PASMA

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

März/April 1988 Nr. 2 D 2856 F

Das CMT-Multicase ist ein wasserfestes Schutzgehäuse aus Polykarbonat für den HP-71 mit integriertem 8-zeiligen LC-Display. Außerdem befinden sich in diesem Gehäuse auch 128 KByte RAM-Disk sowie eine HP-IL /RS-232C-Schnittstelle. Durch den eingebauten Heizwiderstand kann der HP-71 auch im rauhen Außendiensteinsatz bei Minus-Graden verwendet werden.

Verkaufe wegen Systemerweiterung: **HP-IL-Digital-cassette-Drive** (HP 82161A) mit fünf Cassetten für DM 400,–; ThinkJet (HP2225B-IL) mit 1500 Blatt HP-Papier und zwei Köpfen für DM 600,–
W. Meschede, Sorpestr. 4, 5788 Siedlinghausen

Verkaufe: **Kass. Laufwerk 82161A**, 3 Kass. 500,– DM, Kartenleser 82104A, 65 Karten, 250,– DM
☎089/2714396, abends

Suche 41C-Programmsammlung
Cardiac/Pulmonary (00041-90097) Small Business (00041-90137), oder deren Kopie.
Wolfgang Knell, Prozessionsweg 29, 4720 Beckum

Verkaufe HP41 CCD-Modul, komplett mit Zubehör wegen Systemaufgabe gegen Gebot. ☎06131/364994

HP Portable Plus, 256KB RAM, 4MB ROM-Disk mit Lotus 123, MS-WORD u.a. 2900,–, HP ThinkJet IL 700,– HP9114 Disk 800,–, HP 28 CD 350,–, ☎ 08104/484

Verkaufe **HP 9121 Diskdrive** 3½ Zoll Doppellaufwerk unbenutzt. VB 900,–
Harry Schmidt, ☎ 02182/9640 ab 17 Uhr

Verkaufe: HP75C, Assembler-Handbuch + Kassette, Text 75-Programm zusammen DM 1.050,–
Dig. Cassettenlaufwerk HP 82161A mit 8 neuen Kassetten, zusammen DM 650,–
Think-Jet IL, DM 700,–
Video-Interface HP 82163B, DM 150,–
alle Einheiten mit Handbüchern, orig. Zubehör und Verp.
PRISMA-Heft, Jahrg. 1983-1987, kompl. DM 3,50/Heft
Peter Reichetseder, ☎05943/83-140
(Büro) oder 05921/35250 (priv.)

Biete an:
HP41-Hardware: CCD-Modul (Version A) 180.00 DM, Quad-Ram 110,00 DM, X-Funktion 110.00 DM, Struktural-Analysis-Modul (HP41-15021) 70.00 DM je mit Handbüchern, Kurzanleitungen, Overlays in sehr gutem Zustand.
HP71-Software: DISASSEMBLER-FORTHFILE, die Ausgabe gleicht der des Assemblers, wahlweise auf Display, Printer oder als Textfile in RAM oder Massenspeicher. Die Textfiles können vom Assembler direkt neubearbeitet werden. ca. 22 KB-RAM erforderlich. HP-Kassette (ohne), ca. 75 Seiten Beschr., Beisp., Tabellen 240.00 (215.00) DM
REALMAT-LEXFILE, neun Befehle zur Erweiterung der Matrizen-Bearbeitung: addieren, subtrahieren, kopieren von Unter- in größere Hauptmatrizen, löschen von Matrixspalten und -Reihen, sortieren in auf- und absteigender Reihenfolge, spiegeln von quadratischen Matrizen an der Hauptdiagonalen, Berechnung des Polynomwertes angelehnt an den MAT-Befehl PROOT. ca. 1.6KB
HP-Kassette (ohne), ca. 30 Seiten Listings und Beispiele 110.00 (85.00) DM
jeweils zuzüglich 10.00 DM Versandkosten.
Wolfgang Dittrich (2069), Josephinenstr. 4, 4630 Bochum 1
☎ 0234/590547 mögl. zw. 17.00 und 19.00 Uhr

Hiermit biete ich folgende Hardware zum Verkauf an:
HP-71B mit HHP-Speichererweiterung 32K, HP-Speichererweiterung 8K, HP-Texteditor-Modul, HP-Forth-Assembler-ROM, alles mit Original-Handbüchern.
Grabau Videointerface GR7IL, HP-Videointerface
Diskettenlaufwerk HP-9114, mit W&W-Netzteil (keine Akku-Probleme!!)
Tintenstrahldrucker HP-2225B
Diverse Literatur und Utilities auf Diskette
HP-71-Basic leichtgemacht von J.K. Horn, HP71-Software Developers Handbook, HP71-Users Library Games, HP71-Users Library Utilities, HP71-Users Library Math.
Zur gesamten Hardware werden die Originalhandbücher mitgeliefert.
Preisvorstellung bei Gesamtabnahme DM 4.500,– als VB.
Ich bin tagsüber telefonisch erreichbar in der Regenbogen-Apotheke, Inh. Ingo Logemann, Wedeler Landstr. 53K
2000 Hamburg 56, ☎ 040/811033

Verkaufe: **Forth/Assembler** ROM 82331A, Preis 200,– DM; GRABAU, **GR7** Grafik-Video-Controller, Seriennr. 124, für IL, 4 Seiten Bildspeicher, Preis 1000,– DM; Armin Thesen, Im Schwalg 14, 6530 Bingerbrück, ☎ 06721/32753

Suche preisgünstiges Videointerface z.B. HP92198 (Mountain) / PACSCREEN, Manfred Hammer (2743) Oranienstr. 42, 6200 Wiesbaden, ☎ 06121/375294

HP-41 CX. Wer hat versucht zu programmieren bzw. arbeitet schon mit einem Programm, welches die „Technische Analyse von Aktientrends" nach Edwards/Magee beinhaltet? Gedankenaustausch, Programm-Preise etc. erwünscht. Bernd Knebel (363), Im Lämmle 13
7534 Birkenfeld 2, ☎ 07082/20426

Suche für private Zwecke Gillardon-Finanzprogramme für das HP-41 System (Original od. Kopie).
J. Rimböck, Spalenring 123, CH-4055 Basel

Verkaufe: Prisma 9/83, Key Notes 3/82, mehrere Jahrgänge HP Journal, Standard-Sammlung, HP 32 E Bed. Handbuch, HP Allg. Handbuch gegen Gebot.
Video/F 82163 B (150,–), XF 82180 A (100,–), TIME 82182 A (100,–), Netzteil 82066 (15,–).
Suche: Defekte Akkus und Netzteile, Key Notes 1983.
Peter Wintner, ☎ 089/883221

Verkaufe **HP75C**, da doppelt mit Software für 600,– DM
Suche Developmentmodul, RS232 Schnittstelle für HP41
Ralf Rosche, ☎040/6451920

Verkaufe (da doppelt) **HP71B** 1A Zustand mit **HPIL-Modul** 82401, **Akkus**, mit original Zubehör, (Tasche, Schablone, Handbücher und Kurzanleitungen) und interner Documentation. Ulrich Bunse, 4630 Bochum, ☎ 0234/705052

HP-System-Verkauf, HP41CV DM 175,–; IL-Modul DM 130,–; Advantage-ROM DM 90,–; X-Fkt. Modul DM 85,–; X-Mem. Modul DM 85,–; Time-Modul DM 85,–; Finanz-Modul DM 50,–; Statistik-Modul DM 50,–; Home-Management DM 50,–; Standard-Modul DM 40,–. Alle Preise incl. Porto; **Suche HP41CX** und Kartenleser. CCD. N. Heucke, ☎ 05373/6829

Verkaufe HP125 PC, bestehend aus Monitor, Tastatur, Doppelfloppy (5¼'') HP82901M, IB- und ser. Schnittstelle, 2 St. HP-IB-Kabel, Software, sehr ausf. Dok. VB 1300,– DM; **HP-IL-Converter 82166B** (von IL- auf parallel Schnittst. u. vise versa). Neu, ungebr. orig. verp. 250,– DM **NC-Akku-Pack 88014A** vom HP9114 Disc Drive, fast neu 60,– DM, Thinkjet CPU-Boards, IL und parallel, zum ausschlachten, 25,– DM; **2 Doppel-MM f. 41C** zusammen 50,– DM; Wer tauscht mein **IL-Thinkjet** gegen Centronic-Thinkjet? Wer hilft mir bzgl. **Hardware 41CX**? Ich rufe zurück! Gysbert Hageman, Alter Weg 1, 6653 Blieskastel 2 ☎ 06842/2805. (Geschäftl. 06842/3041/3042) CCD 3389

Verkaufe: **IL-ThinkJet HP2225B** mit ca. 1000 Bl. Endlospapier 12'' DM 950,–; **IL-Diskettenlaufwerk HP-9114B** mit 2 Disketten DM 1070,–; **IL-Plotter HP 7470A** mit div. Staedtler-Tuschespitzen, 0,3-0,7 und Marsonic 580 U-Schallreinigungsgerät DM 2125,–; **HP-75C mit I/O-ROM** und Graphics-Solutions auf M-Karte DM 1500,– **Plottermodul für HP-41** DM 180,–; **Real-Estate-Pac, Home-Management-Pac, Securites-Pac und Advantage-Pac** je DM 65,–; Alle Geräte mit Original-Bedienungshandbüchern. Konrad Albers, Südring 42, 2300 Klausdorf/Schwentine, ☎ tagsüber 0431/92055 nach 18.00 Uhr 0431/79494

Verkaufe HP-41CV und HP-IL Modul sowie jede Menge interessante Literatur (Keith Jarret, Wickes etc.) für HP-41; Preis VB; Matthias Rabe, Tel. (05204) 4379

Grabau Video-Interface GR7 mit 64KB Bildschirmspeicher zusammen mit HP9'' Monitor (grün) 1200,– DM, HP-75C zusammen mit 8KB RAM-Erweiterung, I/O-ROM Mathe-ROM und Graphik-Paket für 1200,–DM; Heinrich Sauer, 7314 Wernau am Neckar, Junckerstr. 86,
☎ 07153/3492

Verkaufe für HP-75C: ROM-Simulator HP82713A original verpackt 150,– DM und 2 Memory-Module (8kB) HP82700A je 70,– DM; Uwe Berg, ☎ (06174) 62389

Verkaufe für HP-41: CCD Modul 200,– DM; PPC Modul 100,– DM; Mathe Modul 40,– DM;
Michael Krocker, ☎ 06152/39130

Für HP41: WAND 1F 230,– DM; CCD-Modul 200,– DM; Magnetkarten 0,50 DM/Stück (gebraucht); IL-Dev.-Modul 200,– DM; Für HP 71: Kartenleser 330,– DM,
☎ 069/553520

ORTSGRUPPE HAMBURG
HP41/MS-DOS

Trifft sich regelmäßig

18.30 Uhr jeden 1. Mittwoch im Monat
Meldungen bitte an Heiko Hiller (1948)
Tel.: 040/38 07 29 83 (Dienst)
oder 04122/ 5 13 65 (Privat)

HP 41 CV	DM 170,–
Magnetkartenleser	DM 200,–
Time-Modul	DM 90,–
X-Function-Modul	DM 90,–
X-Memory-Modul	DM 100,–
Mathe-Modul	DM 40,–
Standard-Modul	DM 40,–
bei Gesamtabnahme	DM 590,–

Karlheinz Jünemann (1072), Rugewisch 18
2000 Hamburg 63, ☎ 040/5321111

Verkaufe: **CCD-Modul** mit Benutzerhandbuch für HP41C/CV/CX zum Preis von 200,– DM.
P. Jochen, Heilbronner Straße 240, 7410 Reutlingen,
☎ 07121/630163

Verkaufe: **Finanz-Modul HP41** 60,– DM, Thermodrucker 82143A mit Bar-Code-Druck-Möglichkeit 480,– DM, IL-Karte für Epson-Drucker 120,– DM, Standard Modul 25,– DM.
Software für PC: Clipper Summer 87 990,– DM, MS-Makroassembler 5.0 290,– DM, TITAN 150,– DM, Windows 386 V.2.0 350,–DM, ☎ 07121/320156 (tägl.)

Regionalgruppe Köln

Um sich gegenseitig beim Einsatz eines HP-Computers zu unterstützen, ist die Teilnahme an den Treffs der Regionalgruppe von höchstem Nutzen. Hier findet im Regelfall der Interessierte auch für berufliche Anwendungen im Kreis der Teilnehmer einen kompententen Gesprächspartner.

In der letzten Prisma (1/88) wurden die Themen und möglichen Programme unserer Treffen mit allen Terminen veröffentlicht. Hier bitte bei Interessen nochmals auf der zweiten Umschlagseite nachlesen.

Mit dem heutigen Aufruf weisen wir noch einmal auf die Treffen im „Wiesdorfer Treff" Hauptstr. 133-139 in 5090 Leverkusen hin und bitten um rege Teilnahme am 8. Mai 1988 um 15 Uhr.

Telefonische Auskünfte erteilen vorher gerne Werner Weith unter 02056-24005 und Frank Ortmann unter 0214-27218 jeweils abends.

Regionalgruppe Köln

Verkaufe: XF-Modul 70,- DM, Doppel X-Memory (2 in einem Gehäuse) 150,- DM, Drucker HP82143A 400,- DM; A. B. Gemein, 5000 Köln 60, An der Flora 9, Tel. (0221) 7606443

Die PRISMA-Redaktion bittet die Autoren um Einhaltung des neues Satzspiegels für handgezeichnete Formeln, Abbildungen, Skizzen oder Zeichnungen. Dadurch wird viel Zeit und Arbeit gespart - und selbstverständlich auch der Platz im PRISMA besser genutzt.

Zwei Spaltenmaße stehen zur Auswahl:
entweder 57 mm oder 87,5 mm. Aber bitte immer nur für ein Maß entscheiden.

Bei ganzseitigen Abbildungen gelten als maximale Abmessungen:
180 x 260 mm (Breite x Höhe).

Regionalgruppen

Karlsruhe

In Karlsruhe trifft man sich jeden 2. Mittwoch eines Monats zum Erfahrungsaustausch.
Kontakt: Stefan Schwall, (0721) 576756

München

Auch dort trifft man sich an jedem 2. Mittwoch eines Monats in der Gaststätte Schützenring, Weißenburger Platz, 8000 München, S-Bahn-Haltestelle Rosenheimerplatz
Kontakt: Victor Lecoq, (089) 789379

Rhein-Main

In Raunheim (bei Rüsselsheim) treffen sich die Mitglieder an jedem letzten Samstag eines Monates.
Kontakt:
Andreas Eschmann, (06142) 46642

PRISMA

Impressum

Titel:
PRISMA
Herausgeber:
CCD-Computerclub Deutschland e.V.
Postfach 11 04 11
Schwalbacher Straße 50
6000 Frankfurt am Main 1
Verantwortlicher Redakteur:
Alf-Norman Tietze (ant)
Redaktion:
Hans Jürgen Hübner (hjh)
Klaus Kaiser (kk)
Michael Krocker (mik)
Martin Meyer (mm)
Henry Schimmer (hs)
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:
3000
Bezug:
PRISMA wird von allen Mitgliedern des CCD ohne Anforderung übersandt. Ein Anspruch auf eine Mindestzahl von Ausgaben besteht nicht. Der Bezugspreis ist im Mitgliedsbeitrag enthalten.

Nachsendedienst

Die ab sofort gültige neue Anschrift lautet:

Computerclub Deutschland e.V.
PRISMA-Nachsendedienst
Postfach 11 04 11
D-6000 Frankfurt am Main 1

Inhalt

April, April ...

Ein unglücklicher Zufall hat uns, d.h. eigentlich mir, bereits im letzten Heft einen unbeabsichtigten "Aprilscherz" bereitet. Dort war nämlich in meinem Editorial über "Wichtige Clubadressen" zu lesen, daß man diese ab 1988 regelmäßig am Schluß des Heftes finden kann. Nur - dort waren sie nicht. Die Clubadressen wurden überhaupt nicht im letzten Heft abgedruckt. Aber wie so vieles, hatte auch das seinen Grund: Nachdem alle Artikel fertig waren kamen - wie immer - zum Schluß die Barcodes dazu. Beim Seitenumbruch stellte sich dann heraus, daß der Platz nicht ausreicht, und man beschloß kurzerhand, die Clubadressen wieder herauszunehmen. Dabei dachte jedoch niemand der zuständigen "PRISMA-Macher" mehr an den einleitenden Artikel, welcher eben diese zum Gegenstand hatte. Sorry!

Aber nicht verzweifeln, denn in diesem Heft ist - wie jedes Jahr im April - wirklich ein Aprilscherz zu finden. Er ist bestimmt nicht schwer zu finden und wir wünschen viel Spaß bei der Suche.

Alf-Norman Tietze
(verantwortlicher Redakteur)

HP Werksbesichtigung

Im Computerwerk Böblingen

Für interessierte Clubmitglieder bietet sich die einzigartige Gelegenheit, an einer Werksbesichtigung bei Hewlett Packard in Böblingen teilzunehmen. Dort werden in erster Linie HP Computer hergestellt. Der Besichtigungstermin findet statt am:

Montag, den 9. Mai 1988 um 14:00 Uhr

Interessenten schreiben bitte umgehend eine Postkarte mit kompletter Anschrift, Mitgliedsnummer und Telefon an die Geschäftsstelle in Frankfurt:

Computerclub Deutschland e.V.
Stichwort: HP Werksbesichtigung
Postfach 11 04 11
6000 Frankfurt am Main 1

Einsendeschluß ist der 2. Mai 1988

Leider ist die Teilnehmerzahl auf maximal 20 Personen beschränkt. Bei mehr Interessenten wird die PRISMA-Redaktion die Teilnehmer auslosen.

Stefan M. Schwall
(Beirat)

CeBIT '88 Tendenzen

Die diesjährige CeBIT fand nun zum zweiten Mal als eigenständige - von der Hannover Industrie Messe getrennte - Messe statt. Zu Beginn dieser Trennung im letzten Jahr war noch viel Kritik und Unverständnis darüber zu vernehmen, doch jetzt hat sich dieses Konzept als erfolgreich erwiesen. Die Belegung von mehr als 18 Hallen durch die Informations- und Telekommunikationsbranchen macht deutlich, daß die Trennung von Industrie und CeBIT sinnvoll ist. Fast 20 % mehr Besucher gegenüber dem Vorjahr (insgesamt 480 000) dokumentieren die wichtige Bedeutung von Informations- und Kommunikationstechniken in unserer Gesellschaft und somit auch die Bedeutung dieser Investitionsgüter-Messe.
Im Vergleich dazu hat die Internationale Automobil Ausstellung in Frankfurt - eine Konsumgütermesse - nur (!) gut die 4,5-fache Besucherzahl aufzuweisen.

Der Trend auf der diesjährigen CeBIT ging eindeutig in Richtung Software. Die Neuerungen bei der Hardware (da ein bißchen schneller, dort eine interessante neue Grafik-Karte mit besserem Bildschirm usw.) sind bei näherer Betrachtung geradezu langweilig gegenüber der sich abzeichnenden "Explosion" auf dem Softwaresektor: Experten- und Datenbanksysteme, Tabellenkalkulationen oder mathematische Lösungswerkzeuge, raffinierte Textverarbeitung und Desktop Puplishing (DTP), Grafik-Software (Präsentation und CAD), Computerunterstützte Produktion und Industrie (CAM, CAI), Computerintegriertes Management (CIM), gezielte Branchenlösungen ... etc. - es ließen sich noch viele weitere Details aufzählen. Eine wesentliche Voraussetzung für neue Software sind geeignete Entwicklungswerkzeuge (Programmiersprachen). Auch auf diesem Bereich tat sich - ganz unscheinbar - einiges mehr, als bei der Hardware. Mit etwas Phantasie kann man sich vorstellen, daß in Zukunft beim Kauf eines teueren Programmpaketes der Computer als Rabatt gratis dazugegeben wird. Schlechte Aussichten für Hardware-Hersteller, aber ein enormes Wachstumspotential auf dem Software-, Beratungs- und Schulungs-Sektor. Know-How ist gefragt.

Zusammenfassend konnte ich meinen (privaten) "Messe-Bummel" als spannend bezeichnen. Es gab viel zu entdecken und zu erfahren - hoffentlich auch wieder im nächsten Jahr. Die Hannover-Messe CeBIT '89 findet vom 8. bis 15. März statt.

Alf-Norman Tietze
(Redaktion)

Neuer Rechner-Befehlssatz

Instruction	Meaning	Bedeutung
BAH	*Branch and hang*	Verzweige und bleibe hängen
IIB	*Ignore interrupt and branch*	Mißachte Interrupt und verzweige
TDB	*Transfer and drop bits*	Übertrage und verliere Bits
DO	*Divide and overflow*	Teile und erzeuge Überlauf
DC	*Divide and conquer*	Teile und herrsche
SRZ	*Subtract and reset to zero*	Subtrahiere und setze auf Null
WIC	*Write invalid character*	Schreibe unzulässiges Zeichen
RAST	*Read and shred tape*	Lese und zerfetze das Band
CMI	*Clobber monitor immeadeately*	Klotze den Bildschirm sofort voll
SRSD	*Seek record and scar disk*	Suche Datensatz und zerschramme die Platte
BST	*Backspace and strech tape*	Gehe rückwärts und strecke das Band
RIRG	*Read inter-record gap*	Lies Aufzeichnungslücke
UDR	*Update and delete record*	Aktualisiere und lösche Datensatz
SSB	*Scramble status byte*	Verwürfle das Statusbyte
EDR	*Execute destructive read*	Führe zerstörenden Lesevorgang durch
EIO	*Execute invalid opcode*	Führe ungültigen Befehlscode aus
EP	*Execute programmer*	Führe Hinrichtung des Programmierers durch
ERM	*Erase reserved memory*	Lösche reservierten Speicher
PBD	*Print and break disk*	Drucke und sprenge die Platte
MLR	*Move and lose record*	Verschiebe und verliere Datensatz
CRN	*Convert to roman numerals*	Wandle in römische Zahlen um
IWP	*Ignore write protect notch*	Mißachte Schreibschutzkerbe
HCF	*Halt and catch fire*	Halte an und fange Feuer
WPO	*Wad up printer output*	Stopfe den Drucker-Auslaß zu
IEOF	*Ignore end-of-file*	Mißachte das Datei-Ende
DWIT	*Do what I'm thinking*	Tu was ich denke
PPL	*Perform perpetual loop*	Führe Endlos-Schleife aus
ZD	*Zap directory*	Lösche alle Verzeichnis-Einträge
DSH	*Destroy sector header*	Zerstöre Sektor-Vorspann
DAP	*Deselect active peripheral*	Gebe aktives Peripherie-Gerät frei
SMD	*Spontaneous memory dump*	Selbsttätiger Speicherausdruck
	(use only when payroll checks are loaded)	(nur anwenden, wenn Gehalts-Schecks eingelegt sind)
ACQT	*Advance clock to quitting time*	Stelle Uhr auf Feierabend

Der zweite Turmbau zu Babel

von Dr. Ralf Kern (1100)

1. Teil

Bei dem folgenden Beitrag handelt es sich um die überarbeitete und erweiterte Fassung eines Artikels, der im Heft 5/86 der Zeitschrift „micro" erschienen ist.

Das Computer zum „Denken" nur Nullen und Einsen verwenden, hat sich auch unter Laien schon herumgesprochen. Umso erstaunlicher ist die Tatsache, daß die Rechner inzwischen Befehle in fast normalem (meist englischem) Klartext „verstehen". Die Vermittlerrolle zwischen Mensch und Computer spielen die Programmiersprachen, von denen es inzwischen eine geradezu babylonische Vielfalt gibt, und die zugehörigen Übersetzer. Von ihnen soll im folgenden Beitrag die Rede sein.

Die ersten Computer wurden tatsächlich so programmiert, wie sie (auch heute noch) „denken", also mit Nullen und Einsen, die ihnen auf Lochstreifen (Loch = 1, kein Loch = 0) oder mittels Schalttafeln eingegeben wurden. Um diese Programmierweise zu illustrieren, sei nur die Befehlsfolge angeführt, mit der ein Mikroprozessor (hier der Z 80) die primitive Aufgabe löst, zwei ganze Zahlen aus dem Speicher zu holen, zu addieren und an einer anderen Speicherstelle wieder abzulegen:

1110 1101 0101 1011 0001 0000 0000 0000

0010 1010 0001 0000 0000 0000

0001 1001

0010 0010 0001 0000 0000 0100

Es besteht kein Zweifel daran, daß ein solches Programmierverfahren äußerst kompliziert, zeitraubend und vor allem fehleranfällig war. Völlig undenkbar, auf diese Weise Software im heute gewohnten Umfang zu erstellen – denken wir nur an so gewaltige Software-Pakete wie Symphony, Framework oder „Frage und Antwort"!

„Mnemonics" statt Bitfolgen

Bald jedoch kam man auf die glänzende Idee, menschenfreundliche Gedächtnisstützen („mnemonics") für die sich wiederholenden Befehle einzuführen, also Wörter oder eher Abkürzungen, die man sich besser merken und lesen kann.

Außerdem wurde die Möglichkeit vorgesehen, Programm- und Speicherstellen mit geeigneten Namen zu benennen, um so die Übersichtlichkeit eines Programmes zu erhöhen. Unser einfaches Beispiel sieht dann z.B. so aus:

```
        LD      DE, (SUMMAND1)
        LD      HL, (SUMMAND2)
        ADD     HL, DE
        LD      (SUMME), HL
        ...

        ORG     1000H
SUMMAND1: DS    2
SUMMAND2: DS    2
SUMME:    DS    2
```

Damit wurde für die Verständlichkeit eines Programms ein Riesenschritt vorwärts getan, denn selbst ein Laie kann in dem Beispiel am Befehlswort „ADD" erkennen, daß dort eine Addition stattfindet. Programmiersprachen, die mit solchen Abkürzungen und Namen arbeiten, heißen Assemblersprachen, und um die damit geschriebenen Programme nicht von Hand in 0-1-Folgen umsetzen zu müssen, hat man Übersetzungsprogramme erstellt, sogenannte Assembler. Sie transformieren für den Menschen verständliche Assemblerprogramme automatisch in Maschinenprogramme, also Binär-Folgen, wie sie der Rechner letztlich verlangt.

Die Bedeutung dieser Erfindung liegt vor allem darin, daß eine Assemblersprache menschlichen Denkgewohnheiten und -strukturen ein wenig mehr als die nackte Maschinensprache entgegenkommt, obgleich sie immer noch sehr von den Eigenheiten des jeweiligen Rechners (Befehls-, Registersatz) bestimmt wird. Dahinter steckt die erste Stufe eines Abstraktionsprozesses, die weg vom Denken in Binärzahlen hin zum Umgang mit Befehlsabkürzungen und frei definierbaren Namen geführt hat. Wir werden sehen, daß dieser Abstraktionsprozeß bei anderen Programmiersprachen noch eine Reihe weiterer Stufen durchschritten hat.

Wesentliche Kennzeichen einer Assemblersprache ist es, daß bei der Übersetzung keine Strukturumwandlung (etwa mit Umstellung von Befehlen) nötig ist. Stattdessen werden nur Befehlsnamen durch die entsprechenden Zahlencodes und andere Namen durch geeignete Speicher- und Programmadressen ersetzt. Somit muß ein Assembler bei der Übersetzung immer nur eine Programmzeile bearbeiten.

FORTRAN bringt den Rechnern „Mathe" bei

Mit dem Übergang zu Assemblersprachen wurde ohne Frage ein großer Fortschritt erreicht, und es konnten auch schon größere Programmierprojekte angepackt werden. Trotzdem ist immer noch viel Übung nötig, um z.B. längere mathematische Berechnungen fehlerfrei in ein Assemblerprogramm zu kleiden. Daher gab es bald einen weiteren Sprung nach vorne, als in den Forschungslaboratorien von IBM unter der Leitung von J. W. Backus 1954 ein „Formelübersetzer" (englisch: FORmulare TRANslator) vorgestellt wurde – die „problemorientierte Programmiersprache" FORTRAN war geboren. Unser kleines Beispiel lautet in FORTRAN so:

```
INTEGER SUMME,
SUMMAND1, SUMMAND2

...

SUMME = SUMMAND1 + SUMMAND 2
```

Es zeigt deutlich, daß man nun eine Formel in der gewohnten mathematischen Schreibweise programmieren kann und daß damit wieder ein großer Schritt weg vom Denken in Rechnerstrukturen hin zu menschlichen Denkweisen gelungen ist. Dadurch wurde in erster Linie mathematisch geübten Wissenschaftlern und Technikern ein weiter Bereich der Computeranwendung erschlossen, der ihnen mit Assemblerprogrammierung unzugänglich geblieben wäre.

Mit FORTRAN gewann man noch einen Vorteil in einer ganz anderen Richtung: Assembler- und Maschinenprogramme können nur auf dem Computertyp laufen, für den sie entwickelt worden sind, denn es gibt für jeden Rechnertyp eine eigene Assemblersprache. Bei jedem Modellwechsel mußten also alle Programme neu geschrieben werden, und die alte Software ebenso wie die darin hineingesteckte Arbeit wurde wertlos – ein Unding unter Kosten- und Zeitgesichtspunkten.

FORTRAN hingegen bot ein ganz neuartiges, „problemorientiertes" Programmiermodell an; nicht mehr von Registern, Speicherstellen, Lade- und Addierbefehlen ist in FORTRAN die Rede, sondern von ganz- und reellzahligen Variablen, mathematischen Operatoren und Funktionen. Die sind aber nicht mehr von einem bestimmten Computertyp abhängig, so daß man FORTRAN-Programme auf jedem beliebigen Rechner zum Ablauf bringen kann – vorausgesetzt, es gibt dafür einen FORTRAN-Übersetzer. Wegen dieser (Über)tragbarkeit nennt man FORTRAN-Programme auch „portabel". Mit den portablen, problemorientierten Programmiersprachen war die zweite Abstraktionsstufe erreicht worden.

Übersetzer für problemorientierte oder, wie man auch sagt, „höhere" Program-

miersprachen werden im Unterschied zu Assemblern „Compiler" genannt. Sie müssen manchmal recht komplizierte Strukturumwandlungen durchführen, bevor sie das Maschinenprogramm erzeugen können. Ein einfaches Beispiel mag das erläutern: Bei der Befehlszeile

A = 20.0*B – 5.5 + C*D

muß der Compiler die mathematische Regel berücksichtigen, daß Multiplikationen Vorrang vor Additionen haben und daß man bei Subtraktionen die Operanden nicht vertauschen darf. Daher muß zur Erzeugung des Maschinenprogramms die Befehlsfolge intern umgestellt werden. Zuerst kommen also die Befehle für die Berechnung der Produkte 20.0*B und C*D, bevor die Subtraktion durchgeführt werden darf. All diese Überlegungen hat bei Assemblerprogrammen der menschliche Programmierer anzustellen, bei problemorientierten Programmsprachen entlastet ihn der Compiler davon.

Im allgemeinen können sich Compiler nicht auf die Übersetzung jeweils einer Programmzeile beschränken, sondern müssen selbständig entscheiden, welchen Bereich des Programmtexts sie jeweils bearbeiten müssen.

Obwohl FORTRAN nun mehr als dreißig Jahre alt ist, wird noch kräftig in FORTRAN programmiert. Dahinter steht einerseits die Marktmacht des Computerriesen IBM, die für diese Kontinuität (und damit auch für das Weiterleben einer überalterten Sprache) gesorgt hat, andererseits ein riesiger Fundus an Programmen im wissenschaftlich-technischen Bereich mit Werten in Milliardenhöhe. An die Umstellung dieser Programme ist nicht zu denken, zumal die Programmierer gar nicht damit nachkommen, für alle neuen Anwendungsmöglichkeiten Programme zu schreiben. Allerdings muß man auch berücksichtigen, daß FORTRAN im Laufe der Zeit mehrere „Häutungen" durchgemacht und sich dabei mehr oder weniger an moderne Erfordernisse angepaßt hat, über FORTRAN II und IV zu FORTRAN 77; an einer Spezifikation der Weiterentwicklung FORTRAN 8x wird derzeit gearbeitet. Trotz aller Modernisierungsbemühungen muß FORTRAN nach heutigen Maßstäben als veraltet gelten; nicht alle zur strukturierten Programmierung notwendigen Sprachmittel werden geboten, und auch die in modernen Programmiersprachen so wichtigen Zeiger (pointer) gibt es in FORTRAN nicht.

ALGOL 60 – die Sprache mit Bildungswert

Den nächsten Meilenstein erreichte die Entwicklung der Programmiersprachen mit ALGOL 60 (ALGOrithmic Language, erste Spezifikation 1958, überarbeitete Version 1960). ALGOL war an einer Reihe von (vor allem europäischen) Universitäten entwickelt worden und enthielt als wesentliche Neuerung eine formale Syntaxdefinition. Das bedeutet, daß die Regeln zum Aufbau eines ALGOL-Programms nicht mehr mit Worten, sondern in Formeln einer programmiersprachenähnlichen Sprache angegeben wurden. Das erleichtert sowohl dem, der ALGOL lernt, als auch dem, der ALGOL-Compiler erstellt, die Arbeit wesentlich. Außerdem gestattet ALGOL im Gegensatz zu FORTRAN Rekursion, d. h. daß Unterprogramme sich selbst (direkt oder indirekt) beliebig oft aufrufen dürfen, sowie dynamische Variablenfelder, deren Größe erst während des Programmablaufs festgelegt wird.

ALGOL, das die Entwicklung vieler anderer Sprachen (z.B. Pascal) entscheidend beeinflußt hat, war an Universitäten recht weit verbreitet, konnte sich aber in der Industrie nie durchsetzen, da IBM in ALGOL einen Konkurrenten zum eigenen Zögling FORTRAN sah und ALGOL daher durch einfaches Ignorieren blokkierte – es gab von IBM nie einen ALGOL-Compiler. An den Universitäten ist ALGOL inzwischen durch Pascal völlig verdrängt worden und besitzt, wie ein Kenner einmal zutreffend formulierte, ähnlich wie Latein nur noch „Bildungswert".

COBOL – auch nicht totzukriegen

Ein anderer „Dinosaurier" unter den Programmiersprachen, der bis heute überlebt hat, ist COBOL (COmmon Business-Oriented Language). Diese Sprache wurde 1959 für kommerzielle Anwendungen konzipiert und in einer späteren Version streng und allgemeinverbindlich genormt. COBOL bietet vor allem hierarchisch aufgebaute Datenstrukturen („records") und lehnt sich in der Formulierung der Anweisungen eng an die englische Sprache an. Das erste der FORTRAN-Beispiele lautet in COBOL formuliert:

ADD SUMMAND1 TO
SUMMAND2 GIVING SUMME.

Dieser Versuch, sich der menschlichen Denkweise und der natürlichen Sprache zu nähern, gilt jedoch als mißlungen, da er nur zu weitschweifigen, umständlichen Programmen führt, ohne wirklich die Flexibilität der natürlichen Sprache nachbilden zu können. Außerdem sind COBOL-Programme und -Compiler für ihre Ineffizienz berüchtigt, d. h. für die Länge des erzeugten Codes sowie der Übersetzungs- und Laufzeiten. Dennoch ist COBOL nicht auszurotten, da ähnlich wie im Fall von FORTRAN Milliardenwerte in COBOL-Programmen stecken, die nicht von heute auf morgen überflüssig gemacht werden können.

Auf der Suche nach der „eierlegenden Wollmilchsau"

Gegen Ende der sechziger Jahre machte man verschiedene Versuche, die „eierlegende Wollmilchsau" der Programmiersprachen zu schaffen, eine Sprache also, die gleichermaßen für alle Problembereiche (kommerzielle, technische, wissenschaftliche Aufgaben, Systemprogrammierung u. a.) geeignet sein sollte. Bekannteste Ergebnisse sind ALGOL 68 und PL/I (Programming Language 1). Während sich ALGOL 68 – wieder vom universitären Bereich her kommend – nicht durchsetzen konnte, hat PL/I, von IBM initiiert und mit der zugehörigen Marktmacht im Rücken, größere Bedeutung erlangen können. (Kleine Bemerkung am Rande: die von IBM so bewußt ignorierte Sprache ALGOL hat sichtlich zur Entwicklung von PL/I mehr beigetragen als FORTRAN!)

