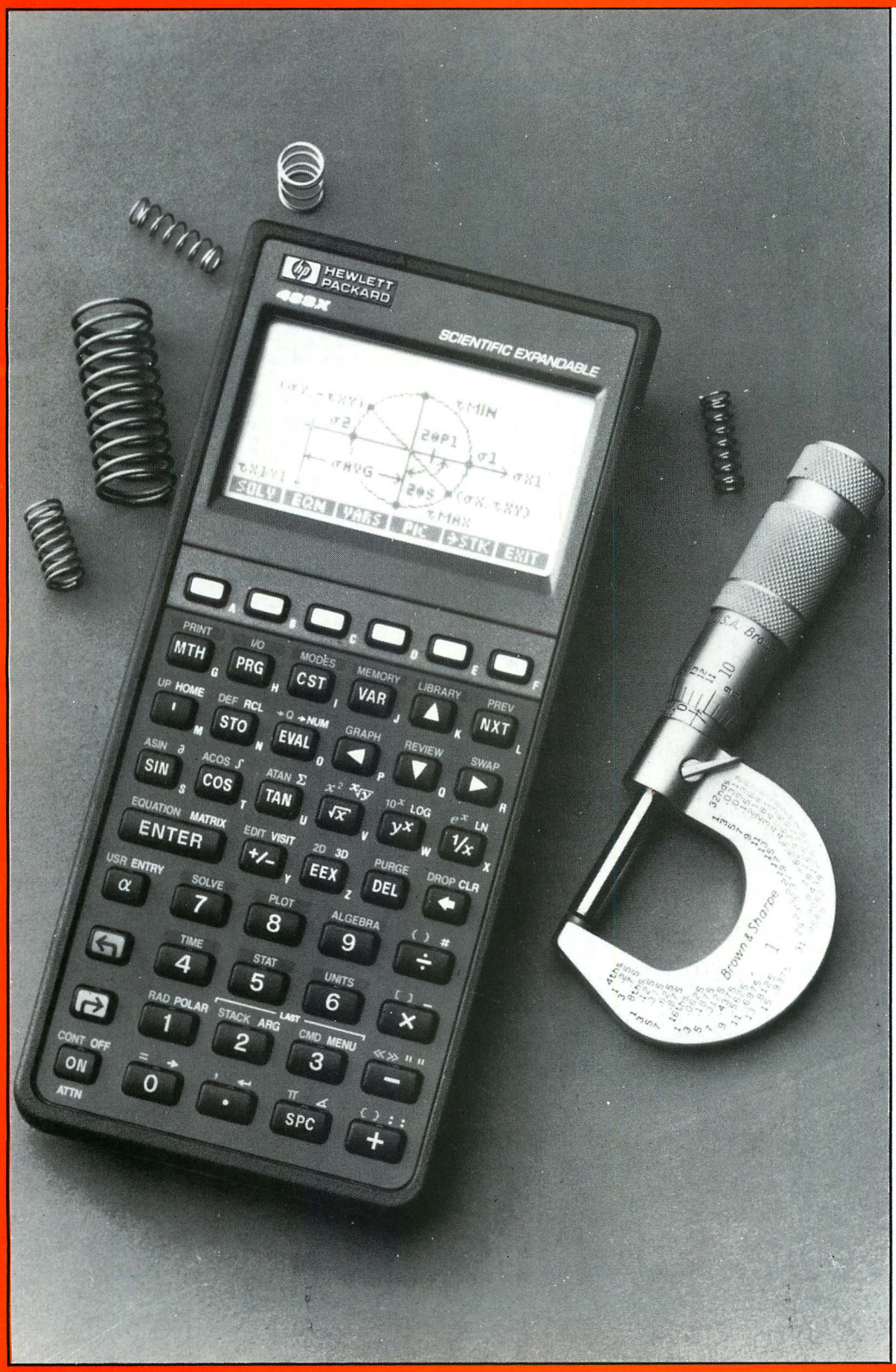
PRISMA

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

Januar / Februar 1990 Nr. 1 **D 2856 F**



Ganz aktuell:
Der neue HP-48 SX .

W&W SOFTWARE PRODUCTS

Bei uns gibt`s keine Probleme, nur Lösungen!

HEWLETT PACKARD COMPUTER PRODUKTE PARTNER-LÖSUNGEN

POSTSCRIPT Lösungen!

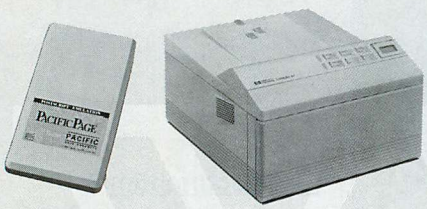
ConoDesk 6000 DM 5.998,-
PostScript PC-Karte mit 2,5MB und Direktanschluß; die z.Zt. schnellste Lösung!

