von zwei p's geglaubt, einem lokalen, und einem globalen (welches natürlich nicht existiert). Der Kern des Programmpaketes Trigonometrie (Vorschlag: alle Programme in ein Directory mit dem Namen TRIG, Anordnung der Programme wie im Beispiel zu PRDIR) ist das Programm **TRI**.

Andere Programme können es aufrufen, wenn sie zuvor die 6 Variablen A,B,C, $\alpha$, $\beta$, $\gamma$ mit Daten füttern: Eine 0 für Unbekannte, und (mind. 3) Zahlenwerte für die bekannten Stücke (=Seiten oder Winkel) des Dreiecks.

TRI rechnet im eingestellten Winkelmo-

dus, so daß auch gegebene Winkel in diesem Winkelmaß angegeben werden müssen. Nach Programmende steht in Ebene 1 ein FLag: Eine-1, falls sich aus den Eingaben zwei unterschiedliche Dreiecke konstruieren lassen; nur in den letzten beiden Fällen enthalten die 6 Variablen alle Stücke des Dreieckes in richtiger Folge. Es gibt insgesamt 20 Möglichkeiten, genau drei Stücke eines Dreieckes zu benennen. Der Fall WWW (W=Winkel, hier: 3 gegebene Winkel) ist allerdings nicht eindeutig, da sich beliebig viele Dreiecke mit drei gleichen Winkeln konstruieren lassen. Betrachtet man den Fall SSS (S=Seite, hier: 3 gegebene Seiten) zunächst auch nicht, so bleiben 18 Möglichkeiten übrig, die man durch zyklische Vertauschung (das macht **ZYKL**) auf nur 6 Fälle zurückführen kann. das Programm verwendet zur Lösung 7 Formeln und eben die zyklische Vertauschung von Seiten und Winkeln. In allen Fällen berechnet TRI den dritten Winkel aus der Winkelsumme, die im Dreieck 180° (in Altgrad=DEG-Modus) beträgt, und welche TRI durch -1 ACOS erzeugt, sodaß - je nach Winkelmodus - 180, 3.14 oder 200 als Winkelsumme erscheint. In manchen Fällen existiert entweder kein Dreieck, welches zu den Eingaben passt, oder bei der Eingabe wurden mehr als 3 Stücke gegeben, die nicht zusammenpassten- diesen speziellen Fall prüft TRI mit Hilfe der Moivreschen Formeln (Zeilen 54-60); In einigen Fällen gibt es sogar zwei Lösungen. Diese Gefahr besteht im Fall SSW (zwei Seiten und ein Winkel, der einer Seite gegenüberliegt). Wenn in diesen (6 von 19) Fällen tatsächlich ei zweites Dreieck existiert, gibt TRI eine 1 statt einer 0 in Ebene 1 zurück. Die unbekannte Seite kann nun zwei Längen haben. TRI wählt die größere, und damit auch das Dreieck mit der größeren Fläche.

Δ ist ein interaktives Programm für die Handarbeit am Dreieck (das Schachbrett im Listing soll ein Dreieck, ein Buchstabe C mit blauer Vortaste, sein, was der 82240A IR-Drucker nicht darstellen kann). Nach dem Start schaltet sich das SOLVR-Menü ein, und mit einem Tastandruck kann man die bekannten Stücke speichern (jede Menütaste "schluckt" den Wert in Ebene 1), wobei man nur die bekannten Stücke eingibt, die Unbekannten nimmt das Programm automatisch als 0 an. Ganz klar, daß man immer die kanonischen Bezeichnungen des Dreiekkes wählt, also wie in der Mathematik üblich ist: Seite A liegt dem Winkel ∝ gegenüber u.s.w., Seiten und Winkelwerden im Gegenuhrzeigersinn benannt. Bevor man mit CONT startet, kann man mit REVIEW noch einmal alle 6 Stücke überprüfen und erhält danach die die Unbekannten als benannte Zahlen (='tagged objects') zurück; REVIEW hilft auch nach TRI, blättert man auf die zweite Seite des TRI-Menüs, stehen hier alle 6 Stücke des Dreieckes.

Die Programme HOEHE; SHALB (=Seiten HALBierende), WHALB (=Winkel-HALBierende) und FLKR (FLäche des dreieckes & radien des in- und umKReises) können erst nach TRI verwendet werden, und geben die gesuchten Größen als bekannte Zahlen aus. Höhen stehen auf der Seite, deren Namen sie tragen, senkrecht. Die Höhen müssen nicht unbedingt im Dreieck liegen. Hat das Dreieck einen stumpfen (>90°), dann liegen zwei der Höhen außerhalb des Dreieckes. TRI verarbeitet nur reelle Zahlen (Type=0), also keine symbolischen Variablen. Diese Bedienungsanleitung gilt auch für die HP-28S Programmversionen TRI, ECK3 (entspricht Δ, organisiert aber nur die Eingabe) und ZYKL.

Das Programm **GLN** nimmt aus Ebene 2 ein Polynom in beliebiger (!) Form, und sucht dessen Koeffizienten heraus. Dabei muß in Ebene 1 der 'Name' der Variablen stehen, und das Programm gibt eine Liste zurück die alle Koeffizienten (als solche sind auch Name und Funktionen zugelassen) enthält. Das erste Element dieser Liste enthält das absolute Glied des Polynoms, Nr. 2 enthält den Koeffizienten des linearen Gliedes u.s.w.; Die Liste enthält mindestens ein Element und bei mehr als einem ist das letzte niemals eine Null. Natürlich kann man GLN auch mit einer transzendenten Funktion (z.B. SIN) in Ebene 2 füttern. Dann sucht das Programm die Taylorentwicklung dieser Funktion heraus. Da diese im allgemeinen unendlich viele Glieder hat, erfolgt das Programmende erst bei Speicherüberlauf.

Das Programm **OF** berechnet den Ostersonntag aus der Jahreszahl in Ebene 1. Eine wichtige Ergänzung des TIME-Menüs, schließlich hängt von Ostern beispielsweise Karneval ab (in Klammern: Eingabe für DATE+ um auf diesen Tag zu kommen), mit Weiberfastnacht (-52), Rosenmontag (-48), Aschermittwoch (-46), dann natürlich die kirchlichen/gesetzlichen Feitage wie Karfreitag (-2), Ostermontag (+1), Christi Himmelfahrt (+39), Pfingstmontag (+50) und Fronleichnahm (+60).

Sicher hat sich schon so mancher gefragt, welchen Regeln die Umstellung von Sommerzeit (MESZ) und Winterzeit (MEZ) folgt. Die Lösung liegt im Zeitgesetz vom 25.07.1978 (s. Bundesgesetzblatt 1978, S. 1110 - 1111), nämlich keinen. Dieses Gesetz verlangt nur, daß die Zeitumstellung an einem Sonntag zwischen dem 1. März und 20. Oktober erfolgt. Im Übrigen ermächtigt es die Bundesregierung, das genaue Datum, Uhrzeit und die Bezeichnung der im Herbst doppelt erscheinenden Stunde durch eine Rechtsverordnung festzulegen.

Diese hat verordnet - zuletzt im BGBl Nr. 9 vom 7.03.1989: **am 25.03 1990, 31.03.1991 und am 29.03 1992 um 2:00 h ist die die Uhr auf 3:00 h vorzustellen, am 30.09.1990, 29.09.1991 und am 27.09.1992 stellt man die Uhr von 3:00 h auf 2:00B (2 Uhr erscheint an diesen Tagen zweifach, beim ersten mal heißt sie 2:00A, beim zweiten mal 2:00B) zurück. Ein Vorschlag für die entsprechenden Steueralarme liegt bei. Mit meiner ungenauen Rechneruhr, sie genehmigt sich eine Sekunde in zwei Tagen, rücke ich ebenfalls mit einem Steueralarm (jeweils nächstens um 500 h) zu Leibe.**

Die hervoragenden Möglichkeiten, mit den Funktionen INPUT und TMENU Eingabe und Bedienung eines Programmes zu steuern, zeigt **DKAL** (lcDKALender). Dieses Programm ruft zunächst →**MJ** auf, ein Unterprogramm, welches in Ebene 1 einen Textstring als Überschrift der Eingabe erwartet, und dann nach dem gewünschtem Monat und Jahr fragt. Dabei bietet →MJ den aktuellen Monat bzw. das aktuelle Jahr an, so daß man mit einem einfachen ENTER davonkommt, falls der Anwender gerade diese Daten wünscht. Im weiteren Verlauf gibt DKAL den Monat aus (s. Ausdruck) und erzeugt ein Menu, welches drei Möglichkeiten bietet: Letzten Monat anzeigen, nächsten Monat anzeigen oder aussteigen. Die Programmiertechnik ist dabei folgende: DKAL legt ein "schlafendes" Programm auf den Stack, welches den Kalender anzeigt und die Menütasten anlegt. Auch die Menütasten erhalten kleine Programme, die das schlafende Programm kopieren und aktivieren.

Einige Vorschläge zu den Zweitfunktionen der Ziffer 1, die sich mit ASN und den folgenden Programmen verbessern lassen (Progarmme nicht im VARS-Menü speichern, sondern gleich komplett zuweisen). RAD schaltet zwischen DEG und RAD Modus hin und her. Nimmt man nun das unter ASN, RAD abgedruckte Programm in Ebene 2 und eine 82,2 in Ebene 1, dann schaltet die RAD-Taste zwischen DEG, RAD und RAD hin und her. POLAR wechselt zwischen kartesischen und Zylinderkoordinaten. Speichert man das Programm ASN, POLAR in Ebene 2, eine 82,3 in Ebene 1, dann schaltet POLAR zwischen kartesischen und Zylinder- und Kugelkoordinaten hin und her.

Wenn man ein Programm aus dem VARS-Menü mit ASN zuweisen will, nimmt man dessen Namen entweder alleine oder als Programm mit nur einem Befehl, den Namen. Der unterschied: Der Name allen läßt sich in Progarmme oder algebraische Ausdrücke einschließen. Für Programme wie PRP ist dagegen sinnvoller, diesen Namen als einzigen Befehl eines Programmes einzugeben, denn dieser bricht automatisch die algebraische Eingabe ab.

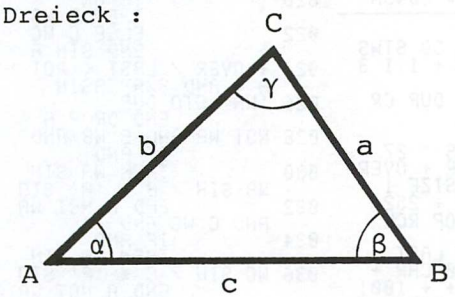
## Byteverbrauch in HP-48SX-Programmen

| | |
|---|---|
| Einfache Befehle & Operationen (z.B. SIN, OR) | 2,5 |
| Namen (lokal & global) | 3,5 + n |
| 'Name' (lokal & global, mit '') | 8,5 + n |
| → ... ≪, →... ' (Definition lokaler Variabler) | 7,5 + Namen |
| Zahlen, ganzzahlig von -9...9 | 2,5 |
| übrige reelle | 10,5 |
| komplexe (...;...) | 18,5 |
| binäre (# ...), Wordsize egal | 13,0 |
| ...FOR...NEXT/STEP | 5 + lokaler Name |
| ...START...NEXT/STEP | 5,0 |
| IF...THEN*...END | 12,5 |
| IF...THEN*...ELSE*...END | 20,0 |
| IFERR*...THEN*...END | 17,5 |
| IFERR*...THEN*...ELSE*...END | 25,0 |
| WHILE...REPEAT*...END | 12,5 |
| DO...UNTIL...END | 7,5 |
| CASE...THEN*...END...END | 20 + 10 je THEN*...END |
| '...' (algebraische Objekte im Programm) | 5,0 |
| IFTE(...;...;...) Test in algebraischen Objekten | 12,5 |
| Strings "..." | 5 + n |
| Listen {...} | 5 + Inhalt |
| Vektoren [...] , reell/komplex | 12,5 + 8/16 je Zahl |
| Matrizen [[...]] , reell/komplex | 15 + 8/16 je Zahl |
| Programme ≪...≫ , im Stack | 10,0 |
| in Namen gespeichert | 11 + Name |
| in einem anderen Programm | 12,5 |
| Benannte Objekte : ... : ... | Name (=3,5 + n) + Objekt |

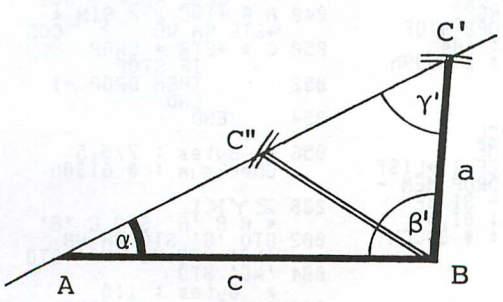
n = Anzahl einfacher Alphazeichen (Buchstaben).
* = Befindet sich hier genau ein Objekt mit 2,5 Bytes, so spart man 2,5 Bytes (= Struktur -5. Objekt +2,5).