Gerade an der ausdrücklich angestrebten Universalität von PL/I hat sich die Kritik entzündet: Es gibt nur wenige Programmierer, die den vollen Sprachumfang beherrschen und praktisch einsetzen können; die PL/I-Compiler und auch die übersetzten PL/I-Programme sind ineffizient. (Diese Situation hat sich aber durch Einführung einer Teilmenge des Sprachumfangs gebessert, die als „ANSI Subset 6" standardisiert wurde.) Niklaus Wirth von der ETH Zürich, der Schöpfer von Pascal, hat es so ausgedrückt: „Die Idee des Schweizer Offiziersmessers hat ihre Verdienste, aber wenn man es übertreibt, wird das Messer zum Mühlstein."

Pascal – elegant und einfach

Wirth wählte daher den Weg „zurück zur Natur": Über die Vorläufer PL360, ALGOL W und Euler gelangte er zum Entwurf der Sprache Pascal, die er möglichst sparsam mit Sprachmitteln und Eigenschaften ausstattete – gerade so viel, wie für universelle Einsetzbarkeit benötigt wurde. Damit erreichte er eine hohe Effizienz der Pascal-Compiler. Zwei Merkmale von Pascal stechen besonders hervor: Zum einen die Vielfalt von Datentypen, verbunden mit der Möglichkeit für den Programmierer, eigene, maßgeschneiderte Datentypen zu definieren; zum anderen die Sprachmittel zur Ablaufsteuerung wie z.B. „while – do" oder „repeat – until", die den Weg zur strukturierten Programmierung eröffnen.

Hinter dem Schlagwort „Strukturierte Programmierung" verbirgt sich ein Sammelsurium von Methoden, die zum Ziel haben, die Möglichkeiten moderner Programmiersprachen in einer Weise zu nutzen, die die Erstellung von Software erleichtert, schneller und sicherer macht. Dazu hat man psychologische Erkenntnisse über den geistigen Vorgang des Programmierens berücksichtigt und versucht, diesen Vorgang wieder einen Schritt weiter in Einklang mit menschlichen Denkgewohnheiten zu bringen. Im einzelnen lassen sich u. a. diese methodischen Merkmale der strukturierten Programmierung anführen:

- Modularisierung: Man zerlegt ein großes Problem in einzelne, leichter zu lösende Teilprobleme, die dann auch von verschiedenen Mitgliedern einer Arbeitsgruppe bearbeitet werden können. Manchmal lassen sich dann Lösungsteile in anderen Programmen weiterverwenden.

- Schrittweise Verfeinerung (Top-Down-Methode): Man skizziert zunächst die Problemlösung in groben Schritten und detailliert die einzelnen Schritte durch Aufbereitung immer feinerer Einzelheiten so lange, bis das arbeitsfähige Programm fertiggestellt ist.

- Vermeidung von Sprungbefehlen („GOTO"): Sprungbefehle behindern die Übersicht besonders in größeren Programmen und erschweren Programmänderungen. Es läßt sich mit mathematischer Strenge beweisen, daß man jeden Sprungbefehl durch Konstruktionen mit den Sprachmitteln IF – THEN – ELSE und WHILE – D0 ersetzen kann. Die Problematik von Sprungbefehlen, die der Holländer E. W. Dijkstra als erster klar erkannt hat, wird weiter unten mit einem BASIC-Beispiel deutlich demonstriert.

- Benutzung von Nassi-Shneiderman-Diagrammen anstelle von Flußdiagrammen, denn erstere erleichtern die Umsetzung eines graphisch veranschaulichten Lösungswegs in ein Programm ohne Sprungbefehle.

Der Nutzen von Datentypen, die der Programmierer selbst definiert, soll nun an einem Beispiel verdeutlicht werden: Stellen wir uns vor, daß in einem Programm die Verteilung der Umsätze eines Restaurants auf die sieben Wochentage analysiert werden soll, wobei die Werktage besonders zu berücksichtigen sind. In Sprachen wie ALGOL oder FORTRAN müssen die Wochentage geeignet codiert werden, etwa Montag durch 1, Dienstag durch 2 usw. bis Sonntag durch 7. In Pascal definiert man die problemorientierten Datentypen „Woche" und „Werkwoche" und deklariert „Wochentag" und „Werktag" als zugehörige Variable. Der folgende Programmausschnitt illustriert dies:

```
TYPE

WOCHE  =  (MONTAG, DIENSTAG,
          MITTWOCH, DONNERSTAG,
          FREITAG, SAMSTAG,
          SONNTAG);

WERKWOCHE = MONTAG . . SAMSTAG;

...

VAR

WOCHENTAG: WOCHE;
WERKTAG:   WERKWOCHE;

...
```

Im späteren Verlauf des Programms sind dann Konstruktionen der Art

```
WOCHENTAG := DIENSTAG;

....

FOR WOCHENTAG := MONTAG TO
                 SONNTAG DO
BERECHNE (UMSATZ (WOCHENTAG));
```

zulässig, während der Compiler eine unlogische Zuweisung wie

```
WERKTAG := SONNTAG;
```

schon bei der Übersetzung als widersinnig entlarvt und abweist, so daß im übersetzten Programm ein solcher Fehler schon gar nicht mehr auftreten kann.

Man erkennt an dem Beispiel auch deutlich, daß durch die Freiheit bei der Definition von eigenen Datentypen wieder ein Abstraktionsschritt in Richtung menschlicher Denkstrukturen gelungen ist.

Im Mikrocomputer-Bereich breitete sich Pascal zunächst in der UCSD-Version (Univesity of California, San Diego) von Ken Bowles aus, und zwar vor allem auf Apple-Computern. Auf CP/M-Systemen setzte sich vorwiegend Pascal MT+ durch, und seit ein paar Jahren gibt es den ganz großen Renner, der in bezug auf Effizienz, Komfort und Preis kaum zu schlagen ist: Turbo-Pascal. Man schätzt, daß vom Turbo-Pascal-Compiler über 250000 Kopien verkauft worden sind, mehr als von allen anderen Pascal-Compilern zusammen – von den Schwarzkopien gar nicht zu reden. Damit ist Turbo-Pascal zu einem De-factro-Standard geworden, doch das ist nicht unbedingt Anlaß zur Freude. Denn Turbo-Pascal erweitert zum Teil recht eigenwillig den Pascal-Standard und stützt sich zudem kräftig auf die Hardware-Architektur IBM-kompatibler PCs ab – mit der Folge, daß die Portierbarkeit auf andere Rechner, ein wichtiger Vorteil von höheren Programmiersprachen, bei Turbo-Pascal-Programmen verlorengeht.

Mit dem Aufbruch in die Mikrocomputer-Welt fand auch ein Neubeginn im Bereich der Programmiersprachen statt. Mit Mikrocomputern befaßte sich nämlich ein ganz anderer Anwenderkreis als etwa mit Mini- oder Großrechnern, und der legte weniger Wert auf Kompatibilität mit FORTRAN, dafür umso mehr auf leichte Erlernbarkeit. Wichtige Vorbedingung war außerdem der geringe Speicherbedarf von Compilern und Programmen. Dies führte zum Aufstieg der Sprachen Pascal und BASIC, und deswegen erlebte auch die Assemblerprogrammierung eine Renaissance.

Muttersprache der Mikrocomputer: BASIC

BASIC (Beginners All-purpose Sybolic Instruction Code) war 1963/64 von Kurtz und Kemeny konzipiert worden, um auch Studenten, die eigentlich nicht mit EDV zu tun haben, einen Einblick in die Computerwelt zu ermöglichen. Wichtige Ziele waren demgemäß geringer Sprachumfang, leichte Erlernbarkeit und Dialogorientierung. Mit letzterer war BASIC seiner Zeit um ein Jahrzehnt voraus, da Dialoge damals nur umständlich über Fernschreiber geführt werden konnten und das Zeitalter der interaktiven Programme so richtig erst mit der Einführung von Bildschirmen als Ein/Ausgabegeräten anbrach. Bedingt durch die Dialogorientierung liegt die Stärke von BASIC im Umgang mit Texten, also mit Zeichenketten, und bei den interaktiven Ein/Ausgabeanweisungen.

Als Nachteil von BASIC muß man ansehen, daß strukturierte Programmierung nicht möglich ist; insbesondere lassen sich einem Programm abgeschlossene, fremde Unterprogramme nicht problemlos hinzufügen. Auch die für BASIC charakteristische Zeilennumerierung erweist sich als ausgesprochen lästig bei der Wertung, Weiterentwicklung und Korrektur von BASIC-Programmen. Hierzu wieder ein einfaches Beispiel:

```
500 GOTO 1020

...

720 GOTO 1020

...

1000 LET X = Y + Z

1010 GOTO 1200

1020 PRINT „Hierher!"

...

1530 GOTO 1020

...
```

Will man in diesem Programm vor der Zeile 1020 eine zusätzliche Befehlszeile einschieben, etwa mit der Zeilennummer 1015, so müssen die Sprungbefehle zur Zeile 1020, die sich im Beispiel auf den Zeilen 500, 720 und 1530 befinden, unbedingt zu „GOTO 1015" geändert werden. Dadurch entsteht bei größeren, unübersichtlichen Programmen eine „todsichere" Fehlerquelle, die bei strukturierter Programmierweise nicht existieren würde.

Zusammen mit BASIC (vor allem in den vorherrschenden Microsoft-Dialekten) wurde eine vorher wenig verbreitete Form von Übersetzern für höhere Programmiersprachen populär, nämlich die der Interpreter. Im Gegensatz zu den Com-

pilern, die aus einem Hochsprachen-Programmtext ein äquivalentes Maschinenprogramm erzeugen, arbeiten Interpreter den Programmtext Zeile für Zeile ab und führen dabei das Programm sofort aus. Während also ein Compiler aus der Programmzeile

SUMME = SUMMAND1 + SUMMAND2

die ganz zu Anfang dargestellte Binärzahlenfolge erzeugt, verfährt ein Interpreter ganz anders: Er liest die Zeile und ruft bei Auffinden des Gleichheitszeichens das für Wertzuweisungen zuständige Unterprogramm auf. Dieses wiederum ruft ein Unterprogramm für Variablen auf, das die Speicheradresse von SUMME ermittelt und (später, nach Durchführung der Addition) den errechneten Wert dorthin speichert. Dann wird die rechte Seite analysiert und beim Antreffen des Pluszeichens eine Additionsroutine aufgerufen, die ihrerseits zur Beschaffung der Werte der Variablen SUMMAND1 und SUMMAND2 ein (anderes) Variablen-Unterprogramm aufruft und anschließend diese Werte addiert. Die Summe wird schließlich, wie schon erwähnt, der Variablen SUMME auf der linken Seite zugewiesen.

Wir sehen also: Ein Interpreter erzeugt nach der Analyse des Programmtextes keine Maschinenbefehle, sondern führt die bei der Analyse erkannten Anweisungen sofort aus. Er beansprucht weniger Speicherplatz als ein Compiler (ganz wichtig für den Einsatz in den ersten Mikrocomputern) und bietet bei Programm-Änderungen wesentlich mehr Komfort, da das Programm nach der Änderung sofort wieder ablauffähig ist, während bei der Arbeit mit einem Compiler nach einer Änderung erst wieder der gesamte Übersetzungsvorgang durchlaufen werden muß.

Andererseits dauert die Programmausführung mittels eines Interpreters wesentlich länger als bei einem compilierten Programm. Der Zeitvorteil liegt bei BASIC gewöhnlich bei einem Faktor von 2 bis 3, kann aber in Extremfällen bis zu 10 erreichen!

Die Fortsetzung mit der Programmiersprache C folgt im nächsten Heft

PRINT#–Bug

Der PRINT#–Bug des HP 71B

Beim Anlegen eines Datensatzes für meine eigene Buchhaltung auf einer Diskette stieß ich auf ein Phänomen, das mir viel Ärger machte.

Bekanntlich erlaubt ja das Betriebssystem des „71B" das Abspeichern von numerischen und Stringdaten im selben Record.

Ich wollte also in einem Datenfile **auf der Diskette** Records mit etwa dieser Abfolge speichern:

H,A$,B$,C$,D$,E,F,G

Anm.: Diese Reihenfolge ergab sich durch die Verwertung vorhandener Formulare.

Das Abspeichern der Daten mit „PRINT#" auf die Diskette verlief ohne jedes Problem, d.h. ohne irgendeine Fehlermeldung. Erst beim Lesen der Daten mit „READ#" erschien unerwartet, nachdem etwa 50 Records gelesen waren, die Fehlermeldung Nr. 29: „Record Ovfl". Ich erspare es mir zu beschreiben, welche Wege ich beschritt, welche Maßnahmen ich versuchte – auch andere Clubfreunde wußten am Ende keinen Rat mehr – um dem Übel abzuhelfen.

Um es vorwegzunehmen: die genannte Zahl von etwa 50 „gesunden" Records vor dem ersten fehlerhaften Record ist völlig irrelevant.

Umfangreiche Untersuchungen führten zu den unterschiedlichsten Ergebnissen: Von 2 „gesunden" Records vor dem Fehler bis zu über 150 vor dem Fehler. Die Größe der kreierten Records selbst bestimmt zwar die Anzahl der „gesunden" Records mit.

Ihr Einfluß geht jedoch in beide Richtungen:
So verursachte die größte Dimensionierung der Records die wenigsten „gesunden". Damit wird natürlich jeder so kreierte Datenfile unsicher und unbrauchbar. Ein Kopieren der so infizierten Datenfiles auf dem Monitor zeigte dann auch, daß schon beim Abspeichern eine „End of Record"-Marke in die verseuchten Records eingedrungen war, die da garnicht hingehörte!

Und zwar ohne eine Fehlermeldung!

Damit wurde auch klar, daß ein solcher Record von einem formatierten „READ#"-Befehl nicht mehr gelesen werden konnte.

Ich wandte mich deshalb an das „HP-Support-Zentrum Ratingen", das seit einiger Zeit „HP-Bad-Homburg" in der Betreuung der Taschenrechner-Benutzer abgelöst hat. Nach einigen Telefonaten und Briefen, in dem ich das nicht ganz alltägliche und nicht ganz einfache Problem darstellte, konnte mir das dortige Team helfen:

COPY der HP-Antwort an mich:

Der PRINT#–Bug

1. Auftreten des Fehlers:
   Schreibt die PRINT#–Anweisung einen String-Eintrag, gefolgt von einem anderen Eintrag, und endet der String-Eintrag auf der externen Einheit ein Byte vor dem Ende eines phys. Records, dann wird das erste Byte des folgenden Eintrags ersetzt durch die End-of-Record-Marke (Hex EF). Auf alle nachfolgenden Daten kann dann nicht mehr zugegriffen werden (Verhinderung durch Record-Overflow-Meldung).

   Es ist also eine Abhängigkeit zwischen der logischen Record-Länge (angegeben beim Kreieren des DATA-Files) und der physikalischen Record-Länge der externen Einheit (in der Regel 256 Bytes) vorhanden.

2. Vermeiden des Fehlers:
   Entsprechend den og. Hinweisen ergeben sich folgende Lösungsvorschläge:
   – Aufteilen der PRINT#–Anweisung in einige Teilanweisungen, so daß kein Eintrag einem String-Eintrag in derselben Anweisung folgen kann.

   z.B. Aufteilung der Anweisung

   40 PRINT#1,X;H,A$,B$,C$,D$,E,F,G

   geschieht wie folgt:

   40 PRINT#1.X;H,A$
   41 PRINT#1;B$ (wobei der Pointer weiter auf Record „X" bleibt)
   42 PRINT#1;C$
   43 PRINT#1;D$
   44 PRINT#1;E,F,G

   Einschränkungen:
   – ein Test auf einen Record-Overflow kann nicht mehr erfolgen.
   – die Recordgröße muß so dimensioniert sein, daß Platz für eine End-of-Record-Marke vorhanden ist (d.h. mindestens 1 Byte mehr als die minimal mit Hilfe der Anforderungen berechneten logischen Record-Länge).
   – Diese Abhängigkeit zwischen log. und phys. Record-Länge kann ausgenutzt werden dadurch, daß man als logische Record-Länge einen Teiler der phys. Record-Länge wählt (d.h. einen Teiler von 256 : wie 4, 8, . . . 128 oder 256 selbst).

   Anmerkung:
   Das damalige Problem benötigte 71 Bytes, d.h. als nächstgrößeren Teiler müßte man 128 Bytes als logische Record-Länge wählen. Dabei wird der Nachteil offenkundig:
   Mman verschenkt 57 Bytes pro Record!

   Weitere Nachteile:
   – keine Verarbeitung einer log. Record-Länge von > 256 Bytes
   – Verschenken von Speicherplatz (siehe Beispiel).

3. Die Behebung des Fehlers geschieht durch eine ROM-Version 2CDCC.

Soweit die HP-Darstellung, die darüberhinaus noch einige Vorschläge enthielt, die aber hier nicht relevant sind.

Welchen der vorgeschlagenen Wege er benutzen will, muß der Anwender wohl selbst herausfinden. Die Aufteilung in Einzelteile dürfte nur geringfügig mehr Programmier-Aufwand bringen.

Über die unter 3. angekündigte neue ROM-Version gibt es noch keine konkreten Informationen. Wir müssen also – insbesondere wir Benutzer der alten Ausführung – noch eine Weile mit dem Behelf leben.

Ich weiß nichts darüber, wie weit der beschriebene „PRINT#–Bug" in unserem Club bekannt ist. Es war meine Absicht, andere vor Schaden oder zumindest „Zeitverlust" zu bewahren.

Vielleicht hat auch der eine oder andere Kenner oder Könner einen besseren Vorschlag zu machen.

Ich freue mich über jede Zuschrift!

Bitte an:

Georg K. Heise
Lindelbrunnweg 10
6728 Germersheim

HP-28 ROM für HP-71B

Gerade hat sich der HP-28C auf dem Pocketcomputer-Markt eine hervorstechende Stellung erobert, da ist auch schon ein Modul mit gleichen Möglichkeiten für den HP-71B erhältlich, was Kenner der Materie sicherlich nicht überraschen wird, da beide Systeme über den gleichen Prozessor verfügen.

Wie der HP-28C verfügt jetzt auch der HP-71B mit dem neuen Advanced-Algebraic-Applications-ROM über einen Algebra-Modus, mit dem man Funktionen entwickeln und vereinfachen, sowie auch einzelne Variablen isolieren kann. Auf diesen Modus kann man natürlich auch über BASIC zugreifen, was durch BASIC-Instruktionen unterstützt wird. Im großen und ganzen enthält das Modul alle 28er-Funktionen, bis auf den Funktionsplot, und zusätzlich einen Befehlssatz von 73 BASIC-Instruktionen. Diese Befehle unterstützen hauptsächlich die Mathematik, die strukturierte BASIC-Programmierung und Utilities.

Das Modul enthält einen 64 kByte-Lexfile, der allerdings nur in Verbindung mit dem Mathe-ROM benutzt werden kann, woraus man schon die Leistungsfähigkeit des Moduls erahnt. Das Advanced-Algebraic-Applications-Modul (kurz „Triple-A-ROM" genannt) ist bereits in USA mit der Bestellnummer 82491A für 275 US-Dollar zu haben und soll Ende April auch in Deutschland erhältlich sein. Der Preis hierzulande ist bis jetzt allerdings nicht bekannt. Man darf das neue Modul also mit großer Spannung erwarten, denn es erweitert die Leistungsfähigkeit des HP-71B erheblich.

Anton Prilsch (1488)

. . . und noch ein Lex-File: Combarr

227 Bytes, HP-71

Permutationen und Variationen werden in manchen Bereichen der Mathematik und Statistik verwendet. Der vorliegende LEX-File beschäftigt sich mit diesen und liefert bequem zu bedienende Funktionen, die das Errechnen von Variations- und Permutationsmöglichkeiten zum Kinderspiel machen.

Das Programm stammt von den SIG-Mitgliedern Laurent Istria und Guy Toublanc und wurde unter anderem bereits im SIG-Journal „JPC" Nr. 25 und 41 veröffentlicht, sowie im CHHU-Chronicle V2N7p46. Die Größe des Files ist 227 Bytes. Es gibt die beiden folgenden Keywords:

Filename: COMBARR
Filesize: 227 Bytes
ID# Hex: E1

Word	Token	Char
ARR	14	F
COMB	15	F

Die Funktionen lauten ARR(*n*,*p*) und COMB (*n*,*p*). Dabei sind *n* und *p* beliebige positive numerische Argumente. Beide Keywords sind Funktionen, die ein numerisches (positiv und ganzzahliges) Ergebnis liefern in Abhängigkeit der beiden Argumente *n* und *p*.

ARR liefert die Anzahl der möglichen Variationen aus *n* Elementen zur *p*-ten Klasse, ohne Wiederholung. Hat man also z.B. 3 verschiedene Schachfiguren, wie etwa Turm, Läufer und Springer und will diese auf 4 Felder verteilen, so gibt es genau ARR(4,3) =24 verschiedene Möglichkeiten, dies zu tun, wie man leicht selbst ausprobieren kann. Die Funktion ARR(*n*,*p*) ist mathematisch gleichwertig mit dem Ausdruck FACT(*n*)/FACT(*n*-*p*).

COMB liefert die Anzahl der möglichen Permutationen von zwei Mengen von Elementen deren eine p und die andere n-p Elemente enthält. Innerhalb einer Menge sind alle Elemente einander gleich. Hier kann man als Beispiel nehmen: 3 Bauern, die auf 4 Felder des Schachbretts verteilt werden sollen. Da wir die Bauern voneinander nicht unterscheiden gibt es hier nur COMB(4,3)=4 Möglichkeiten dies zu tun. Die Funktion COMB(*n*,*p*) ist ersetzbar durch den Ausdruck FACT(*n*)/(FACT(*p*)*FACT(*n*-*p*)).

Hexdump-Listing: COMBARR

(Zum Abtippen benötigen Sie ein Hex-Ladeprogramm, wie z.B. MAKEFILE aus Prisma 7/86!)

COMBARR L ID#E1 227 Bytes

	0123	4567	89AB	CDEF	ck
000:	34F4	D424	1425	2502	8A
001:	802E	0033	5132	0178	76
002:	5A10	01E4	1510	0000	A9
003:	F020	0000	0000	0000	9C
004:	0A20	00F9	00B2	000F	C8
005:	5142	5254	1734	F4D4	65
006:	2451	1FF8	8228	4059	E1
007:	0882	2850	7B11	8F00	B3
008:	4D09	7982	17F7	8018	FD
009:	F9D3	D08F	C14D	0228	C7
00A:	FA74	D054	1860	F0AF	9C
00B:	2218	0CE6	8D02	68FA	1E
00C:	74D0	4606	6F08	6062	6D
00C:	A4E8	F363	C08F	C14D	46
00E:	0218	FA74	D059	08FE	2D
00F:	04D0	1321	BBB8	F214	F6
010:	0AF0	AF12	EB05	8FC2	5A
011:	9E08	F189	E08F	5F3D	C6
012:	08FA	34C0	8FC1	4D08	8F
013:	6090	8FCA	4C08	FC29	72
014:	E08F	E04D	08F7	23C0	21
015:	9792	28FE	04D0	8F7E	8F
016:	3D08	F723	C08F	7E3D	82
017:	061A	F8F1	89E0	1BBB	D3
018:	8F21	4613	48F4	99C0	BB
019:	8D83	2F08	FDF8	E094	5A
01A:	CD13	2110	9B63	18FF	73
01B:	B6C0	8F4F	6C04	008D	2B
01C:	91FB	0			65

Michael Fiedler
Friedrichstr. 17
6070 Langen

Neu: 64 KByte RAM Modul

Corvallis Micro Technology Inc. - kurz CMT - hat wieder ein neues RAM-Modul für den ganz normalen HP-71 Port auf den Markt gebracht.

Trotz gleicher Größe mit dem bisher bekannten 32 KByte RAM-Modul bietet es den doppelten Speicherplatz und ist zusätzlich (!) noch von einer Lithiumbatterie gepuffert. Dadurch bleibt der Speicherinhalt des Moduls auch dann erhalten, wenn es aus dem HP-71 herausgenommen wird. Soviel Komfort bei dieser Kompaktheit hat aber auch seinen Preis, der im Fachhandel bei 699,- DM liegt.

Alf-Norman Tietze
(Redaktion)

GRAPADAT

Liebe Clubfreunde,

im Prisma 5/87 ist eine elegante Adressenverwaltung von CHRISTIAN ROTHE ausführlich dargestellt.

Sie ist in der dortigen Programmierung jedoch nur für Grabau GR7 geeignet. Ich habe mich deshalb mit dem Autor ins Benehmen gesetzt und das Programm so variiert, daß es sowohl mit dem GR7 als auch mit dem „ADVANCED PAC-SCREEN", läuft.
(Natürlich auch ohne eines der beiden Interfaces – also auch ohne Monitor)

Die Bedienungsanleitung ist die gleiche, wie im Prisma 5/87 beschrieben. Allerdings mit folgender Änderung:

Das Programm wird mi Hauptmenue nicht durch die Eingabe von „Ende" sondern durch Eintippen von „X" (groß oder klein egal) verlassen! Im Hauptmenue wird hier auch ein „X" gefordert.

Die Änderung war erforderlich, weil ich anstelle von „INPUT" oder „LINPUT" mit dem in solchen Fällen komfortableren „ON POS (,) GOTO" gearbeitet habe, wodurch z.B. auch das „ENDLINE" entfällt.

Zusätzlich habe ich eine Drucker-Überprüfung eingesetzt, die z.B. auch fehlendes Druckpapier meldet.

Die Textzeilen sind fast ausschließlich dem Originalprogramm entnommen. Sie erscheinen auf dem Monitor vollständig. Beim Listen im Drucker allerdings fallen die Umlaute und das „ß" aus. Viel Spaß allen Anwendern und herzliche Grüße allen Clubfreunden.

```
1 !                      Die einfache Adressenverwaltung
2 !                           auf  dem   HP 71B
3 !                         von  CHRISTIAN  ROTHE
4 !                           18.02.1986
5 !              variiert fr GRABAU GR 7 und PAC-SCREEN (adv)
6 !                           von GEORG K. HEISE
7 !                             20.11.1987
8 !                           Name : GRAPADAT
9 !
10 ON ERROR GOTO 11 @ COPY KEYS TO 'RKEY'
11 ON ERROR GOTO 12 @ PURGE KEYS @ OFF ERROR
12 DEF KEY "#43",""; @ POKE "2F441","F" @ RESET HPIL @ STANDBY 6 @ DESTROY ALL @
 CFLAG ALL
13 OFF ERROR
14 IF MEM<6000 THEN BEEP 1400,.1 @ DISP "Zu wenig Speicher frei" @ GOTO 459
15 INTEGER I,K,L,P,Q,U,V,W,D0,D1,D2,I1,I2,I3
16 USER ON @ OPTION BASE 1 @ INTEGER T(999) @ CFLAG -27
17 RESTORE IO @ D1=DEVADDR("%48") @ D2=DEVADDR("%64")
18 E1$=CHR$(27)
19 IF D1<1 THEN D1=0
20 IF D2<1 THEN D2=0
21 D0=MAX(D1,D2) @ IF D0=0 THEN DELAY .5,.5 ELSE DELAY 0,0
22 DIM F$(8)[20],E$(8)[60],S$(8)[60],B$[200],L$(2)[480],T$[60]
23 IF D0 THEN GOSUB 'G1' @ GOSUB 381 ELSE GOTO 25
24 GOTO 26
25 DISP "Die einfache Adressenverwaltung" @ WAIT 1
26 DISP "'N'eue Datei anlegen  oder  'W'eiterarbeiten an vorhandener Datei ? N/W
 :" @ DISP
27 ON POS("NW",UPRC$(KEY$))+1 GOTO 27,28,29
28 GOTO 'NEU'
29 A$="S1" @ ON ERROR GOTO 'ERR'
30 'S1': ASSIGN #3 TO * @ LINPUT "Alter Datenfile ? ";Z1$
31 IF D0 THEN GOSUB 381
32 DISP "Lesen der Indexdaten " @ ASSIGN #3 TO Z1$&"I:%16"
33 A$="S2"
34 'S2': READ #3,0;I1,I2,I3
35 FOR I=1 TO I3 @ READ #3;T(I) @ NEXT I
36 ASSIGN #3 TO *
37 ASSIGN #3 TO Z1$&":%16"
38 READ #3,0;I1,I2,I3
39 READ #3;F$()
40 'MENUE': CFLAG ALL @ IF D0 THEN GOSUB 391
41 ON ERROR GOTO 41 @ DISP "Bitte whlen Sie !" @ A$=UPRC$(KEY$) @ IF A$="" THEN
GOTO 41
42 SFLAG -27
43 ON POS("EAZDSLTBPX",A$)+1 GOTO 41,56,196,147,259,195,120,110,299,359,45
44 'ENDE':
45 'X': IF D0 THEN GOSUB 'G3' @ GOSUB 413
46 RESTORE #3 @ PRINT #3,0;I1,I2,I3,F$()
47 ASSIGN #3 TO * @ DISP "Speichern d.Indexdaten"
48 ON ERROR GOTO 'ERROR' @ A$="ENDE1"
49 'ENDE1': ASSIGN #3 TO Z1$&"i:%16"
50 PRINT #3;I1,I2,I3
51 FOR I=1 TO I3 @ PRINT #3;T(I) @ NEXT I
52 ASSIGN #3 TO * @ POKE "2F441","0" @ DEF KEY "#43"
53 PRINTER IS '%35' @ PRINT E1$&"&k0S" @ PWIDTH 80 @ IF D0 THEN GOSUB 'G2'
54 USER OFF @ CFLAG -27 @ PUT "#43" @ MERGE 'RKEY' @ PURGE 'RKEY'
55 DISP E1$&"E" @ DISP "Ende" @ END
56 'E': IF D0 THEN GOSUB 'G1' @ PRINT TAB(29);
57 DISP "Eingabeteil"
```

```
58 IF D0 THEN PRINT
59 ON ERROR GOTO 'ERROR'
60 IF I1=I3 THEN BEEP 1400,.1 ELSE GOTO 62
61 DISP "Alle Datenstze belegt" @ WAIT 1.5 @ GOTO 40
62 FOR I=1 TO 8
63 DISP "Eingabe "&F$(I);
64 LINPUT E$(I) @ IF E$(I)#"" THEN E$(I)=E$(I)&" "
65 E$(I)[1,1]=UPRC$(E$(I)[1,1])
66 IF E$(I)="M " THEN 40
67 IF E$(I)="? " THEN 63
68 IF I=3 AND E$(3)#"" THEN E$(3)=" "&E$(3)[1,LEN(E$(3))-1]
69 NEXT I
70 IF D0 THEN PRINT E1$&"B";TAB(22);
71 DISP "Datensatz wird gespeichert!"
72 L$(1)=UPRC$(E$(1)&E$(2)&E$(3)&E$(4)&E$(5)&E$(6)&E$(7)&E$(8))
73 ON ERROR GOTO 'ERROR' @ A$="E8"
74 IF I1=0 THEN 91
75 'E8': RESTORE #3,T(I1)/10 @ READ #3,T(I1)/10;S$()
76 L$(2)=UPRC$(S$(1)&S$(2)&S$(3)&S$(4)&S$(5)&S$(6)&S$(7)&S$(8))
77 IF L$(1)>=L$(2) THEN P=I1+1 @ GOTO 109
78 U=0 @ O=I1 @ A$="E2"
79 FOR Z=1 TO 8
80 IF O-U<=10 THEN 88
81 L=IP((U+O)/2)
82 'E2': READ #3,T(L)/10;S$()
83 L$(2)=UPRC$(S$(1)&S$(2)&S$(3)&S$(4)&S$(5)&S$(6)&S$(7)&S$(8))
84 IF L$(1)<L$(2) THEN O=L
85 IF L$(1)>L$(2) THEN U=L
86 IF L$(1)=L$(2) THEN BEEP @ DISP "Datensatz schon vorhanden !" @ WAIT .5 @ GOT
O 40
87 NEXT Z
88 A$="E3"
89 V=U+20 @ IF V>I1 THEN V=I1
90 IF U=0 THEN U=1
91 IF I1=0 THEN P=1 @ GOTO 103
92 FOR I=U TO V
93 'E3': READ #3,T(I)/10;S$()
94 L$(2)=UPRC$(S$(1)&S$(2)&S$(3)&S$(4)&S$(5)&S$(6)&S$(7)&S$(8))
95 IF L$(1)<=L$(2) THEN 97
96 NEXT I @ I=I-1
97 IF L$(1)=L$(2) THEN BEEP @ DISP "Datensatz schon vorhanden !" @ WAIT .5 @ GOT
O 40
98 P=I @ IF P<1 THEN P=1
99 A$="E4"
100 FOR I=I1 TO P STEP -1
101 Z=FP(T(I+1)/10)*10 @ T(I+1)=IP(T(I)/10)*10+Z
102 NEXT I
103 A$="E6" @ FOR I=1 TO I1+1 @ IF FP(T(I)/10)=0 THEN T(I)=T(I)+1 ELSE 105
104 T(P)=FP(T(P)/10)*10+I*10 @ GOTO 106
105 NEXT I
106 'E6': RESTORE #3,I @ PRINT #3,I;E$()
107 A$="E7" @ I1=I1+1 @ I2=I2-1
108 GOTO 40
109 IF L$(1)=L$(2) THEN BEEP @ DISP "Datensatz schon vorhanden !" @ GOTO 40 ELSE
 103
110 'T': IF D0 THEN GOSUB 'G1'
111 GOSUB 'p9' @ IF NOT D0 THEN WAIT 5
112 DISP "Druck einer Daten-Teilliste"
113 IF D0 THEN PRINT
114 INPUT "Startnummer :";U
```

```
115 INPUT "Endnummer :";0
116 IF U>0 THEN BEEP 1400,.1 @ DISP "Falsche Eingabe !!!" @ GOTO 'T'
117 IF U<1 THEN U=1
118 IF 0>I1 THEN 0=I1
119 GOTO 125
120 'L':
121 U=1 @ 0=I1 @ IF D0 THEN GOSUB 'G1'
122 GOSUB 'P9' @ IF NOT D0 THEN WAIT 5
123 DISP "Druck einer Daten-Gesamtliste"
124 IF D0 THEN PRINT USING "Z/"
125 DISP "Ausdruck mit laufender Nummer J/N ?"
126 ON POS("JN",UPRC$(KEY$))+1 GOTO 126,127,128
127 SFLAG 2
128 IF D0 THEN PRINT USING "#,3/,Z3A";""
129 DISP "Druck der Datenliste !" @ PRINTER IS :%35 @ PWIDTH INF
130 PRINT CHR$(27)&"&k2S"&CHR$(27)&"&s1C"
131 B$="-----------------------------------"
132 B$=B$&B$&B$&B$
133 C$=CHR$(124) @ STD
134 PRINT TAB(10);F$(1);TAB(28);C$;TAB(36);F$(2);
135 PRINT TAB(52);C$;TAB(62);"Straße";TAB(86);C$;
136 PRINT TAB(97);F$(5);TAB(120);C$;TAB(127);F$(6)
137 PRINT TAB(28);C$;TAB(52);C$;TAB(86);C$;TAB(120);C$&CHR$(13)&B$
138 ON ERROR GOTO 'ERROR'
139 FOR I=U TO 0
140 A$="LZ"
141 'LZ': READ #3,T(I)/10;E$()
142 IF FLAG(2) THEN PRINT USING "#,3D";I
143 PRINT E$(3)&" "&E$(1);TAB(28);C$&E$(2);TAB(52);C$&E$(4);TAB(86);C$&E$(5);TAB
(120);C$&E$(6)
144 NEXT I
145 PRINT CHR$(27)&"&k0S"
146 GOTO 40
147 'Z': OFF ERROR @ IF D0 THEN GOSUB 'G1'
148 DISP "Lschen von Datenstzen" @ IF D0 THEN PRINT
149 FOR I=1 TO 8
150 DISP F$(I);
151 LINPUT S$(I) @ S$(I)=UPRC$(S$(I))
152 IF S$(I)="M" THEN 40
153 IF S$(I)="?" THEN 150
154 IF I=3 AND S$(3)#"" THEN S$(3)=" "&S$(3)
155 NEXT I
156 IF D0 THEN PRINT
157 DISP "Soll rckgefragt werden J/N ?"
158 ON POS("JN",UPRC$(KEY$))+1 GOTO 158,160,159
159 SFLAG 1
160 ON ERROR GOTO 'ERROR' @ A$="Z1"
161 A$="ZZ"
162 IF D0 THEN PRINT E1$&"E"
163 GOSUB 452 @ FOR I=1 TO I1
164 'ZZ': READ #3,T(I)/10;E$()
165 FOR K=1 TO 8 @ IF S$(K)="" THEN 167
166 IF POS(UPRC$(E$(K)),UPRC$(S$(K)))=0 THEN 168
167 NEXT K @ GOTO 172
168 NEXT I
169 IF FLAG(3) THEN 40 ELSE IF D0 THEN PRINT USING "#,/,Z1A";""
170 DISP "Datensatz nicht gefunden !!!     " @ GOTO 40
171 IF D0 THEN PRINT TAB(18);
172 IF FLAG(1) THEN SFLAG 3 @ GOTO 187
173 IF D0 THEN PRINT TAB(18);CHR$(10);
```

```
174 SFLAG 3 @ DISP "Zu lschenden Datensatz gefunden !!!" @ WAIT .6
175 IF D0 THEN DISPLAY IS :NULL
176 DISP E$(3)&E$(1)&" "&E$(2) @ DISP E$(5)&" "&E$(4)
177 IF D0 THEN GOSUB 'G2' @ PRINT E1$&"E"
178 IF D0 THEN SFLAG 14 @ GOSUB 350 @ PRINT
179 DISP "Soll wirklich gelscht werden J/N ?"
180 ON POS("JN",UPRC$(KEY$))+1 GOTO 180,187,181
181 IF FLAG(4) THEN CFLAG 4,14 @ GOTO 303
182 IF FLAG(1) THEN 168
183 IF D0 THEN PRINT
184 DISP "Soll weitergesucht werden J/N ?  oder Menue M ?"
185 ON POS("MJN",UPRC$(KEY$))+1 GOTO 185,40,186,40
186 PRINT E1$&"E" @ GOSUB 452 @ GOTO 168
187 IF D0 THEN PRINT TAB(24);CHR$(10);
188 DISP "Lschen d. Datensatzes" @ WAIT .3 @ T(T(I)/10)=T(T(I)/10)-1 @ I1=I1-1 @
 I2=I2+1
189 FOR Z=I TO I1
190 ON ERROR GOTO 'ERROR'
191 A$="Z3"
192 R=FP(T(Z)/10)*10 @ T(Z)=IP(T(Z+1)/10)*10+R
193 NEXT Z
194 GOTO 181
195 'S': SFLAG 9
196 'A': OFF ERROR @ IF D0 THEN GOSUB 'G1'
197 IF FLAG(9) THEN DISP "Suchen"; @ GOTO 199
198 DISP "ndern";
199 DISP " von Datenstzen" @ IF D0 THEN PRINT
200 FOR I=1 TO 8
201 DISP F$(I);
202 LINPUT S$(I) @ S$(I)=UPRC$(S$(I))
203 IF S$(I)="M" THEN 40
204 IF S$(I)="?" THEN 201
205 IF I=3 AND S$(3)#"" THEN S$(3)=" "&S$(3)
206 NEXT I
207 ON ERROR GOTO 'ERROR' @ A$="A1"
208 A$="A2"
209 GOSUB 452 @ FOR I=1 TO I1
210 'A2': READ #3,T(I)/10;E$()
211 FOR K=1 TO 8 @ IF S$(K)="" THEN 213
212 IF POS(UPRC$(E$(K)),UPRC$(S$(K)))=0 THEN 214
213 NEXT K @ GOTO 217
214 NEXT I
215 IF FLAG(3) THEN 40 ELSE IF D0 THEN PRINT USING "#,/,21A";""
216 DISP "Datensatz nicht gefunden !!!     " @ WAIT .5 @ GOTO 40
217 IF D0 THEN PRINT TAB(18);CHR$(10);
218 SFLAG 3 @ IF FLAG(9) THEN DISP " Gesuchten"; @ GOTO 220
219 DISP "Zu ndernden";
220 DISP " Datensatz gefunden !!!     " @ WAIT 1.6
221 IF D0 THEN DISPLAY IS :NULL
222 IF FLAG(9) THEN DISP USING "17a,4d";"Datensatznummer :",I @ WAIT 2
223 DISP E$(3)&E$(2)&" "&E$(1) @ DISP E$(5)&" "&E$(4)
224 ON ERROR GOTO 224 @ IF D0 THEN GOSUB 'G2' @ PRINT E1$&"E"
225 IF FLAG(9) AND D0 THEN PRINT USING "17a,4d";"Datensatznummer :",I
226 SFLAG 14 @ IF D0 THEN GOSUB 350 @ PRINT CHR$(10);TAB(23);
227 IF FLAG(9) THEN PRINT @ GOTO 244
228 DISP "Was wollen Sie ndern ?"
229 IF D0 THEN DISPLAY IS :NULL
230 STD @ FOR P=1 TO 8 @ DISP STR$(P)&"-"&F$(P)&" "; @ IF P=4 THEN DISP
231 NEXT P
232 IF D0 THEN GOSUB 'G2' @ PRINT TAB(9);"Die Optionen 1 bis 8 oder";
```

```
233 DISP "  N   - 'Nichts'      M   - 'Men'  ?"
234 X$=UPRC$(KEY$) @ IF LEN(X$) THEN 235 ELSE 234
235 IF X$="M" THEN 40
236 IF X$="N" THEN 240
237 X=VAL(X$) @ IF X<1 THEN X=1 ELSE IF X>8 THEN X=8
238 IF D0 THEN PRINT
239 GOSUB 247
240 IF D0 THEN PRINT E1$&"E" @ GOSUB 350 @ PRINT
241 DISP "Soll noch etwas gendert werden J/N ?"
242 ON POS("JN",UPRC$(KEY$))+1 GOTO 242,221,243
243 IF FLAG(4) THEN CFLAG 4,14 @ GOTO 303
244 DISP "Soll weitergesucht werden J/N ?  oder Menue M ?"
245 ON POS("MJN",UPRC$(KEY$))+1 GOTO 245,40,246,40
246 PRINT E1$&"E" @ GOSUB 452 @ GOTO 214
247 OFF ERROR
248 DISP F$(X)&" alt : "&E$(X)
249 DISP F$(X)&" neu ";
250 LINPUT E$(X) @ E$(X)[1,1]=UPRC$(E$(X)[1,1])
251 IF E$(X)="M" THEN 40
252 IF E$(X)="?" THEN 247
253 IF X=3 AND E$(X)#"" THEN E$(X)=" "&E$(X)
254 IF X#3 AND E$(X)#"" THEN E$(X)=E$(X)&" "
255 'A3': A$="A3" @ ON ERROR GOTO 'ERROR'
256 RESTORE #3,T(I)/10
257 PRINT #3,T(I)/10;E$()
258 RETURN
259 'D': OFF ERROR @ IF D0 THEN GOSUB 'G1'
260 GOSUB 'p9' @ IF NOT D0 THEN WAIT 5
261 DISP "Drucken von Datenstzen" @ IF D0 THEN PRINT
262 FOR I=1 TO 8
263 DISP F$(I);
264 LINPUT S$(I) @ S$(I)=UPRC$(S$(I))
265 IF S$(I)="M" THEN 40
266 IF S$(I)="?" THEN 263
267 IF I=3 AND S$(3)#"" THEN S$(3)=" "&S$(3)
268 NEXT I
269 IF D0 THEN PRINT
270 DISP "Soll rckgefragt werden J/N ?"
271 ON POS("JN",UPRC$(KEY$))+1 GOTO 271,273,272
272 SFLAG 1
273 IF FLAG(1) THEN DISP "Telefonnummer und Sonstiges drucken J/N ?"
274 ON POS("JN",UPRC$(KEY$))+1 GOTO 274,276,275
275 IF NOT FLAG(1) THEN 276 ELSE SFLAG 2
276 ON ERROR GOTO 'ERROR' @ A$="D1"
277 A$="D2"
278 IF D0 THEN PRINT E1$&"E"
279 GOSUB 452 @ FOR I=1 TO I1
280 'D2': READ #3,T(I)/10;E$()
281 FOR K=1 TO 8 @ IF S$(K)="" THEN 283
282 IF POS(UPRC$(E$(K)),UPRC$(S$(K)))=0 THEN 284
283 NEXT K @ GOTO 287
284 NEXT I
285 IF FLAG(3) THEN 40 ELSE IF D0 THEN PRINT USING "#,/,21A";""
286 DISP "Datensatz nicht gefunden !!!     " @ GOTO 40
287 SFLAG 3
288 PRINTER IS :%35
289 PRINT CHR$(27)&"&k3S"&E$(3)&" "&E$(2)&E$(1) @ PRINT " "&E$(4) @ PRINT " "&E$
(5)
290 IF FLAG(2) THEN PRINT @ PRINT @ GOTO 284 ELSE IF FLAG(1) THEN X$="J" @ GOTO
294
```

```
291 IF D0 THEN GOSUB 'G3' @ PRINT @ PRINTER IS :%35
292 DISP "Auch die Telefonnummer und Sonstiges drucken J/N ?"
293 ON POS("JN",UPRC$(KEY$))+1 GOTO 293,294,295
294 PRINT " "&E$(6) @ PRINT " "&E$(7) @ PRINT " "&E$(8) @ GOTO 295
295 PRINT @ IF FLAG(1) THEN 284
296 PRINT @ DISP "Soll weitergesucht werden J/N ? oder Menue M ?"
297 ON POS("MJN",UPRC$(KEY$))+1 GOTO 297,40,298,40
298 PRINT E1$&"E" @ GOSUB 452 @ GOTO 284
299 'B': IF D0 THEN GOSUB 'G1'
300 OFF ERROR @ DISP "Blttern in der Datei "
301 IF I1=0 THEN DISP "Keine Daten vorhanden" @ WAIT .5 @ GOTO 40
302 INPUT "Startnummer ?";Q @ IF Q<1 OR Q>I1 THEN 302
303 ON ERROR GOTO 'ERROR' @ A$="B1" @ IF Q>I1 THEN Q=I1
304 WINDOW 1,22 @ 'B1': READ #3,T(Q)/10;S$()
305 IF D0 THEN DISPLAY IS :NULL @ PRINT E1$&"E"&'Datensatz Nr.'; @ PRINT USING "
dddd";Q
306 FOR K=1 TO 8 @ E$(K)=S$(K) @ NEXT K @ DISP USING "ddd";Q
307 IF D0 THEN GOSUB 350
308 W=1 @ WINDOW 5,22
309 T$=S$(1) @ L=LEN(T$)-17 @ DISP T$
310 C$=KEY$
311 IF C$="#47" THEN 344
312 IF C$="#48" THEN 345
313 IF C$="#104" THEN 331
314 IF C$="#103" THEN 334
315 IF C$="#50" THEN 337
316 IF C$="#51" THEN 339
317 IF C$="#159" THEN Q=1 @ GOTO 303
318 IF C$="#160" THEN Q=I1 @ GOTO 303
319 IF C$="#162" THEN W=78 @ GOTO 337
320 IF C$="#163" THEN W=-55 @ GOTO 337
321 IF UPRC$(C$)="A" THEN SFLAG 4 @ WINDOW 1 @ I=Q @ GOTO 224
322 IF UPRC$(C$)="M" THEN WINDOW 1 @ GOTO 40
323 IF UPRC$(C$)="Z" THEN SFLAG 4 @ I=Q @ WINDOW 1 @ GOTO 177
324 IF UPRC$(C$)#"D" THEN 310
325 IF S$(3)="" THEN S$(3)=" "
326 PRINTER IS :%35
327 PRINT CHR$(27)&"&k3S" @ PRINT "Datensatz Nr."; @ PRINT USING "#,dddd";Q
328 SFLAG 14 @ GOSUB 350 @ CFLAG 14
329 IF D0 THEN GOSUB 'G3'
330 GOTO 310
331 Q=Q+1
332 L=1 @ IF Q>I1 THEN Q=1
333 GOTO 303
334 Q=Q-1
335 IF Q<1 THEN Q=I1
336 GOTO 303
337 W=W-1
338 IF W<1 THEN W=8
339 IF C$="#51" THEN W=W+1
340 IF W>8 THEN W=1
341 T$=S$(W)
342 L=LEN(T$)-17
343 DISP T$ @ GOTO 310
344 L=L-2
345 L=L+1 @ IF L>LEN(T$)-18 THEN L=LEN(T$)-18
346 IF L<1 THEN L=1
347 IF LEN(T$)<=18 THEN 310
348 DISP T$[L,L+17]
349 GOTO 310
```

```
350 PRINT USING "/"
351 FOR G=1 TO 8
352 IF FLAG(14) THEN PRINT USING "#,d,3A";G;" - "
353 PRINT USING "#,17A";F$(G)
354 PRINT ":  "&E$(G)
355 NEXT G
356 IF FLAG(14) THEN RETURN
357 PRINT USING "2/,30A";"Bitte eine Taste drcken !!!"
358 RETURN
359 'P': IF D0 THEN GOSUB 'G1'
360 DISP "Statusinformationen"
361 IF D0 THEN PRINT
362 GOSUB 379
363 DISP "Dateiname :";UPRC$(Z1$) @ IF D0=0 THEN WAIT 1
364 GOSUB 379
365 STD @ DISP "Dateigre :";(I3+1)*512;"Bytes" @ IF D0=0 THEN WAIT 1
366 GOSUB 379
367 DISP "Indexdateigre : 8192 Bytes" @ IF D0=0 THEN WAIT 1
368 GOSUB 379
369 DISP "Maximale Anzahl der Datenstze :";I3 @ IF D0=0 THEN WAIT 1
370 GOSUB 379
371 DISP "Davon belegt :";I1 @ IF D0=0 THEN WAIT 1
372 GOSUB 379
373 DISP "Und unbelegt :";I2 @ IF D0=0 THEN WAIT 1
374 GOSUB 379
375 DISP "Freier Hauptspeicher :";MEM;"Bytes" @ IF D0=0 THEN WAIT 1
376 GOSUB 379
377 IF D0 THEN DISP @ GOSUB 379 @ DISP "Bitte eine Taste drcken !!!";
378 IF D0=0 THEN 40 ELSE IF KEY$="" THEN 378 ELSE DISP @ GOTO 40
379 IF D0 THEN DISP TAB(12);
380 RETURN
381 E1$=CHR$(27) @ GOSUB 'G1'
382 P$="************************"
383 PRINT USING "2/" @ PRINT P$&P$&P$
384 IF D1 THEN PRINT E1$&CHR$(54);
385 IF D2 THEN PRINT TAB(17);E1$&"FE1";
386 PRINT CHR$(31);
387 PRINT " Die einfache  Adressenverwaltung "; @ PRINT CHR$(31)
388 IF D1 THEN PRINT E1$&CHR$(53);
389 IF D2 THEN PRINT E1$&"FE0";
390 PRINT P$&P$&P$ @ PRINT @ RETURN
391 GOSUB 'G1' @ DISPLAY IS :NULL
392 P$="****************************"
393 PRINT USING "2/" @ PRINT TAB(5); @ PRINT P$&P$
394 IF D1 THEN PRINT TAB(5);E1$&CHR$(54);
395 IF D2 THEN PRINT TAB(16);E1$&"FE1";
396 PRINT CHR$(31)&CHR$(31);
397 PRINT "  H A U P T M E N U E  "; @ PRINT CHR$(31)&CHR$(31);
398 IF D1 THEN PRINT E1$&CHR$(53);
399 IF D2 THEN PRINT E1$&"FE0";
400 PRINT TAB(5); @ PRINT P$&P$ @ PRINT USING "3/"
401 PRINT TAB(11);"E  -  Eingabe von Datenstzen"
402 PRINT TAB(11);"A  -  ndern  von Datenstzen"
403 PRINT TAB(11);"Z  -  Lschen von Datenstzen"
404 PRINT TAB(11);"D  -  Drucken von Datenstzen"
405 PRINT TAB(11);"S  -  Suchen  von Datenstzen"
406 PRINT TAB(11);"L  -  Druck einer vollstndigen Datenliste"
407 PRINT TAB(11);"T  -  Druck eines Teils der Datenliste"
408 PRINT TAB(11);"B  -  Blttern in der Datei"
409 PRINT TAB(11);"P  -  Statusinformationen"
```

```
410 PRINT @ PRINT TAB(11);"X  -  Ende :  Daten speichern und Ende des Programms"
411 PRINT @ PRINT TAB(14);"B i t t e  w h l e n  S i e  ! ! !";
412 RETURN
413 P$="**********************" @ PRINT E1$&"E";
414 PRINT USING "2/" @ PRINT TAB(17); @ PRINT P$&P$;
415 IF D1 THEN PRINT TAB(17);E1$&CHR$(54);
416 IF D2 THEN PRINT TAB(26);E1$&"FE1";
417 PRINT CHR$(31)&CHR$(31);
418 PRINT "   Programmende   "; @ PRINT CHR$(31)&CHR$(31);
419 IF D1 THEN PRINT E1$&CHR$(53);
420 IF D2 THEN PRINT E1$&"FE0";
421 PRINT TAB(17); @ PRINT P$&P$
422 PRINT USING "3/,17A,43A";"";"Die Indexdatei wird jetzt abgespeichert !!"
423 RETURN
424 'NEU': IF FLAG(-27) THEN 40
425 A1$=Z1$ @ DISP "Wie soll die neue Datei heien "; @ LINPUT Z1$ @ Z1$=UPRC$(Z1
$)
426 IF Z1$="M" AND A1$#"" THEN 40 ELSE IF Z1$="M" AND A1$="" THEN RUN
427 IF Z1$="?" THEN 424
428 IF LEN(Z1$)>7 THEN DISP "Name zu lang !!! Maximal 7 Buchstaben !" @ GOTO 424
429 DISP "Wieviele Datenstze soll die Datei haben "; @ LINPUT Z2$ @ Z2$=UPRC$(Z2
$)
430 IF Z2$="?" THEN 429
431 IF Z2$="M" THEN 40
432 D=IP(VAL(Z2$))+1
433 IF D>1000 THEN BEEP 1400,.10 @ DISP "Zu gro !! Maximal 999 Datenstze !!  " @
 GOTO 429
434 GOSUB 381 @ DISP "Die Datei wird angelegt !!"
435 A$="N1" @ ON ERROR GOTO 'ERR'
436 'N1': CREATE DATA Z1$&":%16",D,512
437 DESTROY T @ INTEGER T(999) @ OFF ERROR
438 STD @ FOR I=1 TO 8
439 READ F$(I)
440 DATA "Nachname","Vorname","Titel","Strae","Stadt","Telefon","Sonstiges 1","S
onstiges 2"
441 NEXT I
442 'N2': A$="N2" @ CREATE DATA Z1$&"i:%16",32,256
443 ON ERROR GOTO 'ERROR'
444 'N3': A$="N3" @ ASSIGN #3 TO Z1$&":%16"
445 'N4': A$="N4" @ PRINT #3,0;0,D-1,D-1
446 'N5': A$="N5" @ PRINT #3;F$()
447 I1=0 @ I2=D-1 @ I3=D-1
448 GOTO 40
449 'ERROR': IF ERRN=255039 THEN A$
450 RESET HPIL @ STANDBY 6 @ WINDOW 1,22 @ IF ERRN=255043 THEN A$
451 BEEP 1400,.10 @ DISP ERRM$ @ OFF ERROR @ GOTO 40
452 IF D0 THEN PRINT USING "#,/,25A";""
453 DISP "Suchen in der Datei " @ RETURN
454 'ERR': IF ERRN=255022 THEN DISP "Datei nicht gefunden !" @ GOTO 459
455 IF ERRN=255031 OR ERRN=255030 THEN DISP "Datei schon vorhanden!" @ GOTO 459
456 IF ERRN=255017 THEN DISP "Datei zu gro fr die Diskette" @ GOTO 459
457 IF ERRN=255039 THEN A$
458 DISP ERRM$ @ PRINTER IS :%35 @ PRINT E1$&"&k0S" @ IF D0 THEN GOSUB 'g2'
459 USER OFF @ POKE "2F441","0" @ DEF KEY "#43" @ MERGE 'KEYS'
460 PRINTER IS :%35 @ DISP E1$&"E"&"Ende" @ END
461 'G1': IF D1 THEN PRINTER IS :%48 @ DISPLAY IS :%48
462 IF D2 THEN PRINTER IS :%64 @ DISPLAY IS :%64
463 PRINT E1$&"E" @ RETURN
464 'G2': IF D1 THEN DISPLAY IS :%48
465 IF D2 THEN DISPLAY IS :%64
```

Fortsetzung des Listings auf Seite 51

Funktion:

BROWSE ist Utility zum seitenweisen Anzeigen von Textdateien. Im Gegensatz zum DOS-Befehl TYPE kann man bei BROWSE im Text mit den Cursortasten vor und zurück blättern.

Format:

[*d:*][*Pfad*]BROWSE [[*d:*][*Pfad*]*Dateiname*]

Typ:

Intern Extern

***

Funktion:

DOSKEY ist ein speicherresidentes Programm, das dem DOS-Promt zu mehr Komfort verhilft. Dazu gehören ein Befehlsstack und die Belegung der Funktionstasten mit Befehlen.

Format:

[*d:*][*Pfad*]DOSKEY

Typ:

Intern Extern

***

Hinweise:

Nach dem Aufruf von DOSKEY werden alle DOS-Befehle in einem Puffer gespeichert. Er hat Platz für 15 Befehle. Mit den Cursortasten (Auf/Ab) kann man in den vorherigen Befehlen blättern.

Mit den anderen Tasten des Cursorblocks kann man die aktuelle Befehlszeile editieren.

Die Funktionstasten sind mit folgenden Befehlen belegt:

F 1 – Schaltet die Vordergrundfarbe um
F 2 – Schaltet die Hintergrundfarbe um
F 3 – dir
F 4 – type
F 5 – copy
F 6 – delete
F 7 – chdir
F 8 – path
F 9 – browse
F10 – cls

F11 – backup
F12 – restore

F11 und F12 stehen nur bei einer MF-Tastatur zur Verfügung. Bei F1 und F2 kann man, wenn man die Taste zusammen mit Shift drückt, auch rückwärts in der Farbpalette blättern.

Damit der BROWSE-Befehl funktioniert, muß das Programm BROWSE.COM von dieser INFO im Suchpfad von DOS liegen.

Funktion:

CO ist ein Utility zum Manipulieren von Dateigruppen. Man kann Dateien kopieren, löschen und verschieben.

Format:

[*d:*][*Pfad*]CO [*d:*][*Pfad*] [*/E*] [*/S*] [*/D*] [*/T*] [*/O*]

Typ:

Intern Extern

***

Hinweise:

Nach dem Aufruf meldet sich CO mit Dateiliste und Menü. Hier sind alle Funktionen per Tastendruck abrufbar.

Wird beim Abruf kein Verzeichnis angegeben nimmt CO das aktuelle Verzeichnis.

Die Sortierung der Dateien kann gleich beim Aufruf gewählt werden:

/E – sortiert nach Extension
/S – sortiert nach Größe
/D – sortiert nach Datum
/T – sortiert nach Zeit
/O – sortiert nicht.

Ohne Angabe sortiert CO nach Namen.

Funktion:

PRN2FILE leitet die gesamte Druckerausgabe in eine Textdatei um. Die Textdatei kann mit einem normalen Editor editiert werden.

Format:

[*d:*][*Pfad*]PRN2FILE [*Dateiname*] [*/Pn*] [*/Bn*] [*/U*]

Typ:

Intern Extern

***

Hinweise:

PRN2FILE ist ein speicherresidentes Programm. Es sollte vor anderen residenten Druckerutilities geladen werden, um Konflikte zu vermeiden.

Die Druckdatei kann auch gewechselt werden, dazu ist PRN2FILE mit einem anderen Dateinamen aufzurufen, bzw. ohne Dateiname um die Druckumleitung abzuschalten.

Beim Aufruf von PRN2FILE sind folgende Parameter möglich:

[*/Pn*] Wählt Drucker LPTn aus (LPT1 – LPT3). Standardeinstellung LPT1.
[*/Bn*] Puffergröße in KByte. Gibt die Größe des Puffers an, in dem die Daten zwischengespeichert werden. Wird der Puffer zu klein gewählt kann es zu einem Überlauf kommen. Standardeinstellung 4 KBytes.
[*/U*] Entfernt PRN2FILE aus dem Speicher. Die Druckerausgaben werden dann wieder zum Drucker geleitet.

MARYS II

Da liegt nun dieses geheimnisumwitterte Kästchen vor mir, Kästchen ist wohl doch etwas untertrieben: der HP41, den ich gerade daneben gelegt habe sieht dagegen etwas schmächtig aus, MARY ist etwa 2,5 cm länger als der 41'er mit Kartenleser.

Die Ähnlichkeit mit dem HP41 ist wirklich verblüffend, dieselbe Grundform, nur überall gleichmäßig dick, die eigentlich recht praktische Pultform wurde aus Platzgründen nicht übernommen, unter dem gesamten Gerät befinden sich die 6 Modulsteckplätze und ein Drittel des Bodenplatzes nimmt das Batteriefach in Anspruch, hier haben 5 Mignonzellen Platz.

Wie man im letzten PRISMA auf dem Foto sehen konnte, ist das Tastaturlayout fast komplett vom HP41 übernommen worden, auf den Tasten stehen aber in demselben Blau die Buchstaben, was auf den ersten Blick natürlich etwas verwirrt. Die Funktionen, beim HP41 in weiß auf den Tasten, stehen bei MARY ebenfalls in weiß, allerdings über den Tasten. Die mit SHIFT zu erreichenden und ab HP41CV in Gold gehaltenen Funktionen sind gerade oberhalb der weißen Funktionsnamen zu finden, die Aufteilung beim HP41 war doch leichter zu lesen . . .

(Das beschriebene Tastaturlayout ist das der Schablone für die HP41CX-Emultation)

Bei all der Kritik am Tastaturlayout darf aber natürlich nicht die eigentliche Funktion dieses Rechners vergessen werden, er ist eigentlich ein winziger PC, zumindestens, was seine Bedieneroberfläche angeht, dazu aber später. Die sogenannte HP41-Emulation ist nämlich ein Modul (128kByte), in dem ein Programm steht, das einen HP41CX !! (den mit HP-IL Modulfunktionen für Drucker und Massenspeicher) simuliert, natürlich mit größerem Display und noch einigen sehr interessanten Funktionen, dazu aber später, die genauen Daten des Rechners hatte ich ja schon im letzten Heft beschrieben.

Ich möchte mich zuerst mit der HP41CX-Emulation befassen, soweit mir dies aufgrund der zur Verfügung stehenden Hardware möglich ist, ich glaube, daß das wohl sehr viele interessieren wird. Diese besteht aus einem CMT MC-II, so heißt das Gerät offiziell, einem MC-41M Emulatormodul, ein 128kB RAM-Modul und das Betriebssystem-ROM, das ebenfalls einen Steckplatz belegt, Betriebssystemupdates sind also jederzeit durch Austausch dieses Moduls möglich, hoffentlich nicht so oft wie bei IBM mit ihrem MS-DOS.

Bleiben noch drei Steckplätze für weitere (RAM/ROM/EPROM-) Module übrig, sechs hat der MC-II insgesamt.

Das Handbuch für das Emulatomodul ist gut 170 Seiten stark, DIN A4 und sehr gut und sauber geschrieben, ordentlich gesetzt, was ja bei einigen Modulen für den HP41 oft nicht der Fall war. Alle Befehle, Eigenschaften und Eigenarten dieses Moduls sind wirklich vorbildlich, wenn auch noch in Englisch, beschrieben, ideal auch für Anfänger, oder auch gerade für diese . .

Das Handbuch für das Gerät selbst ist mit gut 100 Seiten zwar etwas kleiner aber keineswegs kurz gehalten, auch hier wurde gute didaktische Arbeit geleistet.

Ich suche nun also die Taste zum Einschalten, ganz rechts unten steht auch ON. Nach dem Draufdrücken erscheint

```
F:              size:
41M.S                  7
OFF.S                  6
KEYBEEP.S              9
TIME.S                 6
DATE.S                 6
MODE.S                 6
MARK COPY DEL VIEW RUN
```

in der Anzeige.

F: ist dabei die Anzeige, welches Laufwerk bzw. in diesem Fall welcher Port des MC-II gerade angezeigt wird. Die Angabe .S hinter den Files besagt dabei, das es sich dabei um sogenannte Script Files handelt, also Dateien, in denen ein oder mehrere Kommandos als ganz normaler Text abgespeichert sind. Bei den PC-Benutzern klingelt es jetzt wahrscheinlich recht heftig, das können doch nur BATCH-Files sein, völlig richtig.

41M.S ist invers dargestellt, d.h. wenn ich den ganz rechten Softkey F5 (RUN) drücke, müßte eigentlich der HP41 Emulatormodus anspringen. Gedacht, getan, das Display wechselt schlagartig:

```
<<<<<<< 41M >>>>>>>
ver 1.01      Jan 19
© 1987,88     C.M.T

0.0000
```

sehe ich auf dem Display.

Einige Zahlen eingetippt, alles ganz normal, nur daß alle eingegebenen Zahlen und Funktionen also Protokoll nach oben scrollen, d.h. die letzten 6 Aktionen sind noch auf dem Display zu sehen.

Ich schalte in den ALPHA-Modus, gebe beliebige Zeichen ein, 21 passen gleichzeitig nebeneinander auf das Display, dann wird der Rest wie beim HP41 nach links geschoben; beim 24. Zeichen ertönt der gewohnte Warnton, alles wie gehabt.

Drücken wir noch einmal die Taste LCASE (Lowercase), ab sofort werden nur Kleinbuchstaben eingegeben, dies ist etwas angenehmer als der Trick mit dem USER-Modus bei CCD-Modul.

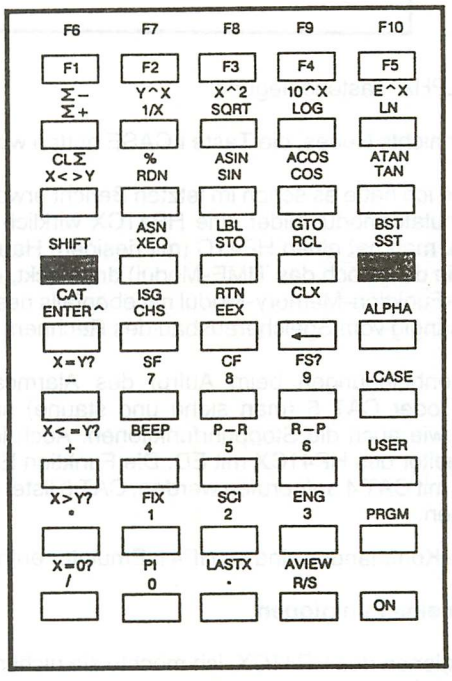


Bild 1: Normales Tastenfeld

Auf dem Bild 1 sehen wir nicht nur die uns gewohnten Tasten, ALPHA, PRGM, USER und ON sind nach rechts unten gerutscht. Die Taste LCASE ist neu und aktiviert den Kleinschreibmodus.

In der ersten Reihe prangen 5 (10) neue Funktionstasten, denen alle bisher bekannten Funktionen und zusätzlich noch Sonderbuchstaben zugewiesen werden können, die man häufiger im ALPHA-Modus eingeben will. Dies geschieht auch mit ASN, nur daß dann statt eines Funktionsnamens oder Labels der 3stellige Dezimalcode des Buchstabens angegeben wird, so einfach ist das.

Bei Aufruf einer Funktion wie STO__ erscheinen ja immer eine bestimmte Anzahl von sogenannten Eingabepromts, d.h. der maximalen Anzahl adressierbarer Register, beim HP41 sind es also 99.

Beim 41'er Emulator sind es maximal 65536 ! ! (je nach Speicherausbau). Mit der Taste EEX kann die Anzahl der Eingabestellen schrittweise erhöht werden, mit CHS geht es dann wieder rückwärts.

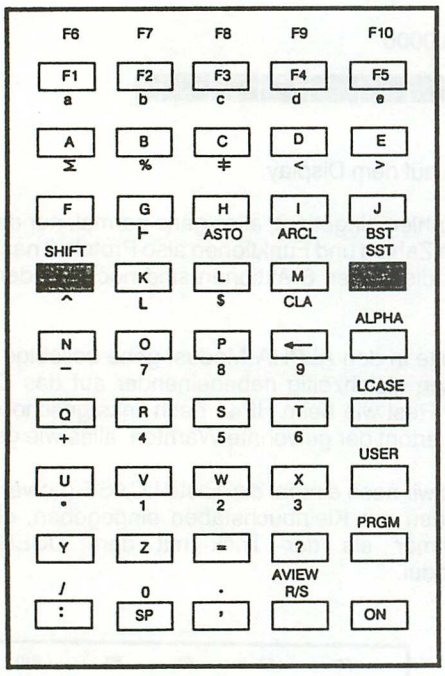


Bild 2: ALPHA-Tastenbelegung

Auch hier nichts Neues, die Taste LCASE hatten wir ja schon.

Ich glaube ich hatte es schon im letzten Bericht erwähnt, dieses HP41-Emulatormodul bildet eine HP41CX wirklich vollständig nach, d.h. man hat einen HP41C (mit riesigem Hauptspeicher) vor sich, in dem noch das TIME-Modul) drinsteckt, ebenso das Extendet-Funktion-Memory-Modul mit ebenfalls riesigem Speicher, abhängig vom Speicherausbau des Rechners natürlich.

Die Tastenbelegungen beim Aufruf des Alarmcatalogs mit ALMCAT oder CAT 5 (man siehe und staune) sind ebenso identisch wie auch die Stoppuhrfunktionen. Auch identisch ist der Texteditor des HP41CX mit ED. Die Funktion EMDIR kann ebenfalls mit CAT 4 aufgerufen werden, CAT 6 listet die Tastenbelegungen.

Folgende Kommandos sind im HP41-Emulator enthalten:

1. Allgemeine Funktionen

Alle Funktionen des HP41CX, ich möchte sie nicht noch einmal im Detail aufzählen.

2. Druckerfunktionen

MACA
MACCHR
MACK
MAN
MLIST
MPRA
MPRP
MPRREG
MPRREGX
MPRSTK
MPRΣ
MPRX
MSKPCHR
NORM
PROFF
PRON
TRACE
TRACES

Diese Funktionen sind im Prinzip dieselben wie im IL-Modul, das M vorneweg soll sie von diesen unterscheiden, da alle Ausgaben dieser Funktionen auf die deklarierte Druckerschnittstelle gehen, das kann die erste wie auch die zweite serielle Schnittstelle oder auch das Display selbst sein, man kann also ein Programm auch ohne Drucker mitprotokollieren. Auf den Rest möchte ich an dieser Stelle nicht näher eingehen.