**Siehe auch Seite 20: "Zusatz zur Byte Tabelle"**

Kanonische Bezeichnungen im Dreieck :




Zwei Lösungen im Fall SSW (hier: a, c und α gegeben), falls: c•sin α < a < c.




```
48SX ▒
    ≪ { ZYKL A B C α β
002 γ 3 } DUP OBJ→
      START 0 SWAP STO
004   NEXT STEQ → 1
      ≪ 30 MENU
006 "Nach Eingabe CONT"
    PROMPT { } 2 7
008     FOR n 1 n GET
          IF DUP RCL
010       THEN DROP
          ELSE +
012       END
        NEXT
014   » 0 MENU TRI
      CASE DUP -1 SAME
016     THEN DROP2
    "Geht nicht" DOERR
018     END
        THEN 1 FREEZE
020 "2 Dreiecke" 1 DISP
      END
022   END 1
      DO GETI RCL
024 LASTARG →TAG ROT
    ROT
026   UNTIL -64 FS?
      END DROP2
028 »  Bytes : 307
    Checksum : # 58C7h
```

Nach Eingabe CONT

```
4:
3:              A: 3
2:              B: 4
1:            γ: 90
 A | B | C | α | β | γ
```

```
                USER
≡E MATH TRIG ) 09.06.90   20:13:55
4:
3:                  C: 5
2:     α: 36,8698976458
1:     β: 53,1301023542
 △ | HOEHE | SHALB | WHHL | FLKR | TRI
```

```
48SX TRI
    ≪ DEPTH → n
002   ≪
        IFERR 0 1 5
004       START
            CASE β y
006 AND α NOT AND
            THEN -1
008 ACOS β - y -
            END A NOT
010 α OR
            THEN α
012         END B C
    AND
014         THEN B SQ
    C SQ + A SQ - 2 B *
016 C * / ACOS
            END B β
018 AND DUP C y AND OR
            THEN
020           IF OR
              THEN B
022 β
            ELSE C
024 y
            END SIN
026 A * OVER / LASTARG
    < ROT A < AND SWAP
028 ASIN
            END DROP
030 α
          END 'α' STO
032       CASE α NOT
    A OR
034         THEN A
            END B β
036 AND
            THEN α
038 SIN B * β SIN /
            END C y
040 AND
            THEN α
042 SIN C * y SIN /
            END B C
044 AND
            THEN B SQ
046 C SQ + 2 B * C * α
    COS * - √
048         END A
          END 'A' STO
050 ZYKL
        NEXT ZYKL
052   THEN DEPTH n -
    DROPN -1
054   ELSE
        IF A B + y 2
056 / SIN * -9 RND α β
    - 2 / COS C * -9
058 RND ≠
        THEN DROP -1
060     END
      END
062   »
    »  Bytes : 669
064 Checksum : # FDD7h
```

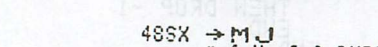
```
48SX ZYKL
    ≪ A B 'A' STO C 'B'
002 STO 'C' STO α β 'α'
    STO γ 'β' STO 'γ'
004 STO
    »  Bytes : 109
006 Checksum : # 25B5h
```

```
48SX HOEHE
    ≪ "Ha " "Hb " "Hc "
002 1 3
      START B γ SIN * 4
004 ROLL →TAG ZYKL
      NEXT
006 »  Bytes : 73
    Checksum : # 4F72h
```

```
48SX SHALB
« "Sa " "Sb " "Sc "
002  1 3
     START B SQ C SQ +
004  2 * A SQ - J 2 / 4
     ROLL →TAG ZYKL
006    NEXT
     » Bytes : 97,5
008  Checksum : # BBDCh

48SX WHALB
« "Wα " "Wβ " "Wγ "
002  1 3
     START α 2 / COS 2
004  * B INV C INV + / 4
     ROLL →TAG ZYKL
006    NEXT
     » Bytes : 95
008  Checksum : # 98B4h

48SX FLKR
« A α SIN / 2 /
002  "Ru " →TAG B C + A
     - α 2 / TAN * 2 /
004  "Ri " →TAG A B * γ
     SIN * 2 / "Fl "
006  →TAG
     » Bytes : 124,5
008  Checksum : # 91D5h

48SX DKAL
« RCLF STD
002    « CLLCD 3 FREEZE
     DUP 32 DATE+ FP 1 +
004  DUP2 DDAYS 3 *
     " Mo Di Mi Do Fr Sa
006  So"
     "
008  18,101582 7 PICK
     DDAYS 7 MOD SUB DUP
010  DUP + + +
     "  1  2  3  4  5  6
012    7  8  9 10 11 12
     13 14 15 16 17 18 1
014  9 20 21 22 23 24 25
      26 27 28 29 30 31"
016  1 4 ROLL SUB + 1 7
       FOR n 1 21 SUB
018  LASTARG + OVER SIZE
     SUB SWAP n DISP
020    NEXT DROP SWAP
     -2 DATE+ FP 1 +
022    « IF1 OVER EVAL
       »
024    « DROP OVER
     EVAL
026    » 1 2
       START 4 ROLL
028  DUP 0 TSTR 8 12 SUB
     ROT 2 →LIST ROT
030    NEXT ( ) DUP
     DUP ( EXIT (
032    « DROP DROP2
     STOF 2 MENU
034    » ) ) 6 →LIST
     TMENU
036    »
     "Anzeige des Monats
038  :"
     →MJ 10000 / + 1 % 1
040  + OVER EVAL
     » Bytes : 533
042  Checksum : # 10B8h
```

(Kalender für Dez. 90)

```
 Mo Di Mi Do Fr Sa So
                 1  2
  3  4  5  6  7  8  9
 10 11 12 13 14 15 16
 17 18 19 20 21 22 23
 24 25 26 27 28 29 30
 31
```




```
48SX →MJ
     « ( V -9 ) DUP2
002    ":Monat:  " 1 DATE
     FP %T IP + + INPUT
004  OBJ→ ROT ROT
       ":Jahr:   " 1 DATE
006  %T FP 10000 * + +
     INPUT OBJ→
008  » Bytes : 111
     Checksum : # E4C7h

48SX MINV
     « ( 3 4 )
002    IF OVER TYPE POS
       THEN DUP SIZE 1
004  GET IDN OVER /
     LASTARG 3 PICK RSD
006  RCI / +
       ELSE INV
008    END
     » Bytes : 80
010  Checksum : # 31FCh
```

```
48SX PRP
« ( ) + RCLF STD 8
002  SQ STWS HEX SWAP 1
     1 3 PICK SIZE
004    START GETI 8 OVER
         CASE TYPE SAME
006        THEN DATE
     TIME TSTR 1 19 SUB
008        END DUP VTYPE
     8 SAME
010        THEN 27 CHR
     253 CHR + OVER + 27
012  CHR + 252 CHR +
     SWAP RCL SWAP
014        END EVAL 8
     TVARS 15 TVARS +
016      IF DUP SIZE
         THEN PRP 0
018      END UPDIR
       END DUP
020    IF TYPE 2 SAME
       THEN CR "48SX "
022  SWAP + PR1 DROP DUP
     "Bytes : " + SWAP
024  BYTES 10 CHR +
     "Checksum : " +
026  SWAP + + 1001
         DO DUP 2 MOD
028  NOT OVER "
     IFTE "  " + ROT + 2
030  OVER 10 CHR POS DUP
           CASE NOT
032          THEN DROP
     OVER SIZE
034          END DUP
     26 >
036          THEN DROP
     OVER SIZE SUB 1 24
038          END 1 -
           END SUB
040  LASTARG + OVER SIZE
     SUB SWAP PR1 DROP
042  SWAP 1 + OVER SIZE
       UNTIL NOT
044    END DROP2
     ELSE DROP
046  END
   NEXT DROP2 STOF
048  » Bytes : 576,5
     Checksum : # 962h

48SX PRDIR
« CR PATH DUP SIZE
002  GET " DIR :" + 3
     →GROB PR1 DROP "  "
004  0 → P
     «
006    « VARS
         IF DUP SIZE
008      THEN 1 1
     LASTARG
010        START GETI
     DUP VTYPE 5 PICK
012  OVER 100 + " " + 2
     OVER SIZE SUB + 3
014  PICK + PR1 DROP
           IF 15 ==
016        THEN EVAL
     ROT "  " + P EVAL
018  UPDIR 3 OVER SIZE
     SUB ROT ROT 0
020          END DROP
           NEXT
022      ELSE OVER
     " (Kein Eintrag) "
024  + PR1
       END DROP2
026      » DUP 'P' STO
     EVAL
028  » DROP
     » Bytes : 301,5
030  Checksum : # 7F4Ch
```

TRIG DIR :

```
08 ▣
08 HOEHE
08 SHALB
08 WHALB
08 FLKR
08 TRI
00 A
00 B
00 C
00 α
00 β
00 γ
08 ZYKL
06 EQ
```

(Alarm für Winterzeit)

```
{ 31,031991 2
« 29491200 CLKADJ OFF
» 0 }
```

```
48SX GLN
« -1 → v n
002    «
         WHILE DUP 0
004  SAME NOT
         REPEAT v DUP2 (
006  0 ) + ! 'n' INCR !
     / COLCT ROT ROT ∂
008      END n 1 +
       »
010    WHILE DUP2 SWAP 0
     SAME AND
012    REPEAT SWAP NOT -
       END →LIST +
014  » Bytes : 145
     Checksum : # 8565h

2: 'X^4-2*X*X^3+SIN(A)*X
^3+X^2-10'
1: 'X'

 GLN
1: { -10 0 1 'SIN(A)' -1
}

48SX OF
« DUP DUP2 1900 ≥
002  OVER 2100 < AND /
     19 DUP2 MOD DUP 10
004  < ROT ROT * 24 + 30
     MOD DUP 28 SAME ROT
006  AND 5 ROLLD DUP 6 *
     5 + ROT 7 MOD 4 * +
008  ROT 4 MOD DUP + + 7
     MOD + SWAP 1000000
010  / 22,03 + OVER
     DATE+ ROT ROT DUP
012  35 SAME SWAP 34
     SAME ROT AND OR 7 0
014  IFTE - "So " →TAG
     » Bytes : 286
016  Checksum : # 7030h

1991 OF
So : 31,031991

48SX ASN,RAD
« ( RAD GRAD DEG )
002  -17 FS? LASTARG 1 -
     FS? 2 * + 1 + GET
004  EVAL
     » Bytes : 63
006  Checksum : # CD1Eh

48SX ASN,POLAR
«
002    CASE -16 DUP FC?
         THEN SF
004      END 1 + DUP
     FC?C
006    THEN SF
       END 1 - CF
008  END
   » Bytes : 68
010  Checksum : # C643h
```

```
28S PRP
« RCLF STD 8 SQ STWS
002  HEX SWAP ( ) + 1 1 3
     PICK SIZE
004    START GETI DUP CR
     TYPE 6
006      IF SAME
         THEN "28S " 27
008  CHR + 253 CHR + OVER
     →STR 2 OVER SIZE 1 -
010  SUB + 27 CHR + 252
     CHR + PR1 DROP RCL
012      END →STR
     "   Bytes : " LAST
014  LNG →STR + 10 CHR +
     OVER CHKSUM + + 1001
016      DO DUP 2 MOD NOT
     OVER →STR "  "
018  IFTE " " + ROT + 2
     OVER 10 CHR POS DUP
020  4 PICK SIZE 2 + IFTE
     1 - SUB LAST + OVER
022  SIZE SUB SWAP PR1
     SIZE 24 / CEIL ROT +
024  OVER SIZE
       UNTIL NOT
026    END DROP2
   NEXT DROP2 STOF
028  » Bytes : 404
     Checksum : # 3899h

28S LNG
«
002    IFERR RCL
       THEN CLMF
004    END 31 CF 1 →LIST
     MEM SWAP DROP MEM -
006  5,5 SWAP - 31 SF
     » Bytes : 81,5
008  Checksum : # 33CCh
```

```
'PRP' LNG
1: 404

28S CHKSUM
« RCLF STD HEX SWAP
002  8 SQ STWS
       IFERR RCL
004    THEN CLMF
       END →STR DUP 1
006  OVER SIZE 2 OVER LN
     2 LN / CEIL ^ SWAP -
008  SUB +
       WHILE DUP SIZE 2 >
010    REPEAT 1 OVER SIZE
     DUP 4 ROLLD 2 / SUB
012  LAST + 4 ROLL SUB
     XOR
014    END DUP 2 2 SUB
     NUM SWAP NUM 256 * +
016  R→B →STR SWAP STOF
       "Checksum : " SWAP +
018  » Bytes : 206,5
     Checksum : # D117h

1: "GEHT AUCH MIT «»"

 CHKSUM
1: "Checksum : # B9C9h"

28S ECK3
« « ZYKL A B C WA WB
002  WC 3 ) LIST→
       START 0 SWAP STO
004    NEXT STEQ 24 MENU
     HALT 23 MENU
006    IF TRI
       THEN LAST 1 SAME
008  "2 Dreiecke"
     "Geht nicht" IFTE 1
010  DISP
       END
012  » Bytes : 165
     Checksum : # 8C2Bh

28S TRI
« DEPTH → n
002    «
         IFERR 0 1 5
004        START WA NOT
     WB AND WC
006          IF AND
             THEN -1 ACOS
008  WB - WC - 'WA' STO
           END WA NOT A
010  AND B AND C
           IF AND
012        THEN B SQ C
     SQ + A SQ - 2 / B /
014  C / ACOS 'WA' STO
         END B WB AND
016  DUP C WC AND OR WA
     NOT A AND
           IF AND
018        THEN
             IF OR
020          THEN B WB
             ELSE C WC
022          END SIN A
     * OVER / LAST < ROT
024  A < AND SWAP ASIN
     'WA' STO DUP
026        END DROP A
     NOT WA AND B WB AND
028          IF AND
             THEN WA SIN
030  WB SIN / B * 'A' STO
           END A NOT WA
032  AND C WC AND
           IF AND
034        THEN WA SIN
     WC SIN / C * 'A' STO
036        END A NOT WA
     AND B C AND
           IF AND
038        THEN B SQ C
     SQ + 2 B * C * WA
040  COS * - J 'A' STO
           END ZYKL
042      NEXT ZYKL
       THEN DEPTH n -
044  DROPN -1
       ELSE RCLF 9 SCI
046  A B + WC 2 / SIN *
     →STR WA WB - 2 / COS
048  C * →STR * SWAP
       IF STOF
050    THEN DROP -1
       END
052  END
   »
054  » Bytes : 775,5
     Checksum : # 6130h

28S ZYKL
« A B 'A' STO C 'B'
002  STO 'C' STO WA WB
     'WA' STO WC 'WB' STO
004  'WC' STO
     » Bytes : 115
006  Checksum : # 3BDEh
```

Ralf Pfeifer (116)
Rubensstr. 5, 5000 Köln 50

# Input-Output Board

## für den HP-IL Converter HP82166A Teil 1

**HP41, IL-Modul, Extended I/O, CCD Modul, HP-IL Converter HP82166A**

### Hardware

Das im folgenden beschriebene Input/Output Board für den HP-IL Converter HP82166A soll ein Beispiel sein, diesen leistungsfähigen Baustein für die Datenein- und ausgabe in Meß- und Regelanwendungen einzusetzen.
Die vorgestellte Grundschaltung kann für eigene Applikationen (z.B. 16 Bit Datenbus) modifiziert werden.

Das I/O-Board verfügt über zwei 8-Bit Ports für die Dateneingabe sowie noch einmal weiteren zwei 8-Bit Ports für die Datenausgabe.

Die Adressierung der Ports erfolgt mit den Convertersignalen PWRDN, DCLO und GETO sowie den Handshakeleitungen DAVI und DAVO.
Die dazu nötigen Gatter werden durch zwei Monoflops ergänzt, welche die notwendigen Pulslängen für das System generieren.

Der Anschluß des HP-IL Converters erfolgt ganz einfach mit einem Flachbandkabel und daran angeschlagenen Pfostensteckern.
Ebenfalls mit einem Pfostenstecker werden die notwendigen Versorgungsspannungen und Systemmassen vom Netzteil zugeführt. Dies sind +15V, - 15V, +5V, - 5V sowie IL-Referenzmasse, Digitalmasse und Analogmasse. Zur besseren Störunterdrückung werden die Masseleitungen erst am Netzteil zusammengeführt.

Die vier I/O-Ports erlauben die flexible Anpassung des Systems an diverse Aufgaben. Die Adressenzuordnung erfolgt auf den aufgesteckten Karten mittels Jumper.
Vorgestellt werden 8-Bit Ein- und Ausgabeschaltungen basierend auf dem 74HC574. Dieser Baustein kann für Systemtests mit DIP-Schaltern (bei den Inputs) oder LED's (bei den Outputs) verdrahtet werden, oder er dient als Grundschaltung für eigene Anwendungen wie Ansteuern von Optokopplern, Relais, Digitallogik, Schnittstellen etc. ).