3. Filesystemfunktionen

MCREATE
MINA
MIND
MOUTA
MREADK
MREADP
MREADR
MREADRX
MREADSB
MSEEKR
MWRTAP
MWRTAPV
MWRTK
MWRTP
MWRTPV
MWRTR
MWRTRX
RCVFL
SENDFL

} Empfangen/Senden von Files mit Hilfe von KERMIT

Auch hier lauter bekannte Namen, ebenfalls mit einem M für 41M wie Mary vorne dran. Die ganzen Funktionen beziehen sich im IL-Modul auf Massenspeicher wie z.B. das Kassettenlaufwerk. Hier wirken diese auf den internen Speicher des MC-II, es werden also in seinem normalen RAM Files wie auf der Kassette angelegt, die dann von anderen Programmen verarbeitet werden können, sehr gut für Meßdatenverarbeitung. Datenfiles lassen sich natürlich dann mittels des im Betriebssystem enthaltenen Filetransferprogramms KERMIT zu jedem beliebigen Rechner über eine der beiden seriellen Schnittstellen übertragen und dann auf einem größeren Rechner verarbeiten.

Mit dem Befehl MODE, die MS-DOSler kennen ihn wahrscheinlich, lassen sich die beiden seriellen Schnittstellen auch vom Emulator aus direkt einstellen.

Der Befehl SOFTKEY kreiert zusätzliche Softkeys, die dann in der untersten Zeile direkt unter den Statusanzeigen im Display stehen, sehr praktisch für menügeführte Programme, diese Funktion findet sich bei den IBM-Programmen leider nur sehr selten, wer aber einmal mit Softkeys gearbeitet hat, weis diese über alle Maßen zu schätzen.

Mit Hilfe des Befehls SPAWN kann jedes beliebige Programm des MC-II ausgeführt werden, man kann also mal schnell den eingebauten Texteditor MCED aufrufen, sich ein Paar Notizen machen, diese speichern und wieder in den Emulator zurückgehen als wäre nichts gewesen, alle Daten sind unverändert.

Ebenso können ja z.B. in C eigene Programme wie ein Displayplotprogramm geschrieben werden, die dann vom „41'er" aus aufgerufen werden können und durch die Compilierung natürlich wesentlich schneller als ein 41'er Programm sind.

SYSTEM bringt einen dann wieder zu guter Letzt zurück in das eigentliche Betriebssystem des MC-II.

Ich möchte jetzt kurz einige Unterschiede zum HP41 beleuchten, schreibe aber trotzdem nicht den APPENDIX D aus der Bedienungsanleitung ab:

Das Display ist viel größer, na klar.

Alle Flags können gesetzt und gelöscht werden, ihre Zahl ist auf 60 gestiegen. Man kann also z.B. die Funktion ON durch SF 44 aktivieren . . .

Der Zahlenbereich geht bis E±308, ich kann also ruhig FACT 100 ausführen, kein Problem. Außerdem werden 15 Stellen nach dem Komma angezeigt und berechnet.

Laut Hersteller laufen die Programme 5-10 Mal schneller ab als im HP41, ich habe es nicht nachgeprüft, es geht aber alles wesentlich schneller.

(Unter-)Programme können als XROM-Funktionen im normalen Hauptspeicher des MC-II abgelegt werden und werden natürlich auch schneller gefunden als ein ALPHA-Label. Ich konnte leider das Verfahren hierfür aus Zeitgründen nicht ausprobieren, es ist mit Sicherheit sehr interessant.

Es können bis zu 20 Unterprogrammebenen angesprungen werden, beim HP41 sind es 6.

Wie schon erwähnt, die Liste ist nicht vollständig, das waren nur die meiner Meinung nach wichtigsten Punkte.

So, jetzt wollen wir uns noch kurz dem eigentlichen MARYS II zuwenden, er steckt auch noch voller interessanter Details.

Noch einmal als Wiederholung folgende Punkte:

Das Betriebssystem, das CMT-ROS (ROS = **R**AM **O**perating **S**ystem) steckt mit seinen 42kByte im Betriebssystem-ROM, das einen der 6 zur Verfügung stehenden Ports belegt. Der Kommandointerpreter belegt noch einmal 26kByte, MCED, das ist der ASCII-File-Editor, 29kByte und zu guter Letzt KERMIT, das Filetransferprogramm 15kByte. Die außerdem noch enthaltenen Batch-Files fallen vom Platz nicht in's Gewicht, erleichtern die Bedienung aber nicht unerheblich (dazu gehörte auch 41M.S, mit dem ich zu Anfang die HP41 Emulation aufgerufen hatte). Alles in allem sind also im ROM fast die gesamten 128kByte belegt (genaugenommen bleiben 16kByte übrig).

Für den Betrieb ist mindestens ein 128kByte RAM-Modul erforderlich, es gehört immer zum Lieferumfang. Das Betriebssystem zwackt davon 32kByte für sich selbst und noch einmal 32kByte für Programme ab, bleiben 64kByte für Texte, neue Programme und Speicher für die HP41 Emulation in der Grundausstattung übrig. Bei den derzeit erhältlichen 128kByte RAM-Modulen ist also ein maximaler Speicherausbau auf 512kByte möglich, wenn das HP41 Emulations-ROM nicht im Rechner steckt, sonst sind es eben 128kByte weniger.

Die RAM-Module sind mit einer Litiumbatterie ausgerüstet, man kann also getrost das Modul aus dem Rechner nehmen, nach dem Wiedereinsetzen sind noch alle Daten bis zu einem Jahr externem Aufenthalt gespeichert.

Die beiden seriellen Schnittstellen an der Kopfseite des Rechners sind als 6-polige Modembuchsen amerikanischer Bauart ausgeführt, man findet auf beiden die Leitungen RXD, TXD, RTS, CTS und zweimal GND.

Es lassen sich alle Baudraten von 110 bis 19200 einstellen, dies geht aber leider immer nur für beide gemeinsam. Die anderen Einstellungen wie Parität oder XON/XOFF sind für jede Schnittstelle einzeln einstellbar.

Unter dem Rechner befindet sich ein 32-poliger Expansionsport, an dem die Prozessorsignale für Peripheriegeräte zur Verfügung stehen, eine RAM-DISK und ein HP-IL Interface wurden im Handbuch angeführt.

Das Tastenfeld seht ihr in Bild 3, dazu ist, glaube ich, nicht viel zu sagen.

Der Kommandointerpreter kennt zwei Funktionsmodi, den gewohnte normale Zeileneingabemodus und eine menügeführte Eingabe. Bei dieser wird das Direktory ständig angezeigt, das aufzurufende File/Programm wird mit den Cursortasten angewählt und mit dem Softkey RUN gestartet. Softkeys gibt es insgesamt 5, diese werden invers in der untersten Displayzeile angezeigt und wechseln bei verschiedenen Kommandos ihre Belegung.

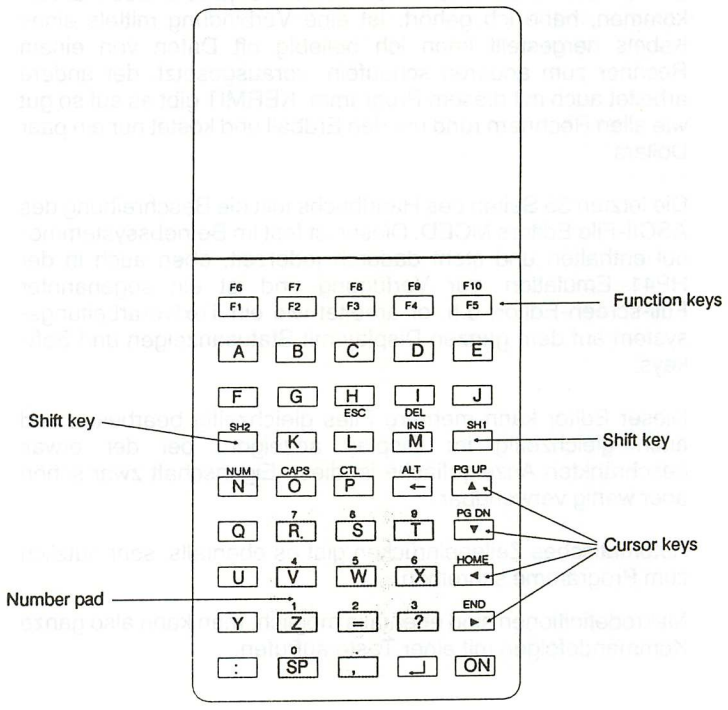


Bild 3: Normale Tastatur

Im Zeileneingabemodus werden alle Eingaben in einem 512Byte großen Stack gespeichert und können mit Cursor up und down wieder in die Eingabezeile geholt werden, ganz schön komfortabel. Wildkards wie * oder ? sind ebenfalls erlaubt, in menügesteuerter Eingabeweise kann man Files markieren und dann gemeinsam z.B. kopieren.

Folgende Kommandos sind implementiert:

CD	Direktorywechsel
CHKDSK	Laufwerksparameter anzeigen
CHMOD	Schreib-Lesemodus eines Files ändern
CLS	Bildschirm löschen
CMD	der Kommandointerpreter selbst
COPY	Files kopieren
CTTY	Ein/Ausgaben extern
DATE	Datum anzeigen/ändern
DEL	Files löschen
DIR	Inhaltsverzeichnis anzeigen
FORMAT	Laufwerksparameter setzen (RAM-Modul)
KERMIT	Filetransferprogramm
KEYBEEP	Tongenerator ein/ausschalten
LINE	Zeileneingabemodus anwählen
MENU	Menüeingabemodus anwählen
MODE	Serielle Schnittstellen setzen
REN	Files umbenennen
TIME	Zeit anzeigen/ändern
TYPE	Fileinhalt anzeigen
VER	Betriebssystemversion anzeigen
VIEW	Fileinhalt anzeigen (=TYPE)

Was ist wohl CTTY?

Das habe ich mich auch gefragt, ein Blick in's Handbuch löste das Rätsel höchst erfreulich. Mit diesem Befehl kann ein externes Terminal oder ein Rechner mit Terminalemulation Tastatur und Display des MC-II ersetzen, man kann also alle Eingaben über die natürlich bequemere PC-Tastatur machen, die Bildschirmausgaben erfolgen aber trotzdem in 21 Spalten/Zeile.

Über KERMIT kann man eine Verbindung zu einem anderen Computer über eine der beiden seriellen Schnittstellen herstellen, dies gilt natürlich auch für zwei MARYs, wenn sie sich einmal miteinander unterhalten wollen, soll ja ab und zu mal vorkommen, habe ich gehört. Ist eine Verbindung mittels eines Kabels hergestellt kann ich beliebig oft Daten von einem Rechner zum anderen schaufeln, vorausgesetzt, der andere arbeitet auch mit diesem Programm. KERMIT gibt es auf so gut wie allen Rechnern rund um den Erdball und kostet nur ein paar Dollars.

Die letzten 35 Seiten des Handbuchs füllt die Beschreibung des ASCII-File Editors MCED. Dieser ist fest im Betriebssystemmodul enthalten und steht dadurch jederzeit, eben auch in der HP41 Emulation, zur Verfügung und ist ein sogenannter Full-screen-Editor, d.h. er arbeitet wie ein Textverarbeitungssystem auf dem ganzen Display mit Statusanzeigen und Softkeys.

Dieser Editor kann mehrere Files gleichzeitig bearbeiten und auch gleichzeitig im Display anzeigen, bei der etwas beschränkten Anzeigefläche ist diese Eigenschaft zwar schön aber wenig verwendbar.

Automatisches Zeileneinrücken gibt es ebenfalls, sehr nützlich zum Programme schreiben.

Makrodefinitionen sind ebenfalls möglich, man kann also ganze Kommandofolgen mit einer Taste aufrufen.

Eine HELP-Taste für Anfänger und Gelegenheitsbenutzer zeigt Kommandoinformationen an, die Funktion UNDO rettet so manchen Text vor Irrtümern oder Vertippen.

So, ich hoffe nicht allzuviele Fragen aufgeworfen zu haben, die Bedienungsanleitung wollte ich ja nun auch nicht abschreiben.

Résumé

Mir ist dieser sogenannte Nachfolger des HP41 in der kurzen Testzeit doch sehr an's Herz gewachsen, er bietet auf engstem Raum eine Fülle von all den Funktionen und Eigenschaften, die das tägliche Leben und Arbeiten mit diesen kleinen Rechenzwergen erleichtern.

Mary hat wohl das Zeug, ein neuer Standard in der mobilen Meßdatenerfassung und auch Verarbeitung zu werden, Speicherplatz steht ja genügend zur Verfügung, ebenso verschiedene Schnittstellen.

Constant Memory und die aus dem HP41 bekannten Alarmbefehle ermöglichen ein unbeobachtetes Arbeiten des Gerätes auch über einen längeren Zeitraum hinweg (Ausschalten kann er sich ja auch immer selbst), es erschließen sich so einige Anwendungsgebiete vor meinen Augen, warten wir es also nicht nur ab!

Martin Meyer (1000)
Redaktion

Division ohne Grenzen

Kurzbeschreibung des Programms „Abdivision"

Dieses Programm führt eine Division natürlicher Zahlen durch, bei der

1. Dividend und Divisor je BELIEBIG viele Ziffern haben dürfen – wobei

2. Dividend und Divisor in anderen Zahlensystemen als dem Dezimalsystem dargestellt sein können.

3. Wahlweise werden der Rest oder die Nachkommastellen bestimmt, falls die Division nicht aufgeht.

4. Der Algorithmus für diese Division verwendet nur die vier Grundrechenarten und keine Stringfunktionen, sodaß auch programmierbare Taschenrechner mit ihm arbeiten.

Erläuterung des Algorithmus.

Die Ziffern von Dividend und Divisor werden in Gruppen zusammengefaßt. Damit wird im Grunde ein anderes Zahlensystem als das übliche Dezimalsystem mit der Basis B=10 verwendet. Wenn man etwa in der Zahl 315 864 302 die Ziffern so zusammenfaßt (von hinten jeweils vier): 3|1586|4302, so kann man sie als Darstellung im Zahlensystem mit der Basis B=10000 auffassen. Die „Ziffern" dieser Zahl sind 3;1586; 4302.

Umständliche Formulierungen werden vermieden und die Begriffe werden geklärt, wenn das Wort Ziffer seine übliche Bedeutung behält, und die „Ziffern" in vom Dezimalsystem verschiedenen Zahlungssystemen anders, etwa mit Zuffern, bezeichnet werden. Für die Zuffern werden hier aber nicht (wie sonst bei B=16) neue Zeichen erfunden, sondern die Zuffern werden mit Hilfe der Ziffern dezimal dargestellt. So wären etwa für B=23 die Zuffern 1; 2; 3; ...;22.

Wählt man Systeme mit Basiszahlen, die Potenzen von 10 sind, so kann man die in ihnen dargestellten Zahlen „dezimal" lesen. Das vereinfacht sehr den Umgang mit ihnen. Deshalb wird der Anwender dieses Programms im Normalfall

1. B als Zehnerpotenz und
2. B so groß wie möglich wählen, um möglichst viele Ziffern zusammenzufassen. Damit wird die Verarbeitungszeit bei gegebener Ziffernzahl erhöht.

B=100000 wird auf den meisten Rechnern möglich sein. Das Produkt zweier Zuffern ist ja höchstens zweizuffrig, sodaß auch das Produkt $(B-1)^2$ (also das Quadrat der größten Zuffer unter $B=10^5$) noch kleiner als 10^{10} ist und ganzzahlig verarbeitet werden kann.
Abweichend vom Normalfall kann jede Basis B mit $2 <= B <= 10^5$ verwendet werden, aber natürlich mit Zuffern $0 <= Zuffer <= B-1$.

Für die folgende Erläuterung lege ich der Einfachheit halber die Basis B=10 zugrunde, aber sie gilt für jede zulässige Basis. Deshalb wird auch durchgängig das Wort Zuffer verwendet.

Die Herleitung ist nicht streng mathematisch. Die an sich einfachen Zusammenhänge werden an Hand von Beispielen dargelegt.

Der Dividend muß mindestens zweizuffrig und ≥ als der Divisor sein.

Die Zuffern von Divisor und Dividend werden in dieser Reihenfolge in ein Feld A% eingelesen. Die letzte Zuffer des Divisors und die erste Zuffer des Dividenden sind durch zwei leere Speicherplätze getrennt. Im Verlauf der Rechnung bleibt der Divisor erhalten, der Dividend wird nach und nach vom Quotienten überschrieben und geht verloren. Diese sparsame Verwendung von Speicherplatz sollte ermöglichen, den Algorithmus ohne große Umstände auf programmierbare Taschenrechner zu übertragen. Selbstverständlich ist es nicht schwer, den Algorithmus so abzufassen, daß Dividend, Divisor und Quotient eigene Felder erhalten.

Der Algorithmus im Überblick:

Beispiel 1.	78503:249 Nr. = 3 Zahl der Divisorzuffern Nd = 5 Anzahl der Dividendenzuffern.
2490078503 Feld A%	Es wird zuerst untersucht, ob der Divisor <= der Zahl ist, die aus den ersten Nr. Zuffern des Dividenden gebildet wird. Ist das der Fall, wie hier (249 <= 785), so wird der Teilquotient zwei Stellen links von der ersten Zuffer des Dividenden hineinaddiert, sonst eine Stelle links. Das Feld A% sieht dann so aus:
2493078503 – 6 2493018503 – 12 2493006583 – 36	Nun werden die Teilprodukte 3*2, 3*4 und 3*9 nacheinander von der 7, von der 8 und von der 5 subtrahiert. Wie der Teilquotient berechnet und wie die Subtraktion im Einzelnen durchgeführt wird, erfahren Sie weiter unten. Hier genügt es, zunächst ,,herkömmlich" zu subtrahieren.
2493002983	Jetzt geht es wieder von vorne los. Der ,,reduzierte" Dividend ist 2983.

Der Algorithmus im Detail:

Zur Berechnung des Teilquotienten müßten eigentlich alle Zuffern des Divisors und entsprechend viele des Dividenden verwertet werden. Das geht aber nicht, weil – bei $B=10^5$ – nur aus zwei Zuffern gebildete Zahlen ganzzahlig verarbeitet werden können. Deshalb muß der Algorithmus mit weniger Information auskommen.

Der Teilquotient wird berechnet, indem zunächst die Zahlen durcheinander dividiert werden, welche sich je aus den beiden ersten Zuffern des Dividenden und des Divisors ergeben. Muß der Teilquotient zwei Stellen weiter links neben der ersten Zuffer des Dividenden hineinaddiert werden (s. oben), so wird vom Ergebnis der Division sofort der ganzzahlige Anteil genommen, sonst erst nach Multiplikation mit der Basis B. Im Beispiel 1. ist der Teilquotient Int (78/24)=3.

Beispiel 2.	33785 : 345
3450033785 Feld A%. 3450933785 – 27 345090685 – 36 345090325 – 45 345090280	345 > 337! Teilquotient = Int(33*B/34) = 9 Die Teilprodukte 9*3, 9*4 und 9*5 werden subtrahiert: Der ,,reduzierte" Dividend ist jetzt 280.

Der so gebildete Teilquotient kann zunächst zu klein oder zu groß sein. Aber eine Abschätzung, die ich hier nicht durchführe zeigt, daß der Fehler bei der oben beschriebenen Berechnung des Teilquotienen für jede Basis klein, genauer: von der Größenordnung 1 ist und deshalb leicht korrigiert werden kann. (Dieser Sachverhalt macht den Algorithmus erst möglich.)

Der Teilquotient ist zunächst zu klein.

Beispiel 3.	1080 : 120
120001080 Feld A%	Teilquotient Int(10*B/12)=. (Richtig ist aber 9).
120081080 – 8 120080280 – 16 1200800120 – 0 120080120	Die Teilprodukte 8*1, 8*2 und 8*0 werden wie sonst subtrahiert. Danach ist der reduzierte Dividend 120. 120 <= 120, Deshalb wird der neue Teilquotient Int (12/12)=1 zwei Stellen links von der 1 – wie oben dargestellt – hineinaddiert und
120090120 – 1 120090020 – 2 120090000 – 0 120090000	die Teilprodukte 1*1, 1*2 sowie 1*0 subtrahiert: Damit hat sich ,,von allein" der richtige Teilquotient ergeben.

Der Teilquotient ist zunächst zu groß

Beispiel 4.	11022 : 111
11100111022 Feld A%	111>110, deshalb Teilquotient Int(11*B/11)=10=B, wo doch nur Zuffern erlaubt sind, die <B sind. In diesem Fall wird der Teilquotient sofort um 1 erniedrigt.
Beispiel 5.	42093 : 109
1090042093 Feld A% 1094042093 – 4 1094002093 – 0 1094002093 – 36 geht nicht ...	109<=420, deshalb Teilquotient Int(42/10)=4. Wie sich herausstellen wird, ist aber 3 richtig. Zuerst werden wieder die Teilprodukte 4*1, 4*0 und 4*9 subtrahiert: ... also war der Teilquotient 4 zu groß. Er wird um 1 erniedrigt, und die zuviel subtrahierten Teilprodukte 1*1 und 1*0 werden wieder addiert:
+ 1 1093012093 + 0 1093012093 ...	 Jetzt hat das Feld dieselbe Gestalt, als wäre sofort der richtige Teilquotient 3 gefunden und die Teilprodukte 3*1 sowie 3*0 subtrahiert worden. Jetzt kann das noch fehlende Teilprodukt 3*9 subtrahiert werden:
– 27 1093009393	 Jetzt wird der nächste Teilquotient bestimmt usw.

Einzelheiten zur Subtraktion der Teilprodukte

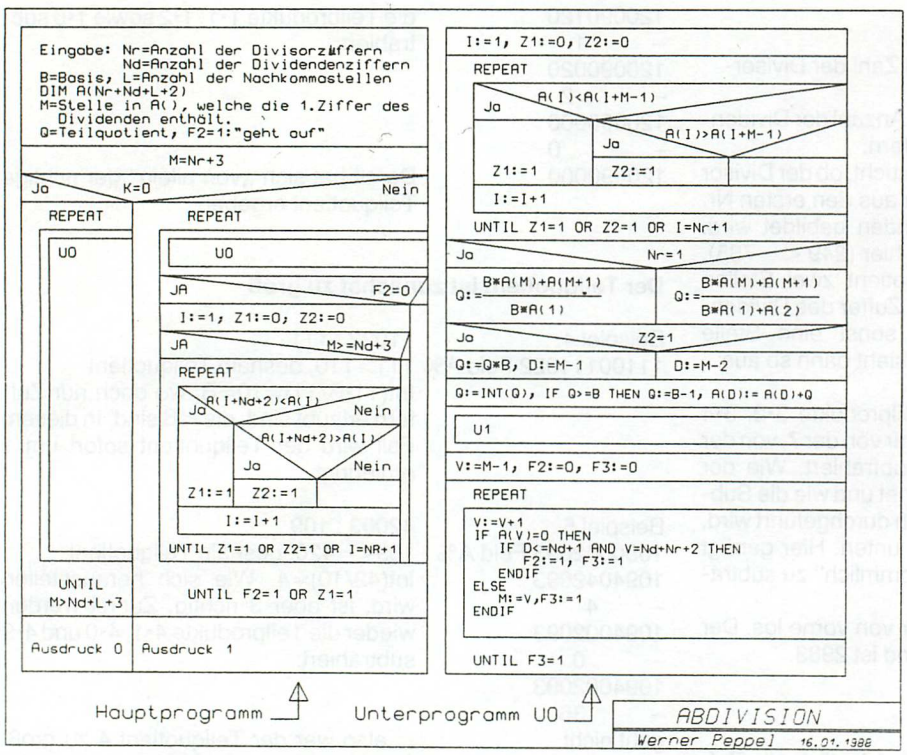
Zu behandeln ist nur der Fall, daß ein Teilprodukt nicht subtrahiert werden kann, weil die Differenz negativ wird.

a) Das Teilprodukt ist einzuffrig, z.B.: 2*3=6

...5 8 4 2 Feld A% – 6 ...5 8-2 2 + 10 ...5 8 8 2 Übertrag = 1 – 1 ...5 7 8 2	Ist die Differenz < Null, hier –2, so wird B addiert und ein Übertrag geschaffen, der von der vorhergehenden Stelle subtrahiert wird. Dabei kann sich wieder ein Übertrag ergeben. Dieser Fall wird unten besprochen.

b) Das Teilprodukt ist zweizuffrig, z.B.: 4*9=36

Der Rechner spaltet das Teilprodukt in zwei Zuffern, die ,,Zehnerzuffer" 3 und die ,,Einerzuffer" 6 auf. Dabei wird die ,,Zehnerzuffer" 3 sofort als Übertrag angesehen, der sich noch um 1 erhöhen kann, wenn nämlich die Subtraktion von 6 eine negative Differenz ergibt. Anschließend wird der Übertrag von der vorhergehenden Stelle subtrahiert, wobei sich wieder ein Übertrag – jetzt aber höchstens =1 – ergeben kann. Ist das der Fall, so wird geprüft, ob dieser Übertrag von weiter links stehenden Zuffern des Dividenden subtrahiert werden kann. Nur wenn das möglich ist, kann die Subtraktion des Teilprodukts abgeschlossen werden. Andernfalls war der Teilquotient zu groß und muß korrigiert werden. (s. oben).



Erläuterungen zum Struktogramm ABDIVISION

A: Gemeinsames Feld für Divisor und Divident

D: Stelle im Feld A, in die der Teilquotient hineinaddiert wird

J: Stelle in Feld A, von der die „Einer-Zuffer" des Teilprodukts subtrahiert wird

K=0 Nachkommastellen werden berechnet

K=1 Rest wird berechnet

Nd+1 Stelle in Feld A, hinter der das Komma steht

U Übertrag

U0 ist der Kern des Programms. Es bestimmt den richtigen Teilquotienten und subtrahiert alle nötigen Teilprodukte, bis (K=0) die Division aufgeht oder die gewünschte Zahl von Nachkommastellen berechnet worden ist bzw. bis (K=1) die Division aufgeht oder der letzte Rest kleiner als der Divisor ist.

In U0 wird zunächst festgestellt, ob der Divisor kleiner oder gleich der Zahl ist, die aus den ersten Nr. Zuffern des Dividenden gebildet wird. Ist das der Fall, wird der Teilquotient zwei Stellen links von der Stelle M (im Feld A) hineinaddiert, sonst eine Stelle links von M.

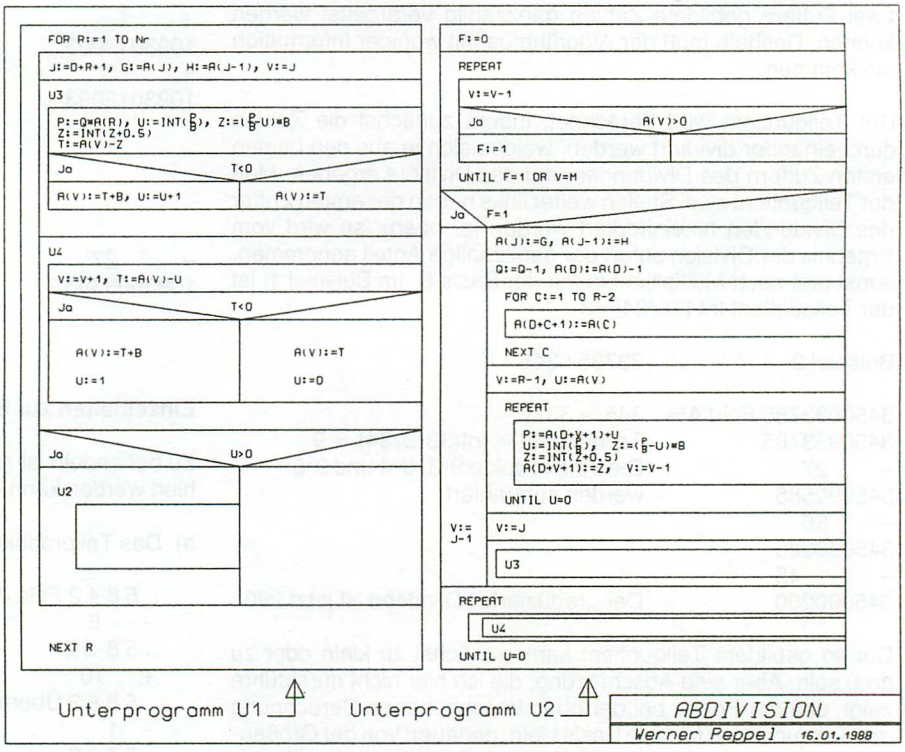
In **U1** werden die Teilprodukte subtrahiert.
Falls nötig wird der Teilquotient korrigiert.

In **U3** wird das Teilprodukt in die „Zehner–" und „Einerzuffer" zerlegt.
Die Einerzuffer wird subtrahiert.

In **U4** wird die „Zehnerzuffer" subtrahiert.

U2 Falls ein Übertrag U verbleibt, ist die Subtraktion des Teilprodukts noch nicht abgeschlossen. Es wird geprüft, ob sie überhaupt möglich ist. Dazu wid festgestellt, ob links von der Stelle J-1 noch Zuffern des Dividenden stehen. Andernfalls war der Teilquotient zu groß und muß korrigiert werden.

Ps.: Die Strukturprogramme sind zwar nicht ganz normgerecht, dafür aber mit viel Liebe erstellt.



Werner Peppel
An der Halde 3
7325 Boll

Hinweis

Den dritten Teil meiner Serie „Barcodes, was ist das", habe ich einfach mangels Zeit nicht mehr fertigbekommen, den gibt's im nächsten Heft. Dies betrifft auch die zu diesem Thema angekündigte Buchbesprechung zu dem Buch Barcodes mit dem HP-IL System. Als Ersatz für diese Beiträge habe ich Marys II und den Bericht zum HEPAX-Modul fertiggestellt, das ist doch auch was.

Martin Meyer (1000)
Redaktion

Testbericht des HP-41CY

Testbericht: „HP-41CY TURBO"

„Der neue HP-41CY mit 64kRAM!" Diese Anzeige in PRISMA 2/87 klang eigentlich sehr vielversprechend. Darum entschloß ich mich, mit der Fa. W&W Software Products Kontakt aufzunehmen, was zur Folge hatte, daß ich kurz darauf die Bestellung eines HP-41CY einreichte. Nach etwa dreimonatiger Wartezeit, wegen der hohen Nachfrage und einigen angeblichen Schwierigkeiten beim Rechnerumbau, wurde mir Anfang Oktober 1987 endlich der Taschenrechner zugesandt. Der Rechner wurde in der HP-41CX Verpackung geliefert, in der sich zusätzlich zu den Original HP-41CX-Handbüchern noch ein RAMBOX-Handbuch (im DIN A5-Format) sowie eine Zusatzanleitung für die Rechner HP-41CW und HP-41CY (vier DIN A4-Blätter) befanden. Anscheinend war es zuviel Arbeit, ein dem Rechner angemessenes Handbuch zur Beschreibung der Zusatzfunktionen mitzuliefern.

Der HP-41CY, ein von der Firma W&W Software Products getunter HP-41CX, unterscheidet sich äußerlich nur durch einen Mikro-Schalter an der linken Gehäusewand des Rechners, der zur Umschaltung der Rechengeschwindigkeit dient, sowie durch ein mitgeliefertes Staubschutzoverlay vom HP-41CX. Seine wesentlichen Verbesserungen gegenüber dem Grundmodell sollen im folgenden kurz beschrieben werden.

Der TURBO-Modus:

Durch Umschalten des Mikro-Schalters in den TURBO-Modus wird die Rechengeschwindigkeit verdoppelt, was alle Betriebsarten des Rechners ohne Einschränkung betrifft. Ausschalten muß man den TURBO-Modus nur beim Aufzeichnen von Magnetkarten, da diese sonst nicht mehr lesbar sind. Wegen des doppelt so hohen Stromverbrauchs bei eingeschaltetem TURBO-Modus sollte der Rechner nicht mit dem HP-Akku betrieben werden, weil die Kapazität dieses Akkus zu gering ist, sondern entweder mit den im Lieferumfang enthaltenen Batterien (Lebensdauer von etwa drei Monaten) oder dem von W&W Software Products angebotenen Akku.

Der Zusatzspeicher:

Der HP-41 kann allgemein einen maximalen ROM-Speicherbereich von 64 kByte verwalten. Diese 64kByte unterteilen sich in sechzehn 4kByte Adressblöcke (im folgenden auch Pages genannt), die von hex 0 bis hex F durchnumeriert sind. Davon sind die Adressblöcke hex 0 - 7 für das HP-41 Betriebssystem und einige interne Erweiterungen reserviert. Mit den Adressblöcken hex 8 - F werden die Modulschächte PORT 1 bis PORT 4 verwaltet, womit maximal 32kByte ROM-Speicher in Form von Anwender-Modulen (CCD-ROM, Advantage-ROM, Math-Modul, usw.) zur Verfügung stehen. Diese Adressblöcke werden auch vom Zusatzspeicher des HP-41CY genutzt. Steckt man ein ROM-Modul in den Rechner, so wird gleichzeitig jener Teil des Zusatzspeichers abgeschaltet, dessen Adressblock identisch mit dem des Modules ist. Das trifft aber nicht für die Verwendung des HP-IL Modules und der X-MEMORY Module zu, da diese gänzlich anders verwaltet werden und deshalb in diesem Sinne keinen PORT belegen.

Wie oben angegeben, kann der HP-41 nur 32kByte Modulspeicher verwalten. Durch sogenanntes „page-switching" (Befehl „PG <>") besteht beim HP-41CY die Möglichkeit, die Speicherkapazität zu verdoppeln, indem einfach zwischen zwei eingebauten 32kByte-RAMBOXEN hin- und hergeschaltet wird, d.h. es ist nicht möglich, den gesamten 64kByte Speicher gleichzeitig zur Verfügung zu haben, sondern jeweils nur einen 32kByte Block bzw. acht Pages. Diese Einschränkung wird jedoch durch geschickte Einteilung der Pages unter Verwendung der Befehle „PG <>", „PG01" und „PG10" umgangen. Dadurch besteht die Möglichkeit, Programme und Daten sowohl von der RAMBOX 0 als auch von der RAMBOX 1 zur Verfügung zu haben.

Zur einwandfreien Programmverwaltung muß bei der Vergabe der XROM-Nummern an die einzelnen Pages darauf geachtet werden, daß diese nicht mit den XROM-Nummern der in Verwendung befindlichen Module übereinstimmen, da sonst Konflikte auftreten.

Es können Programme, Daten, Tastenbelegungen und Buffer-Files in die initialisierten Pages des Zusatzspeichers geladen werden. Die Programme sind dann, im Gegensatz zum X-Memory Modul, innerhalb des Speichers lauffähig, womit der Hauptspeicher des HP-41 (319 Register) für Daten und andere Programme freibleibt. Es besteht sogar die Möglichkeit, komplette Module mit einer Massenspeichereinheit zu laden und im Zusatzspeicher abzulegen. Zur Verwaltung des Zusatzspeichers befindet sich für RAMBOX 0 und RAMBOX 1 in Page 8 je ein eigenes Betriebssystem, das nicht gelöscht werden kann. Dadurch werden zwei Pages eingebüßt, was aber bei der Größe des Speichers sicher nicht ins Gewicht fallen dürfte.

Der Zusatzspeicher wird auch bei Durchführung eines „MEMORY LOST" nicht gelöscht, sofern man die nachfolgende Frage „CLR RAMBOX?" nicht mit „J" oder „Y" beantwortet.

Befehle zur Speicherverwaltung:

1. Allgemeine Hilfsfunktionen:

„BUFLNG?"
(Buffer Length)
ermittelt die Anzahl der in einem I/O-Buffer belegten Register

„KEYAS?"
(No. of KEY-Assignment Register)
Berechnet die Anzahl der durch Tastenzuordnungen belegten Register

„PG?"
(Page Contents)
ermöglicht programmgesteuert Angaben über den Inhalt eines beliebigen 4kByte Blockes zu erhalten

„FNC?"
(Description of Function)
ermöglicht programmgesteuert Informationen über einzelne Funktionen einer Page zu gewinnen

„XQ>XR"
(Transform XEQ to XROM)
wandelt alle XEQ-Befehle in einem Programm, zu denen ein entsprechendes globales Alpha-LBL in einem ROM oder im erweitertem RAM-Speicher gefunden wird, in einen XROM-Befehl um

„CRDIR"
(Create Direktory Entry)
erlaubt den Zugriff auf den gesamten Speicherbereich einer Diskette für das HP-9114 Diskettenlaufwerk

„SETPRV"
(Set Private)
ermöglicht es, einem Programm auch ohne den Umweg über ein externes Speichermedium, den Private-Status zu verleihen

2. Verwaltungsfunktionen:

„CLPG"
(Clear Page)
erlaubt es, einen 4kByte Block vollständig zu löschen

„INITPG'
(Initialize Page)
eröffnet eine Page, mit der bis zur Ausführung von „ENDPG" gearbeitet werden kann

„FRBYT?"
(Free Bytes in Page)
gibt die Anzahl der noch zur Verfügung stehenden Bytes in einer Page an

„COPYPG"
(Copy Page)
damit läßt sich der Inhalt einer Page in eine andere Page des Speichers kopieren

„WRTPG"
(Write Page)
erlaubt es eine beliebige Page auf einer Massenspeichereinheit aufzuzeichnen

„READPG"
(Read Page)
dient zum Einlesen eines auf einer Massenspeichereinheit aufgezeichneten 4kByte-Blockes
„PGSUM"
(Pagesum)
berechnet die Checksum einer Page
„ENDPG"
(End Page)
schließt die Bearbeitung einer Page ab
„PG<>"
(Page switching)
schaltet jene RAMBOX komplett ein, deren Betriebssystem ausgeschaltet war
„PG01"
(Page switching 0,1)
schaltet die geraden Pages der RAMBOX 0 und die ungeraden Pages der RAMBOX 1 ein
„PG10"
(Page switching 1,0)
schaltet die geraden Pages der RAMBOX 1 und die ungeraden Pages der RAMBOX 0 ein

Anmerkung der Redaktion:
Nach Erfahrung der Redaktion läßt sich das CCD Modul **nicht** mit dem Befehl WRTPG auf dem Digital Cassettenlaufwerk speichern.

3. Filefunktionen

„LDPGM"
(Load Program)
damit lassen sich beliebige Anwenderprogramme in eine Page laden
„CLLSTFL"
(Clear last File)
löscht das jeweils letzte in der Page befindliche Programm
„CRFLDTA"
(Create Data-File)
legt in der spezifizierten Page einen Datenfile an
„LDREG"
(Load Registers)
kopiert alle vorhandenen Datenregister in einen Datenfile
„LDREGX"
(Load Registers by X)
kopiert einen im X-Register festgelegten Datenregisterblock in einen Datenfile
„LDREGXY"
(Load Registers by X and Y)
kopiert einen im X-Register festgelegten Datenregisterblock ab dem im Y-Register angegebenen Datenregister in einen Datenfile
„GTREG"
(Get Registers)
stellt die Umkehrfunktion zu „LDREG" dar
„GTREGX"
(Get Registers by X)
stellt die Umkehrfunktion zu „LDREGX" dar
„GTREGXY"
(Get Registers by X and Y)
stellt die Umkehrfunktion zu „LDREGXY" dar
„CLRFL"
(Clear File)
damit läßt sich der Inhalt eines Datenfiles komplett löschen
„CRFLKEY"
(Create KEY-File)
legt ein KEY-File zur Ablegung von Tastenbelegungen an
„LDKEY"
(Load KEY-Assignments)
überträgt die zur Zeit gültigen Tastenzuordnungen in den spezifizierten KEY-File
„GTKEY"
(Get KEY-Assignments)
löscht die Tastenzuordnungen und kopiert die im KEY-File abgespeicherten Tastenzuordnungen in den Rechner
„CRELBUF"
(Create Buffer-File)
legt einen Buffer-File in der Page an
„LDBUF"
(Load Buffer)
überträgt den Inhalt eines I/0-Buffers in einen BUFFER-File
„GTBUF"
(Get Buffer)
kopiert den Inhalt eines BUFFER-Files in einen I/0-Buffer
„PECT"
(Protect a File)
schützt File gegen unbeabsichtigtes Löschen
„UNPTCT"
(Unprotect a File)
hebt den Fileschutz wieder auf

Garantie und Service-Leistungen:

Durch Umrüstung eines original HP-Rechners in einen HP-41CY geht die Garantie der Firma Hewlett Packard auf diese Rechner verloren! Im gleichen Umfang übernimmt aber die W&W Software Products GmbH diese Garantie. Eine Reparatur in einem Hewlett Packard Service Center ist für diesen Rechner nicht möglich.

Da ich vor dem Kauf dieses Rechners einen HP-41CV besaß, sind auch mir die vom RAM-Speicher gesetzten Grenzen des HP-41 bekannt, die vor allem bei Bearbeitung großer Datenmengen, wie zum Beispiel der Berechnung komplexer Matrizen, bald erreicht werden. Seitdem ich jedoch den neuen Rechner verwende, weiß ich erst, was man alles speichern kann, ohne sich darüber den Kopf zerbrechen zu müssen. Außerdem wird der Motor des Magnetkartenlesers nur mehr selten einem Dauerbetrieb ausgesetzt.

Ich finde, der HP-41CY ist der ideale Taschenrechner für all jene, die zur Bearbeitung ihrer Programme zusätzlich externe Speichermedien, z.B. Kassetten- oder Diskettenlaufwerk, benötigen, da einerseits der langwierige Lade- und Speichervorgang weitgehend entfällt und andererseits die Programme durch die erhöhte Rechengeschwindigkeit, vor allem bei iterativen Berechnungen, in wesentlich kürzerer Zeit ablaufen. Im großen und ganzen stellt der HP-41CY für mich eine gelungene und sinnvolle Erweiterung der 41er Serie dar.

Sollten sich Fragen ergeben, werde ich diese gerne beantworten.

Günter Krainz (CCD 3025)
Papiermühlgasse 36/45
A-8020 Graz

Lamé'sche Reihe

20 Zeilen, 41 Bytes, HP-41C, Drucker

Gabriel **Lamé** * 22.07.1795 in Tours, † 1.05.1870 in Paris, Französicher Mathematiker, 1820-31 Professor in Petersburg, dann in Paris. Mathematische Physik, insbesondere Potentialtheorie.

Jeweils die Summe zweier benachbarter Zahlen ergibt die nächste.

Durch Zeile 5 (mit 6 und 7) wird verhindert, daß die Zahlen am Zeilenende zerrissen und nicht als Ganzes ausgedruckt werden (s. Literatur).

Das Interessante an dieser Reihe ist, daß sich das Verhältnis benachbarter Zahlen immer mehr dem „Goldenen Schnitt" nähert:

$$\frac{7778742049}{4807526976} = 1{,}618033989$$

$1 : x = x : (1+x)$

$x = 1{,}61.....$, s. oben

Nach Anhalten des Programmes für weitere Verwendung muß der Drucker entweder aus- und eingeschaltet werden, oder es muß
PRBUF oder
CLRDEV durchgeführt werden, damit der Druckbuffer entleert wird.

Zeile 5 (mit 6 und 7) kann auch für A-Daten verwendet werden, wenn sie durch Zwischenräume getrennt sind. Sie werden dadurch am Zeilenende nicht zerrissen.

SF17/OUTA/.....PRBUF oder
ACA/.....PRBUF.

Literatur: C.E. Reinstein, HP-41/HP-IL SYSTEM DICTIONARY, S.36, Corvallis Software INC. 1982, Stichwort PARSE MODE.

Wurde nach dem Programm PBC im CCD-Handbuch 7.40 ausgedruckt.
Zeile -5- scheint aber überflüssig zu sein, da der Rechner nach der 4. Zeile WORKING signalisiert.


```
01♦LBL "LAME"
02 FIX 0
03 CF 29
04 CLRDEV
05 "E&k1h"
06 OUTA
07 PRBUF
08 CLX
09 STO 00
10 SIGN
11 STO 01
12♦LBL 00
13 RCL 00
14 RCL 01
15 STO 00
16 +
17 STO 01
18 ACX
19 GTO 00
20 END
```

LAME

005:F5 1B 26 6B 31 68

LAME-sche Reihe

```
ΣX
 1 2 3 5 8 13 21 34 55
89 144 233 377 610 987
1597 2584 4181 6765
10946 17711 28657 46368
75025 121393 196418
317811 514229 832040
1346269 2178309 3524578
5702887 9227465 14930352
24157817 39088169
63245986 102334155
165580141 267914296
433494437 701408733
1134903170 1836311903
2971215073 4807526976
7778742049 1,+10 2,+10
3,+10 5,+10 9,+10 1,+11
2,+11 4,+11 6,+11 1,+12
```

Martin Hochenegger
Heidelberger Landstraße 97
6100 Darmstadt 13

Verbindung von HP-41 zu MS-DOS Rechnern

Besitzer eines PCs, der unter MS-DOS läuft, können aufatmen! Es gibt eine Möglichkeit, einen IL-Taschencomputer mit dem PC zu verbinden, bzw. IL-Peripheriegeräte vom PC her zu benutzen. Dies, man staune, von HP! Und erst noch für weniger als 400 DM!!

Wie das Wunder heißt? – HP 82973A HP-IL Interface Card.

Dies ist eine Schnittstellenkarte, die einen halben 8-Bit-Slot in einem Vectra oder Kompatiblen belegt. Eigentlicher Verwendungszweck ist, den HP-Portable + der keine Laufwerke besitzt, mit dem stationären PC zu verbinden. Mit der Karte wird eine 5¼'' Diskette mit Treiberprogrammen und Softmanual ausgeliefert.

Die Organisation läuft nach dem Master-Slave-Prinzip, d.h. ein Gerät tritt als IL-Controller und das andere als Device auf.

Für den HP-41 läßt sich der PC so als Videodisplay nutzen, oder man kann – via PC – einen Matrixdrucker ansteuern. Die Option „Laufwerk" wird leider nicht genutzt, da der 41er nicht MS-DOS „kann".

Wird beim PC in der Datei CONFIG.SYS der Treiber HPIL.SYS eingebunden, kann der PC als IL-Controller auftreten und z.B. den Thinkjet oder die 9114 Diskettenstation benutzen. Sogar der 82162 Thermodrucker und das 82161 Kassettenlaufwerk folgen willig dem großen Bruder (die Frage nach der Zweckmäßigkeit einer solchen Verbindung lasse ich offen).

Wieder ein Wermutstropfen: der HP-41 läßt sich so nicht ansprechen, da er selbst immer als Controller auftreten will.

Wo also liegt der Nutzen dieser Karte?

1. In der Standardanwendung: PC <> Portable +
2. Verwendung von IL-Peripherie am PC
3. Videointerface und Drucker für HP-41
4. **Spezielles:** Datenübertragung vom HP-41 > PC

Obwohl man nicht **direkt** die Laufwerke des PC nutzen kann, gibt es eine Möglichkeit, Daten vom 41er in eine Diskettendatei zu schaufeln! Dazu begeben wir uns etwas in die DOS-Niederungen und lassen das (unfreundliche aber leistungsfähige) Betriebssystem für uns arbeiten:

Die Symbole < und > haben für DOS eine spezielle Bedeutung. Mit ihnen kann man die Richtung des Datenflusses angeben. Für unseren Zweck wollen wir die (Bildschirm)Ausgabe umlenken. Dazu verwenden wir das Symbol >.

Wir rufen das HPLINK Programm mit der Befehlsfolge

CS80 > Beispiel.txt auf.

Auf dem Bildschirm erscheint nur noch der Cursor, der plötzlich eingefroren ist. Kunststück, haben wir doch die Bildschirmausgabe in der Datei Beispiel.txt „gefangen"!

Nun muß nur noch durch Eingabe einer „3" (die auch nicht angezeigt wird) auf „DISPLAY" umgeschaltet werden. Ab sofort wird jede Information, die über die IL kommt, in die Datei geschrieben. So lassen sich Programmlistings oder Text übertragen, den man nacher bequem auf dem PC mit der Textverarbeitung der Wahl bearbeiten kann.

Besitzt man gar ein Extended I/0 Modul, so kann man mit der Funktion OUTP Programmcode als Folge von ASCII-Zeichen senden und diese mit einem Programm auf dem PC in Barcodes umrechnen und auf einem beliebigen Drucker in Windeseile ausdrucken!

(Ein solches Programm ist zur Zeit in Arbeit, wird voraussichtlich aber noch eine Weile bis zur Vollendung benötigen.)

Ist die Übertragung abgeschlossen, tippt man auf der PC-Tastatur ein „e" und die „Umleitung" ist beendet.

Für Assembler-Programmierer auf dem PC werden noch Beispiel-Sources für die Programmierung von HP-IL Primitives mitgeliefert, die über den Interrupt 54h aufgerufen und für eigene Applikationen auf der IL-Schleife verwendet werden können.

Ich bin überzeugt, daß ich erst einen Teil der Möglichkeiten entdeckt habe und daß man mit einem „besseren" Controller (z.B. HP-71) und mehr Fachwissen noch mehr erreichen könnte!

Tips nimmt gerne entgegen:

A. Jakob
St. Georgen-Str. 219
CH-9011 St. Gallen

Utilities

1) **Zwei Werte in einem Register**
2) **NONEXISTENT**
3) **DATA ERROR**
4) **Vergleichsopertionen**

Zwei Werte in einem Register

Trotz der beachtlichen Registerzahl auch im erweiterten Speicher, der Rambox oder extern biete ich Euch hier ein Programm, mit dem Ihr jedes Register mit zwei Zahlen belegen könnt (Koordinaten, Buchhaltung usw.). Die beiden Zahlen werden im Y- und X-Register bereitgestellt (also mit ENTER eingegeben) und mit XEQ „CD" gestartet. Im Stack bleibt „Schrott" zurück und im Alpharegister 6 unleserliche Zeichen, die es aber „in sich haben". Sie können vom Benutzer dann mittels ASTO.. abgespeichert und über ARCL.. zu gegebener Zeit wieder geholt werden. Der umgekehrte Weg: Mit diesen Zeichen im Alpharegister startet man XEQ „DC" und die ursprünglichen Zahlen stehen wieder im Y- und X-Register. Bedingungen: Keine negativen Zahlen (es wurden vorsorglich zwei „ABS"s eingebaut) und keine Zahl darf größer als 16.451.579 in der FIX-0-Version (164.515,79 bei FIX 2) sein. Meine Routine arbeitet mit FIX 2 und rundet vorsichtshalber beide Werte.

Wie ist das möglich? Bekanntlich gibt es 255 Möglichkeiten, Zahlen in ASCII-Zeichen umzuwandeln. Da führende Nullen im Alpharegister ignoriert werden, wird mit 254 Möglichkeiten gearbeitet. Ich wandle die Zahl in ein 254iger System um. Wie oft steckt 254^2 in der umzuwandelnden Zahl? = 1. Ziffer. Wie oft steckt 254 im Rest? = 2. Ziffer und der dann verbleibende Rest ergibt die 3. Ziffer. Die Addition von 1 erfolgt, um den ASCII-Code 00 zu vermeiden. Jede Zahl läßt sich so in drei Alphazeichen umwandeln. Die zweite Zahl wird dann ebenso umgewandelt und einfach angehängt. Sechs Alphazeichen faßt bekanntlich ein Register. Die größte umzuwandelnde Zahl in der ganzzahligen Version ergibt sich aus $254^3 + 254^2 - 1 = 16.452.579$. Gibt man eine zu große Zahl ein, kommt DATA ERROR. Auch bei der Rückumwandlung bleibt im Stack „Schrott" zurück, es wird jedoch alles im Stack „herumjongliert", also keine Register benötigt.
Diese Routine ist eine Gemeinschaftsarbeit zusammen mit meinem Kollegen Peter Göddel aus 6721 Gommersheim.

```
01*LBL "CD"
02 FIX 2
03 CLA
04 ABS
05 STO 00
06 RDN
07 ABS
08 XEQ 00
09 RCL 00
10*LBL 00
11 RND
12 2
13 10↑X
14 *
15 2
16*LBL 01
17 254
18 RCL Y
19 Y↑X
20 ST/ T
21 R↑
22 INT
23 *
24 ST- Z
25 RDN
26 LASTX
27 1
28 +
29 XTOA
30 RDN
31 DSE X
32 GTO 01
33 X=0?
34 GTO 01
35 RTN
36*LBL "DC"
37 XEQ 02
38 STO 00
39 XEQ 02
40 RCL 00
41 X<>Y
42 STOP
43*LBL 02
44 254
45 STO Y
46 X↑2
47 ATOX
48 1
49 -
50 *
51 RCL Y
52 ATOX
53 1
54 -
55 *
56 +
57 ATOX
58 1
59 -
60 +
61 2
62 10↑X
63 /
64 END
```

NONEXISTENT

Ich habe die Programmiergewohnheit, gleich mit einem Wert im X-Register per Taste einzusteigen. Als Beginn eines Programms bietet sich wohl allgemein die Taste A an. Nun kommt es beim besten Routinier mal vor, daß er vergaß, das USER-Tastenfeld einzuschalten. Was passiert? Die Register 11 bis 16 sind kaputt! Abhilfe für Nichtstatistiker (wohl die meisten von Euch): Verschiebt von Anfang an die Statistikregister hinter den SIZE-Vorhang, dann erzeugt der oben erwähnte Tastendruck NONEXISTENT aber er richtet keinen Schaden an. Beispiel:

SIZE 317	= Registerzuweisung 0 - 316
309 Σ REG IND X	= Stat. Reg 309 - 315
SIZE 309	= Registerzuweisung 0 - 308.

Dieser Vorschlag hat noch einen Vorteil für Nichtstatistiker: Wenn man aus irgendeinem Grund in seinem Programm eben diese Meldung „NONEXISTENT" erzeugen möchte, genügt der Befehl Σ+. Die Programmausführung bleibt dann auch dort hängen und läßt sich mit R/S nicht mehr weiterbewegen – nur mit einem Labelaufruf woanders hinziehen.

DATA ERROR

Will man in einer Schleife absichtlich ein DATA ERROR erzeugen, bei dem man aber sofort mit einem neu eingegebenen Wert per R/S weiterfahren kann, so bietet sich an

– CLX $\frac{1}{x}$ $\frac{1}{x}$ (X-Register geht verloren)
oder
– 0 $\frac{1}{x}$ $\frac{1}{x}$ (Stack wird angeboten).

Obwohl kein Stop programmiert ist, hält das Programm mit DATA ERROR an, weil der Kehrwert von 0 unendlich groß ist. Mit dem dann richtig eingegebenen Wert (außer 0!) kann man mit R/S starten. Mit

– 1 CHS $\sqrt{x}$ x^2 (Stack wird angehoben)

kann man zwar 0 verarbeiten, aber keine negativen Neueingaben.

Vergleichsoperationen

Die fehlenden Vergleichsoperationen werden wie folgt umschifft:

X>=0? entspricht X≠0? X>0? und
X>=Y? entspricht X≠Y? X>Y?

Sollen zwei Bedingungen mit **und** verbunden werden, so muß die zweite Abfrage verneint gestellt werden.

Beispiel:
Wenn X = 0 **und** Flag 5 gesetzt ist, soll nach 01 verzweigt werden:
X = 0 ? FC? 05 GTO 00 GTO 01 LBL 00.

Sollen zwei Bedingungen mit **oder** verbunden werden, so muß die erste Abfrage verneint gestellt werden.
Beispiel:
Wenn X ≠ 0 **oder** Flag 5 gesetzt ist, soll nach 01 verzweigt werden:
X ≠ 0 ? FS? 05 GTO 01.

Jürgen Schatz
Carl-Theodor-Straße 12
6710 Frankenthal

Das HEPAX-Modul

Und wieder ist ein neuer Stern am Himmel des HP41 aufgegangen, ein neues Modul ist uns erschienen, was mag es wohl verheißen?

Klingt wie Weihnachten, das wäre es auch beinahe geworden, der mir überlassene Prototyp funktionierte leider auf Anhieb schon einmal garnicht, Frust. Was tun, sprach Zeuss und schickte das begehrte Stück wieder nach Copenhagen zurück. Das Modul kam Mitte Januar wieder, diesmal „voll" funktionstüchtig, leider zu spät für die Februarausgabe von PRISMA.

Nun erst einige einleitende Worte zum Thema HEPAX-Module, dies ist nämlich nicht nur ein einziges:

Die **HE**wlett **PA**ckard e**X**tension Module bestehen aus

Standard HEPAX-Modul,	(mit 8k RAM)
Erweitertes HEPAX-Modul,	(mit 16k RAM)
HEPAX Memory-Modul,	(8k RAM)
HEPAX Doppel-Memory-Modul.	(16k RAM)

Es handelt sich hier also um ein Modulsystem zur Speichererweiterung des HP41 in größerem Stil als bislang gewohnt. Zusätzlich ist das HEPAX-Modul eine MLDL-Einheit, mit der man mit Hilfe eines eingebauten Editors ebenso wie eines Disassemblers eigene Funktionen für den HP41 in Maschinensprache schreiben kann, das ist etwa dasselbe wie die LEX-Files des HP71 (**L**anguage **EX**tension Files = Spracherweiterungen).

Durch den auf 64kByte begrenzten Adressraum des HP41 und die maximale Anzahl von 10 Buffern im RAM des Rechners kann auch nicht immer die maximal mögliche RAM-Größe erreicht werden, dazu ein Beispiel vorweg:

Das erweiterte HEPAX-Modul benötigt 5 (3 bei Standart) Buffer, das HP-IL Modul 2, bleiben noch 3 übrig.

Ein HEPAX Doppel-Memory braucht 4 Buffer, sein kleiner Bruder nur 2, den kann ich also noch dazustecken. Das TIME-Modul paßt also mit 1 gerade noch in die 10 verfügbaren Buffer Man ist hier also einigen Einschränkungen unterworfen, Tabelle 1 zeigt die von Modulen benötigten Buffer.

Modul	benötigte Buffer
HP-IL Drucker en.	2
HP-IL Drucker dis.	1
HP 41 Memory	0
X-Memory	0
TIME	1
HP82143A Drucker	1
HP82242 IR Drucker-Modul	1
Kartenleser	1
Barcodeleser	1
4k Applikat. Modul	1
8/12k Applik. Modul	2
Standart HEPAX	3
Erweitertes HEPAX	5
HEPAX Memory	2
HEPAX Doppel-Memory	4

Tabelle 1: Bufferzuweisung für HP41 Peripheriemodule

Ich hoffe das genügt als Entscheidungshilfe.

In der folgenden Aufzählung der Befehle des HEPAX-Moduls werde ich zu jedem Befehl einen kurzen Kommentar über dessen Funktion abgeben, das ist wahrscheinlich die schnellste und effizienteste Methode den Leistungsumfang dieses Moduls darzustellen. Ein Teil der Befehle ist sowieso aus dem X-Function Modul bekannt, da das HEPAX-Modul seinen Speicher logisch genauso in Files organisiert wie das X-Function Modul mit dem Unterschied, daß auch **W**rite **A**ll Files und Key-Assignment-Files (Tastenzuweisungen) wie auf dem Kassettenlaufwerk oder der Floppy möglich sind, sichern von Files gegen Überschreiten oder Löschen ist ebenfalls möglich!

Der Speicher des HEPAX-Systems ist gegen MEMORY LOST geschützt, Programme können außerdem noch in diesem direkt gestartet werden, das hört sich recht gut an. Beim X-Function Modul muß man ja die darin abgelegten Programme erst einmal wieder in den Hauptspeicher holen, um sie auszuführen. Dies ermöglicht natürlich wesentlich längere und somit in der Regel komfortablere Programme, man muß ja sonst immer so mit Text geizen, da dieser zu viel Speicherplatz gefressen hat. Die Anzahl der maximal möglichen Datenregister ändert sich dadurch allerdings nicht, es stehen aber natürlich im Extremfall alle 319 Datenregister für die Speicherung von z.B. Matrizen zur Verfügung.

Befehl	Erläuterung
Allgemeine Funktionen	
HCLFL	löscht den Inhalt eines Daten- oder Textfiles
HEPDIR	Anzeige des Inhaltsverzeichnisses des HEPAX-Memories
HEPDIRX	Anzeige einer Fileinformation des Inhaltsverzeichnisses des HEPAX-Memories
HEPROOM	Anzeige des freien Speicherplatzes im HEPAX-Memory
HFLSIZE	Anzeige der Filegröße
HPURFL	löscht einen File im HEPAX-Memory ohne den Fehler der ersten X-Function Module, die den gesamten X-Memory Bereich gelöscht haben, wenn man nach dem Löschen mit PURFL nicht den Pointer mit z.B. EMDIR neu gesetzt hat
HRCLPT	holt den Zeiger des aktuellen Daten/Textfiles
HRCLPTA	holt den Zeiger eines beliebigen Daten/Textfiles
HREADFL	kopiert einen beliebigen File von einem IL Massenspeicher in das HEPAX-Memory, das HEPAX Filesystem benutzt genau dieselben Formate wie die Files auf IL Massenspeichern wie z.B. dem Kassettenlaufwerk. D.h. das HEPAX-Memory ist wie eine RAM-Disk im HP41 selbst!
HRENAME	Benennt eine File im HEPAX-Memory um
HRESZFL	variiert die Größe eines bereits existierenden Files im HEPAX-Memory, man muß also nicht mehr im voraus genau wissen, wie groß ein Daten/Textfile sein soll, man kann ihn mit Hilfe dieser Funktion jederzeit vergrößern, verkleinern natürlich auch
HSEC	sichert eine File im HEPAX-Memory gegen Überschreiben oder Löschen
HSEKPT	setzt den Zeiger des aktuellen Daten/Textfiles im HEPAX-Memory
HSEKPTA	setzt den Zeiger eines beliebigen Daten/Textfiles im HEPAX-Memory
HUNSEC	hebt die Sperre von HSEC wieder auf
HWRTFL	kopiert jeden File im HEPAX-Memory auf ein IL Massenspeichermedium
Datenfilefunktionen	
HCRFLD	erzeugt einen Datenfile im HEPAX-Memory
HGETR	liest alle Daten eines Datenfiles im HEPAX-Memory
HGETRX	liest bestimmte Daten eines Datenfiles im HEPAX-Memory
HGETX	liest ein Datum des aktuellen Datenfiles in's X-Register
HSAVER	speichert die Datenregister des HP41 in einen HEPAX Datenfile
HSAVERX	speichert bestimmte Datenregister des HP41 in einen HEPAX Datenfile
HSAVEX	speichert den Inhalt des X-Registers in einen HEPAX Datenfile

TEXTFILEFUNKTIONEN

HAPPCHR hängt die im ALPHA-Register befindlichen Buchstaben an den aktuellen HEPAX Textfile an

HAPPREC hängt die im ALPHA-Register befindlichen Buchstaben an den aktuellen HEPAX Textfile als neuen Record an

HARCLRC hängt einen Record aus einem HEPAX Textfile an den Inhalt des ALPHA-Register

HASROOM gibt die Anzahl der unbenutzten Bytes eines HEPAX Textfiles zurück

HCRFLAS erzeugt einen neuen HEPAX ASCII-File (Textfile). Dieser hat genau dasselbe Format und denselben Aufbau wie die ASCII-Files des X-Funktion Moduls. Man kann also einen ASCII-File im X-Memory mit SAVEAS auf Kassette speichern und wieder mit HREADFL in das HEPAX Memory holen, es ist alles kompatibel.

HDELCHR löscht ein oder mehrere Textzeichen aus dem aktuellen HEPAX Textfile

HDELREC löscht im aktuellen HEPAX Textfile den Record, auf den gerade der Zeiger zeigt

HGETREC liest einen Record aus dem aktuellen HEPAX Textfile in das ALPHA-Register

HINSREC fügt die im ALPHA-Register befindlichen Textzeichen in den aktuellen HEPAX Textfile ein

HINSREC fügt die im ALPHA-Register befindlichen Textzeichen als neuen Record in den aktuellen HEPAX Textfile ein

HPOSFL sucht einen String im aktuellen HEPAX Textfile

Allgemeine Funktionen

HGETA liest einen Write All File aus dem HEPAX-Memory. Man kann sich also hier im RAM praktisch mehrere komplette Rechnerkonfigurationen abspeichern, d.h. man hat in einem File die finanzmathematischen Programme und Belegungen, im nächsten die für geometrische Berechnungen und im dritten die Haushaltskasse oder das Fahrtenbuch. Für einen solchen File muß man circa 2,5k rechnen, im Standard HEPAX-Modul ließen sich danach locker 2 komplette Belegungen abspeichern, es bliebe noch genügend Platz für anderes.

HGETK liest Tastenbelegungen aus dem HEPAX-Memory

HSAVEA speichert einen kompletten HP41 wie mit WRTA des HP-IL Moduls in das HEPAX-Memory

HSAVEK speichert die aktuellen Tastenbelegungen in das HEPAX-Memory

HSAVEMC speichert eine vom Anwender definierte Maschinensprachefunktion in das HEPAX-Memory, mehr weis ich auch nicht

HSAVEP speichert ein normales Programm in das HEPAX-Memory, dies kann dort auch gestartet werden

PRIVATE macht ein Programm im HEPAX-Memory PRIVATE, dieses kann nur noch gestartet oder gelöscht werden, lesen, kopieren und Einzelschrittabarbeitung sind nicht mehr möglich

ROMNAME versieht einen 4kByte Block des HEPAX-Memory mit einem Namen, das HEPAX-Memory ist nämlich physikalisch wie ein Applikationsmodul aufgebaut und hat eine Wortbreite von 10 Bit statt 8 Bit wie im Hauptspeicher des HP41

M-Code Funktionen

CLRAM löscht einen Block oder Teile davon im HEPAX-Memory

CODE wandelt eine Hexadezimalzahl im ALPHA-Register in den entsprechenden Wert in's X-Register um

COPYROM kopiert einen 4k Byte großen Block ROM in das HEPAX-Memory

DECODE schreibt das hexadezimale Äquivalent des X-Registers in das ALPHA-Register

DECODYX schreibt den im X-Register spezifizierten Teil des Y-Register als hexadezimales Äquivalent in das ALPHA-Register

DISASS listet einen Speicherbereich des HP41 in der Form Adresse, Wortwert, Mnemonic im Display oder auf einem Drucker auf; die Ausdrucke haben Jacobs/DeArras Format. Das Ausgabeformat läßt sich mit Hilfe der Flags 0-4 noch in weiten Bereichen variieren, dazu als Beispiel die Listings 1 bis 4:

LISTING 1

alle Flags geloescht

```
0000 201 ?NC GO
0001 006 ->0180
0002 2B5 ?NC GO
0003 006 ->01AD
0004 04E C=0 ALL
0005 270 RAM SLCT
0006 0F8 READ 3(X)
0007 158 M=C ALL
0008 398 C=ST XP
0009 056 C=0 XS
000A 284 CLRF 7
000B 3D8 C<>ST XP
000C 14C ?FSET 6
000D 05B JNC 0018 +0B
000E 08C ?FSET 5
000F 04B JNC 0018 +09
```

LISTING 2

Flag 0 gesetzt,
Flag 2 und 3 geloescht

```
0000 201
0001 006
0002 2B5
0003 006
0004 04E
0005 270
0006 0F8
0007 158
0008 398
0009 056
000A 284
000B 3D8
000C 14C
000D 05B
000E 08C
000F 04B
```

LISTING 3

Flag 0 und 2 gesetzt,
Flag 3 geloescht

```
0000 201 "x"
0001 006 "Γ"
0002 2B5 "5"
0003 006 "Γ"
0004 04E "N"
0005 270 "p"
0006 0F8 "x"
0007 158 "X"
0008 398 "ö"
0009 056 "V"
000A 284 "α"
000B 3D8 "X"
000C 14C "L"
000D 05B "["
000E 08C "μ"
000F 04B "K"
```

LISTING 4

Flag 0 und 3 gesetzt,
Flag 2 geloescht

```
0000 201 "A"
0001 006 "F"
0002 2B5 "5"
0003 006 "F"
0004 04E "N"
0005 270 "0"
0006 0F8 "8"
0007 158 " "
0008 398 " "
0009 056 "V"
000A 284 "D"
000B 3D8 " "
000C 14C "μ"
000D 05B "["
000E 08C "L"
000F 04B "K"
```

Listing 1 ist das interessanteste mit Ausdruck der Mnemonics, Listing 2 ist lediglich der Adressinhalt hexadezimal ausgegeben, Listing 3 stellt ASCII-Daten dar, Listing 4 sogenannte ROM-Daten, darunter konnte ich mir nichts vorstellen, die Assemblerfreaks werden schon etwas damit anfangen können

HEXEDIT ist nur zur Maschinenspracheprogrammierung erforderliche Speichereditor. In der Anzeige erscheint die zu editierende Adresse links im Display, in der Mitte steht der momentane Wortwert und rechts sind drei Prompts für den neuen Wert, alle Anzeigen in hexadezimaler Form. Die Eingabetastatur ist in Bild 1 zu sehen.

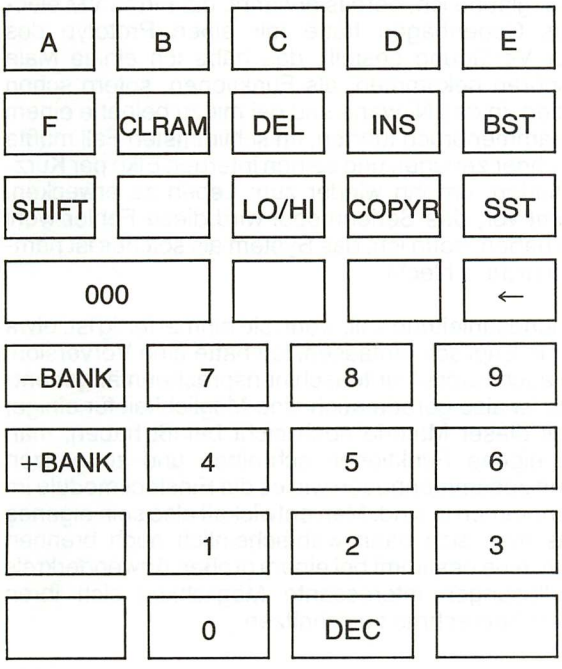


Bild 1: Tastenbelegung bei HEXEDIT

HPROMPT fordert zur Eingabe hexadezimaler Zahlen mit Kommentar linksbündig im Display, dies ist vergleichbar mit PMTH des CCD-Moduls

Andere Funktionen

HEPAX Multifunktionsaufruf weiterer 17 selten gebrauchter Funktionen für Maschinenspracheprogrammierung durch Eingabe deren Nummer, siehe Tabelle 2

HEPAXA wie HEPAX, nur Aufruf der Funktion durch ihren Namen

RAMTEST testet RAM-Bereiche der HEPAX Module

RAMTOG schaltet den Schreibschutz für HEPAX Speicherbereichsblöcke ein und aus

READROM liest ein auf einem IL-Massenspeicher abgelegtes ROM in das HEPAX-Memory

ROMTEST testet jedes ROM auf korrekte Prüfsumme

WRTROM schreibt eine Kopie eines HP41 ROM-Moduls auf einen Massenspeicher (z.B. Kassettenlaufwerk)

XF Multifunktionsaufruf mit Hilfe von Nummern; hiermit sind alle Funktionen des X-Functionmoduls und der CX X-Functions des HP41CX gemeint, das sind insgesamt 33 weitere Funktionen für die Leute, die noch den HP41C oder CV haben, siehe dazu Tabelle 3

XFA ist dasselbe wie XF, nur werden hier die Funktionen mit Namen aufgerufen

Ihr werdet Euch mit Sicherheit fragen, was soll denn das mit den Multifunktionen. Die Sache ist ganz einfach:
Im HEPAX-Modul stecken wesentlich mehr als die normalerweise pro Einsteckmodul maximal möglichen 64 Funktionen, die Multifunktionen allein sind schon 50 Funktionen. Also hat man Funktionen, die viele, weil sie z.B. einen HP41CX haben, nicht brauchen oder die insgesamt eher selten gebraucht werden, in einen anderen Adressbereich des Moduls gelegt, der über die Funktionen HEPAX bzw. HEPAXA und XF bzw. XFA erreichbar ist. Das Verfahren ist recht einfach:

Man ruft z.B. XF mit XEQ ALPHA XF ALPHA auf, in der Anzeige steht jetzt

XF _ _ _

Tippt man nun 003 ein, so wird die Funktion RCLFLAG ausgeführt. Der Aufruf geht natürlich auch mit dem Namen, dafür ist die Funktion XFA zuständig. Bei ihr erscheint

XFA _

Mit ALPHA RCLFLAG ALPHA passiert genau dasselbe wie bei XF 003.