Die Einbindung des Systems in die "analoge Welt" ermöglichen Wandlerkarten mit den 8-Bit Wandlern ADC 0804 und DAC 0808. Die notwendige Referenzspannung von 2,5V wird auf dem I/O-Board erzeugt, die ±15V Versorgungsspannung ermöglicht die Verwendung von Operationsverstärkern und analogen Baugruppen. Mit den Wandlern ist der Betrieb spannungsgesteuerter Baugruppen wie Verstärkern, Filtern usw. möglich, ebenso die analoge Meßwerterfassung von z.B. Temperatur, Druck Schall o.ä..

### Software

Zur Programmierung des Statusregisters des IL-Converters benötigt der HP41 das Extended I/O Modul.

Gewählt wurde für den Betrieb des IL-Converters mit dem I/O-Board negative Handshakelogik, positive Datenlogik, 8 Bit Datenformat bidirektional, Strobed Output, 100µs DAVO-Timeintervall und DAVO Timeout disable (Converterstatusregister R02). Stausregister R03 wird auf 256 Zeiteinheiten gesetzt.
Somit können, in Verbindung mit den Monoflops, Handshakelogikzustände zu Testzwecken mit einfachen LED's überprüft werden.

Mit den Programmlabels *IN1*, *IN2* sowie *OUT1*, *OUT2* können die Steckports direkt adressiert werden; ebenso können Datenbytes ein und aus dem ALPHA-Register gelesen werden. Hierbei ist das führende Dummy-Byte ("D") zu beachten. Diese wird für die Funktionen INAN und OUTAN benötigt, da man sonst kein 0-Byte schicken kann, es würde im ALPHA-Register unterdrückt.

```
01 LBL "IN/OUT"
   CLA
   "D@-*-" (68 64 0 16 0)
   64
   FINDAID
   SELECT
   LAD
   0
   DDL
10 4
   OUTAN
   UNL
   ADRON
   64
   FINDAID
   SELECT
   MANIO
   SF 17
   RTN
```

```
20 LBL "IN1"
   CLRDEV
   TRIGGER
   1
   INAN
   RTN
   LBL "IN2"
   PWRDN
   TRIGGER
   1
30 INAN
   RTN
   LBL "OUT1"
   CLRDEV
   1
   OUTAN
   RTN
   LBL "OUT2"
   PWRDN
   1
40 OUTAN
   END
```

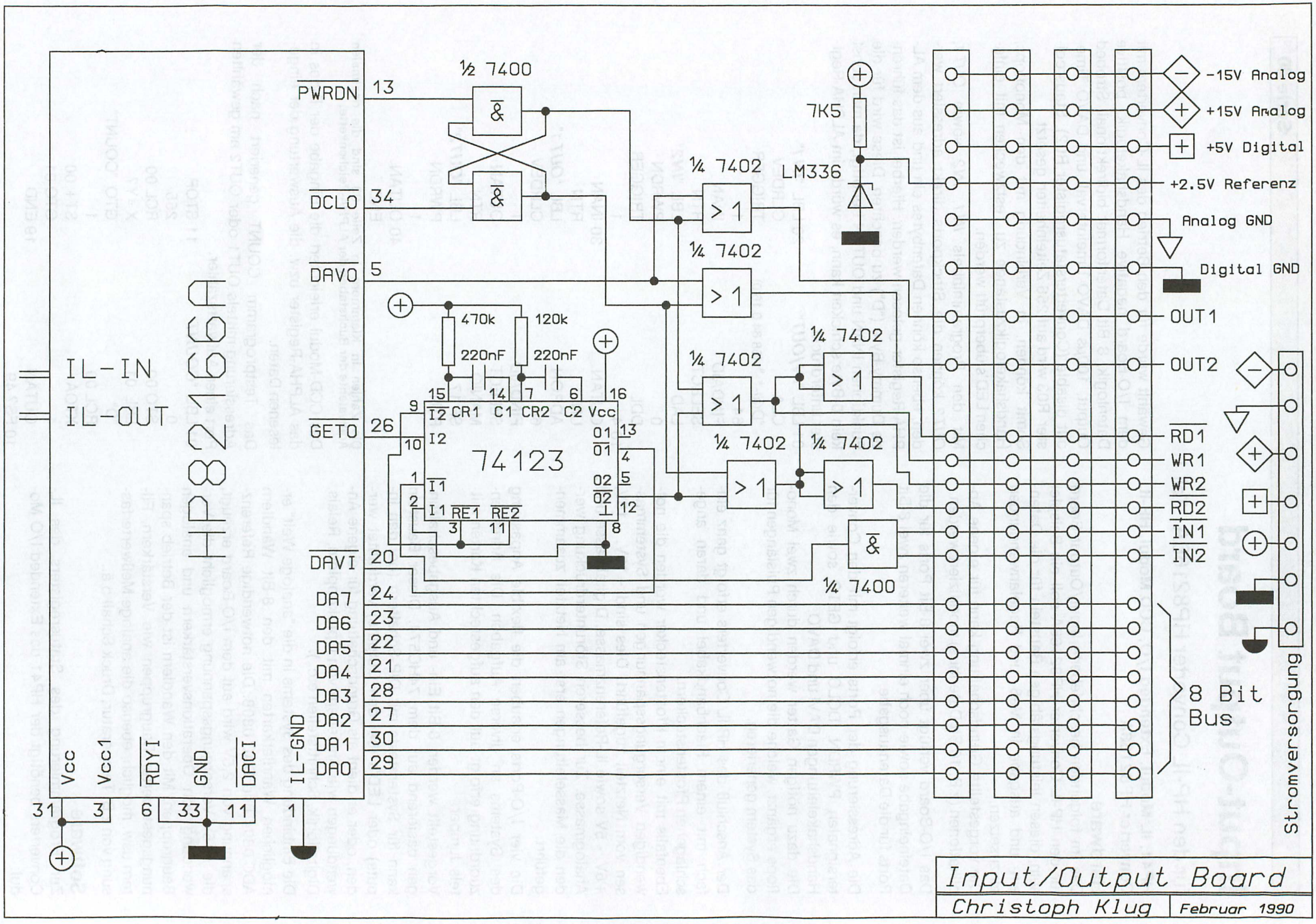
**Die Zahlen in Klammern in Zeile 03 sind die dezimalen Äquivalente der Buchstaben der ALPHA-Zeichenkette.**

Das CCD-Modul erleichtert die Eingabe der Bytes in das ALPHA-Register bzw. die Auswertung der eingelesenen Daten.

Das Testprogramm COUNT generiert nach der Adressierung mittels OUT1 oder OUT2 am gewählten Port einen Aufwärtszähler.

```
01 LBL "COUNT"
   0
   STO 00
   LBL 01
   "D"
   RCL 00
   XTOA
   1
   OUTAN
10 FS? 49
```

```
11 STOP
   255
   RCL 00
   X=Y?
   GTO "COUNT"
   1
   ST+ 00
   GTO 01
19 END
```


HP82166A
IL-IN
IL-OUT
PWRDN 13
DCLO 34
DAVO 5
GETO 26
DAVI 20
DA7 24
DA6 23
DA5 22
DA4 21
DA3 28
DA2 27
DA1 30
DA0 29
Vcc 31
Vcc1 3
RDYI 6
GND 33
DACI 11
IL-GND 1
½ 7400
¼ 7402
¼ 7400
7K5
LM336
470k
120k
220nF
220nF
74123
CR1 C1 CR2 C2 Vcc
RE1 RE2
-15V Analog
+15V Analog
+5V Digital
+2.5V Referenz
Analog GND
Digital GND
OUT1
OUT2
RD1
WR1
WR2
RD2
IN1
IN2
8 Bit Bus
Stromversorgung
Input/Output Board
Christoph Klug
Februar 1990

Mit dem Programm OUTV2 kann der D/A-Wandler auf dem Output-Port 2 angesteuert werden. Dazu wird **vor** der Programmausführung die gewünschte Spannung 0 . . 10V im X-Register abgelegt.

```
01 LB "OUTV2"        05 XTOA
   25,5                 GTO "OUT2"
   *                 07 END
   "D"
```

**Bei Werten > 10 Volt erfolgt die Meldung DATA ERROR.**

Nun noch drei Programme zum Testen des IL-Converters selbst:

**INST** schreibt die 19 Statusregisterbytes in das ALPHA-Register,

**OUTST** setzt die 19 Statusregisterbytes mit den im ALPHA-Register befindlichen Bytes,

**TRAN** schreibt den Inhalt des ALPHA-Registers in den Converter-Transferbuffer, löscht dann das ALPHA-Register und lädt den Inhalt des Converter-Transferbuffers in das ALPHA-Register zurück.

```
01 LBL "INST"
   64
   FINDAID
   ADROFF
   TAD
   0
   DDT
   19
   INAN
   UNT
11 END
```

```
01 LBL "OUTST"
   64
   FINDAID
   ADROFF
   LAD
   0
   DDL
   19
   OUTAN
   UNL
11 END
```

```
01 LBL "TRAN"
   "ALPHA DATA ?"
   AON
   PROMPT
   AOFF
   ALENG
   STO 01
   64
   FINDAID
10 STO 00
   ADROFF
   LAD
   1
   DDL
```

```
15 RCL 01
   OUTAN
   "  ALPHA EMPTY"
   AVIEW
   RCL 00
20 TAD
   RCL 01
   1
   -
   INAN
   PROMPT
   UNT
27 END
```

**Alle drei Routinen verwenden ein führendes Dummy-Byte.**

## Literaturangaben

Über weitere IL-Converter Anwendungen gibt das Buch von G. Friedmann "Control the world with HP-IL", Heldermann Verlag Berlin, mit vielen Tips und Anregungen Auskunft.

Lesenswert, aber doch harte Kost, sind die HP-Manuals und Supplements zum Converter selbst.

Notwendig sind die Datenblätter der verwendeten IC's, vor allem, wenn Spannungsbereiche der Wandler variiert werden.

Christoph Klug
Leibnitzstraße 19
3200 Hildesheim

# Programmierkurs HP-28 ?!

Nachdem ich wieder einmal die alten Prismazeitschriften durchgesehen habe, kam ich auf die Idee, ob der CCD wohl mal einen Kurs im Programmieren des HP 28 SD anbieten könnte. Es bestehen doch für Umsteiger, vom HP-41 zum HP-28, gewisse Schwierigkeiten (mir geht es wenigstens so). Mit dem beiliegenden Handbuch kann man seine Probleme leider nicht so ohne weiteres lösen.

Für den HP-41 gibt es einige praktische Beispiele, somit kann auch der etwas weniger Bewanderte manches nachvollziehen. Z.B.: Quadrat. Gleichung mit Abfrage und auch Anzeige durch den Prompt/View Befehl. (siehe Bedienungs- u. Programmierhandbuch zum HP-41 C Aug. 1979, Seite 179-181 usw.).

Wenn grundlegende Begriffe an praktisch nachvollziehbaren Beispielen begreiflich gemacht werden, kann sich jeder Interessierte seine Lösungen selbst erarbeiten. Das Problem liegt doch in der Umsetzung der vorhandenen 41er Programme in die neue Eingabetechnik. Dabei kommt es weniger auf elegante und bytesparende Lösungen an, sondern wie man eine ganze Reihe von bekannten Tastenbefehlen für den 28er aufbereitet.

Beispiel: STO 5 + RCL 6 - R/S Goto View Aview...usw. Es sind oft die einfachsten Dinge, die einen manchmal scheitern lassen. Zu den Listings wäre es zweckmäßig, die dabei verwendeten Formeln und auch ein Musterbeispiel mit der Lösung anzugeben (als Eingabekontrolle).

Sollte sich jemand um Hilfe für den 28 SD an HP wenden, so kann er sich die Zeit und die anfallenden Telefonkosten sparen. Es ist z.Zt. niemand dafür da. Nicht in Ratingen, nicht in Bad Homburg, nicht in München. Man wird an den Händler verwiesen, aber HP kann keine Hilfe geben.

Die Freude über den neuen Rechner hält dann auch nicht lange an. Die ganzen Programme vom 41er müssen erst überarbeitet und dann mühsam eingetippt werden. Sehr schade ist auch, daß man weder den Magnetkartenleser noch den optischen Lesestift benutzen kann.

Reiner Ziegler
Am Schenkenfeld 35
8707 Veitshöchheim

# ANUMDEL-Problem

Für Steuerungszwecke setze ich vor manche Textzeilen im Ext-Memory Kennzahlen, die bei der Textverarbeitung isoliert und entfernt werden müssen.

Hierzu dient die Funktion ANUMDEL im EXT I/O.

Folgt dieser Kennzahl jedoch ein E, so wird dieses vom Rechner als Zahlenbestandteil mit erfaßt und aus dem Text entfernt (siehe Handbuch S. 27/28).

Folgende Methode löst das Problem:

0-Bytes sind linksbündig nicht existent.

| | | |
|---|---|---|
| Alpha-Register | "10EMIL" | "10 EMIL" |
| | XEQ "ANUMDEL" | |
| Alpha-Register | "MIL" | "EMIL" |
| X-Register | 10 | 10 |

Dr. Martin Hochenegger
Heidelberger Landstraße 97
6100 Darmstadt 13

# Druck und Display-Utilities

## für den HP28

| | |
|---|---|
| **PRD** | **PRint Dsiplay** |
| **PID** | **Positioniere Im Display** |
| **FLG** | **Flags** |
| **PR1B** | **PRint 1: Big** |
| **PR2B** | **PRint 1: und 2: Big** |
| **IKEY** | **Tastaturabfrage-SUB** |
| **MERGE** | **Programme in andere einfügen** |
| **VERL** | **VERknüpfe Listen** |

Da ich erst seit einigen Wochen den Infrarotdrucker mein Eigen nennen kann war ich bei der Darstellung von Ergebnissen allein auf das Display des Rechners angewiesen. Um dort übersichtlich anzeigen zu können habe ich zwei Funktionen geschrieben, die mir dabei helfen konnten.

Die erste Funktion ***PRD*** (PRint Display) schiebt einen beliebigen String, in normal oder invers, von unten in die Anzeige; das Display "scrolled up".
Vor der Ausführung muß der String in Ebene 1: stehen.
Ist *Flag 30* gesetzt, so wird *invers* angezeigt, ansonsten normal.

Seit ich den Drucker habe, wurde die Funktion von mir noch erweitert: Ist der erste Character des Strings ein Escape (Zeichen $27_{10}$), so wird der gesamte Rest als Grafikstring interpretiert und angezeigt. Dadurch kann man die Funktion anstelle von PR1 benutzen und so Druckpapier sparen.

Für die meisten Anwendungen ist dies ausreichend, außer wenn man Strings betrachtet, in denen Grafik- und Normalzeichen gemischt vorkommen oder Zeichencodes von $250\text{-}255_{10}$ verwendet werden.

Die zweite Funktion ***PID*** (Positioniere Im Display) erlaubt die Positionierung eines beliebigen Strings, unterstrichen oder nicht, invers oder normal an beliebiger Stelle im Display. Der alte Inhalt an dieser Stelle wird überschrieben.
Vor der Ausführung müssen der String in Ebene 2: und die kombinierte Positions-/Darstellungsangabe in Ebene 1: stehen.

Bedeutung der Zahl:
(Displaypositionen = 0 bis 91)

| | | |
|---|---|---|
| Normal | nicht unterstrichen: | 1..91 |
| Normal | unterstrichen: | 100..191 |
| Invers | nicht unterstrichen: | -.01..-91 |
| Invers | unterstrichen: | -100..-191 |

Es ist auch möglich Zwischenpositionen zu erreichen, indem man eine Nachkommastelle dazunimmt. Die Verschiebungen in *Anzahl Pixel* sind unter dem Programm abgedruckt.
Um ab Position 0 "Invers nicht Unterstrichen" anzeigen zu können muß daher -0.01 verwendet werden.

Mit dieser Funktion ist es möglich eine "Bildschirmmaske" zu erstellen, in der ein Programm die Anzeigen, je nach Bedarf natürlich, neu überschreibt. Durch inverse Anzeigen oder unterstrichen können auch sehr gut Eingaben von Ergebnissen getrennt werden, was der Übersichtlichkeit zugute kommt. Außerdem kann man dadurch das Display "voller packen".

Das Programm ***FLG*** (Flags) zeigt alle 64 Flags im Display an und demonstriert auch die Verwendung von *PID*.
Bei mir kam dann sehr schnell der Wunsch auf, die optisch gut dargestellten Anzeigen auch etwas größer auf Papier zu bringen, da einige Brillenträger (das soll keine Diskriminierung sein) im Kollegenkreis Schwierigkeiten beim Lesen der Ausdrucke hatten.

Dafür schrieb ich ***PR1B*** (Print 1 Big) und ***PR2B*** (Print 2 Big).

Vor dem Aufruf des Programms müssen ein oder zwei mit LCD→ gewonnene Displayabzüge im Stack stehen.

Es mögen vielleicht einige fragen, warum *PR1B* und *PR2B* nicht in einem Programm realisiert wurden.
Dazu folgendes: Ich habe die Laufzeit von *PR1B* von anfangs 15min. bis auf ca. 4min. herunterdrücken können, indem ich die Drehung der Grafik mit Flagabfragen bewältigte. Da die inverse Schleife 4480 mal durchlaufen wird wäre ein zusätzlicher Test, ob "DUP+" oder "DUP+DUP+", eine große Verzögerung. Eine einfache Flagabfrage ist nämlich nicht realisierbar, da im Programm durch RLB und STOF alle Flags verändert werden.
42 SF in Zeile 014 stellt sicher, daß mindestens eine Binärwortlänge von 32 Bit vorhanden ist, damit die Additionen und RLB richtig funktionieren.
Wer dennoch nur ein Programm abtippen möchte, der sollte sich für *PR2B* entscheiden und ca. 6½ min. Laufzeit in Kauf zu nehmen, da für einen von beiden Displayabzügen auch ein Leerstring (" ") eingegeben werden kann.

***IKEY*** habe ich als extra Programm ausgelagert, weil ich diese Befehlsfolge in sehr vielen Programmen benötige.

Das Programm ***MERGE*** ist in der Lage ein Programm in ein anderes einzubinden. Dazu wird das erste mit RCL in den Stack geholt. Mit EDIT setzt man nun die Zeichenfolge "« »" an die Position, an der das zweite Programm dann stehen soll. Nun holt man das zweite Programm ebenfalls mit RCL in den Stack und führt *MERGE* aus.
Danach steht das zweite Programm, ohne Programmarken, an der gekennzeichneten Stelle, fertig zum Abspeichern mit STO.

Das Programm ***VERL*** (Verknüpfe Listen) ist in der Lage in zwei gleich langen und gleichartigen Listen die einzelnen Elemente jeweils mit dem gleichen Rechenzeichen zu verknüpfen.
Das Programm erleichtert mir die Arbeit beim Umschreiben von 41er Programmen. Programmschleifen, die dort Register bearbeiten, um z.B. Zahlen zu addieren, sind mit Listen und GETI und PUTI etwas umständlich. Also schreibe ich jetzt eine Schleife, in der die nötigen Summanden in eine Leerliste gebracht werden und verknüpfe alte und neue Liste mit "+".

*VERL* arbeitet theoretisch mit allen Funktionen, die zwei Objekte im Stack benötigen, aber nur eines zurückgeben. So z.B. "MAX" oder

"DUP SUB", um eine bestimmtes Zeichen eines Listenelements zu erhalten.

*Noch ein Tip, um Ordnung zu halten:*

Bei jedem Programmausdruck lasse ich erst eine Datum/Uhrzeit-Zeile drucken. Damit ist dann immer ersichtlich, welches die neueste Version ist.

Aber wohin mit den Ausdrucken ? Mit dem Aufkleben der Ausdrucke auf DIN A4 Papier hatte ich viel Pech, da der Kleber (Pritt Stift) fast immer das Papier durchdrang und die geschwärzten Stellen verblassen ließ. Deshalb bin ich dazu übergegangen sie in Negativ-Ablageblätter zu schieben. Von der Firma "hama-photo-video" gibt es diese Blätter aus Pergamin (Nr.9122) mit einer Höhe von 60mm. Auf einer Seite sind 4 Reihen, in die jeweils 85 Druckzeilen hineinpassen. Der Satz mit 25 Blättern kostete mich 6,40.- DM; dies kann man also aushalten.

```
IKEY
«
  DO KEY
  UNTIL 1400 .1 BEEP
.5 WAIT
  END
»
```

Anzeige von FLG

```
FLG
« STD 0 3
  FOR X "" 1 16
    FOR Y X 16 * Y +
FS? →STR Y 4 MOD 0
== "  " IFT +
    NEXT + + + + 1
22 SUB X 1 + DISP
  NEXT " 4" -4 PID
" 8" -10 PID "12"
-16 PID "20" -27 PID
"24" -33 PID "28"
-39 PID "36" -50 PID
"40" -56 PID "44"
-62 PID "52" -73 PID
"56" -79 PID "60"
-85 PID
»
```

```
PID
« LCD→ 3 PICK 4 DISP
LCD→ 412 DUP 6 ROLL
SIZE 6 * + 2 - SUB 3
PICK 0
  IF <
  THEN NOT
  END 3 PICK ABS 99
  IF >
  THEN "" OVER SIZE
1 SWAP
    FOR A OVER A A
SUB NUM 128 + CHR +
    NEXT SWAP DROP
ROT ABS 100 - ROT
ROT
  END OVER 1 5 ROLL
6 * ABS DUP 137 /
FLOOR - SUB 2 ROLL +
DUP SIZE 1 + 548 4
ROLL 3 ROLLD SUB +
→LCD
»
```

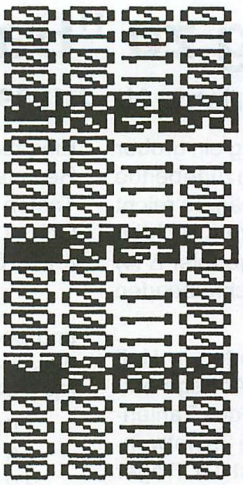
```
PRD
« DUP LCD→ 138 548
SUB OVER 1 1 SUB NUM
27
  IF ==
  THEN SWAP 3 139
SUB
  ELSE SWAP 4 DISP
LCD→ 412 548 SUB
  END 30
  IF FS?
  THEN NOT
  END + "" 4 DISP
→LCD
»
```

So 29.04.1990 ※ 16:01:51

```
   MERGE
001 « →STR DUP SIZE 1 -
    2 SWAP SUB SWAP →STR
003 DUP DUP "«»" POS 2 -
    1 SWAP SUB ROT +
005 SWAP DUP "«»" POS 3
    + OVER SIZE SUB +
007 STR→
    » 106.5 Bytes
```

So 29.04.1990 ※ 16:05:34

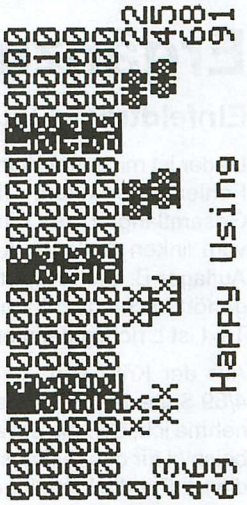
```
   VERL
001 « → o
      « LIST→ 1 + ROLL
003 LIST→ → z
      « z 1
        FOR x x ROLL x
005       2 * ROLL o STR→ x 2
007 * 1 - ROLLD -1
        STEP z →LIST
009     »
    »
011 » 123 Bytes
```



Ausdruck von PR1B

```
DEMO
« CLLCD "XX" 3 PID
"XX" 108 PID "XX"
-13 PID "XX" -118
PID "XX" 26.5 PID
"XX" 131.5 PID "XX"
-36.5 PID "XX"
-141.5 PID "0" 0 PID
"22" 21 PID "23" 23
PID "45" 44 PID "46"
46 PID "68" 67 PID
"69" 69 PID "91" 90
PID "Happy Using" 74
PID PRLCD
»
```

```
PR2B
« SWAP RCLF 3 ROLLD
1 2
  START DUP 1 137
SUB 0 CHR DUP DUP +
+ DUP 4 ROLLD + OVER
138 274 SUB + 3 PICK
+ OVER 275 411 SUB +
3 PICK + SWAP 412
548 SUB + SWAP +
SWAP
  NEXT CLLCD
"Printer ready ? "
-49 PID IKEY DROP 4
CHR PR1 DROP 27 CHR
128 CHR + 140 4
  FOR A DUP 4 5
    FOR E 0 420
      FOR B 42 SF
# 0d 0 3
        FOR D E PICK
B A + D - DUP SUB
NUM + RLB
      NEXT RRB
STOF 1 8
      FOR C 0 C
        IF FS?
        THEN 192 +
        END C 8 +
        IF FS?
        THEN 48 +
        END C 16 +
        IF FS?
        THEN 12 +
        END C 24 +
        IF FS?
        THEN 3 +
        END CHR
DUP + +
    NEXT 140
  STEP
NEXT PR1 DROP -4
STEP DROP2 DROP
STOF
»
```



Ausdruck von PR2B

```
2: « DUP + "«»" PR1
»
1: « 2 FIX →STR
»
MERGE  ( bei "«»" )
1: « DUP + 2 FIX →STR
PR1
»

3: { 1 2 3 4 5 6 }
2: { 3 2 1 3 2 1 }
1: "A"
VERL (verknüpfe Liste)
1: { 1 A 3 64 25 6 }
```

```
PR1B
« RCLF SWAP 4 CHR
PR1 DROP DUP 1 137
SUB 0 CHR DUP DUP +
+ DUP 4 ROLLD + OVER
138 274 SUB + 3 PICK
+ OVER 275 411 SUB +
3 PICK + SWAP 412
548 SUB + SWAP +
CLLCD
"Printer ready ? "
-49 PID IKEY DROP 27
CHR 128 CHR + 140 4
  FOR A DUP 0 420
    FOR B 42 SF # 0h
0 3
      FOR D 4 PICK B
A + D - DUP SUB NUM
+ RLB
      NEXT RRB STOF
1 8
      FOR C 0 C
        IF FS?
        THEN 192 +
        END C 8 +
        IF FS?
        THEN 48 +
        END C 16 +
        IF FS?
        THEN 12 +
        END C 24 +
        IF FS?
        THEN 3 +
        END CHR DUP
+ DUP + +
    NEXT 140
  STEP PR1 DROP -4
STEP DROP2 STOF
»
```

Wolfgang Führer
Sachsenstraße 166
4350 Recklinghausen

# Ergänzung zu "Baustatik mit HP 28S"

## Einfeldträger (aus Prisma Nr. 2/1989, Seite 39)

Leider ist mir in der Zeichnung ein grober Fehler unterlaufen. Die einzugebende Gesamtlänge L des Einfeldträgers reicht vom linken Auflager A bis zum rechten Auflager B. Die Kragarmlänge (hier Δ l4) gehört nicht dazu! Im beschreibenden Text ist L richtig definiert.

Aus der Kritik von Josef Herten in Heft 4/89 Seite 18 und aus einer Anfrage entnehme ich, daß ein ergänzendes Zahlenbeispiel für das Bild aus Seite 39 notwendig ist. Für die Lasteingabe ist der Träger in 4 Falkfelder mit den Längen Dl1 bis Dl4 eingeteilt, die an jedem Belastungssprung bzw. bei den Einzellasten anfangen und enden. Das Programm arbeitet nur mit Falkfeldern. Für jedes Falkfeld müssen die 4 Werte Dl, ql, qR, P eingegeben werden.

Das Beispiel auf Heft 4/89 besitzt eine andere Lasteingabe getrennt nach Lastarten. Wegen seiner 6 Einzellasten wäre für mein Programm hierbei eine Aufteilung in 6 Falkfelder und die Eingabe von 6 x 4 Werten für die Lasten erforderlich. Folgende Werte seien für das Beispiel in Heft 2, Seite 39 gegeben:

F = 4 (Zahl der Falkfelder)

L = 4,8 m (Länge des Trägers zwischen den Auflagern A und B; Zeichnungsangabe L ist hier falsch, siehe obige Berichtigung)

MA = 2,2 KNm (Randmoment am linken Auflager)

MB = 0 (Randmoment am rechten Auflager)

1. Falkfeld: DL1 = 1,5 m
qL = 1,2 KN/m
qR = 1,5 KN/m
P1 = 3,0 KN

2. Falkfeld: DL2 = 1,9 m
qL = 0 KN/m
qR = 0,5 KN/m
P2 = 2,0 KN

3. Falkfeld: DL3 = 1,4 m
qL = 0,5 KN/m
qR = 0,5 KN/m
P3 = 5,0 KN

4. Falkfeld: DL4 = 1,3 m
qL = 0,5 KN/m
qR = 0,5 KN/m
P4 = 0

(Durch den gewählten Programmablauf kann ohne weiteres ein rechter Kragarm berücksichtigt werden; ein linker Kragarm müsste über die Momenteingabe MA und die Addition von QL zum Auflagerdruck A berücksichtigt werden).

Eingabe in den HP 28 und Ausrechnung:
Rechner einschalten und dann folgende Tasten drücken:
ON mit ▲ gleichzeitig (Systemstop) α EIN α ENTER (führt zur Anzeige in Zeile 1

"F L MA MB"

und bedeutet eine Eingabeaufforderung dieser 4 Werte in den Stack mit jeweils ENTER-Taste)

4 ENTER
4.8 ENTER
2.2 ENTER
0 ENTER
Rote Taste CONT

(führt zur Anzeige in Zeile 1

"| [ DL QL QR PL"

und bedeutet Eingabeaufforderung für die je 4 Werte der einzelnen Falkfelder in eine Matrix, bei diesem Beispiel 4 x 4 Matrix)

[ [ 1.5 Space
1.2 Space
1.5 Space
3.0 [ 1.9 Space
0 Space
0.5 Space
2.0 [ 1.4 Space
0.5 Space
0.5 Space
5.0 [ 1.3 Space
0.5 Space
0.5 Space
0 rote Taste CONT.