Nummer	Name	Funktion
001	AND	logisches AND von X und Y Register
002	BCAT	Catalog der Modulblöcke 3 - F (so ähnlich wie CAT 8 - F des CCD-Moduls
003	BCD-BIN	schreibt den im X-Register befindlichen Dezimalwert als binäres Bitmuster in dasselbe
004	BIN-BCD	ist die Umkehrung von BCD-BIN
005	CTRAST	variiert den Displaykontrast der neueren 41'er mit sogenanntem Halfnut-Display
006	DELETE	arbeitet wie DEL der HEXEDIT-Funktion
007	INSERT	arbeitet wie INS der HEXEDIT-Funktion
008	NOT	invertiert das X-Register
009	OR	logisches OR von X und Y Register
010	ROTXY	rotiert das Y Register um X Nybbles
011	SHIFTYX	schiebt das Y Register um X Bit
012	XOR	logisches XOR von X und Y Register
013	X + Y	bitweise Addition von X und Y Register
014	X - $	konvertiert den Inhalt des X-Registers als Text in das ALPHA Register (wie das geht weiß ich auch noch nicht)
015	X − Y	bitweise Subtraktion von X und Y Register
016		nicht in der vorläufigen Bedienungsanleitung enthalten
017		nicht in der vorläufigen Bedienungsanleitung enthalten

Tabelle 2: Multifunktionen von HEPAX

Nummer	Name	Funktion
000	ALENG	berechnet die Anzahl der im ALPHA-Register befindlichen Zeichen
001	AROT	rotiert die Zeichen im ALPHA-Register wie im X-Register spezifiziert
002	ATOX	speichert den Dezimalcode des linken Zeichens im ALPHA-Register in das X-Register
003	RCLFLAG	ruft Status der Flags 00-43 in das X-Register
004	XTOA	hängt dezimal dargestelltes Zeichen an den Inhalt des ALPHA-Register
005	CLRGX	löschte einen bestimmten Datenregisterblock
006	PSIZE	programmierbares SIZE
007	REGMOVE	verschiebt Datenregister blockweise
008	REGSWAP	vertauscht Datenregister blockweise
009	SIZE?	ermittelt die zur Verfügung stehenden Datenregister
010	REG?	ermittelt die noch freien Programmregister
011	STOFLAG	speichert einen mit RCLFLAG ermittelten Flagstatus
012	X <>F	tauscht dezimales Äquivalent im X-Register mit dem Status der Flags 00-07
013	PASN	programmierbares ASN
014	CLKEYS	löscht alle Tastenbelegungen
015	ANUM	wandelt Zahl im ALPHA-Register in Zahl im X-Register

016	POSA	ermittelt die Position eines im X-Register stehenden Buchstabencodes im ALPHA-Register
017	X=NN?	indirekter Registervergleich mit dem X-Register
018	X=NN?	indirekter Registervergleich mit dem X-Register
019	X<NN?	indirekter Registervergleich mit dem X-Register
020	X<=NN?	indirekter Registervergleich mit dem X-Register
021	X>NN?	indirekter Registervergleich mit dem X-Register
022	X>=NN?	indirekter Registervergleich mit dem X-Register
023	GETKEY	warten auf Tastendruck und speichern des Tastencodes im X-Register
024	GETKEYX	wie GETKEY, nur durch bestimmte Tasten möglich
025	PCLPS	programmierbares CLP
026		nicht in der vorläufigen Bedienungsanleitung enthalten
027		nicht in der vorläufigen Bedienungsanleitung enthalten
028		nicht in der vorläufigen Bedienungsanleitung enthalten
029		nicht in der vorläufigen Bedienungsanleitung enthalten
030		nicht in der vorläufigen Bedienungsanleitung enthalten
031		nicht in der vorläufigen Bedienungsanleitung enthalten
032		nicht in der vorläufigen Bedienungsanleitung enthalten

Tabelle 3: Multifunktionen von XF

Resumé

Man sieht an Hand der Tabelle 3, daß man einen HP41CV mit Hilfe dieses Moduls recht leicht in einen HP41CX verwandeln kann, wenn man noch das TIME-Modul dazusteckt, das war auf alle Fälle eine gute Idee für die vielen Leute, die noch einen „alten Hasen" haben.

Ich hatte es, glaube ich, bereits erwähnt, die Firma VM electronics aus Copenhagen hatte mir einen Prototyp des Moduls zur Verfügung gestellt, das habe ich einige Male bitter zu spüren bekommen, als Funktionen, sofern schon implementiert, in das Nirwana und bei mir zu beinahe einem Nervenzusammenbruch führten, im schlimmsten Fall mußte ich den Rechner zerlegen und seinen internen Elko per Kurzschluß entladen, um ihn wieder zum Leben zu erwecken. Naja, kommt vor, das Serienmodul wird diese Fehler wohl nicht mehr haben, hoffe ich, das System als solches ist nämlich wirklich nicht schlecht.

Die Bedienungsanleitung soll, wenn sie einmal fertig ist, etwa 100 Seiten in Englisch umfassen, ich hatte eine Vorversion. Darin sind auch Kapitel für Maschinenspracheanfänger enthalten, dies ist also gerade auch eine Möglichkeit für einige, die sich mit dieser Materie noch nicht befaßt haben, man kann jetzt eigene Funktionen schreiben und zu ganzen Bibliotheken zusammenbauen wie es die Einsteckmodule im Grunde genommen ja sind. Man entwickelt also sein eigenes Modul, das man sich dann wahrscheinlich auch brennen lassen kann, eine bestimmt bei einem großen Anwenderkreis für Speziallösungen interessante Möglichkeit sich ihren eigenen Taschenrechner zu schnitzen.

Martin Meyer (1000)
Redaktion

Koordinatentransformation

246 Zeilen, 494 Bytes, 71 Regs., CCD

Hinweis zur Nomenklatur: Vektoren haben eine einfache Unterstreichung. Beispiel $\underline{P}$, $\underline{P'}$und $\underline{T}$ (Translationsvektor). Matrizen besitzen eine doppelte Unterstreichung wie die Transformationsmatrix $\underline{\underline{T}}$.

Die meisten von Euch werden wissen, was eine Koordinatentransformation ist. Und viele werden sich mit den Transformationsprogrammen aus dem Mathematikmodul oder dem Advantage Pac beschäftigt haben, und eventuell Schwierigkeiten gehabt haben, den Rotationswinkel bzw. das Vorzeichen dieses Winkels zu bestimmen. Ich weiß es jetzt und will es auch nicht für mich behalten: Eine normale Schraube (Rechtsgewinde) dreht sich in der anzugebenden Rotationsachse (a, b, c) bei positiv zählendem Winkel (ROT∢) in den Ursprung des **neuen** Koordinatensystemes hinein. Anders ausgedrückt, blickt man entlang der Rotationsachse zum **neuen** Koordinatenursprung, so läuft der positive Winkel im Uhrzeigersinn um.

– Ist die Rotationsachse durch (a, b, c) = (1, 1, 1) festgelegt, so wird die Drehung doch anders ausgeführt als bei der Rotationsachse (a, b, c) = (−1, −1, −1), und dies obwohl die angegebenen Achsen auf einer Geraden liegen!!

Im weiteren Verlauf meines Beitrages möchte ich weitere Betrachtungen zu Koordinatentransformationen mitteilen, und gleichzeitig eine für mich offene Frage an Euch weitergeben, um damit vielleicht nochmal eine rege Diskussion zu entfachen, wie es z.B. bei Primzahlenbestimmung, SIZE-Routinen, Rundung von Geldbeträgen (Franz Riedlinger) usw. der Fall war. Ich möchte die 41-Gruppe, und vielleicht nicht nur diese Gruppe, wachrütteln, damit der Umfang unserer Clubzeitschrift nicht abnimmt, sondern durch rege Beiträge zunimmt.

Es gibt eine Reihe von denkbaren Koordinatentransformationen, bei denen eine Translation (Verschiebung) und mehrere Rotationen nacheinander durchgeführt werden. Beispiel hierfür ist die sogenannte Eulertransformation, die in der technischen Mechanik (Kinematik) bei der Umrechnung von raumfesten in körperfeste Koordinaten angewendet wird.

Jede Rotationsformation kann wie folgt geschrieben werden:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} & T & \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Wird die Transformationsmatrix $\underline{\underline{T}}$ mit dem Ursprungsvektor $(x,y,z,)^T$ multipliziert, so entsteht der neue Vektor $(u,v,w)^T$. Bei einer Translation (Verschiebung) wird vom Ursprungsvektor der Translationsvektor (x_0,y_0,z_0) abgezogen. Durch die weiter unten noch erklärten „Eulerwinkel" kann die Matrix $\underline{\underline{T}}$ aufgestellt werden. Ich möchte nun aus dieser Matrix oder aus den Eulerwinkeln die Größen bestimmen, die ich für die Transformationsprogramme aus dem Mathe-Modul oder dem Advantage Pac benötige. Das bedeutet, ich will den Rotationswinkel und den Rotationsvektor, also die Achse (a,b,c), wissen.

Dies soll Eure Aufgabe sein, doch zunächst gebe ich noch einige Erläuterungen zur 2-dimensionalen und zur Eulertransformation.

Ein rechtwinkliges Koordinatesystem X-Y ist durch Translation und Rotation in ein anderes rechtwinkliges System U-V zu überführen. Vgl. Abb. 1.

Der Translationsvektor $\vec{r_0} = (x_0, y_0)$ ist in x-y-Koordinaten anzugeben. Der Rotationswinkel ϕ ist in mathematisch positiver Richtung zu zählen (Gegenuhrsinn). Die aus der Zeichenebene zeigende Z-Achse komplettiert die beiden anderen Achsen zu einem Rechtssystem, so daß sich die Z-Achse bei einer Drehung mit wachsendem ϕ wie eine Rechtsschraube bewegt. Für die neuen Koordinaten u-v erhält man:

$$u = (x-x_0)\cdot\cos\phi + (y-y_0)\cdot\sin\phi$$
$$v = (x-x_0)\cdot\sin\phi + (y-y_0)\cdot\cos\phi$$

für die Rücktransformation gilt:

$$x = x_0 + u\cdot\cos\phi - v\cdot\sin\phi$$
$$y = y_0 + u\cdot\sin\phi + v\cdot\cos\phi$$

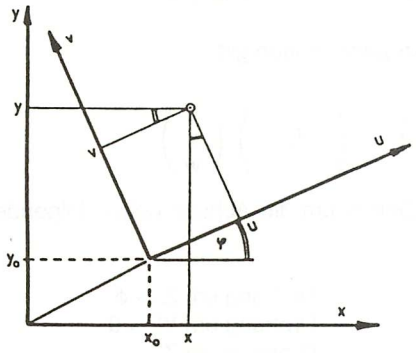


Abb. 1

In Polarkoordinaten hat die Transformation folgende Form: Zunächst ergibt sich die Zeigerlänge $L = \sqrt{x^2 + y^2}$ und der entsprechende Winkel $\alpha = \arctan(\frac{y}{x})$. Eine Rotation bedeutet eine Winkelveränderung bei gleicher Zeigerlänge L. Aus dem neuen Winkel $\alpha-\phi$ werden die neuen Koordinaten u−v berechnet mit: $u = L\cdot\cos(\alpha-\phi)$ und $v = L\cdot\sin(\alpha-\phi)$.

Mit den Additionstheoremen:

$\cos(\alpha-\phi) = \cos\alpha\cdot\cos\phi+\sin\alpha\cdot\sin\phi$ und

$\sin(\alpha-\phi) = \sin\alpha\cdot\cos\phi-\cos\alpha\cdot\sin\phi$

ergibt sich nach Einsetzen

$u = L\cdot(\cos\alpha\cdot\cos\phi + \sin\alpha\cdot\sin\phi)$ und

$v = L\cdot(\sin\alpha\cdot\cos\phi - \cos\alpha\cdot\sin\phi)$.

Verwendet man nun noch $x=L\cdot\cos\alpha$ und $y=L\cdot\sin\alpha$, so erkennt man die Gleichheit der beiden Transformationen. Der Übersicht wegen wurde der Translationsvektor hier fortgelassen.

Das Programm TRANS2 berechnet sowohl die Hin- als auch die Rücktransformation zwischen den beiden Systemen. Zunächst ist der Translationsvektor (x_0, y_0) einzugeben, wobei nach dem y_0-Wert die ENTER-Taste nicht betätigt werden darf. Anschließend wird der Rotationswinkel ϕ abgefragt.

Soll die Transformation vom X-Y-System ins U-V-System erfolgen, so ist die Abfrage in Zeile 07: XY → UV J:N mit J(a) zu beantworten. Woraufhin x,y abgefragt werden (auch hier nach y nicht „ENTER(N)!"). Die angezeigten Werte u und v stehen, während sie angezeigt werden, jeweils im X-Register, ansonsten im Y-Register.

– Für die Rücktransformation gilt die entsprechende Umkehrung. D.h. in Zeile 07 ist die Abfrage XY → UV J:N mit N(ein) zu beantworten. Danach wird u und v abgefragt und (x,y) berechnet.

Verwendete Register:

00: ϕ
11: X_0
12: Y_0

Flag:(gesetzt)

00: Transformation von XY nach UV
21: nur bei Verwendung des Druckers (wegen AVIEW)

Auf die Verwendung der Speicher 1-10 wird bei solch kleinen Programmen immer verzichtet, da diese Register für Handrechnungen angenehmer sind und von den Programmen nicht überschrieben werden sollen!

Die hier durchgeführten Transformationen haben in der Matrixschreibweise die folgende Gestalt:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x-x_0 \\ y-y_0 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Im Grunde bietet sich die Matrizenrechnung geradezu an, zumal die Transformationsmatrix nur einmal abgespeichert werden muß. Bei der Umrechnung von UV nach XY würde die Matrix invertiert, was mit dem Modul Advantage Pac in wenigen Sekunden vollzogen ist. Aber es würden insgesamt 4 Matrizen benötigt. Platzbedarf im RAM-Bereich 14 Register und im XM-Bereich 18 Register! Das ist offensichtlich weit mehr als nötig.

Mit den Polarkoordinaten geht es, wie zuvor beschrieben, platzsparender und schneller. Auch bei der 3-dimensionalen Koordinatentransformation kann das Schema angewendet werden.

Eulertransformation

Bei der sogenannten Eulertransformation werden 3 Rotationen nacheinander durchgeführt. Die Anwendung findet sich beispielsweise in der technischen Mechanik, wenn die Kinematik rotierender Körper gegenüber ortsfesten Systemen zu bestimmen ist.

Das Rechtssystem X−Y−Z wird dabei zunächst um den Winkel ϕ (PHI) um die Z-Achse gedreht. Es entsteht das System X'−Y'−Z'. Anschließend wird mit dem Winkel θ (THETA) um die (neue) Y'−Achse gedreht. Die Y'−Achse wird häufig auch als Knotenlinie bezeichnet.

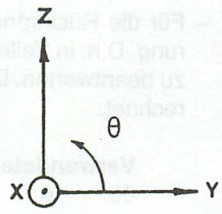
– In dem hieraus entstandenen X"−Y"−Z"−System wird schließlich noch einmal um die Z"−Achse geschwenkt. Der letzte Rotationswinkel wird χ(CHI) genannt.

Um die X-Achse findet keine Rotation statt. Die positive Drehrichtung für die Winkel können der nachstehenden Abb. 2 entnommen werden. Die Matrizen der einzelnen Rotationen sind mit angegeben. Auch die Drehung um die X-Achse ist der Vollständigkeit wegen aufgeführt, obwohl sie für die Eulertransformation nicht benötigt wird.

– Die jeweils dritte Achse, die nicht in der Zeichenebene liegt, zeigt, nach oben, d.h. sie läuft auf den Betrachter zu, was durch den umkreisten Punkt dargestellt ist.

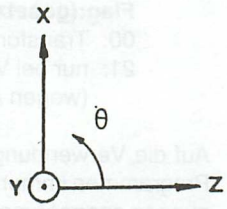
Drehung um die X-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Drehung um die Y-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Drehung um die Z-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

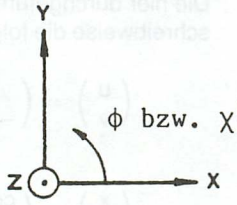


Abb. 2

Der Aufbau der 3x3-Matrix wird leicht verständlich, wenn diese Matrix mit der 2x2-Matrix der 2-dimensionalen Transformation verglichen wird. Die beiden cos- und sin-Terme entsprechen einander. Wird um die Achse A_i gedreht, so verändert sich deren Koordinate a_i nicht ($a'_i = a_i$). Damit wird in der Transformationsmatrix das Hauptdiagonalelement $t_{ii} = 1$. Die beiden Ursprungskoordinaten a_j haben keinen Einfluß auf a'_i, aber auch a_i hat keinen Einfluß auf die beiden Ergebniskoordinaten a'_j. Und damit werden alle Elemente in der i-ten Spalte und i-ten Zeile zu 0, außer natürlich dem Element t_{ii}. Die beiden anderen Hauptdiagonalelemente t_{jj} der Matrix $\underline{\underline{T}}$ sind $\cos\phi$ (Rotationswinkel = ϕ). Die beiden verbleibenden Elemente sind $+\sin\phi$ und $-\sin\phi$. Und zwar hat der obere rechte Sinusterm das positive Vorzeichen, wenn die nach rechts zeigende Achse in der Tripelreihenfolge X-Y-Z vor der nach oben zeigenden Achse steht. Dies ist bei Rotationen um die X- und um die Z-Achse der Fall. Bei Rotationen um die Y-Achse ist der obere rechte Term $-\sin\phi$ und der untere linke $+\sin\phi$. (Vgl. Abb. 2)

Im nächsten Schritt wird die gesamte Rotationsmatrix $\underline{\underline{E}}$ für die Eulertransformation bestimmt. Es gilt $\underline{\underline{E}} = (\underline{\underline{Z}}\ \underline{\underline{Y'}})\ \underline{\underline{Z''}}$. Hierbei tragen die Einzelmatrizen den Namen der Achse, um die sie eine Drehung bewirken. Abb. 3 ist ein Schema der Abb. 4. Es erleichtert die Durchführung von Matrizenmultiplikationen, denn Spaltenelemente der 1. Matrix werden mit Zeilenelementen der 2. Matrix multipliziert und aufaddiert, wobei die Verlängerung von der jeweiligen Spalte bzw. Zeile auf das Element der Ergebnismatrix weisen.

	$\underline{\underline{Z}}$
$\underline{\underline{Y'}}$	$\underline{\underline{Z}}\,\underline{\underline{Y'}}$
$\underline{\underline{Z''}}$	$\underline{\underline{E}}$

Abb. 3

In Abb. 4 ist die Matrix $\underline{\underline{E}}$ stark umrandet wiedergegeben. Eine Transformation vom X-$\overline{Y}$-Z – System ins U-V-W – System lautet in Matrizenschreibweise:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \left(\underline{\underline{E}} \right) \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Für die Rücktransformation gilt:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \left(\underline{\underline{E}}^{-1} \right) \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Für die Rotationen um die Achsen gelten folgende Winkelbezeichnungen:

Drehung um Z → ϕ
Drehung um Y' → θ
Drehung um Z'' → χ

Die Eulertransformation ist, wie die 2-dimensionale Transformation, mit Polarkoordinaten wesentlich einfacher und schneller zu bewerkstelligen, als mit den oben aufgeführten zahlreichen Multiplikationen von Sinus- und Cosinuswerten. Die Koordinate, um deren Achse gedreht wird, verändert sich während der Rotation nicht. Der Wert steht im Z-Register. Die beiden anderen Koordinaten sind in den Registern X und Y des Stacks so angeordnet, daß die Polarkoordinatenumrechnung ein Rechtssystem beschreibt. Vergleiche Abb. 2. Die Koordinate der nach oben zeigenden Achse steht im Y-Register, und die Koordinate der nach rechts weisenden Achse befindet sich im X-Register.

Die gesamte Rotationsmatrix für die Drehungen um die Z-, Y'- und Z''-Achsen

Abb.4	$\cos\phi$ $-\sin\phi$ 0	$\sin\phi$ $\cos\phi$ 0	0 0 1
$\cos\theta$ 0 $-\sin\theta$ 0 1 0 $\sin\theta$ 0 $\cos\theta$	$\cos\phi\cdot\cos\theta$ $-\sin\phi$ $\cos\phi\cdot\sin\theta$	$\sin\phi\cdot\cos\theta$ $\cos\phi$ $\sin\phi\cdot\sin\theta$	$-\sin\theta$ 0 $\cos\theta$
$\cos\chi$ $\sin\chi$ 0 $-\sin\chi$ $\cos\chi$ 0 0 0 1	$+\cos\phi\cdot\cos\theta\cdot\cos\chi$.. $-\sin\phi\cdot\sin\chi$ $-\cos\phi\cdot\cos\theta\cdot\sin\chi$.. $-\sin\phi\cdot\cos\chi$ $\cos\phi\cdot\sin\theta$	$\sin\phi\cdot\cos\theta\cdot\cos\chi$.. $+\cos\phi\cdot\sin\chi$.. $-\sin\phi\cdot\cos\theta\cdot\sin\chi$.. $+\cos\phi\cdot\cos\chi$ $\sin\phi\cdot\sin\theta$	$-\sin\theta\cdot\cos\chi$ $\sin\theta\cdot\sin\chi$ $\cos\theta$

Das Programm TTRANS3 berechnet 3-dimensionale Transformationen (Eulertransformation) und die entsprechenden Rücktransformation. Auch eine translatorische Bewegung vor der Rotation, bzw. nach der Rotation bei Rücktransformation, kann berücksichtigt werden.

Nach dem Starten des Programmes wird dieser Translationsvektor $(X_0, Y_0, Z_0)^T$ abgefragt. Anschließend sind die 3 Winkel ϕ (PHI), θ (THETA) und χ (CHI) einzugeben. Wie bei dem Programm TTRANS2 wird dann in Zeile 07 abgefragt, ob die Transformation vom XYZ-System ins UVW-System oder umgekehrt erfolgen soll. D.h. hier ist mit J(a) oder N(ein) zu antworten. Bei J(a) werden die Koordinaten x,y,z abgefragt und u,v,w berechnet. Für die Umkehrung gilt entsprechendes.

Verwendete Register:

00: X_0
11: Y_0
12: Z_0
13: ϕ (PHI)
14: θ (THETA)
15: χ (CHI)

Flag:(gesetzt)

00: Transf. von XYZ nach UVW
21: nur bei Verwendung des Druckers (wegen AVIEW)

Auf die Verwendung der Register 1-10 wurde auch hier aus vorgenanntem Grund verzichtet.

Wer das CCD-Modul nicht besitzt, kann die Funktion PMTK durch lokale ALPHA-Labels umgehen. Hierzu Zeile 6-10 löschen, bzw. durch PROMT, GTO 00 ersetzen. Weiterhin LBL 01 durch LBL A, SF 00 und LBL 02 durch LBL C, CF 00 ersetzen. Ist x,y,z gegeben, so wird die Taste A betätigt. Für die Rücktransformation ist die Taste C zu wählen.

Damit die Ausführungszeit von PMTK nicht zu lange dauert, wurde die Abfrage nach Verschiebungsvektor und Winkeln (LBL 06) zum Programmende verschoben. Wenn die Funktion PMTK nicht verwendet wird, so können die Zeilen 95-111 in Zeile 2 und folgende verschoben werden. 4 Bytes werden damit eingespart.

Bei dem Programm TTRANS2 kann die Funktion PMTK genauso ersetzt werden.

Bei den Transformationsprogrammen aus dem Mathematik-Modul oder dem Advantage-Pac wird eine im Raum liegende Rotationsachse (A,B,C) und ein Rotationswinkel θ benötigt. In Vektorschreibweise ergibt sich die Transformation vom alten Vektor $\underline{P}$ aus dem XYZ-System zum neuen Vektor $\underline{P}'$ in dem UVW-System wie folgt:

$$\underline{P} = \{(\underline{P} - \underline{T}) \cdot \underline{n}\}\, \underline{n}\,(1-\cos\theta) + (\underline{P} - \underline{T})\cos\theta + \{(\underline{P} - \underline{T}) \times \underline{n}\}\sin\theta$$

$\underline{T}$ ist ein Translationsvektor, der vor der Rotation von dem Vektor $\underline{P}$ subtrahiert wird. Für die Rücktransformation vom neuen System ins alte Koordinatensystem gilt die Beziehung

$$\underline{P} = \{(\underline{P}' \cdot \underline{n})\, \underline{n}\,(1-\cos\theta) + \underline{P}'\cos\theta + (\underline{P}' \times \underline{n})\sin(-\theta)\} + \underline{T}$$

$\underline{n}$ ist der Einheitsvektor der Rotationsachse mit den Komponenten $(a,b,c)^T$. Die Auswertung der vektoriellen Schreibweise liefert die Transformationsmatrix $\underline{\underline{T}}$

$$\underline{\underline{T}} = \begin{bmatrix} a^2(1-\cos\theta) + \cos\theta & ab(1-\cos\theta) + c\cdot\sin\theta & ac(1-\cos\theta) - b\cdot\sin\theta \\ ab(1-\cos\theta) - c\cdot\sin\theta & b^2(1-\cos\theta) + \cos\theta & bc(1-\cos\theta) + a\cdot\sin\theta \\ ac(1-\cos\theta) + b\cdot\sin\theta & bc(1-\cos\theta) - a\cdot\sin\theta & c^2(1-\cos\theta) + \cos\theta \end{bmatrix}$$

In Matrixschreibweise lautet die Hintransformation:

$$\underline{P}' = \underline{\underline{T}}\,(\underline{P} - \underline{T})$$

Die Rücktransformation ergibt sich zu:

$$\underline{P} = \underline{\underline{T}}^{-1}\underline{P}' + \underline{T}$$

Die Frage ist nun, ob aus den 3 Eulerwinkeln der Rotationsvektor $(A,B,C)^T$ bzw. dessen Einheitsvektor $(a,b,c)^T$ und der Rotationswinkel eindeutig bestimmt werden kann, so daß die sich ergebende Rotationsformation identisch ist.

Axonometrische Parallelprojektionen

Zur einfachen Darstellung von Körpern werden axonometrische Parallelprojektionen verwendet. Obwohl die Projektion den Körper nur aus einer Richtung zeigt, können doch komplizierte Konturen auch von Laien gut erkannt werden. Deshalb finden sich diese Darstellungen auch in fast jeder Bedienungsanleitung und den sogenannten Explosionszeichnungen.

Im Gegensatz zur Fluchtpunktperspektive bleiben bei der Parallelprojektion alle parallelen Linien des Körpers auch in der Darstellung parallel. Rechte Winkel erscheinen in der Projektion nicht als solche, aber sie bewirken beim Betrachter den Eindruck der Rechtwinkligkeit.

Die beiden folgenden Programme TISOMET und TDIMET erleichtern dem im Zeichnen Ungeübten die Anfertigung solcher Parallelprojektionen.

Isometrie

Die einfachste Parallelprojektion ist die isometrische Darstellung. Vgl. Abb. 5. Die Z-Achse (Höhenkoordinate) ist vertikal angeordnet, während die X- und Y-Achsen je einen 30° Winkel zur Horizontalen bilden. Der Maßstab aller drei Achsen ist gleich.

Die x,y,z Koordinaten des Körpers werden auf das X,Y,Z Liniennetz der Abb. 5 übertragen. D.h. man verfolgt x Einheiten entlang die X-Achse, dann bewegt man sich von dort y Einheiten parallel zur Y-Achse. Und schließlich geht man von diesem Punkt z Einheiten parallel zur Z-Achse. Dieser Vorgang wird für jeden markanten Körperpunkt wiederholt.

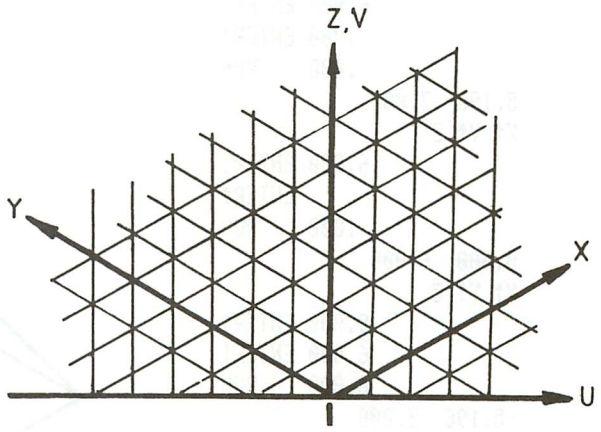


Abb. 5

Legt man in den Ursprung des XYZ-Systemes ein rechtwinkliges UV-System, bei dem die U-Achse horizontal und die V-Achse vertikal verläuft, so können die einzelnen Punkte als UV-Koordinaten berechnet werden. Somit kann auf Millimeterpapier ein Körper perspektivisch dargestellt werden, ohne daß zuvor das XYZ-Liniennetz gezogen wird. Mathematisch ist dies wie

jede andere Projektion eine Koordinatentransformation. Hier wird ein 3-dimensionaler Vektor $(X,Y,Z)^T$ auf die 2-Dimensionalität des Papierblattes abgebildet. Man kann dies schreiben als:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Die Transformationsmatrix setzt sich aus folgenden Gliedern zusammen:

$t_{11} = \cos 30°$ $\quad t_{12} = -\cos 30°$ $\quad t_{13} = 0$

$t_{21} = \sin 30°$ $\quad t_{22} = \sin 30°$ $\quad t_{23} = 1$

Die Matrix $\underline{\underline{T}}$ lautet damit:

$$\underline{\underline{T}} = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix}$$

Beispiel:
Eine vierseitige Pyramide der Höhe 5 cm und einer Seitenlänge der Grundfläche von 6 cm ist isometrisch darzustellen. Die Koordinaten x,y,z der 5 Eckpunkte und deren u,v Koordinaten ergeben sich zu:

Punkt	x	y	z		u	v
1	0	0	0		0,000	0,000
2	6	0	0		5,196	3,000
3	6	6	0		0,000	6,000
4	0	6	0		−5,196	3,000
5	3	3	5	(Spitze)	0,000	8,000

Die Abb. 6 ist das Ergebnis der Berechnung.

```
                XEQ "ISOMET"
X↑ Y↑ Z
                0.000 ENTER↑
                      ENTER↑
                      ENTER↑
                        RUN
0.000  0.000
X↑ Y↑ Z
              6.000 ENTER↑
               .000 ENTER↑
               .000    RUN
5.196  3.000
X↑ Y↑ Z
              6.000 ENTER↑
                    ENTER↑
               .000    RUN
0.000  6.000
X↑ Y↑ Z
              0.000 ENTER↑
              6.000 ENTER↑
              0.000    RUN
-5.196  3.000
X↑ Y↑ Z
              3.000 ENTER↑
              3.000 ENTER↑
              5.000    RUN
0.000  8.000
X↑ Y↑ Z
```

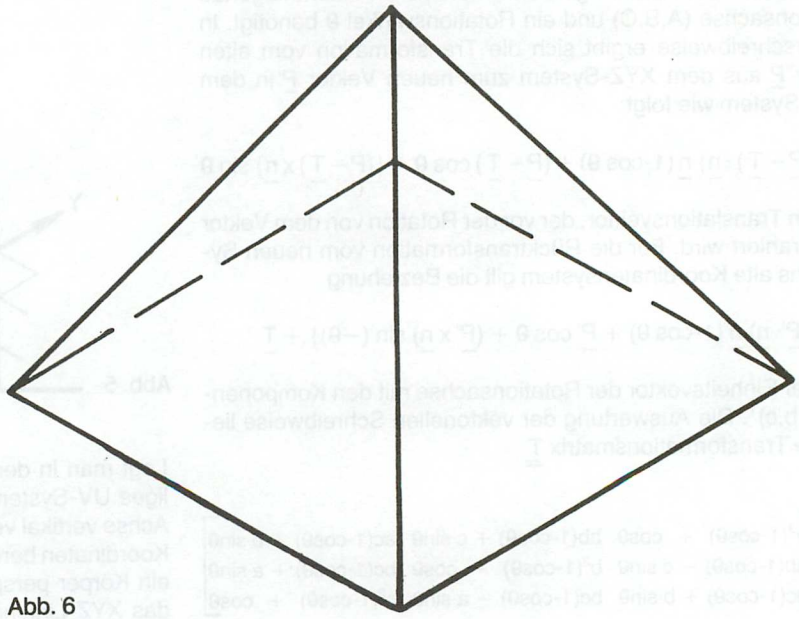


Abb. 6

Dimetrie

Die zuvor beschriebene isometrische Darstellung ist zwar sehr einfach zu zeichnen, hat aber den Nachteil, daß bei regelmäßigen Körpern häufig Linien der Körperrückseite von Kanten der Frontseite verdeckt werden. Dies tritt insbesondere dann auf, wenn quadratische Körperflächen in der XY-Ebene verlaufen und von der vorderen und hinteren Ecke Kanten nach oben oder unten gehen. Die isometrische Projektion erreicht damit nicht die gewünschte Transparenz. Bei der dimetrischen Parallelprojektion kommt die Überlegung von Körperkanten viel seltener vor.

Die Z-Achse (Höhenkoordinate) ist auch hier vertikal angeordnet. Die beiden X- und Y-Achsen bilden gegenüber der Horizontalen einen Winkel von 7° bzw. 42°. Der Maßstab der in 42°-Richtung verlaufenden Achse ist halb so groß wie die Maßstäbe der beiden anderen Achsen. Man hat die Möglichkeit, die 42°-Achse auf die rechte oder auf die linke Seite zu legen, wodurch der Körper entweder mehr von rechts oder mehr von links gesehen dargestellt wird.

Die Konstruktion der einzelnen markanten Körperpunkte im XYZ-Liniennetz geschieht nach den gleichen Regeln wie bei der Isometrie. Analog zur isometrischen Darstellung können die XYZ-Koordinaten in UV-Koordinaten der Papierblattebene umgerechnet werden. Wiederum fallen die Koordinatenzentren zusammen. Die Transformationsmatrix für die Berechnung der UV-Koordinaten hat die gleiche Gestalt wie bei der isometrischen Projektion. Hier muß nur die rechte oder linke Position der 7°-Achse sowie der halbierte Maßstab der 42°-Achse berücksichtigt werden.

Für die Darstellung „7° RECHTS" werden die folgenden Matrixelemente eingesetzt:

$t_{11} = \cos 7°$ $\quad t_{12} = -\frac{1}{2}\cos 42°$ $\quad t_{13} = 0$

$t_{21} = \sin 7°$ $\quad t_{22} = -\frac{1}{2}\sin 42°$ $\quad t_{23} = 0$

Bei der Projektion „7° LINKS" werden die Beträge der 1. und 2. Spalten ausgetauscht, wobei die Vorzeichen beibehalten werden. Die Matrix $\underline{\underline{T}}$ erhält somit folgende Einträge:

$t_{11} = \frac{1}{2}\cos 42°$ $\quad t_{12} = -\cos 7°$ $\quad t_{13} = 0$

$t_{21} = \frac{1}{2}\sin 42°$ $\quad t_{22} = \sin 7°$ $\quad t_{23} = 1$

Als Beispiel für beide dimetrischen Darstellungen wird wieder die Pyramide verwendet, so daß isometrische und dimetrische Darstellung direkt miteinander verglichen werden können.

Im Vergleich von iso- und dimetrischer Projektion gilt prinzipiell: Die dimetrische Darstellung ist kleiner, wobei der „Blickpunkt" zu einer Seite hin und tiefer versetzt liegt. Andere Körpergeometrien können die Behauptung des tieferen Blickpunktes besser belegen. Die Pyramide wurde hier nur der Einfachheit halber als Beispiel gewählt.

Die Bedienung der Routinen ᵀISOMET und ᵀDIMET ist denkbar einfach gehalten. Nach dem Programmstart sind die Körperkoordinaten XYZ einzugeben und R/S zu betätigen. Hierbei darf nach der Z-Koordinate die ENTER-Taste **nicht** gedrückt werden. Bei beiden Programmen steht das Ergebnis als UV-Koordinaten im Alpha-Register bzw. U im X- und V im Y-Register des Stacks.

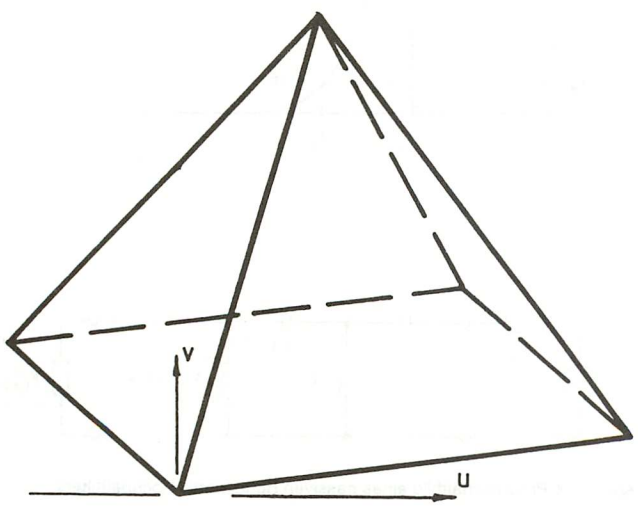
Soll bei der dimetrischen Darstellung die 7°-Achse rechts sein, so muß die Abfrage in Zeile 7 (7'RECHTS J:N) be(J)aht werden. Flag 00 hält die Entscheidung fest. Für die Projektion 7° links wird die obige Abfrage ver(N)eint. Wer das CCD-Modul nicht besitzt, kann die PMTK-Funktion wie folgt umgehen. Nach dem PROMPT in Zeile 7 steht über der Taste „D" das „J" für ja und über der Taste „E" das „N" für nein. Mit den lokalen ALPHA-Labels D und E wird die PMTK Funktion sozusagen simuliert. Im Klartext: Soll die 7°-Achse rechts sein, so ist die Taste D zu betätigen, anderenfalls die Taste E drücken.

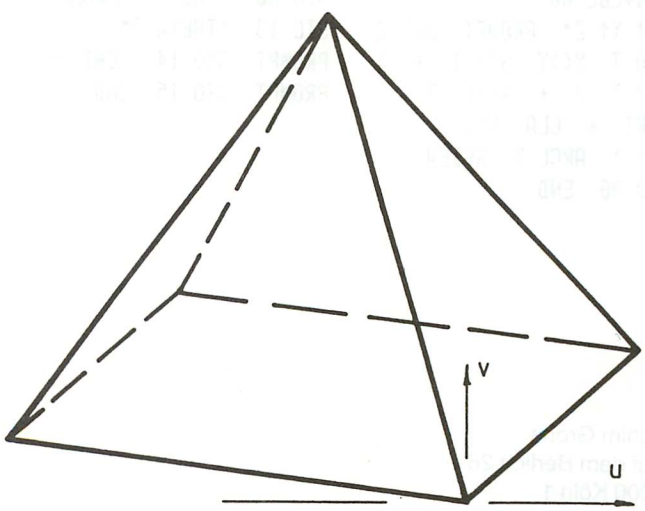
Beide Routinen verändern nur Flag 00 und den Stackinhalt.

```
7'RECHTS

          XEQ "DIMET"
X↑_Y↑_Z
          0,000 ENTER↑
                ENTER↑
                ENTER↑
                   RUN
0,000  0,000
X↑_Y↑_Z
          6,000 ENTER↑
          0,000 ENTER↑
          0,000    RUN
5,955  0,731
X↑_Y↑_Z
          6,000 ENTER↑
          6,000 ENTER↑
          0,000    RUN
3,726  2,739
X↑_Y↑_Z
          0,000 ENTER↑
          6,000 ENTER↑
          0,000    RUN
-2,229  2,007
X↑_Y↑_Z
          3,000 ENTER↑
          3,000 ENTER↑
          5,000    RUN
1,863  6,369
X↑_Y↑_Z
```

```
7'LINKS

          XEQ "DIMET"
X↑_Y↑_Z
          0,000 ENTER↑
                ENTER↑
                ENTER↑
                   RUN
0,000  0,000
X↑_Y↑_Z
          6,000 ENTER↑
          0,000 ENTER↑
          0,000    RUN
2,229  2,007
X↑_Y↑_Z
          6,000 ENTER↑
          6,000 ENTER↑
          0,000    RUN
-3,726  2,739
X↑_Y↑_Z
          0,000 ENTER↑
          6,000 ENTER↑
          0,000    RUN
-5,955  0,731
X↑_Y↑_Z
          3,000 ENTER↑
          3,000 ENTER↑
          5,000    RUN
-1,863  6,369
X↑_Y↑_Z
```

```
 02♦LBL 01
FIX 3  SF 21  SF 27
"7 RECHTS:J N"  PROMPT
GTO 01

 09♦LBL D
SF 00  GTO 00

 12♦LBL E
CF 00

 14♦LBL 00
"X↑ Y↑Z"  .END.


 01♦LBL "DIMET"
CF 00  FIX 3  SF 21  1 E
"7 RECHTS JN"  PMTK
X=Y?  SF 00

 10♦LBL 00
"X↑_Y↑_Z"  PROMPT  X<> Z
FS? 00  X<>Y  2  /  42
X<>Y  P-R  X<> T  +
X<>Y  7  X<>Y  P-R  R↑
-  FC? 00  CHS  X<> Z  +
X<>Y  CLA  ARCL X  "⊢"
ARCL Y  AVIEW  GTO 00
END
```





```
01♦LBL "TRANS2"
XEQ 06

03♦LBL 00
CF 00  "XY↑UV JN"   E
PMTK  X=Y?  SF 00
GTO IND X

11♦LBL 01
"X↑_Y"  PROMPT  RCL 12
-  X<>Y  RCL 11  -
GTO 03

20♦LBL 02
"U↑_V"  PROMPT  X<>Y

24♦LBL 03
R-P  X<>Y  RCL 00
FC? 00  CHS  -  X<>Y
P-R  FS? 00  GTO 04
RCL 11  +  X<>Y  RCL 12
+  X<>Y

41♦LBL 04
SF 21  "U"  FC? 00  "X"
XEQ 05  X<>Y  "V"
FC? 00  "Y"  XEQ 05
GTO 00

53♦LBL 05
FIX 6  RND  "⊢= "
ARCL X  AVIEW  RTN

60♦LBL 06
"X0↑_Y0"  PROMPT  STO 12
X<>Y  STO 11  "PHI ?"
PROMPT  STO 00  END
```

```
01♦LBL "TRANS3"
XEQ 06

03♦LBL 00
CF 00  "XYZ↑ UVW JN"   E
PMTK  X=Y?  SF 00
GTO IND X

11♦LBL 01
"X↑_Y↑_Z"  PROMPT
RCL 12  -  X<>Y  RCL 11
-  RCL Z  RCL 00  -  R-P
X<>Y  RCL 13  -  XEQ 03
RCL 15  -  X<>Y  P-R
GTO 04

32♦LBL 02
"U↑_V↑_W"  PROMPT  X<> Z
R-P  X<>Y  RCL 15  +
XEQ 03  RCL 13  +  X<>Y
P-R  RCL 00  +  X<> Z
RCL 12  +  X<>Y  RCL 11
+  RCL Z  GTO 04

55♦LBL 03
X<>Y  P-R  RCL Z  R-P
X<>Y  RCL 14  FC? 00
CHS  -  X<>Y  P-R  X<> Z
X<>Y  R-P  X<>Y  RTN

72♦LBL 04
SF 21  "U"  FC? 00  "X"
XEQ 05  "V"  FC? 00  "Y"
XEQ 05  "W"  FC? 00  "Z"
XEQ 05  GTO 00

87♦LBL 05
FIX 6  RND  "⊢= ?"
ARCL X  AVIEW  RDN  RTN
```

```
01♦LBL "ISOMET"
FIX 3  SF 21

04♦LBL 00
"X↑ Y↑ Z"  PROMPT  X<> Z
STO T  X<>Y  ST- T  +  2
ST/ T  /  +  X<>Y  3
SQRT  *  CLA  ARCL X
"⊢  "  ARCL Y  AVIEW
GTO 00  END
```

```
95♦LBL 06
"X0_Y0_Z0"  PROMPT
STO 12  RDN  STO 11  RDN
STO 00  "PHI ?"  PROMPT
STO 13  "THETA ?"
PROMPT  STO 14  "CHI ?"
PROMPT  STO 15  END
```

Achim Grohs
Auf dem Berlich 26
5000 Köln 1

Butterworth Tief- und Hochpaßfilter

105 Zeilen, 26 Reg., 180 Byte, SIZE 007, HP-41 C/CV

Dieses Programm entstand durch Portierung eines BASIC-Programms aus dem Buch „Nolte-Basic: HF-Rechenprogramme" erschienen als Band 9 in der Reihe „Franzis Computer-Bibliothek". Auch die gezeigten Bilder stammen aus diesem Buch. Für das freundlicherweise bereitgestellte Material möchte ich mich an dieser Stelle bedanken.

Dieses Programm berechnet die Elemente eines passiven Tief- bzw. Hochpaßfilter mit Butterworth-Übertragungscharakteristik.

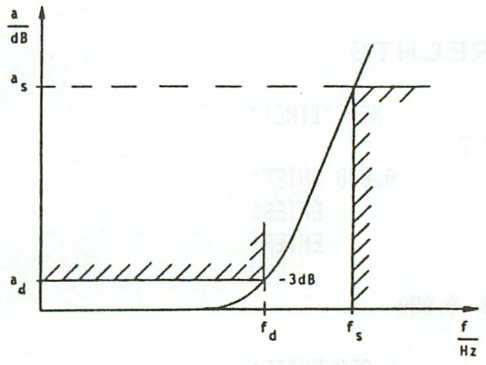


Abb. 1 Dämpfungsverlauf eines Butterworth-Tiefpaßfilters

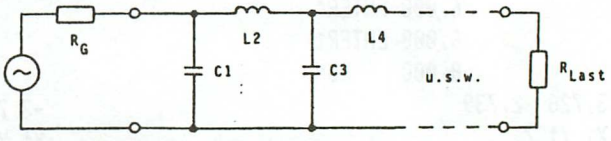


Abb. 2 Prinzipschaltbild eines passiven Butterworth-Tiefpaßfilters

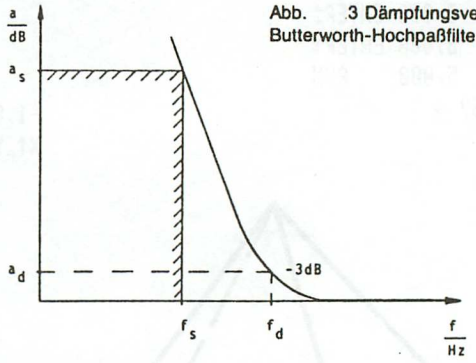


Abb. 3 Dämpfungsverlauf eines Butterworth-Hochpaßfilters

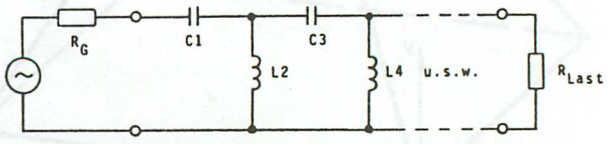


Abb. 4 Prinzipschaltbild eines passiven Butterworth-Hochpaßfilters

Das Programm verwendet die Flag 01 und 29 sowie die Datenregister 00 bis 06 für interne Aufgaben.

Bedienung:
<XEQ> „BTP" für Butterworth Tiefpaßfilter
bzw.
<XEQ> „BHP" für Butterworth Hochpaßfilter

Eingabe des Ein- und Ausgangswiderstands „R0" <R/S> [Ohm]
Eingabe der Betriebsfrequenz „F0" <R/S> [Hz]
Eingabe der Sperrfrequenz „FS" <R/S> [Hz]
Eingabe der Dämpfung bei Sperrfrequenz „A" <R/S> [dB]
Anschließend wird ausgegeben, wie viele Elemente benötigt werden und dann die Werte der Elemente.

```
Beispiel 1 :
     XEQ "BTP"
     R0 =?
      50,0000 RUN
     F0 =?
      145+06 RUN
     FS =?
      430+06 RUN
     A =?
      40,0000 RUN
     N=5
     RUN
     C(1)=13,57E-12 F
     RUN
     L(2)=88,80E-9 H
     RUN
     C(3)=43,90E-12 F
     RUN
     L(4)=88,80E-9 H
     RUN
     C(5)=13,57E-12 F
     RUN

Beispiel 2 :
     XEQ "BHP"
     R0 =?
      50,0000 RUN
     F0 =?
      430+06 RUN
     FS =?
      145+06 RUN
     A =?
      60,0000 RUN
     N=7
     RUN
     C(1)=16,63E-12 F
     RUN
     L(2)=14,84E-9 H
     RUN
     C(3)=4,108E-12 F
     RUN
     L(4)=9,253E-9 H
     RUN
     C(5)=4,108E-12 F
     RUN
     L(6)=14,84E-9 H
     RUN
     C(7)=16,63E-12 F
     RUN
```

```
01♦LBL "BTP"
02 SF 01
03 GTO 00
04♦LBL "BHP"
05 CF 01
06♦LBL 00
07 RAD
08 CF 29
09 "R0"
10 XEQ 14
11 STO 01
12♦LBL 01
13 "F0"
14 XEQ 14
15 STO 02
16 RCL 01
17 *
18 PI
19 *
20 1/X
21 STO 05
22 "FS"
23 XEQ 14
24 STO 00
25 "A"
26 XEQ 14
27 RCL 00
28 RCL 02
29 FC? 01
30 X<>Y
31 X>Y?
32 GTO 01
33 /
34 LOG
35 20
36 *
37 /
38  E
39 STO 06
40 +
41 INT
42 STO 04
43 "N="
44 FIX 0
45 ARCL X
46 PROMPT
47 -2
48 STO 03
49♦LBL 02
50 RCL 05
51 RCL 06
52 .5
53 -
54 RCL 04
55 /
56 PI
57 *
58 SIN
59 FS? 01
60 *
61 FS? 01
62 GTO 00
63 4
64 *
65 /
66♦LBL 00
67 RCL 03
68 "L("
69 X<0?
70 "C("
71 FIX 0
72 ARCL 06
73 "┣)="
74 ENG 3
75 ARCL Y
76 X<0?
77 "┣ F"
78 X>0?
79 "┣ H"
80 PROMPT
81 -1
82 ST* 03
83 RCL 01
84 RCL 03
85 Y↑X
86 ST* 05
87  E
88 ST+ 06
89 RCL 06
90 RCL 04
91 X<Y?
92 GTO 00
93 GTO 02
94♦LBL 14
95 "┣ =?"
96 PROMPT
97 RTN
98♦LBL 00
99 CLST
100 CLA
101 CF 01
102 SF 29
103 DEG
104 FIX 4
105 END
```

Werner Meschede
CCD 2670

Summender HP 41

Antwort auf Hilferuf in Prisma 6/87 S. 61

Nicht nur André Gerards aus Nettetal, sondern auch vielen anderen CX-Besitzern sind schon die eigenartigen Geräusche aufgefallen, die ihr Liebling hin und wieder von sich gibt.

Die Ursache dieses mehr oder weniger lauten Summens bis Zischens ist die Ausführung von sogenannten „synthetischen" Tönen, die den Rechner mit einem nicht gelöschten (CPU-internen) Tonregister zurücklassen.

Um es gleich vorweg zu nehmen: die Abhilfe ist denkbar einfach: die Ausführung von BEEP oder einem „normalen" TONE 0 bis 9 genügt und schon ist wieder Ruhe!

Die Erklärung für den Effekt ist etwas weniger einfach, dafür aber interessant. (Alle, die nicht gerne Falten auf der Stirn kriegen, können jetzt das Heft weglegen)

Hewlett Packard haben beim Schreiben des 41er-Betriebssystems genau 10 einzelne Töne vorgesehen. Für eben diese Töne befinden sich in ROM 1 in einer Tabelle 10 „Frequenzwerte", die von der TONE-Funktion ausgelassen werden. Wird nun der CPU z.B. 10 als Wert für TONE vorgelegt, greift sich die TONE-Funktion einfach den ersten der Tabelle folgenden Maschinensprachbefehl und interpretiert diesen als „Frequenzwert". Alle offiziellen Werte sind ungerade und hinterlassen das Tonregister leer, das heißt, mit Inhalt 00. Die der Tabelle folgenden Befehle besitzen nun zu einem großen Teil geradezahlige Codes, sodaß das Tonregister am Schluß des „Tons" nicht auf Null gesetzt wird. Dadurch liegt dann ständig etwas „auf der Leitung" und der Piezo-Buzzer zuschelt sich eins.

In der amerikanischen Literatur wird zwischen „guten" und „bösen" Tönen unterschieden. Ganz klar, gute Töne sind solche mit ungeraden, böse jene mit geraden Opcode.

Wer einen (Dis)Assembler besitzt, kann die Opcodes ab Ende der Tabelle auf Adresse 16FC (mit dem Opcode 1B0, entsprechend dem TONE „10") auslesen und selbst entscheiden, ob es ein „guter" oder ein „böser" Ton ist.

Je nach ROM-Revision des ROMs 1 ändert sich auch die Tonhöhe und -dauer eines bestimmten TONEs. Für die ROM-Revision F (sie wird im CAT 2 mit TIME 2C angegeben) sind die „guten" Töne (alle Angaben dezimal):

0	1	2	3	4	5	6	7	8	9	15
17	19	23	25	29	34	40	42	46	48	51
52	55	80	85	86	89	105	106	117	119	

A. Jakob
St. Georgen-Str. 219
CH-9011 St. Gallen

Sortieren

:*SORT,*SORTFL, DIRa, INHVERZ

*SORT	Sortieren allgemein
*SORTFL	
DIRa	Sortieren des Directory
INHVERZ	Erstellung eines alphabetischen Inhaltsverzeichnisses mit Seitenzahl

HP-41 CX
Drucker
Extended I/O Modul
CCD Modul
Speichererweiterungsmodule, je nach Sortiervolumen

Zum Sortieren von Alpha (A) oder numerischen Daten(N) gibt es Programme, siehe Lit. (1).

Das CCD-Modul bietet die Funktionen SORT und SORTFL an:
Sort zum Sortieren in den Registern des Rechner-s,
SORTFL zum Sortieren in einem N-Datenfile im Extended Memory (EM), jedoch jeweils nur bis zu ALENG=6.
Programm- und Filenamen können jedoch bis zu sieben Zeichen lang sein. DATCL und DATACLR erscheinen also nur als DATACL.

Um A-Ketten *W* 6 sortiert aufzulisten gibt es zwei Möglichkeiten:
*SORT benützt die Funktion SORT,
*SORTFL die Funktion SORTFL.

Die unten beschriebenen Programme sind lauffähig, müssen aber vom Benutzer auf seine eigenen Verhältnisse (Geräte in der Schleife) abgestimmt werden! (Mehr als ein Drucker oder Speichergerät in der Schleife vorhanden).

	*SORT	*SORTFL
Für eine genügende Anzahl von Registern im Rechner sorgen	X	
Zeile 004 anpassen	X	
Zeile 004 und 007 anpassen		X
PLNG, gepackt	119	132
Zeitdauer in sec für FLLENG 11, ohne Eingaberoutine	11,3	13,4
LBL 05 nicht im RUN-Modus durchtasten, sonst MEMORY LOST		X !

Die eingegebenen A-Ketten können in diesen Programmen bis 48 lang sein.

Nachteil: Sind zwei A-Ketten >6, so werden sie beim Sortieren in der Reihenfolge ausgedruckt, wie sie eingegeben wurden. Dies gilt auch für das Programm DIRa.

DIRa
Für dieses Sortierprogramm wurde die Methode *SORT benützt. Die alphabetische Auflistung des Directorys hat den Vorteil, daß man nicht den ganzen DIR-Ausdruck durchsuchen muß, um eine bestimmte Aufzeichnung zu finden. (Allerdings kann man auch durch FLNAME/FLLENG den Rechner suchen lassen).

Im nachfolgenden Programm wird vorausgesetzt, daß nur ein Drucker und ein Speichergerät in der Schleife vorhanden sind. Andernfalls ist es vor LBL 01 abzuändern.

Es ist für 250 Einträge ausgelegt und berücksichtigt nicht, wenn das EM voll ist.

Nach der Eingabe wird DIRX im Display angezeigt.

Am Endes des Programms wird das EM gelöscht.

14.02.1988

DIRECTORY
NOTEN

Name	TYPE	REGS

INHVERZ
Wem das Inhaltsverzeichnis von PRISMA zu spät erscheint, der kann sich selber eines anfertigen. Da PRISMA DIN A4 hat, so ist dieses Format einzustellen. F28 ist zu setzen und FIX 2 einzustellen. Für genügend Speicherplatz im Rechner und im EM ist zu sorgen!
Bei DIN A4 kann der Text 75,
bei DIN A5 45 lang sein.
Ist der Text jedoch länger, so weist der Rechner diese Eingabe zurück, und der Text ist kürzer zu fassen!
Die Seitenzahl kann bis zu 4-stellig sein.
Nach Beendigung der Eingabe ist bei Abfrage des Textes R/S zu drücken.

FLAG	CLEAR	SET
00	A4	A5
01	CF17	SF17
02	1.	2. Durchgang in *05

Literatur: u.a. J.S. Dearing, Tricks, Tips und Routinen für Taschenrechner der Serie HP-41, deutsche Ausgabe von H. Dalkowski, S. 74ff, Heldermann Verlag, Berlin 1984

PRISMA, Heft 1 1987, Serie 40 betreffend

Inhalt:


```
01♦LBL "*SORTFL"
02 SF 27
03 "A"
04 25
05 CRFLAS
06 "N"
07 10
08 CRFLD
09♦LBL 00
10 "A"
11 RCLPTA
12 CF 23
13 CLA
14 PMTA
15 FC? 23
16 GTO 02
17 APPREC
18 ASTO X
19 24
20 ALENG
21 X=Y?
22 XEQ 01
23 "N"
24 RCLPTA
25 RCL T
26 SAVEX
27 GTO 00
28♦LBL 01
29 "FORTS.="
30 PMTA
31 APPCHR
32 RTN
33♦LBL 02
34 "N"
35 RCLPTA
36 RESZFL
37 SORTFL
38 CLX
39 SEEKPT
40 SF 25
41♦LBL 03
42 "N"
43 RCLPTA
44 GETX
45 FC? 25
46 GTO 05
47 ENTER↑
48 "A"
49 CLX
50 SEEKPTA
51 CLA
52 ARCL Y
53 POSFL
54♦LBL 04
55 GETREC
56 OUTA
57 FS? 17
58 GTO 04
59 GTO 03
60♦LBL 05
61 X<> c
62 STO 64
63 X<> c
```

```
01♦LBL "*SORT"
02 SF 27
03 "A"
04 25
05 CRFLAS
06 1,9
07 STO 00
08 SF 25
09♦LBL 00
10 CF 23
11 CLA
12 PMTA
13 FC? 23
14 GTO 02
15 ASTO IND 00
16 APPREC
17 24
18 ALENG
19 X=Y?
20 XEQ 01
21 ISG 00
22 GTO 00
23♦LBL 01
24 "FORTS.="
25 PMTA
26 APPCHR
27 RTN
28♦LBL 02
29 RCLPT
30 2
31 +
32 PSIZE
33 999
34 +
35  E3
36 /
37 SORT
38 1,9
39 STO 00
40 SF 25
41♦LBL 03
42 CLA
43 ARCL IND 00
44 FC? 25
45 GTO 05
46 CLX
47 SEEKPT
48 POSFL
49♦LBL 04
50 GETREC
51 OUTA
52 FS? 17
53 GTO 04
54 ISG 00
55 GTO 03
56♦LBL 05
57 "A"
58 PURFL
```

```
01♦LBL "DIRa"
02 SF 21
03 SF 27
04 CF 29
05 FIX 6
06 DATE
07 CLA
08 ADATE
09 FMT
10 ACA
11 PRBUF
12 FIX 0
13 ADV
14 SF 12
15 "DIRECTORY"
16 ACA
17 FMT
18 PRBUF
19 CF 12
20 CLA
21 PMTA
22 ACA
23 FMT
24 PRBUF
25 24,061
26 ACLX
27 PRBUF
28 "NAME      TYPE "
29 "├     REGS"
30 OUTA
31 PRL
32 250
33 PSIZE
34 EMROOM
35 "a"
36 CRFLAS
37 1,25
38 STO 00
39 CF 21
40♦LBL 00
41 RCL 00
42 VIEW X
43 DIRX
44 X=0?
45 GTO 01
46 ASTO IND 00
47 APPREC
48 ISG 00
49 GTO 00
50♦LBL 01
51 CLD
52 RCLPT
53 2
54 +
55 PSIZE
56 999
57 +
58  E3
59 /
60 SORT
61 1,25
62 STO 00
63 SF 21
64 SF 25
65♦LBL 02
66 CLA
67 ARCL IND 00
68 FC? 25
69 GTO 03
70 CLX
71 SEEKPT
72 POSFL
73 GETREC
74 FLLENG
75 FLTYPE
76 "├   "
77 ARCL X
78 24
79 RCL Z
80 ACAXY
81 PRBUF
82 ISG 00
83 GTO 02
84♦LBL 03
85 PRL
86 RCL 00
87  E
88 -
89 ACX
90 " Einträge"
91 OUTA
92 "a"
93 PURFL
94 32
95 PSIZE
96 .END.
```

```
01♦LBL "INHVERZ"
02 SF 20
03 SF 21
04 CF 25
05 CF 29
06 2.9
07 STO 00
08 CLX
09 STO 01
10 FIX 0
11 SIGN
12 "DIN A4:A5? 45"
13 PMTK
14 X=Y?
15 SF 00
16 "A"
17 EMROOM
18 CRFLAS
19♦LBL 00
20 CF 23
21 SF 27
22 CLA
23 PMTA
24 FC? 23
25 GTO 01
26 ALENG
27 ST+ 01
28  E
29 FS? 00
30 75
31 FC? 00
32 45
33 X<NN?
34 GTO 07
35 ASTO IND 00
36 APPREC
37 24
38 RCL T
39 X=Y?
40 XEQ 06
41 RCLPT
42 INT
43 SEEKPT
44 "SEITE="
45 PROMPT
46 CLA
47 ARCL X
48 INSCHR
49 ISG 00
50 CLX
51 STO 01
52 GTO 00
53♦LBL 01
54 RCL 00
55 INT
56 1999
57 +
58  E3
59 /
60 SORT
61 STO 00
62 CLX
63 SEEKPT
64 SF 25
65♦LBL 02
66 GETREC
67 FC? 25
68 GTO 03
69 OUTA
70 GTO 02
71♦LBL 03
72 SIGN
73 "OK? AE"
74 PMTK
75 CLA
76 X≠Y?
77 ED
78 "E&k1SE&l1L◢"
79 ACA
80 SF 17
81 "I N H A L T:E=◢"
82 "├E&k0S"
83 OUTA
84 24.095
85 ACLX
86 PRBUF
87 "┼"
88 ACA
89♦LBL 04
90 CLX
91 SEEKPT
92 CLA
93 ARCL IND 00
94 POSFL
95 INT
96 SEEKPT
97 FS? 00
98 8 E1
99 FC? 00
100 48
101♦LBL 05
102 GETREC
103 FS? 17
104 SF 01
105 FC? 02
106 ANUMDEL
107 ALENG
108 ST- Z
109 RDN
110 SF 17
111 OUTA
112 SF 02
113 FS?C 01
114 GTO 05
115 CF 02
116 CLA
117 ACAXY
118 PRBUF
119 ISG 00
120 GTO 04
121 CF 00
122 CF 17
123 "A"
124 PURFL
125 STOP
126♦LBL 06
127 "FORTS="
128 PMTA
129 ALENG
130 ST+ 01
131  E
132 FS? 00
133 75
134 FC? 00
135 45
136 X<NN?
137 GTO 07
138 APPCHR
139 24
140 RCL T
141 X=Y?
142 GTO 06
143 RTN
144♦LBL 07
145 "REC TOO LONG"
146 PROMPT
147 DELREC
148 CLX
149 STO 01
150 GTO 00
```

INHVERZ

```
078:FB 1B 26 6B 31 53 1B
 26 6C 31 4C 0E
081:FF 49 20 4E 20 48 20
 41 20 4C 20 54 3A 1B 3D
 0D
082:F6 7F 1B 26 6B 30 53

087:F3 0A 0A 0F
```

Martin Hochenegger
Heidelberger Landstraße 97
6100 Darmstadt 13

Meßuhr und Schiebelehre . . .

65 Zeilen, 169 Bytes, 25 Regs., SIZE 000, HP-41C, X-I/O, IL, PRINTER

Leider hat sich in meinem Programm „Meßuhr und Schiebelehre am HP41 über RS 232" nach einigen Tagen Gebrauch eine Fehlerquelle gezeigt.

Der Fehler war folgender:
Durch das gesetzte Flag 17 wartete der HP41 nicht, bis alle Datenbytes im IL-Speicher waren, sondern er brach die Übernahme vorzeitig ab. Dadurch entstanden zum Teil falsche Werte, wenn die Zahl 2 im Meßwert enthalten war.

Die Verbindungen der Leitungen 2 und 3 zwischen Interface und Meßgeräte erfolgt so, daß man die weniger als −3 Volt führende Leitung des Meßgerätes-Interfaces mit dem Pin 3 des HP-IL Interfaces verbindet und den Pin 2 vom IL-Interface mit dem dann noch freien Pin 2 oder 3 des Meßgerätes, falls das Meßgerät vom HP41 angesteuert werden soll.

In meinem Anwendungsfall sieht es wie folgt aus:
Die von 4 Schiebelehren-Herstellern gemeinsam verwendete Sylvac-Elektronik auf der Schiebelehre gibt nur ein Low-power-Signal ab, das durch ein spezielles Interface auf den genormten RS232-Pegel gebracht werden muß. Dieses Interface hat einen Fußschalteranschluß mit dem die jeweilige Datenübergabe auf den Signalpin laut Anschlußbild des RS232-Steckers (siehe Heft 1/88) veranlaßt wird. Weiterhin kann das Datenformat in gewissen Grenzen variiert werden.

Die Meßuhr kann zwar direkt über ihre Tastatur bedient werden, es ist aber besser, sie vom HP41 aus fernzusteuern, weil nur dann der Meßwert ohne Erschütterung fehlerfrei gesendet wird. Im Fernsteuerbetrieb erwartet die Meßuhr auf Pin 3 einen Ein-Byte-Befehl ohne LF und CR, sie quittiert diesen im fehlerfreien Betriebsfalle als gleichen Einbyte-Wert auf Pin 2 auch wieder ohne LF und CR. Mit den Befehlen „OUTXB" und „INXB" wickelt der HP41 diesen Befehlsaustausch ab. Die Datenbytes des Meßwertes folgen dem Quittungsbyte sofort ohne ein Trennzeichen und werden mit CR abgeschlossen.

Ein 2-poliger, handbetätigter Umschalter legt den jeweils benutzten RS232-Ausgang auf den HP82164-RS232-Eingang.

Nun zum Programm. Es erwartet auf dem IL-Platz 1 den Drucker und auf dem IL-Platz 2 das HP82164-Interface. Mit dem Label RS2 wird der Betrieb aufgenommen und das HP-Interface auf die Kommunikation mit der Meßuhr eingestellt. Da sich die beiden Meßgeräte nur in der Baud-Rate unterscheiden, muß beim Wechsel der Meßzeuge nur die Baud-Rate und die Datenleitung umgeschaltet werden. Dazu dienen die Label „UHR" bzw. „SCH"iebelehre. Beim Arbeiten mit der Schiebelehre wird mit dem Label „DAS" (**DA**ten **S**chiebelehre) das IL-Interface auf vorhandene Daten im Empfangsregister abgefragt und wenn diese nach der Fußtasterbetätigung da sind mit der Zeile 31 in das X-Register übernommen. Mit dem Label „DAU" (**DA**ten-**U**hr) wird ein Meßwert von der Meßuhr abgerufen.

In der Zeile 37 wird der Code – hier die Zahl 50 – für eine Meßwertübernahme bereitgestellt und mit Zeile 38 gesendet. Mit der Zeile 39 wird der von der Meßuhr quittierte Steuerbefehl wieder aus dem IL-Empfangsspeicher entnommen und in der Zeile 40 mit dem gesendeten verglichen. Im Fehlerfalle wird der Transferspeicher gelöscht (Zeile 42-46) und der Datenabruf wiederholt. Bei der richtigen Antwort der Meßuhr verzweigt das Programm zum Label 02 und holt, nachdem die Zeile 48 den Programmablauf verzögert hat, den Meßwert ins X-Register. Weitere Meßwerte müssen mit dem erneuten Aufruf der Programme „DAS" bzw. „DAU" in den HP41 geholt werden. Da die Meßwerte-Verarbeitung individuell ist, enden die Musterprogramme mit RTN. Da ich bei früheren Programmversionen Timing-Probleme hatte, bei gesetztem Flag 17 wurde der Datensatz wegen der geringen Baudrate der Meßuhr nur teilweise aus dem IL-Speicher geholt, stellte ich fest, daß die Zeile 48 den Programmablauf gerade so viel verzögert, daß die Daten voll im IL-Speicher sind, ehe IND mit der Ausführung fertig ist.

Weil die Meßuhrfunktionen auch über die RS232-Schnittstelle eingestellt werden können, wurde der Programmteil „UHS" (**UH**r **S**teuern) geschaffen. Mit dem PROMT von Zeile 55 kann der jeweilige Befehlscode als Dezimaläquivalent eingetippt und mit R/S abgeschlossen werden. Die Zeile 56 sendet den Befehl, die Zeile 57 holt die Quittung in den HP41 und der prüft in der Zeile 58 auf die richtige Ausführung und zeigt dieses in den Zeilen 62 bis 64 an.

Wenn zwischen den Meßwert-Übernahmen der Drucker benötigt wird, dann ist dieser mit „1 SELECT" zu aktivieren. Alle Programmteile enthalten deshalb zu Beginn das „2 SELECT".

Die Programmteile werden vorzugsweise bestimmten Tasten im User Betrieb zugeordnet, was ja keinen zusätzlichen Speicherplatz benötigt.

Die Synthetiker können die Zeilen 16 bis 18 als „SW1;SS1" zusammenfassen, das Trennzeichen muß das Semikolon – Dez 59 – sein.