Damit ist die Eingabe beendet und das Programm läuft.
Nach 7 Sekunden erscheinen die Auflagerdrücke A = 4.0996 in Zeile 2 und B = 9.7503 in Zeile 1.
Weiter mit roter Taste und CONT.
Nach 2 Sekunden erscheint in Zeile 2 das maximale Feldmoment Mx = 6.8869 und in Zeile 1 sein Ort x = 1.5.
Eingabe eines neuen Trägers:
ON mit ▲ gleichzeitig α EIN α ENTER und neue Eingaben (siehe oben).

Zum Verständnis des Programms noch folgende Erklärungen. Zuerst wird mit Momentbildung der Lasten um Auflager A und Teilen durch L der Auflagerdruck B bestimmt. Aus der Summe der Lasten weniger B ergibt sich Auflagerdruck A. Mit A und den Lasten wird von links nach rechts der Querkraftnullpunkt gesucht und dann, falls vorhanden, das maximale Feldmoment und sein Ort bestimmt.

Variablen sind: **F** Zahl der Felder, **L** Länge zwischen Auflagern, **A** angreifendes Moment an Auflager A, **B** angreifendes Moment an Auflage B, **C** Matrix der Eingabewerte Dl, ql, qR, P (Einzelne Werte der Matrix werden mit 'C' {x i} GET in den Stack geholt), **SM** Summe der Lastenmomente um Auflager A, **S** Summe der DL, **Q** Summe der Querkräfte von Auflager A aus, **ML** Moment an der linken Seite des gerade untersuchten Falkfeldes, **MR** Moment an dessen rechter Seite, **QL** Querkraft an der linken Seite des gerade untersuchten Falkfeldes, **QR** Querkraft an dessen rechter Seite, wenn QL x QR < 0, ist in diesem Feld der Querkraftnullpunkt und das maximale Feldmoment, **QB** Querkraft links von Auflager B, **K** = (rechte Lastordinate - linke Lastordinate) / Falkfeldlänge. SQ bedeutet quadrieren und keine Variable!

R. Beaucamp (1071)
Brockhoffstr. 4
4400 Münster

# Anmerkungen zu ROOT und Vorgabe bei INPUT

## HP48SX

Es ist mir gelungen, die SOLVE-Applikation in Programmen anzuwenden. Die Anleitung ist hier vage und ausführliches Probieren empfiehlt sich sehr. Jedoch sollte stets bedacht werden, daß eine der vorliegenden Aufgabe angepasste eigene Routine viel schneller sein kann.
Dies gilt nicht nur für den 48'er, sondern auch für SOLVE im 41'er Advantage-Modul und FNROOT im 71'er Mathe-Modul.

Der 48'er läßt Vorgabewerte bei INPUT in sehr ähnlicher Weise zu wie auf dem 71'er, wie ich durch Probieren herausfand. (Siehe dazu das Unterprogramm SZ im schon vorgelegten Programm EPH) Allerdings gibt das Handbuch hierfür keine Hinweise.

Beim Probieren mit ROOT fiel mir auf, daß im von mir vorgelegten 48'er Programm EPH im Unterprogramm KP von mir vergessen worden sein kann, die erste Zeile mit π abzuschließen.
Die Zeile muß heißen:

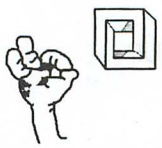
< -3 SF RAD D→R π.

Ich bitte darum, dies zu korrigieren.

Dr. Gerhard Heilmann
Obernhofer Straße 15
5408 Seelbach

# Objekt Handling

## Utilities für den HP28S



**LOAD** kopiert Objekte aus der RAM-Disk (= Directory {ZZZ} in {HOME})
**SAVE** kopiert Objekte in die RAM-Disk
**COPY** kopiert Objekte von einem Directory in eine anderes (das ausgewählte alte Objekt bleibt erhalten)
**MOVE** verschiebt Objekte von einem Directory in eine anderes (das ausgewählte alte Objekt wird gelöscht)
**OBSI** Zeigt die Größe eines beliebigen Objektes in *BYTES* an (geht nicht für Directorys)
**REPL** ersetzt eine Substring durch einen anderen Substring in einem beliebigen Objekt
**INS** fügt ein Programm in ein anderes ein
**Z** geht ein Directory zurück (= CD.. bei DOS/TOS-Rechnern mit Kommandointerpreter)

**Vor der Verwendung der Programme LOAD, SAVE, COPY und MOVE muß eine "RAM-Disk" ZZZ im Verzeichnis { HOME } mit HOME 'ZZZ' CRDIR angelegt werden. Ebenso nicht die für die Ausführung nötigen Unterprogramme vergessen!**

### LOAD

```
« 8 CF          Zeigt an, daß Objekte kopiert werden sollen
 PATH→p         rettet aktuellen Verzeichnispfad
 « ZZZ          springt in die RAM-Disk
  →CML          Unterprogr.: "Objekte auf den Stack laden"
  p →DIR        altes Verzeichnis wiederherstellen
  →CMS »        Unterprogr.: "Objekte vom Stack abspeichern"
»
```

### SAVE

```
« 8 CF          Zeigt an, daß Objekte kopiert werden sollen
 PATH→p         rettet aktuellen Verzeichnispfad
 « →CML         Unterprogr.: "Objekte auf den Stack laden"
  ZZZ           springt in die RAM-Disk
  →CMS          Unterprogr.: "Objekte vom Stack abspeichern"
  p →DIR »      altes Verzeichnis wiederherstellen
»
```

**Anwendung** von **LOAD** und **SAVE**

*Syntax*

<globaler Name> LOAD
<Liste mit globalen Namen> LOAD
<globaler Name> SAVE
<Liste mit globalen Namen> SAVE

Bsp.: { 'TEST1' } SAVE oder
{ 'TEST1' 'NAECHST' 'UEBERN' } SAVE
speichern die angegebenen Objekte in der RAM-Disk ZZZ ab, sie müssen natürlich in meinem aktuellen Verzeichnis zu finden sein.
Um die "zwischengespeicherten" Objekte in ein anderes Verzeichnis zu bekommen muß ich nur in dieses wechseln und
{ 'TEST1' } LOAD oder
{ 'TEST1' 'NAECHST' 'UEBERN' } LOAD
ausführen.
*Als Objekte sind auch ganze Subdirectorytrees, oder zu deutsch Verzeichnisbäume, möglich.*

### COPY

```
« 8 CF          Zeigt an, daß Objekte kopiert werden sollen
 →CML           Unterprogr.: "Objekte auf den Stack laden"
 HALT           Anhalten für Verzeichniswechsel
 →CMS           Unterprogr.: "Objekte vom Stack abspeichern"
»
```

### MOVE

```
« 8 SF          Zeigt verschieben von Objekten an
 →CML           Unterprogr.: "Objekte auf den Stack laden"
 HALT           Anhalten für Verzeichniswechsel
 →CMS           Unterprogr.: "Objekte vom Stack abspeichern"
»
```

**Anwendung** von **COPY** und **MOVE**

*Syntax*

<globaler Name> COPY
<Liste mit globalen Namen> COPY
<globaler Name> MOVE
<Liste mit globalen Namen> MOVE

Bsp.: { 'TEST1' } COPY oder
{ 'TEST1' 'NAECHST' 'UEBERN' } COPY
wechseln in das neue Verzeichnis, z.B. in das HOME-Verzeichnis mit HOME ENTER,
jetzt CONT (SHIFT 1) drücken. Die Objekte befinden sich jetzt in diesem anderen Verzeichnis, in unserem Fall das HOME-Verzeichnis.
MOVE funktioniert genauso, nur daß es die Objekte im Quellverzeichnis nach dem Kopiervorgang löscht.

### Unterprogramme zu LOAD, SAVE, COPY, MOVE

**Diese müssen auch in der RAM-Disk ZZZ vorhanden sein !!**

```
→CML                    Unterprogr.: "Objekte auf den Stack laden"
« →O→L                  Unterprogr.: "eine Liste erzeugen lassen"
 1 ol SIZE FOR i        Schleife von 1 bis Anzahl der zu ladenden Objekte
 ol i GET DUP           hole Objekt-Namen
 IFERR RCL              und Inhalt
  IF 8 FS?              Verschieben o. Kopieren
  THEN OVER PURGE END   Wenn ja, dann das Objekt im Quellverzeichnis löschen
 SWAP
 THEN
  IF ERRN # 12Ah = = THEN  Wenn das Objekt ein Verzeichnis war, dann
   EVAL→dir              springe in dieses Unterverzeichnis uns speichere den
```

```
                          Namen des Unterverzeich-
                          nisses
  « VARS DUP              alle Objekte in diesem Ver-
                          zeichnis herausfinden:
   IF { } ≠               Wenn das Verzeichnis nicht
                          leer ist,
   THEN →CML              dann rufe noch einmal das
                          Unterprogramm selbst auf.
                          Dies nennt man Rekursion !
   ELSE DROP END          Wenn es leer ist, dann
                          nichts tun...
   dir DUP                Namen des Unterverzeich-
                          nisses auf den Stack zu-
                          rückholen
   « CRDIR »              Damit ein neues Unterver-
                          zeichnis erzeugen
   Z                      Unterpr.: "Ein Verzeichnis
                          zurück"
   IF 8 FS? THEN          Verschieben o. Kopieren
   OVER PURGE END         Wenn ja, dann lösche Ob-
  »                       jekt im Quellverzeichnis
  ELSE KILL END           Unterprogramm abbrechen
 END
NEXT
1 ol SIZE                 Schleife von 1 bis Anzahl
                          Objekte; Schleifenzähler-
                          Anzahl für das Unterpro-
                          gramm →CMS im Stack las-
                          sen
'ol' PURGE                nicht mehr benötigte Ob-
                          jektliste löschen
»
```

**→C M S** Unterpr.: "Objekte vom Stack abspeichern"

```
« IFERR                   Fehlerbehandlung, wenn
                          ein Verzeichnis auftaucht
  START                   Schleife von 1 bis Anzahl
                          der Objekte im Unterver-
                          zeichnis
   IFERR                  Fehlerbehandlung, wenn
                          ein Verzeichnis auftaucht
    STO                   Objekt abspeichern
   THEN EVAL EVAL         Wenn es ein Verzeichnis-
                          Objekt war, dann ein Ver-
                          zeichnis erzeugen
    →CMS                  Rufe das Unterprogramm
                          noch einmal selbst auf, das
                          nennt man Rekursion !
   Z                      Unterprogramm: "Ein Ver-
                          zeichnis zurück"
  END
 NEXT
THEN END
»
```

**→O→L** Unterprogr.: "eine Liste erzeugen lassen"

```
« IFERR LIST→ THEN 1 END  Aus einer Liste
 →LIST 'ol' STO           in eine Liste und in "ol"
                          speichern
»
```

**→D I R** Wechselt in einen anderen Verzeichnispfad

```
« DUP SIZE 1 1 ROT START GETI EVAL NEXT DROP2 »
```

Benutzung: In Stackebene 1: muß der komplette Pfad als Verzeichnis stehen, d.h. wenn man sich im Verzeichnis { HOME PRISMA } befindet und { ZZZ } ENTER ->DIR per Softkey aufruft, so landet man in { HOME ZZZ }

**Z** Springt ein Verzeichnis zurück (= CD.. o.ä.)

```
« PATH DUP SIZE 1 - 1 MAX GET EVAL »
```

Benutzung: Einfach das Programm aufrufen, schon ist man eine Verzeichnisebene weiter oben Richtung HOME

**O B S I** Ermittelt Objektgröße in Bytes

```
« →O→L                    Unterprogr.: "eine Liste erzeugen
                          lassen"
 1 DROP                   setze den LAST-Puffer auf einen
                          konstanten Wert
 MEM 'm0' STO             retten des freien Speichers
 ol LIST→                 Liste auf den Stack legen
 1 SWAP                   Schleife von 1 bis zur Anzahl der
 FOR i                    in der Liste befindlichen Objekte
  IFERR                   Fehlerbehandlung für den Fall,
                          daß das Objekt ein Verzeichnis
                          ist
   RCL                    In der Objektliste wurde ein Ver-
                          zeichnis mit angegeben
   " 'm" i →STR + STR→    daraus ein "Dummy-Objekt" ma-
                          chen
   STO                    und abspeichern
  THEN DROP END           das Objekt war ein Verzeichnis,
 NEXT                     also nächstes Objekt
 1 DROP                   setze den LAST-Puffer auf einen
                          konstanten Wert
 MEM m0 - NEG             Differenz zwischen freiem Spei-
                          cher vor und nach dem Verdop-
                          peln des Objekts
 0 ol SIZE FOR i          Schleife von 0 bis Anzahl der Ob-
                          jekte
  " 'm" i →STR + STR→     Namen für oben angelegtes
                          "Dummy-Objekt" erzeugen
  PURGE                   und löschen
 NEXT
 ol →STR SIZE +           Summe des Speicherbedarfs der
                          angegebenen Objekte bilden
 ol SIZE 3 * - 20 -       und den Speicherbedarf der
                          "Dummy-Objekte" abziehen
 'ol' PURGE               nicht mehr benötigte Liste lö-
                          schen
»
```

**→O→L** Unterprogr.: "eine Liste erzeugen lassen"

```
« IFERR LIST→ THEN 1 END  Aus einer Liste
 →LIST 'ol' STO           in eine Liste und in "ol"
                          speichern
```

**Anwendung** von **OBSI**

*Syntax*

<globaler Name> OBSI
<Liste mit globalen Namen> OBSI

Bsp.: 'OBSI' OBSI
Ergebnis: 222

**R E P L** ersetzt einen String durch einen anderen

```
« IF DUP TYPE 6 ≠ THEN    ist es kein (globaler) Name ?
 'rep' SWAP OVER STO      Dann abspeichern
 DUP 4 ROLLD →REPL        Unterprogramm aufrufen
 RCL LAST PURGE           löschen "Dummy-Objekt" 'repl'.
 ELSE →REPL END           Ansonsten nur das Unterpro-
                          gramm aufrufen
»
```

**→R E P L** Unterprogramm für REPL

```
« 0 → a b p z             lokale Variablen erzeu-
                          gen
 « p RCL →STR             Objekt wiederholen und
                          in einen String wandeln
  WHILE                   Schleife
   DUP DUP a POS DUP 'z' STO  zur Änderung des
                          Strings und seiner Grö-
                          ße
  REPEAT                  Wiederhole
```

```
  1 z 1 - SUB                          von Anfang bis zum Auftauchen des Strings
  b +                                  und füge ihn dazu.
  SWAP z a SIZE +                      Neue Größe ermitteln,
  OVER SIZE SUB +                      ergänzen des Restes.
 END
 p RCL                                 Holen des Objekts
 IF TYPE 2 ≠ THEN STR→ END             aus dem String (falls es einer gewesen war)
  p STO DROP                           und abspeichern desselben
» »
```

**Anwendung** von **REPL**

*Syntax*

3: <string1>
2: <string2>
1: <beliebiges Objekt>
REPL

Bsp.: "^"
"!" (wird mit 33 CHR erzeugt)
"Beispiel ^"
REPL
ergiebt "Beispiel !" .
Die Ersetzung erfolgt allerdings nur einmal, d.h. wenn ich in einen Objekt mehrfach ersetzen will, dann muß ich den Vorgang mit demselben Objekt sooft wiederholen, bis alle alten Strings durch neue ersetzt worden sind.