Erhard Ristow
Hersbruckerstraße 188
8500 Nürnberg 30

41 CX Speed UP

SPEED UP für alle neuen HP41CX

Der Aufbau aller neuen HP41CX (zu erkennen an dem runden Displayrahmen) wird durch den internen Aufbau etwas erschwert. Hat man den Rechner geöffnet, sieht man eine kleine auf Beinchen aufgelötete sog. PIGGYPACK-Platine, die auf der linken Seite entlötet wird. Man kann sie dann nach oben biegen und hat so Zugang auf die Oszillator-Bausteine. Man sieht ein kleines rechteckiges, relativ hohes Bauteil mit der Aufschrift 820J, hierbei handelt es sich um eine Spule von 82 mikroH. Man muß nun die Leiterbahnen verfolgen und den Kondensator in der Nähe suchen, der dieser Spule parallelgeschaltet ist. Das ist ein kleiner Chipkondensator in SMD-Technik, meist ohne Aufschrift. Man entlötet ihn und setzt einen kleinen Keramikkondensator (Werkstoff NPO – schwarze Kennung –) oder einen Styroflexkondensator (hohe Güte, kleiner Verlustfaktor, geringe Eigenkapazität) ein. Der Wert sollte etwa um 22pF liegen. Damit lassen sich Geschwindigkeiten von etwa 2.3facher normaler Rechengeschwindigkeit erreichen. Man kann aufgrund der höheren Integration des Rechners ohne weiteres auch auf das 3fache gehen. Allerdings liegen hier noch keine Erfahrungswerte vor. Das SPEED UP muß auch hier wieder schaltbar ausgeführt werden, damit der Magnetkartenleser einwandfrei funktioniert. Ansonsten gilt all das, was bisher schon über SPEED UP's gesagt wurde. Für Rückfragen stehe ich gern zur Verfügung.

Ralf Rosche
Berner Heerweg 155 B
2000 Hamburg 72
Tel.: 040/6 45 19 20

oder
Rainer Gillmann
Möllers Kamp 11
2050 Hamburg 80
Tel.: 040/7 26 12 87

HAPPY TUNING

Funktionswerte von Polynomen

von Herrn André Gerad Rubrik „Serie 40“ im PRISMA 6/87 Seite 61

Der Autor beklagt sich, daß das Programm POLY des MATH 1C für den HP 41 unrichtigte Resultate für die Funktionswerte liefert und führt dafür beispielhaft das Polynom

1. $P(x) = 3x^5 - 2x^4 + x^3 - x - 1$

an. Es verhält sich nun freilich so, daß das genannte Programm nur für Polynome anwendbar ist, bei denen das Argument mit dem höchsten Exponenten den Faktor 1 mit sich führt, mithin also entsprechend normiert ist. Löblicherweise geht dies aus dem MATH 1C-Modul beigelegten Handbuch durchaus unmißverständlich hervor.

Normiert man folglich das oben angegebene Polynom entsprechend, so sind **beide Seiten der Gleichung** (1) durch 3 zu dividieren:

2. $\frac{P(x)}{3} = x^5 - \frac{2}{3}x^4 + \frac{1}{3}x^3 - \frac{1}{3}x - \frac{1}{3}x$

Dieser Sachverhalt sollte übrigens spätestens bei der Ausführung des genannten Programms deutlich werden, denn nach Eingabe der Größe des höchsten Exponenten der Polynoms (hier also „5“) fragt das Programm den Faktor für das Glied mit dem nächstniederen Exponenten ab.

Gibt man nun als das **Polynom** (2) in den Rechner ein, so liefert dieser für das Argument $-0{,}65$ ganz richtig den Wert $-0{,}4432$. Der Autor scheint freilich übersehen zu haben, daß zur Bestimmung von Gleichung (2) **linke und rechte Seite** durch den Faktor des Argumentes mit dem höchsten Exponenten zu dividieren waren (in diesem Falle also durch 3). Folglich sind alle so durch den Rechner gewonnenen Ergebnisse die **Funktionswerte** wieder mit 3 zu multiplizieren, was im vorliegenden Beispiel ganz brav zu dem richtigen Resultat

3. $\frac{P(-0{,}65)}{3} = -0{,}4432$

$\rightarrow P(-0{,}65) = 0{,}4432 \cdot 3 = -1{,}3297$

führt.

Nachdem nun derart klargestellt ist, daß das genannte Programm exakt funktioniert, erscheint natürlich arg zweifelhaft, daß „einige Werte“. . „korrekt bestimmt“ werden. Eher will ich vermuten, daß bei der **Nullstellenbestimmung** innerhalb des gleichen Programms kein Fehler auftritt, was ja angesichts des aufgezeigten Sachverhaltes auch ganz selbstverständlich ist und was der Autor mit an Sicherheit grenzender Wahrscheinlichkeit auch gemeint haben wird.

Insofern werde ich persönlich es vorziehen, auch künftig mit dem im MATH-Modul vorhandenen Programm Funktionswerte zu berechnen, was natürlich gegebenenfalls die Mühe des vorherigen Normierens wie oben beschrieben beinhaltet, andererseits aber keinen zusätzlichen Speicherbedarf erfordert. Keinesfalls aber darf die Aussage im Raume stehen bleiben, das genannte Programm arbeite nicht einwandfrei. Einräumen will ich immerhin, daß das vom Autor angegebene (von mir allerdings nicht überprüfte) Programm geeignet wäre, dem Anwender das Normieren zu ersparen – auf Kosten des Speicherplatzes, wie gesagt.

Zum Schluß noch ein Hinweis für alle: Programme zur Filterberechnung (die auch durchaus industriellen Standards genügen) finden sich im HP-Lösungsbuch „Electrical Engineering“ (vgl. 6/87 S. 62).

Dr. Christof Rohner
Aventinstraße 12
8000 München 5

M-Code Druck Utility

LFX (**L**ine **F**eed by **X**) und PO (**P**aper **O**ut)

Drucker-Utilities in MCODE

Hardware: HP41, RAMBOX o.ä., Drucker

Die beiden in PRISMA 87/6 erschienenen Programme FLINFO und CONT haben mich veranlaßt, in meinen Maschinenprogrammen zu suchen und auch etwas zum (hoffentlich wachsenden!) MCODE-Geschehen beizutragen.

Doch zuerst eine Anmerkung zu FLINFO: Die angegebenen Ansprungadressen in das RAMBOX-Betriebssystem gelten nur für die Revision WW-3F! (mit PGSUM ermitteln.) Die neueste (?) Revision WW-3H hat andere Startadressen, wie man schon durch einen PRFAT-Ausdruck erkennen kann. Also Vorsicht!!

Jetzt eine Beschreibung meines Programms:

LFX: (**L**ine **F**eed by **X**) Diese Funktion bewirkt bei angeschlossenem Drucker (egal ob 82143 A oder ein IL-Drucker) die im X-Register angegebene Zahl an Zeilenvorschüben (max. 96, Zeile 6, HEX 060). Mehr als 96 Zeilenvorschübe sind nicht sinnvoll, da ist Form Feed besser. Es wird nur der absolute Integeranteil der Zahl in X verwendet. Bei nicht angeschlossenem Drucker wird nichts ausgeführt (ähnlich ADV).

PO: (**P**aper **O**ut) Diese Funktion entspricht von der Wirkung dem PPC-Modul-Programm TPO. Sie bewirkt 5 Zeilenvorschübe und wurde für den Thermodrucker geschrieben, um den Papierstreifen nach der letzten bedruckten Zeile abreißen zu können.

Fehlermeldungen für LFX:
ALPHA DATA:
Alpha Daten im X-Register
DATA ERROR:
Zahl in X größer 96

Ich hoffe, der ein oder andere kann die Programme gebrauchen, für Verbesserungsvorschläge bin ich dankbar.

Thomas Mareis (2886)
Cranachstr. 3
8000 München 40

```
FNC: LFX
     CFC5 READ  3(X)
     CFC6 NC XQ BCDBIN
     CFC8 A=C  S&X
     CFC9 LDI
     CFCA HEX  060
     CFCB A<>C  ALL
     CFCC A=A-C  S&X
     CFCD C GO ERRDE
     CFCF JNC +05 CFD4


FNC: PO
     CFD2 LDI
     CFD3 HEX  005  E
     CFD4 C=C+1  S&X
     CFD5 B=0  ALL
     CFD6 C=C-1  S&X
     CFD7 ?C#0  S&X
     CFD8 NC RTN
     CFD9 C<>B  S&X
     CFDA NC XQ 6FDD
     CFDC C=B  S&X
     CFDD JNC -07 CFD6
```

Koordinatenberechnung

249 Zeilen, 308 Bytes, 44 Regs., SIZE 011, HP-41C

Das Programm dient dazu, die Koordinaten eines Himmelsobjekts (Komet, Kleinplanet etc.) anhand von 3 Sternen zu berechnen. Diese 3 Sterne müssen ein Dreieck bilden, in dem sich das Objekt unbekannter Koordinaten befindet.

1. Man benötigt Rektaszension und Deklination der 3 Anhaltssterne, die man um ihre Eigenbewegung korrigiert. (Bei Sternen mit hoher Deklination müssen noch andere Korrekturen angewendet werden.)

2. Man zeichnet die 3 Sterne in ein möglichst großes Koordinaten-System, dessen Ursprung das Objekt unbekannter Koordinaten ist. Nun schreibt man sich die X/Y-Werte der 3 Sterne auf.

3. Eingabe der Daten:

   Stern 1 X1/Y1/D1/R1 (s. Beispiel)
   Stern 2 X2/Y2/D2/R2
   Stern 3 X3/Y3/D3/R3

Mit den nun erhaltenen Daten kann man, mit den entsprechenden Programmen, die schon im PRISMA veröffentlicht wurden, die Bahnelemente des Objekts berechnen.

```
01*LBL "KB"
02 "X1?"
03 PROMPT
04 STO 00
05 "Y1?"
06 PROMPT
07 STO 01
08 "X2?"
09 PROMPT
10 STO 02
11 "Y2?"
12 PROMPT
13 STO 03
14 "X3?"
15 PROMPT
16 STO 04
17 "Y3?"
18 PROMPT
19 STO 05
20 RCL 02
21 *
22 RCL 04
23 RCL 03
24 *
25 -
26 STO 06
27 RCL 04
28 RCL 01
29 *
30 RCL 00
31 RCL 05
32 *
33 -
34 STO 05
35 RCL 00
36 RCL 03
37 *
38 RCL 02
39 RCL 01
40 *
41 -
42 STO 04
43 RCL 05
44 RCL 06
45 +
46 +
47 STO 03
48 1/X
49 RCL 06
50 *
51 STO 00
52 RCL 05
53 RCL 03
54 /
55 STO 01
56 RCL 04
57 RCL 03
58 /
59 STO 02
60 "D1?"
61 PROMPT
62 HR
63 15
64 *
65 STO 05
66 "D2?"
67 PROMPT
68 HR
69 15
70 *
71 STO 07
72 "D3?"
73 PROMPT
74 HR
75 15
76 *
77 STO 09
78 +
79 +
80 3
81 /
82 STO 03
83 "R1?"
84 PROMPT
85 HR
86 STO 06
87 "R2?"
88 PROMPT
89 HR
90 STO 08
91 "R3?"
92 PROMPT
93 HR
94 STO 10
95 +
96 +
97 3
98 /
99 STO 04
100 COS
101 RCL 06
102 SIN
103 *
104 RCL 06
105 COS
106 RCL 04
107 SIN
108 *
109 RCL 05
110 RCL 03
111 -
112 COS
113 *
114 -
115 ASIN
116 STO 06
117 RCL 08
118 SIN
119 RCL 04
120 COS
121 *
122 RCL 08
123 COS
124 RCL 04
125 SIN
126 *
127 RCL 07
128 RCL 03
129 -
130 COS
131 *
132 -
133 ASIN
134 STO 08
135 RCL 10
136 SIN
137 RCL 04
138 COS
139 *
140 RCL 10
141 COS
142 RCL 04
143 SIN
144 *
145 RCL 09
146 RCL 03
147 -
148 COS
149 *
150 -
151 ASIN
152 STO 10
153 RCL 04
154 COS
155 RCL 05
156 RCL 03
157 -
158 SIN
159 *
160 RCL 06
161 COS
162 /
163 ASIN
164 STO 05
165 RCL 04
166 COS
167 RCL 07
168 RCL 03
169 -
170 SIN
171 *
172 RCL 08
173 COS
174 /
175 ASIN
176 STO 07
177 RCL 04
178 COS
179 RCL 09
180 RCL 03
181 -
182 SIN
183 *
184 RCL 10
185 COS
186 /
187 ASIN
188 STO 09
189 RCL 02
190 *
191 RCL 01
192 RCL 07
193 *
194 +
195 RCL 00
196 RCL 05
197 *
198 +
199 STO 05
200 RCL 06
201 RCL 00
202 *
203 RCL 08
204 RCL 01
205 *
206 +
207 RCL 10
208 RCL 02
209 *
210 +
211 STO 07
212 SIN
213 RCL 04
214 COS
215 *
216 RCL 05
217 COS
218 RCL 07
219 COS
220 *
221 RCL 04
222 SIN
223 *
224 +
225 ASIN
226 STO 00
227 HMS
228 "DKL.="
229 ARCL X
230 PROMPT
231 RCL 00
232 COS
233 1/X
234 RCL 05
235 SIN
236 RCL 07
237 COS
238 *
239 ASIN
240 RCL 03
241 +
242 STO 01
243 15
244 /
245 HMS
246 "R.A.="
247 ARCL X
248 PROMPT
249 .END.
```

Stephan Trzeciak
Fritz-Reuter-Straße 24
2190 Cuxhaven

Akkukiller?

Angeregt durch sehr unterschiedliche eigene Erfahrungen und die PRISMA-Veröffentlichung 84.9.32 habe ich mir die Akkus für den HP41 und den Drucker bzw. das Magnetband-Laufwerk mal genauer vorgenommen.

In den Betriebsanleitungen zum Akkusatz HP 82120, zum Drucker und IL-Laufwerk steht, daß die Akkus 12 bis 14 Stunden zu laden sind, wenn man die volle Kapazität erhalten will. Da im Drucker der Akku HP 82033 1200 mAh groß ist, müßte der Ladestrom danach laut der Literatur bei 120 mA liegen. Er beträgt aber in Verbindung mit dem Steckertrafo HP 82066B ca. 270 mA. Dieses ist der Mittelwert aus 8 Geräten, also ein realer Wert und kein Ausreißer. Mit diesem Ladestrom ist der Akku laut der NiCd-Literatur aber schon nach 6,25 Stunden voll geladen. Eine Lade-Entladeprüfung an 4 Akkus HP 82033 bestätigte den Sachverhalt. Ein Überladeschutz ist im Drucker nicht vorhanden. Nun ist mir klar, warum andere den Begriff „Akku-Killer" für den Drucker geprägt haben.

Wer seine Akkus streng nach der Gebrauchsanleitung auflädt oder im Gebrauchsfalle den Drucker dauernd mit dem Steckertrafo puffert, der lädt den Akku schlicht kaputt, der arme Kerl wird dauernd mit einem viel zu hohen Strom thermisch überlastet und muß daher schnell sterben.

Ein Vergessen des Abschaltens nagt aber auch an der Lebensdauer eines Akkus. Wenn die schwächste Zelle entladen ist, wird sie durch den Entladestrom der übrigen Zellen mit falscher Polarität geladen. Das nimmt sie aber sehr übel und quittiert schnell ihren Dienst, wenn es häufiger passiert. Nur in diesem Fall lohnt sich der Ersatz einer einzelnen Zelle des Batteriesatzes, ein so reparierter Akkussatz lebte noch weitere 2 Jahre. Nach ca. 5,5 Stunden ist ein vollgeladener Akku durch den Ruhestrom des Druckers oder Laufwerkes entladen.

Die Untersuchung am Akkusatz HP 82120 für den HP41 zeigte das gleiche Ergebnis. Der ca. 70 mAh große Akkusatz ist auch nach 6,5 Stunden voll geladen, ein weiteres Laden würde ihn nur töten.

Wie die falsche Ladezeit von 12 bis 14 Stunden in die offzielle HP-Betriebsanleitung gelangt ist, habe ich nicht erforscht. Vielleicht sind die Werte für das 110-Volt Netzteil richtig. Es wäre interessant zu erfahren, was HP dazu sagt. Die eigene Erfahrung zeigt, daß die NiCd-Akkus dann sehr lange halten, wenn man sie erst beim Aufleuchten der „BAT"-Anzeige wieder nachlädt, nicht überlädt und das Ausschalten nicht vergißt. Die hohen pulsierenden Entladeströme bis 4 A beim Drucken sollen laut einer Rücksprache mit einem deutschen NiCd-Hersteller und den Angaben in der Literatur nichts ausmachen. Nach 500 Lade-Entladezyklen kann man aber mit dem normalen Ende eines Akkus rechnen.

Wer seine Akkus immer voll geladen in Bereitschaft haben will, der sollte den Dauerladestrom nicht über 1/50 der Nennkapazität hinaus einstellen, also nur mit etwa 20 mA beim Drucker oder 1 mA beim HP 41 puffern, wenn das Gerät selber abgeschaltet ist.

Erhard Ristow
Hersbruckerstraße 188
8500 Nürnberg 30

Akkus erneuern

Ergänzung zu PRISMA 6/87 Seite 33

Liebe PRISMA-Freunde

Akkus erneuern ist gut, Akkus nicht kaputtmachen ist besser.

Die von H. Pfeifer angegebenen Lademethoden sind jedoch wenig Akku-schonend. NiCd-Akkus, auch die mit Sinterzellen, sind nämlich keineswegs so überladefest und unverwüstlich, wie gelegentlich behauptet wird. Eine 15-stündige Ladung kann man ihnen nicht allzuoft zumuten. Nach meinen Messungen ist der Ladestrom bei Verwendung des HP-Ladegeräts 82066 sowohl beim Drucker als auch beim Cassette-Drive im ausgeschalteten Zustand ca. 270 mA, bei zunehmender Ladung etwas abnehmend. Der 15h-Nennladestrom für die HP-Zellen ist aber 120 mA, so daß die Akkus max. 7h am Netz hängen sollten (Schaltuhr verwenden!). Eine kürzere Ladezeit ist besser, da die Lebensdauer der NiCd-Akkus (bei guten Fabrikaten mindestens 500 Zyklen) dadurch überproportional verlängert wird. Noch besser ist es, mit einer Konstantstromquelle von 120 mA 15h, mit Schaltuhr geregelt, zu laden. Bei allen, mir bekannten HP-Geräten mit Akkus, ist eine Gleichstromladung problemlos, man muß nur das Anschlußkabel zwischen Trafo und Gerät aufschneiden und mit Bananensteckern versehen.

Mit freundlichen Grüßen

Dr. Hanspeter Waldmann
Erlenweg 3
7951 Dettingen

CD (Schatz) 1591

Computerclub Deutschland e.V.

Zeile 1 (1- 7) CCD-Barcodes

Zeile 2 (7- 17) CCD-Barcodes

Zeile 3 (17- 26) CCD-Barcodes

Zeile 4 (26- 35) CCD-Barcodes

Zeile 5 (35- 39) CCD-Barcodes

Zeile 6 (39- 48) CCD-Barcodes

Zeile 7 (48- 58) CCD-Barcodes

Zeile 8 (58- 64) CCD-Barcodes

DIMET (Grohs) 1595

Computerclub Deutschland e.V.

Zeile 1 (1- 3) CCD-Barcodes

Zeile 2 (3- 6) CCD-Barcodes

Zeile 3 (6- 11) CCD-Barcodes

Zeile 4 (11- 18) CCD-Barcodes

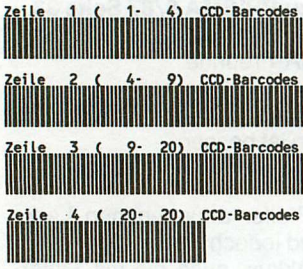
Zeile 5 (18- 29) CCD-Barcodes

Zeile 6 (29- 36) CCD-Barcodes

Zeile 7 (36- 40) CCD-Barcodes

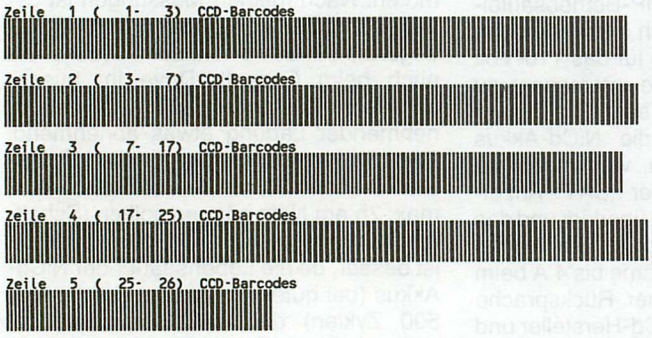
LAME (Hochenegger) 1590

Computerclub Deutschland e.V.



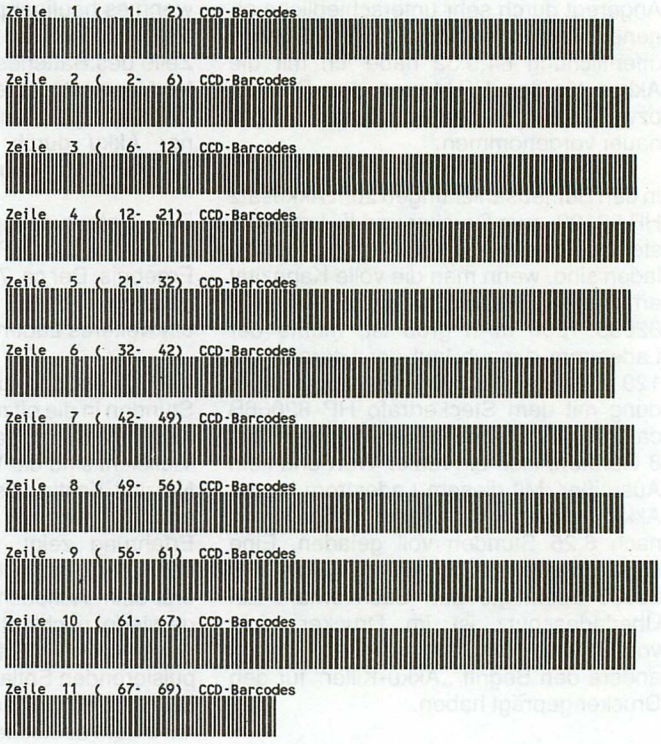
ISOMET (Grohs) 1594

Computerclub Deutschland e.V.



TRANS2 (Grohs) 1592

Computerclub Deutschland e.V.



Clubadressen:

1. Vorsitzender
Prof. Dr. Wolfgang Fritz (125)
Kronenstraße 34
7500 Karlsruhe
GEO1: W.FRITZ

2. Vorsitzender
Erich H. Klee (1170)
Ruhrallee 8
4300 Essen 1
GEO1: E.H.KLEE

Schatzmeister Mitgliederverwaltung
Dieter Wolf (1734)
Pützerstraße 29
6000 Frankfurt 90
☎ 069 / 765912
GEO1: D.WOLF

1. Beisitzer
Werner Dworak (607)
Allewind 51
7900 Ulm
☎ 07304 / 3274
GEO1: W.DWORAK

2. Beisitzer Geowissenschaften
Alf-Norman Tietze (1909)
Thudichumstraße 14
6000 Frankfurt 90
☎ 069 / 7893995
GEO1: A.N.TIETZE

PRISMA-Nachsendedienst
CCD e.V.
Postfach 11 04 11
6000 Frankfurt 1
☎ 069 / 765912

Beirat
Martin Meyer (1000)
Robert-Stolz-Straße 5
6232 Bad Soden 1

Programm-Bibliothek HP-71
Henry Schimmer (786)
Homburger Landstraße 63
6000 Frankfurt 50

Serie 80 Service
Klaus Kaiser (1661)
Mainzer Landstraße 561
6230 Frankfurt am Main 80
☎ 069 / 397852

Beirat MS-DOS Service
Alexander Wolf (3303)
Pützerstraße 29
6000 Frankfurt 90
☎ 069 / 765912

Hardware 41
Winfried Maschke (413)
Ursulakloster 4
5000 Köln 1
☎ 0221 / 131297

Grabau GR7 Interface
Holger von Stillfried (2641)
Alsterkrugchaussee 212
2000 Hamburg 60
☎ 040 / 5116346

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

E-Technik
Werner Meschede (2670)
Sorpestraße 4
5788 Siedlingshausen

Mathematik
Andreas Wolpers (349)
Steinstraße 15
7500 Karlsruhe

Vermessungswesen
Ulrich Kulle (2719)
Schnuckentrift 14
3000 Hannover 51
☎ 0511 / 6042728

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

Regionalgruppe Hamburg
Alfred Czaya (2225)
An der Bahn 1
2061 Sülfeld
☎ 040 / 433668 (Mo.-Do. abends)

Horst Ziegler (1361)
Schüslerweg 18 b
2100 Hamburg 90
☎ 040 / 7905672

Beirat Regionalgruppe Karlsruhe
Stefan Schwall (1695)
Rappenwörtstraße 42
7500 Karlsruhe 21
☎ 0721 / 576756
GEO1:S.SCHWALL

Regionalgruppe Köln
Frank Ortmann (1089)
Okerstraße 24
5090 Leverkusen 1

Regionalgruppe München
Victor Lecoq (2246)
Seumestraße 8
8000 München 70
☎ 089 / 789379

Regionalgruppe Rhein-Main
Andreas Eschmann (2289)
Lahnstraße 2
6096 Raunheim
☎ 61442 / 46642

Beirat
Peter Kemmerling (2466)
Danzigerstraße 17
4030 Ratingen

Beirat
Ulrich Schwaderlap (438)
An den Berken 34
5840 Schwerte 6

Beirat
Günther Schwarz (2658)
Bodelschwinghstraße 34
3408 Duderstadt 1

TRANS3 (Grohs) 1593

Computerclub Deutschland e.V.

Zeile 1 (1- 2) CCD-Barcodes

Zeile 2 (2- 5) CCD-Barcodes

Zeile 3 (5- 12) CCD-Barcodes

Zeile 4 (12- 19) CCD-Barcodes

Zeile 5 (19- 29) CCD-Barcodes

Zeile 6 (29- 34) CCD-Barcodes

Zeile 7 (34- 44) CCD-Barcodes

Zeile 8 (44- 54) CCD-Barcodes

Zeile 9 (54- 65) CCD-Barcodes

Zeile 10 (65- 75) CCD-Barcodes

Zeile 11 (75- 81) CCD-Barcodes

Zeile 12 (81- 86) CCD-Barcodes

Zeile 13 (86- 93) CCD-Barcodes

Zeile 14 (93- 98) CCD-Barcodes

Zeile 15 (98- 106) CCD-Barcodes

Zeile 16 (106- 109) CCD-Barcodes

Zeile 17 (109- 112) CCD-Barcodes

BTP (Meschede) 1596

Computerclub Deutschland e.V.

Zeile 1 (1- 4) CCD-Barcodes

Zeile 2 (4- 9) CCD-Barcodes

Zeile 3 (9- 15) CCD-Barcodes

Zeile 4 (15- 24) CCD-Barcodes

Zeile 5 (24- 32) CCD-Barcodes

Zeile 6 (32- 43) CCD-Barcodes

Zeile 7 (43- 52) CCD-Barcodes

Zeile 8 (52- 62) CCD-Barcodes

Zeile 9 (62- 70) CCD-Barcodes

Zeile 10 (70- 76) CCD-Barcodes

Zeile 11 (76- 82) CCD-Barcodes

Zeile 12 (82- 92) CCD-Barcodes

Zeile 13 (92- 99) CCD-Barcodes

Zeile 14 (99- 105) CCD-Barcodes

INHVERZ (Hochenegger) 1597

Computerclub Deutschland e.V.

Zeile 1 (1- 2) CCD-Barcodes

Zeile 2 (2- 10) CCD-Barcodes

Zeile 3 (10- 12) CCD-Barcodes

Zeile 4 (12- 18) CCD-Barcodes

Zeile 5 (18- 25) CCD-Barcodes

Zeile 6 (25- 32) CCD-Barcodes

Zeile 7 (32- 39) CCD-Barcodes

Zeile 8 (39- 44) CCD-Barcodes

Zeile 9 (44- 52) CCD-Barcodes

Zeile 10 (52- 60) CCD-Barcodes

Zeile 11 (60- 68) CCD-Barcodes

Zeile 12 (68- 73) CCD-Barcodes

Zeile 13 (73- 78) CCD-Barcodes

Zeile 14 (78- 81) CCD-Barcodes

Zeile 15 (81- 81) CCD-Barcodes

Zeile 16 (81- 84) CCD-Barcodes

Zeile 17 (84- 89) CCD-Barcodes

Zeile 18 (89- 97) CCD-Barcodes

Zeile 19 (97- 104) CCD-Barcodes

Zeile 20 (104- 111) CCD-Barcodes

Zeile 21 (111- 118) CCD-Barcodes

Zeile 22 (118- 124) CCD-Barcodes

Zeile 23 (124- 129) CCD-Barcodes

Zeile 24 (129- 136) CCD-Barcodes

Zeile 25 (136- 144) CCD-Barcodes

Zeile 26 (144- 145) CCD-Barcodes

Zeile 27 (145- 151) CCD-Barcodes

DIRA (Hochenegger) 1598

Computerclub Deutschland e.V.

Zeile 1 (1- 4) CCD-Barcodes

Zeile 2 (4- 11) CCD-Barcodes

Zeile 3 (11- 15) CCD-Barcodes

Zeile 4 (15- 21) CCD-Barcodes

Zeile 5 (21- 25) CCD-Barcodes

Zeile 6 (25- 28) CCD-Barcodes

Zeile 7 (28- 29) CCD-Barcodes

Zeile 8 (29- 33) CCD-Barcodes

Zeile 9 (33- 39) CCD-Barcodes

Zeile 10 (39- 47) CCD-Barcodes

Zeile 11 (47- 55) CCD-Barcodes

Zeile 12 (55- 61) CCD-Barcodes

Zeile 13 (61- 69) CCD-Barcodes

Zeile 14 (69- 76) CCD-Barcodes

Zeile 15 (76- 81) CCD-Barcodes

Zeile 16 (81- 90) CCD-Barcodes

Zeile 17 (90- 92) CCD-Barcodes

Zeile 18 (92- 96) CCD-Barcodes

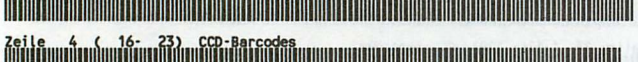
SORTFL (Hochenegger) 1599

Computerclub Deutschland e.V.

Zeile 1 (1- 2) CCD-Barcodes

Zeile 2 (2- 9) CCD-Barcodes

Zeile 3 (9- 16) CCD-Barcodes



Zeile 4 (16- 23) CCD-Barcodes

Zeile 5 (23- 29) CCD-Barcodes

Zeile 6 (29- 34) CCD-Barcodes

Zeile 7 (34- 42) CCD-Barcodes

Zeile 8 (42- 49) CCD-Barcodes

Zeile 9 (49- 57) CCD-Barcodes

Zeile 10 (57- 64) CCD-Barcodes

Zeile 11 (64- 64) CCD-Barcodes

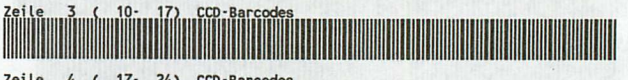
SORT (Hochenegger) 1600

Computerclub Deutschland e.V.

Zeile 1 (1- 3) CCD-Barcodes

Zeile 2 (3- 10) CCD-Barcodes

Zeile 3 (10- 17) CCD-Barcodes



Zeile 4 (17- 24) CCD-Barcodes

Zeile 5 (24- 29) CCD-Barcodes

Zeile 6 (29- 37) CCD-Barcodes

Zeile 7 (37- 44) CCD-Barcodes

Zeile 8 (44- 52) CCD-Barcodes

Zeile 9 (52- 59) CCD-Barcodes

Zeile 10 (59- 59) CCD-Barcodes

KB (Trzeciak) 1601

Computerclub Deutschland e.V.

Zeile 1 (1- 5) CCD-Barcodes

Zeile 2 (5- 11) CCD-Barcodes

Zeile 3 (11- 17) CCD-Barcodes

Zeile 4 (17- 29) CCD-Barcodes

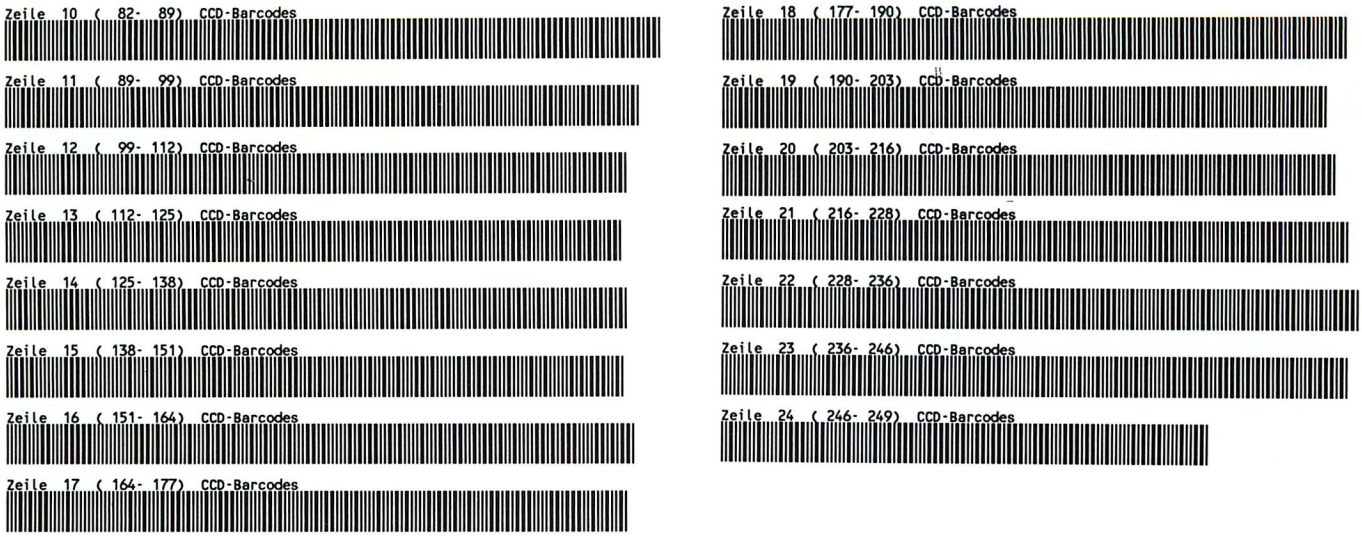
Zeile 5 (29- 42) CCD-Barcodes

Zeile 6 (42- 55) CCD-Barcodes

Zeile 7 (55- 64) CCD-Barcodes

Zeile 8 (64- 72) CCD-Barcodes

Zeile 9 (72- 82) CCD-Barcodes



Fortsetzung von GRAPADAT Seite 18

```
466 RETURN
467 'G3': IF D1 THEN PRINTER IS :%48
468 IF D2 THEN PRINTER IS :%64
469 RETURN
470 'P9': ! Drucker-Pruefung : Papier eingelegt ?
471 P9=SPOLL('%35') @ IF P9<0 OR P9>1200 THEN GOTO 472 ELSE CFLAG 33 @ RETURN
472 IF FLAG(33) THEN GOTO 476
473 DISP @ DISP @ DISP "Drucker prfen ! Papier einlegen und blaue Taste drcken !
"
474 DISP @ DISP "Falls Sie ohne Drucker weiterarbeiten wollen, bitte 'M' drcken
!" @ SFLAG 33
475 DISP @ DISP
476 IF UPRC$(KEY$)="M" THEN POP @ CFLAG 33 @ GOTO 40 ELSE GOTO 471
```

Hinweis:
die HP-71 Umlaute konnten auf dem ThinkJet nicht ausgedruckt werden. Die Programmbibliothek hat jedoch das vollständige BASIC-File "vorrätig".

Dipl.-Ing. Georg K. Heise (CCD 2985)
Lindelbrunnweg 10
6728 Germersheim

Postvertriebsstück
Gebühr bezahlt

D 2856 F

CCD – Computerclub Deutschland
Schwalbacherstraße 50
D-6000 Frankfurt am Main 1

CCD
SSN 0176-8735
PRISMA
März/April 1988 Nr. 2