## I N S

**fügt ein Programm in ein anderes ein**

```
« RCL →STR                  Hole 1.Programm und wandle es in einen String um
 "μ" SWAP ROT               Vorbereitung für REPL
 RCL REPL                   Hole 2.Programm und ersetzte "μ" durch das 2.Programm
»
```

**Anwendung** von **INS**

*Syntax*

2: <Programm1>
1: <Programm2>
INS

Bsp.: « "Test" μ HOME » 'PR1' STO
« 1 DISP » 'PR2' STO
Beide Programmnamen in den Stack zurückholen:
2: 'PR1'
1: 'PR2'
INS ausführen, danach steht
1: « "Test" « 1 DISP » HOME »
im Stack.
Dies kann man nun nach Belieben abspeichern. Das Programm INS macht nichts anderes, als ein in das Programm 1 eingefügtes μ durch ein 2. Programm zu ersetzen. Dies kann sehr praktisch sein, wenn man erst in einem Programm einen Unterprogrammnamen durch μ ersetzt und dann dasselbe durch das Unterprogramm selbst ersetzen läßt.
So erspart man sich das erneute Eintippen.

HP28 Usergroup ULM
Überarbeitung: MM

# Speed

Ändern der Taktfrequenz des HP28S

Diese Programm ändert die Verarbeitungsgeschwindigkeit des HP28S, indem das Speed-Nybble manipuliert wird.
Das Programm SPEED verlangt eine ganze Zahl zwischen 0 und 15, wobei 7 die Standardeinstellung ist. 15 ist damit doppelt so schnell, 0 etwa ein Drittel so schnell wie im Normalbetrieb.

Probleme/Bugs:
Die Taste/Funktion ON/ATTN und manchmal auch die Garbage Collection Routine setzen den Rechner in die Standardgeschwindigkeit zurück.
Da der Rechner höher getaktet wird, verbraucht er natürlich auch etwas mehr Strom, was sich anscheinend nur minimal auf die Lebensdauer der Batterien auswirkt.

**S P E E D**

```
« 'SPEED' → s n                 s ist die Geschwindigkeit, n der Name der Variablen
  « PATH RCWS                   Rettet den Pfadnamen und die Wortlänge auf den Stack
    64 STWS                     Setzt Wortlänge 64 Bit
    s R→B # Fh AND              holt das unterste Nybble aus der lokalen Variable s
    # 1000000000000h *         schiebt es ganz nach links
    # C600302331DFFBCh          und setzt es ein.
    OR { # E60D51FFF00F1h       Dieses Programm muß das erste
    # C808461241131h } +        im Root-Directory sein.
    HOME n RCL                  Holt das Originalprogramm auf den Stack
    SWAP n PURGE n STO          speichert Maschinencode als 1.Variable
    # DFFB7h SYSEVAL            und führt diesen aus.
    n STO STWS CD               Restauriert alten Zustand
  »
»
```

Speichert man das Programm unter dem Namen 'SPEED' im ROOT-Directory ab, so erfolgt der Aufruf durch 15 ENTER und Drücken des Softkeys SPEED, dafür muß natürlich der USER-Mode eingeschaltet sein, sonst fehlen die Softkeys.
Im Stack bleibt { HOME } stehen, falls man sich im Root-Directory befand. Das eventuell vorhandene Programm CD wechselt dann in dieses Verzeichnis, aus dem man das Programm eventuell gestartet hat, ansonsten bleibt es bei der Aufforderung, da das Programm SPEED immer im HOME-Verzeichnis beendet wird.

Die Geschwindigkeitssteigerung macht sich vor allem bei längeren Berechnungen bemerkbar.

Bob Peraino
George Mason Universität
Fairfax, Vancouver 22030
Überarbeitung: MM

# Molmassen beliebiger Verbindungen

Das Programm MOLM für den HP-71 berechnet die Molmassen beliebiger Verbindungen. Dazu werden die LEX-Files ATOMICWT und STRUCLEX benötigt. Nach Start des Programms gibt man die Summenformel einfach so ein, wie sie auf dem Papier steht, also für $KMnO_4$ z.B. KMnO4 oder für $CH_3COOH$ CH3COOH. Das Programm erkennt automatisch, um welche Atome es sich handelt und wieviele vorhanden sind.

```
MOLM    BASIC    609 Bytes

 10 DESTROY M.M$,E$ @ DIM M$(10),E$(10) @ A=1 @ I=1 @ DELAY 0,0
 20 INPUT "-> ";V$ @ DISP "working..."
 30 WHILE I<=LEN(V$)
 40 IF NUM(V$[I,I])>64 THEN
 50 E$(A)=V$[I,I]
 60 IF NUM(V$[I+1,I+1])>96 THEN E$(A)=E$(A)&V$[I+1,I+1]
 70 END IF
 80 I=I+LEN(E$(A))
 90 IF NUM(V$[I,I])<64 THEN
100 M$(A)=V$[I,I]
110 IF NUM(V$[I+1,I+1])<64 THEN M$(A)=M$(A)&V$[I+1,I+1]
120 END IF
130 I=I+LEN(M$(A)) @ A=A+1
140 END WHILE
150 V$="M("
160 FOR I=1 TO A-1
170 IF M$(I)="" THEN M$(I)="1"
180 M=M+VAL(M$(I))*ATOMICWT(E$(I))
185 IF ATOMICWT(E$(I))=0 THEN DISP E$(I);" nicht bekannt !" @ WAIT 1
                                                    @ GOTO 270
190 V$=V$&E$(I)
200 IF M$(I)#"1" THEN
210 IF VAL(M$(I))<10 THEN
220 V$=V$&CHR$(139+VAL(M$(I)))
230 ELSE
240 V$=V$&CHR$(139+VAL(M$(I)[1,1]))&CHR$(139+VAL(M$(I)[2,2]))
250 END IF
260 END IF
270 NEXT I
280 V$=V$&")=" @ DELAY 8 @ DISP V$&STR$(M)&" u" @ A$=KEYWAIT$
                                                    @ GOTO 10
```

```
atomicwt  L ID#5C 1116.000 Bytes

     0123 4567 89AB CDEF ck

000: 1445 F4D4 9434 7545 D2
001: 802E 0044 1282 1198 CD
002: 8980 0C5E 1E10 0000 D5
003: F710 0000 0000 0000 A7
004: 0F10 00FF 1445 F4D4 54
005: 9434 7545 E11F F411 7E
006: 1361 BBB8 F214 43F0 DE
007: 2020 2020 2020 202A 10
008: F58F 83DB 0312 0962 9F
009: 2131 4096 2C18 D029 01
00A: E0AF 414B 8108 1017 24
00B: 16D0 0AF4 15B3 1738 2A
00C: FCBF 3023 79A7 3414 D5
00D: 2000 0000 0000 7227 87
00E: 4142 0000 0000 8687 3E
00F: 01C4 1410 0000 0045 7B
010: 1896 2D41 4200 0000 A2
011: 0000 0342 2514 1000 4A
012: 0000 0084 9933 5141 63
013: 0000 0000 6129 4745 64
014: 1420 0000 0000 0001 9F
015: 2551 4200 0000 0566 BF
016: 9691 0224 1000 0000 F7
017: 0001 8011 4242 0000 37
018: 0000 0337 3154 2400 BD
019: 0000 0008 1210 9942 37
01A: 4200 0000 0408 9802 15
01B: B424 2000 0000 0000 BA
01C: 7422 5241 0000 0000 E6
01D: 0409 9702 3410 0000 61
01E: 0000 1102 1143 4100 40
01F: 0000 0000 8004 4434 D1
020: 2000 0000 0014 2115 44
021: 4342 0000 0000 0210 C9
022: 4164 3420 0000 0000 DD
023: 0015 2C43 4100 0000 75
024: 0003 5453 D434 2000 3F
025: 0000 0000 742F 4341 AF
026: 0000 0000 2339 8525 59
027: 3410 0000 0000 6991 F1
028: 5353 4200 0000 0450 49
029: 9231 5534 1000 0000 0F
02A: 0064 5369 5442 0000 D8
02B: 0000 0052 6125 5420 DC
02C: 0000 0000 6276 1355 31
02D: 4200 0000 0000 0452 2B
02E: 5554 2000 0000 0069 A2
02F: 1510 2641 0000 0030 0D
030: 4899 8154 6410 0000 AD
031: 0000 7485 5D46 4200 D4
032: 0000 0000 0752 2564 0F
033: 2000 0000 0000 3221 F3
034: 4741 0000 0000 0027 30
035: 9644 7420 0000 0000 FA
036: 5275 1547 4100 0000 57
037: 0000 9527 0284 0000 AD
038: 0000 0097 0015 4840 1E
039: 0000 0000 0620 0464 A4
03A: 8420 0000 0000 9487 2B
03B: 1748 4200 0000 0009 6B
03C: 5002 F484 2000 0000 7D
03D: 4039 4610 2942 0000 A0
03E: 0005 4096 21E4 9420 15
```

```
03F: 0000 0000 2841 1259 1E
040: 4200 0000 0002 2291 6D
041: 02B4 1000 0000 0389 01
042: 0932 5B41 0000 0000 4C
043: 0008 3814 C420 0000 EF
044: 0055 0983 194C 4000 D0
045: 0000 0000 1496 25C4 19
046: 2000 0000 0000 0625 44
047: 5C42 0000 0000 0794 E8
048: 7144 D420 0000 0000 2F
049: 0085 274D 4100 0000 CA
04A: 0005 0342 E4D4 1000 D0
04B: 0000 0083 945F 4D41 20
04C: 0000 0000 0049 5902 F7
04D: E410 0000 0007 6004 82
04E: 114E 4100 0000 0779 58
04F: 8922 24E4 1000 0000 86
050: 0460 9294 4E42 0000 6C
051: 0000 0424 4154 E410 B3
052: 0000 0000 9710 294E 46
053: 4100 0000 0000 0785 B0
054: F4E4 2000 0000 0000 F7
055: 5520 5E42 0000 0002 78
056: 8407 3202 F410 0000 D0
057: 0004 9995 135F 4200 10
058: 0000 0000 2091 0205 6D
059: 1000 0000 6737 9031 24
05A: 4052 0000 0009 5301 82
05B: 3224 0520 0000 0000 C9
05C: 0270 2440 5200 0000 18
05D: 0000 4601 D405 2000 F2
05E: 0000 0000 541F 4052 87
05F: 0000 0000 0009 0225 76
060: 0520 0000 0077 0904 EF
061: 1450 5200 0000 0009 4D
062: 0591 5505 2000 0000 1A
063: 0000 4421 4252 0000 49
064: 0004 5206 2224 2510 C0
065: 0000 0008 7645 8542 AD
066: 5200 0000 0070 2681 CC
067: 8425 2000 0000 5509 CB
068: 201E 4252 0000 0000 2C
069: 0002 2255 2520 0000 43
06A: 0000 7010 1023 5100 3C
06B: 0000 0000 6023 2435 BF
06C: 2000 0000 0057 1213 75
06D: 4351 0000 0000 9559 33
06E: 4454 3510 0000 0000 DF
06F: 0698 7943 5100 0000 8B
070: 0055 8082 D435 2000 66
071: 0000 0004 051E 4352 A2
072: 0000 0000 0968 1125 05
073: 3510 0000 0000 0267 76
074: 8144 5200 0000 0974 F2
075: 9081 2445 2000 0000 1D
076: 4529 8513 4451 0000 AC
077: 0000 0000 7954 4520 18
078: 0000 0000 0672 1844 1A
079: 5200 0000 0183 0232 78
07A: 9445 1000 0000 0000 B6
07B: 974C 4452 0000 0000 4E
07C: 0734 02D4 4520 0000 CB
07D: 0024 3986 1025 5200 FB
07E: 0000 0092 0832 0265 1E
07F: 1000 0000 0414 9050 A4
080: 2752 0000 0000 0583 95
081: 8154 8520 0000 0000 FD
082: 0313 1029 5100 0000 27
083: 0095 0988 2495 2000 50
084: 0000 0040 371E 4A51 81
085: 0000 0000 0083 5625 FF
086: A510 0000 0000 0221 E8
087: 9000 0071 3415 E391 1E
088: 2719 1AC0 1631 6E6B 07
089: EFAF 24E0 2016 3AC2 B8
08A: 15EE 1BBB 8F21 4213 72
08B: 08D6 12F0           CF
```

Dennis Föh (2374)
Hermann-Hanker-Str. 17
3400 Göttingen

Wer kann helfen?

# Spezielle HP-71 Tastenzuordnung

Im 41-Translator-ROM existiert ein Keyfile das Tastenzuordnungen für Befehle wie XEQ"" mit Einfügungscursor auf dem hinteren " liefert.

Ein Key-List mit dem ThinkJet im Displaymode zeigt hinter dem Schlußzeichen " die Folge der ASCII-Zeichen 8 (Backspace), 27 (Escape), N (umschalten auf Einfügen).

Diese Abfolge ist ganz einsichtig, jedoch ist mir bisher ihre Erzeugung nie ohne sofortige Ausführung gelungen.

Wer kann dabei helfen?

Dipl.-Ing. Siegbert Förster
Frankenstraße 22
8501 Roßtal

```
DEF KEY '/',' /' &R
&DEF KEY '*',' *' &R
&DEF KEY '-',' -' &R
&DEF KEY '#50',' RCL ';&R
&DEF KEY '#51',' STO ';&R
&DEF KEY ',','-';&R
&DEF KEY '+',' +' &R
&DEF KEY 'fW',' XEQ""&E&N';&R
&DEF KEY 'fR',' GTO""&E&N';&R
&DEF KEY 'fY',' SF ';&R
&DEF KEY 'fU',' CF ';&R
&DEF KEY 'f0',' S+' &R
&DEF KEY 'fP',' S-' &R
&DEF KEY 'f7',' CLS' &R
```

# Funktionen zur Basisumwandlung im HP-71 B

Für eine einfache Umwandlung von Dezimalzahlen in binäre (duale), oktale oder hexadezimale Zahlen (und umgekehrt, auch gemischt), lassen sich im BASIC-Modus die folgenden Schlüsselworte verwenden: BVAL und BSTR$. Erforderlich ist das Mathe-ROM im HP-71B. Nur ganze positive Zahlen können berechnet werden, gebrochene Zahlen werden gerundet. Die Ergebnisse sind stets in ganzen Zahlen. Bei Binärdarstellung werden führende Nullen unterdrückt. BVAL zur Umwandlung von binär, oktal oder hexadezimal in DEZIMAL. BSTR$ zur Umwandlung von DEZIMAL in binär, oktal oder hexadezimal. Die Umwandlung erfolgt intern stets über das Dezimalsystem.

BVAL ("1010", 2) = 10 Dezimaläquivalent v. 1010 binär
BVAL ("17", 8) = 15 Dezimaläquivalent von 17 oktal
BVAL ("46", 16) = 70 Dezimaläquivalent von 46 hexadezimal
BSTR$ (3,2) = 11 Binärdarstellung von 3 dezimal
BSTR$ (72,8) = 110 Oktaldarstellung von 72 dezimal
BSTR$ (88,16) = 58 Hexadezimaldarstell. v. 88 dezimal

Der Wert des Dezimaläquivalents darf nicht größer sein als:
999 999 999 999
Der binäre Stringausdruck darf nicht größer sein als:
1110 1000 1101 0100 1010 0101 0000 1111 1111 1111
Der oktale Stringausdruck darf nicht größer sein als:
16432451207777
Der hexadezimale Stringausdruck darf nicht größer sein als:
E8D4A50FFF

Verknüpfungen:
BSTR$ (BVAL("AF", 16), 2) = 10101111 binär
BSTR$ (BVAL("9F", 16), 8) = 237 oktal
BSTR$ (BVAL("14772", 8) + BVAL ("570", 8), 8) = 15562
(Oktale Summe aus 14772 okt. und 570 okt.)

BSTR$ (BVAL("110", 8) - BVAL ("1010", 2), 2) =
111110 binär (62 dez.)
BSTR$ (BVAL("110", 8) / BVAL ("1010", 2), 2) =
111 binär (7 dez.)
BSTR$ (BVAL("110", 8) * BVAL ("1010", 2), 2) =
1011010000 binär (720 dez.)
BSTR$ (BVAL("110", 8) + BVAL ("1010", 2), 2) =
1010010 binär (82 dez.)

```
10 B$ = "1111" ENDLINE
20 BVAL (B$, 2) ENDLINE =
        15 Dezimaläquivalent des Binärstrings "1111"
30 BVAL (B$&B$, 2) ENDLINE =
        255 Dezimaläquivalent des Binärstrings "11111111"
```

H.A. Wuttke (CCD 2742)
Hainer Weg 271
6000 Frankfurt 70

# Haushalt & CCD-Modul

## Anfängerübungen mit CCD-Feldfunktionen am Beispiel eines Haushalts-Programmes

**170 Zeilen, 530 Bytes, 76 Regs., SIZE 025, HP41CV, 1XF, TIME, CCD, PRINTER, RAMBOX**

### 1. Zielsetzung

- Benutzung der MEMORY-LOST-sicheren Speichermöglichkeit für Daten in der Rambox.
- Anwenden von Feldfunktionen des CCD-Moduls, da mir die enstprechenden Funktionen des PPC-Moduls schon für ähnliche Mißbräuche nützlich waren.
- Die Eingabewerte eines Verarbeitungsganges - z.B. Monatsende - sollen in zwei getrennten Gruppen - z.B. Ausgaben/Einnahmen - summiert werden.
- Einfache Korrekturmöglichkeit für Eingabefehler.
- Protokollierung.
- Wahlweiser Druck eines Auszuges der Kontostände.

### 2. Gebrauch der Feldfunktionen

- Felder:

Felder im RAM erlauben direkten Zugriff. Im XM, in welchem der HP41-Benutzer kaum auf Dauer speichert, steht weniger wertvoller Platz zur Verfügung. Aber der Weg Rambox-XM und umgekehrt führt - für mich und heute - über den Hauptspeicher, braucht also zwei Schritte und mindestens temporär doch Platz im RAM. Einreihige (oder -spaltige) Felder erlauben im RAM Eingabe mittels STO+ und Registernummer (kein Positionieren des Zeigers). Benötigt wird je eine Reihe für die Eingaben und eine für die Summen. Nur letztere soll von und zur Rambox übertragen werden. Also: Zwei einreihige Felder, das Eingabefeld im RAM, das Summenfeld im XM.

- Funktionen:

Eingabe mittels STO+. Da in einem Arbeitsgang mehrere Posten mit gleicher Kontonummer vorkommen können, müsste bei Eingabe mittels R+ die STO+ - Funktion simuliert werden.

Reihensummen mittels SUM, da nur einreihige Felder (sonst RSUM, benötigt aber ein Resultat-Feld). Die mit MDIM angelegten Felder werden mit M- auf Null gesetzt für den Fall, daß sie bereits von früheren Arbeitsgängen her bestehen sollten. (Dann wäre MDIM nur eine Neudimensionierung ohne Nullsetzung).

*M- kann nicht durch CLFL ersetzt werden. Für RAM-Felder ergibt CLFL natürlich FL NOT FOUND. Schlimmer im XM: Hier wird der Header des angesprochenen Files so verändert, daß manche nachfolgenden Funktionen ihn nicht mehr erkennen (Dauerschleife, ON und ENTER). CAT 4 und PURFL funktionieren jedoch weiterhin.*

### 3. Benutzungsanleitung

- Initiieren vor dem ersten Durchlauf:
File HH-24 in der Rambox anlegen (24 Register). An Stelle von PDT (kommt zweimal vor) die eigene Routine für Datums- und Uhrzeitausdruck einbauen (jeder hat seine am liebsten).

- Dialog-Fragen:

a) Konto ? N=R/S
erwartet Eingabe einer Kontonummer zwischen 1 und 24. Eingabe der Nummer Null führt beim Eintippen des Betrages zur Meldung ALPHA DATA, Abhilfe GTO 00. (In REG 00 liegt der Feld-Header). Eingabe einer Nummer über 24: Fehlermeldung und neuerliche Abfrage. Nach Eingabe des letzten Betrages wird die Abfrage mit R/S beantwortet, das Programm läuft weiter zur nächsten Frage:

b) OK ? J=R/S N=0
Nach der Antwort "Ja" (R/S) läuft das Programm weiter, auf die Antwort "Nein" (0 oder eine andere Ziffer) folgt die als "KORREKTUR" gekennzeichnete neuerliche Abfrage gemäss a), der fehlerhafte Wert kann durch den richtigen ersetzt oder durch Eingabe der Differenz korrigiert werden.

c) AUSZ ? J=R/S N=0
Auf "Ja" wird der Auszug der Kontostände nach dem eben abgeschlossenen Arbeitsgang ausgedruckt. Auf "Nein" folgen nur BEEP und die Schlußlinie, die übrigens auch den Ausdruck der Kontostände abschließen.

- Weitere Details:
Feld "R000" dient der Eingabe, daneben zu Beginn und am Ende eines Durchlaufs als Durchgangsstation zwischen Rambox und XM für das die Kontosummen enthaltende "Summenfeld" "A1". Die gewünschte Trennung in zwei Kontengruppen 1-20 bzw. 21-24 hat ihren Preis. Sie wird erreicht, indem für die Summenbildung die Konti 21-24 falls angesprochen nach den Feldern "A2" und "A3" verschoben werden. Im Hinblick auf den Ausdruck der Kontostände wird diese Verschiebung für Feld "A1" rückgängig gemacht.

Gewiss, es geht auch einfacher, aber: So kamen Zielsetzung und Bastlerfreude zu ihrem Recht. M.Meyer mag es mit seinem Haushalts-Programm Nr. 1430 ähnlich ergangen sein.

**Ein Beispiel:**
**Druckroutine für Datum und Zeit**

```
HAUSHALT

DATUM: 18.02.90
ZEIT:  16:56

KONTO              BETRAG
------------------------
K. 20               20,00
K. 1               100,00
K. 24               10,00
??? KTO-NR 31
K. 21               10,00
K. 24               14,00
K. 21               11,00
------------------------
KORREKTUR
K. 1              -100,00
K. 1                1,00
------------------------
ZWΣ 1-20           21,00
TOT 1-20          126,00
ZWΣ 21-24          45,00
TOT 21-24         270,00
------------------------

KONTOAUSZUG
DATUM: 18.02.90
ZEIT:  16:58
K. 1                6,00
K. 2                0,00
K. 3                0,00
K. 4                6,00
K. 5                0,00
K. 6                0,00
K. 7                0,00
K. 8                0,00
K. 9                0,00
K. 10               0,00
K. 11               0,00
K. 12               0,00
K. 13               0,00
K. 14               0,00
K. 15               0,00
K. 16               0,00
K. 17               0,00
K. 18               0,00
K. 19               0,00
K. 20             120,00
K. 21             126,00
K. 22               0,00
K. 23               0,00
K. 24             144,00
++++++++++++++++++++++++
```

**Decodierung**

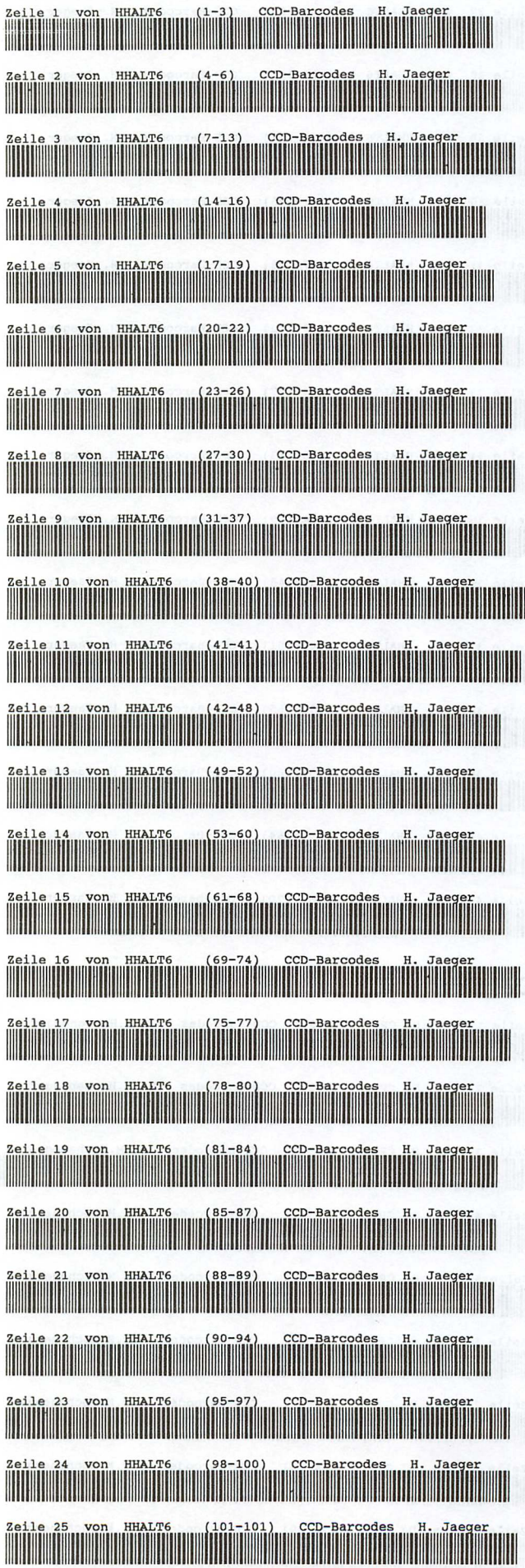
**Header Feld A1 (1,024) im X-Memory vor und nach CLFL**

```
ADRESSE   BYTE-NUMMER
DEZ HEX   66554433221100

188 0BC   00000000000000

189 0BD   01000000000000

190 0BE   40BE0018018018

191 0BF   41312020202020


ADRESSE   BYTE-NUMMER
DEZ HEX   66554433221100

188 0BC   00000000000000

189 0BD   FF0000000000000

190 0BE   40BE0000000018

191 0BF   41312020202020
```

```
                PRP "HHALT6"

 01♦LBL "HHALT6"
SF 12  "HAUSHALT"  AVIEW
CF 12  SIZE?  25  X>Y?
PSIZE  ADV  XEQ "PDT"
ADV  "R000"  1,024
XEQ 03  "A1"  XEQ 03
"HH-24"  GTREGX
"R000,A1"  M+  "R000"
M-  "KONTO"  ACA  12
SKPCHR  "BETRAG"  ACA
PRBUF  PRL  CF 00  CF 22
SF 29  FIX 2

 36♦LBL 00
"KORREKTUR"  FS?C 22
AVIEW  "KONTO ? N=R/S"
PROMPT  FC?C 22  GTO 04
21  X<=Y?  SF 00  CLX
24  X<>Y  X>Y?  GTO 09
"BETRAG ?"  PROMPT
ST+ IND Y  XEQ 01  CF 22
GTO 00

 58♦LBL 01
X<>Y  "K. "  ARCLI  RDN

 63♦LBL 02
0  X<>Y  PRAXY  RTN

 68♦LBL 03
MDIM  M-  RTN

 72♦LBL 04
PRL  "OK? J=R/S N=0"
PROMPT  FS? 22  GTO 00
"R000,A1"  M+  FS? 00
XEQ 06  "R000"  SUM
"ZWΣ 1-20"  XEQ 02  "A1"
SUM  "TOT 1-20"  XEQ 02
FS?C 00  XEQ 08  PRL
"R000"  M-  "A1,R000"
M+  "HH-24"  1,024
LDREGX
"AUSZ? J=R/S N=0"
PROMPT  FS? 22  GTO 10
ADV  "KONTOAUSZUG"
AVIEW  XROM "PDT"
"R000"  CLX  IJ=A

111♦LBL 05
2,001  ?IJ  X=Y?  GTO 10
FRC  E3  *  R>+  XEQ 01
GTO 05

122♦LBL 06
"A2"  1,004  XEQ 03
"A3"  XEQ 03  "R000,A2"
XEQ 07  "A1,A3"  XEQ 07
RTN

133♦LBL 07
1,001  ENTER↑  1,021
ENTER↑  1,024  SWAP  RTN

141♦LBL 08
"A2"  SUM  "ZWΣ 21-24"
XEQ 02  "A3"  SUM
"TOT 21-24"  XEQ 02
"A3,A1"  1,021  ENTER↑
1,001  ENTER↑  1,004
SWAP  RTN

158♦LBL 09
BEEP  "??? KTO-NR "
ARCLI  AVIEW  CF 22
GTO 00

165♦LBL 10
CLX  ACLX  ADV  BEEP
END

                PRP "PDT"

 01♦LBL "PDT"
RCLFLAG  STO L  DMY
CF 28  SF 29  FIX 4
DATE  "DATUM: "  ADATE
AVIEW  FIX 2  "ZEIT:  "
TIME  ATIME24  AVIEW
RCL L  STOFLAG  RTN  END
```

Dipl.-Ing. H. Jaeger (1679)
Weinbergstr. 44
CH-8302 Kloten

## HHALT6



Zeile 26 von HHALT6 (102-105) CCD-Barcodes H. Jaeger

Zeile 27 von HHALT6 (106-108) CCD-Barcodes H. Jaeger

Zeile 28 von HHALT6 (109-114) CCD-Barcodes H. Jaeger

Zeile 29 von HHALT6 (115-121) CCD-Barcodes H. Jaeger

Zeile 30 von HHALT6 (122-126) CCD-Barcodes H. Jaeger

Zeile 31 von HHALT6 (127-128) CCD-Barcodes H. Jaeger

Zeile 32 von HHALT6 (129-132) CCD-Barcodes H. Jaeger

Zeile 33 von HHALT6 (133-137) CCD-Barcodes H. Jaeger

Zeile 34 von HHALT6 (138-143) CCD-Barcodes H. Jaeger

Zeile 35 von HHALT6 (144-145) CCD-Barcodes H. Jaeger

Zeile 36 von HHALT6 (146-148) CCD-Barcodes H. Jaeger

Zeile 37 von HHALT6 (149-151) CCD-Barcodes H. Jaeger

Zeile 38 von HHALT6 (152-155) CCD-Barcodes H. Jaeger

Zeile 39 von HHALT6 (156-160) CCD-Barcodes H. Jaeger

Zeile 40 von HHALT6 (161-164) CCD-Barcodes H. Jaeger

Zeile 41 von HHALT6 (165-170) CCD-Barcodes H. Jaeger

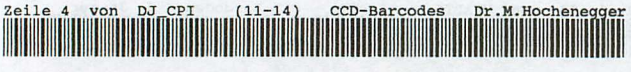
## DJ_CPI

Zeile 1 von DJ_CPI (1-3) CCD-Barcodes Dr.M.Hochenegger

Zeile 2 von DJ_CPI (4-8) CCD-Barcodes Dr.M.Hochenegger

Zeile 3 von DJ_CPI (9-10) CCD-Barcodes Dr.M.Hochenegger

Zeile 4 von DJ_CPI (11-14) CCD-Barcodes Dr.M.Hochenegger



Zeile 5 von DJ_CPI (15-20) CCD-Barcodes Dr.M.Hochenegger

Zeile 6 von DJ_CPI (21-21) CCD-Barcodes Dr.M.Hochenegger

Zeile 7 von DJ_CPI (22-23) CCD-Barcodes Dr.M.Hochenegger

Zeile 8 von DJ_CPI (24-25) CCD-Barcodes Dr.M.Hochenegger

Zeile 9 von DJ_CPI (26-29) CCD-Barcodes Dr.M.Hochenegger

Zeile 10 von DJ_CPI (30-35) CCD-Barcodes Dr.M.Hochenegger

Zeile 11 von DJ_CPI (36-43) CCD-Barcodes Dr.M.Hochenegger

Zeile 12 von DJ_CPI (44-50) CCD-Barcodes Dr.M.Hochenegger

Zeile 13 von DJ_CPI (51-57) CCD-Barcodes Dr.M.Hochenegger

Zeile 14 von DJ_CPI (58-62) CCD-Barcodes Dr.M.Hochenegger

Zeile 15 von DJ_CPI (63-65) CCD-Barcodes Dr.M.Hochenegger

Zeile 16 von DJ_CPI (66-70) CCD-Barcodes Dr.M.Hochenegger

Zeile 17 von DJ_CPI (71-73) CCD-Barcodes Dr.M.Hochenegger

Zeile 18 von DJ_CPI (74-77) CCD-Barcodes Dr.M.Hochenegger

Zeile 19 von DJ_CPI (78-80) CCD-Barcodes Dr.M.Hochenegger

Zeile 20 von DJ_CPI (81-84) CCD-Barcodes Dr.M.Hochenegger

Zeile 21 von DJ_CPI (85-90) CCD-Barcodes Dr.M.Hochenegger

Zeile 22 von DJ_CPI (91-91) CCD-Barcodes Dr.M.Hochenegger

## DJ_ZL

Zeile 1 von DJ_ZL (1-4) CCD-Barcodes Dr.M.Hochenegger

Zeile 2 von DJ_ZL (5-5) CCD-Barcodes Dr.M.Hochenegger

Zeile 3 von DJ_ZL (6-6) CCD-Barcodes Dr.M.Hochenegger



Zeile 4 von DJ_ZL (7-11) CCD-Barcodes Dr.M.Hochenegger



Zeile 5 von DJ_ZL (12-15) CCD-Barcodes Dr.M.Hochenegger



Zeile 6 von DJ_ZL (16-17) CCD-Barcodes Dr.M.Hochenegger



Zeile 7 von DJ_ZL (18-22) CCD-Barcodes Dr.M.Hochenegger



Zeile 8 von DJ_ZL (23-31) CCD-Barcodes Dr.M.Hochenegger



Zeile 9 von DJ_ZL (32-36) CCD-Barcodes Dr.M.Hochenegger



Zeile 10 von DJ_ZL (37-43) CCD-Barcodes Dr.M.Hochenegger



Zeile 11 von DJ_ZL (44-48) CCD-Barcodes Dr.M.Hochenegger



Zeile 12 von DJ_ZL (49-49) CCD-Barcodes Dr.M.Hochenegger

## SERVICELEISTUNGEN

**BEST OF PRISMA**
Schutzgebühr: 30,- DM

**Nachsendedienst PRISMA**
Schutzgebühr: 5,- DM pro Heft für Jahrgänge 1982-86
10,- DM pro Heft für Jahrgänge ab 1987

**Inhaltsverzeichnis PRISMA**
Schutzgebühr: 3,- DM in Briefmarken

**Programmbiliothek HP-71**
Die bislang in PRISMA erschienenen Programme können durch Einsenden eines geeigneten Datenträgers (3,5" Diskette, Digitalkassette oder Magnetkarte) und eines SAFU angefordert werden.

**MS-DOS Inhaltsverzeichnis**
Kann durch das Einsenden einer formatierten 360 kB oder 1,2 MB 5,25"-Diskette oder einer formatierten 720 kB oder 1,44 MB 3,5"-Diskette und einem SAFU angefordert werden.

**ATARI Inhaltsverzeichnis**
Kann durch das Einsenden einer 3,5"-Diskette + SAFU bei Werner Müller angefordert werden.

**UPLE**
Das UPLE-Verzeichnis mit der Kurzbeschreibung der einzelnen Programme sowie den Bezugsbedingungen kann gegen Einsenden von DM 10,- in Briefmarken angefordert werden.

**Programme aus BEST OF PRISMA**

a) Eine Kopie der Programme von BEST OF PRISMA auf Kassette erfordert das Beilegen einer Leerkassette und eines SAFU.
b) Für Barcodes von BEST OF PRISMA-Programmen gibt es folgendes Verfahren:
Schickt eine Liste mit den Namen und der Seitenangabe (der Barcodeseiten) an die Clubadresse, pro Barcodeseite legt bitte 40 Pf., plus 2,40 DM für das Verschicken, in Briefmarken bei. Die Liste der verfügbaren Programme ist in Heft 3/88 auf der Seite 35 abgedruckt, sie kann gegen einen SAFU angefordert werden.

Der Bezug sämtlicher Clubleistungen erfolgt über die Clubadresse, soweit dies nicht anders angegeben ist, oder telefonisch bei Dieter Wolf:

(069) 76 59 12

Die eventuell anfallenden Unkostenbeiträge können als Verrechnungsscheck beigelegt werden, Bargeld ist aus Sicherheitsgründen nicht zu empfehlen; ist dies nicht der Fall, so wird Rechnung gestellt, dies macht die Sache natürlich nicht unbedingt einfacher, bzw. schneller.
Formvorschriften für Schreiben an die Clubadresse gibt es keine; das Schreiben kann durchaus handschriftlich verfasst sein, ein normaler Sterblicher sollte es noch lesen können. Vor allem den Absender und die Mitgliedsnummer deutlich schreiben !
(SAFU = Selbst Adressierter Freiumschlag)

## CLUBADRESSEN

**1. Vorsitzender**
Gerhard Link (3107),
Postfach 1615, 6090 Rüsselsheim,
✆ (06142) 81 51 0, Fax: (06142) 81 57 9, GEO1:G.LINK

**2. Vorsitzender**
Alf-Norman Tietze (1909),
Sossenheimer Mühlgasse 10,
6000 Frankfurt 80, ✆ (069) 34 62 40, GEO1:A.N.TIETZE

**Schatzmeister**
Dieter Wolf (1734),
Pützerstraße 29, 6000 Frankfurt 90,
✆ (069) 76 59 12, GEO1:D.WOLF

**1. Beisitzer**
Norbert Resch (2739),
Tsingtauerstraße 69, 8000 München 82

**2. Beisitzer**
Werner Dworak (607),
Allewind 51, 7900 Ulm,
✆ (07304) 32 74, GEO1:W.DWORAK

**MS-DOS Service / Beirat**
Alexander Wolf (3303),
Pützerstraße 29, 6000 Frankfurt 90,
✆ (069) 76 59 12

**ATARI Service / Beirat**
Dr. Werner Müller (1865),
Schallstraße 6, 5000 Köln 41,
✆ (0221) 40 23 55, MBK1:W.MUELLER

**Regionalgruppe Berlin**
Jörg Warmuth (79), Wartburgstraße 17, 1000 Berlin 62

**Regionalgruppe Hamburg**
Alfred Czaya (2225), An der Bahn 1, 2061 Sülfeld,
✆ (040) 43 36 68 (Mo.-Do. abends)
Horst Ziegler (1361), Schüslerweg 18b, 2100 Hamburg 90,
✆ (040) 79 05 67 2

**Regionalgruppe Karlsruhe / Beirat**
Stefan Schwall (1695), Rappenwörthstraße 42,
7500 Karlsruhe 21, ✆ (0721) 57 67 56, GEO1:S.SCHWALL

**Regionalgruppe Rheinland/Ruhrgebiet**
Jochen Haas (2874), Roßstraße 27, 5000 Köln 30,
✆ (0221) 51 98 70

**Regionalgruppe München / Beirat**
Victor Lecoq (2246), Seumestraße 8, 8000 München 70,
✆ (089) 78 93 79

**Regionalgruppe Rhein-Main**
Andreas Eschmann (2289), Lahnstraße 2, 6906 Raunheim,
✆ (06142) 46 64 2

**Beirat**
Manfred Hammer (2742), Oranienstraße 42, 6200 Wiesbaden

**Beirat**
Peter Kemmerling (2466), Danziger Straße 17, 4030 Ratingen

**Beirat**
Martin Meyer (1000), Kelkheimer Straße 20, 6232 Bad Soden 1

**CP/M-80**
Peter-C. Spaeth, Michaeliburgstraße 4, 8000 München 80

**E-Technik**
Werner Meschede (2670), Sorpestr. 4, 5788 Siedlingshausen

**Grabau GR7 Interface**
Holger von Stillfried (2641), Am Langdiek 13, 2000 Hamburg 61

**Hardware 41**
Winfried Maschke (413), Ursulakloster 4, 5000 Köln 1,
✆ (0221) 13 12 97

**HP-71 Assembler (LEX-Files)**
Matthias Rabe (2062), Teichsheide 13, 4800 Bielefeld,
GEO1:M.RABE

**Mathematik**
Andreas Wolpers (349), Steinstraße 15, 7500 Karlsruhe

**Naturwissenschaften**
Thor Germann (3423), Hobeuken 18, 4322 Spockhövel 2,
✆ (02339) 39 63

**Serie 80**
Klaus Kaiser (1661), Mainzer Landstr. 561, 6230 Frankfurt 80,
✆ (069) 39 78 52

**Vermessungswesen**
Ulrich Kulle (2719), Memeler Straße 26, 3000 Hannover 51,
✆ (0511) 60 42 72 8

**Programmbibliothek HP-71**
Henry Schimmer (786), Homburger Landstr. 63,
6000 Frankfurt 50

**"Clubadresse"**
CCD e.V., Postf. 11 04 11, 6000 Frankfurt 1, ✆ (069) 76 59 12

Postvertriebsstück
Gebühr bezahlt

D 2856 F

CCD - Computerclub Deutschland e
Schwalbacher Straße 50
D-6000 Frankfurt am Main 1

CCD
ISSN 0176-8735
PRISMA
...ai / Juni 1990 Nr. 3

# Shareware

Die auf den ATARI- und MS-DOS-Infodisketten veröffentlichten Shareware-Programme sind NICHT Public-Domain. Das heißt die Programme dürfen weitergegeben und ausprobiert werden, wenn einem aber ein Programm gefällt und man damit arbeitet, dann muß man das Programm auch lizensieren lassen. Das kostet zwischen 10 und 50 US-Dollar, ist also vergleichsweise sehr preiswert.

Das Shareware-Konzept ermöglicht es Programme erst einmal Auszuprobieren, bevor man Geld dafür ausgeben muß. Es vertraut darauf, daß die Benutzer die Programme auch zu den wirklich geringen Kosten lizensieren lassen.

Dieser Vertriebsweg verdient Unterstützung!

Nur wenn die Anwender auch Lizenzgebühren bezahlen, kann sich Shareware mehr und mehr durchsetzen. Die meisten Shareware-Programme sind mit sehr viel Aufwand programmiert worden und werden ständig weiterentwickelt. Viele Shareware-Programmierer leben von ihren Programmen.

Deutsche Anwender stehen oftmals vor dem Problem, wie sie das Geld in die USA bringen.

Wer eine der verbreiteten Kreditkarten (Eurocard, American Express, VISA, ...) sein eigen nennt, hat es am einfachsten: Den Programmen liegt immer eine Datei mit einem vorgefertigten Brief zum Lizensieren bei, der nur noch ausgefüllt werden muß. Einfach die entsprechenden Angaben zur Kreditkarte (Typ, Nummer, etc.) machen, abschicken, fertig.

Wer keine Kreditkarte besitzt, der besorgt sich am besten eine "Money Order" über den entsprechenden Betrag. Die gibt es zum Beipiel bei Niederlassungen amerikanischer Banken oder auch bei den American Express Reisebüros. Die Money Orders werden auf den Empfänger ausgestellt, nur dieser kann etwas damit anfangen. Den Brief ausfüllen, die Money Order zulegen, abschicken, fertig.

Nicht ungeduldig werden! Es dauert schon etwa sechs bis acht Wochen, ehe die Antwort im Kasten ist.

dw

**Nächstes**

## Ortstreffen in Köln

## 22. September 1990,

ab 13 Uhr im Vereinshaus der Humboldt-Siedlung, Frankfurter Straße 855, 5000 Köln 91 (Ostheim)

Verantwortlich und Ansprechpartner:

Dr. Werner Müller,
Schallstraße 6,
5000 Köln 41,
Tel. (0221) 40 23 55

und

Dipl.-Ing. Jürgen Schramm,
Frankfurter Straße 853,
5000 Köln 91,
Tel (0221) 8 90 23 54.