Die PostScript-Komplettlösung
HP LaserJet III + ConoDesk 6000
DM 9.998,-

PACIFIC DATA PRODUCTS

PacificPage DM 2.159,-
PostScript-Cartridge für HP LaserJet II, benötigt 2MB RAM

PacificPage IIP DM 1.699,-
PostScript-Cartridge für HP LaserJet IIP, benötigt 2MB RAM



PostScript-Paket 1 DM 3.599,-
Pacific Page mit 2MB

PostScript-Paket 2 DM 7.998,-
Pacific Page und HP-LaserJet III mit 2MB

PostScript-Paket 3 DM 3.498,-
Pacific Page IIP mit 2MB

PostScript-Paket 4 DM 6.498,-
Pacific Page und HP-LaserJet IIP mit 2MB

CAD/GRAFIK Lösungen!

Laserdrucker als Plotter!
Plotter in a Cartridge DM 1.199,-
HP 7475 Emulation, 100mal schneller!
Paket: Plotter-Cartridge incl. der benötigten 1MB Erweiterung DM 1.998,-

HP-GL Software für alle HP-Drucker

LaserPlotter-Software f. LaserJet	DM	359,-
DeskPlotter-Software f. DeskJet	DM	299,-
JetPlotter-Software f. TJ, QJ, PJ	DM	219,

Neue HP A0 und A1 Plotter P. a. A.

Der PC für CAD: HP Vectra 486 P. a. A.

DRUCKER Lösungen!

Neu!

LaserJet III mit 2MB RAM	DM	5.998,-
LaserJet IIP incl. 2MB Board	DM	4.498,-
LaserJet IID incl. 2MB Board	DM	9.898,-
DeskJet	DM	1.699,-
DeskJet plus	DM	1.999,-
PaintJet Centronics	DM	3.359,-

JETWARE By Computer Peripherals, Inc. — 5 Jahre Garantie

JetMemory-Speichererweiterungen für alle HP-LaserJets

1MB Hauptspeicher-Grundmodul	DM	998,-
2MB Hauptspeicher-Grundmodul	DM	1.866,-
4MB Hauptspeicher-Grundmodul (nicht erweiterbar)	DM	3.498,-
+ 1MB Erweiterungsmodul	DM	1.109,-
+ 2MB Erweiterungsmodul	DM	1.966,-
+ 3MB Erweiterungsmodul (maximal 4MB möglich)	DM	2.588,-

Pacific Data Speichererweiterungen

HP-LaserJet II:		
1-2-4 plus Memory Board 1MB	DM	998,-
1-2-4 plus Memory Board 2MB	DM	1.749,-
1-2-4 plus Memory Board 4MB	DM	3.299,-
HP-LaserJet IIP:		
2 plus 2 Memory Board 1MB	DM	1.099,-
2 plus 2 Memory Board 2MB	DM	1.999,-

GRADCO-Papierzuführung für LaserJets
HCF 1000 DM 1.898,-
Fassungsvermögen: 1000 Blatt DIN A4
(Weitere Papierzuführungen a. A.)

SCHRIFTEN Lösungen!

Super-Set International DM 1.366,-
700 HP äquivalente Schriftzeichen, incl. 13 Treiber für 11 versch. Softwareprogramme!

JetFont-Cartridges (kompatibel zu HP)

JetFont A-Z	ab DM	226,-
JetFont 12/30, Lotica/LICS-L	DM	309,-

25 in 1 Cartridge DM 1.299,-
Enthält alle Schriften aus 25 HP-Cartridges!
Headlines-Cartridge DM 1.299,-
Helvetica/Times Roman von 14-48 Punkt!
Paket: 25 in 1 Cartridge plus Headlines-Cartridge DM 2.489,-

COMPUTER Lösungen!

HP Vectra PCs und HP Vectra 486 PC mit Installationsservice, rufen Sie uns an!

TASCHEN-RECHNER Lösungen!

W&W HP-41 CY TURBO 64 DM 1.099,-
(HP-41 CX mit 64k RAM und doppelter Rechengeschwindigkeit)
W&W HP-41 CY TURBO 32 DM 999,-
(s.o., 32k RAM)

HP-42S	DM	259,-
HP-28S	DM	519,-
HP-19B II (Neu!)	DM	439,-
HP-17B II	DM	279,-
HP-41 CX, der Klassiker!	DM	499,-

HP-48SX (Neu!) DM 799,-
(engl. Dokumentation)

Zubehör

32k RAM	DM	199,-
128k RAM	DM	699,-
Modul Gleichungslöser	DM	279,-
PC-Interface	DM	229,-

Alle Preise verstehen sich inclusive individueller Beratung und Service (Eigene Reparaturwerkstatt, eigene Entwicklung), sowie einer einjährigen Garantie. Alle Jetware-Produkte verstehen sich inclusive 5-jähriger Herstellergarantie.

W&W Software Products GmbH, Odenthaler Str. 214, 5060 Bergisch Gladbach 2, FAX: 32794

HOTLINE: 02202/42021

Ab 1.4.90 Zweigstelle in München: W&W Software Products GmbH, Ammerthalstr. 12, 8011 Kirchheim

Inhalt

Vorschau

auf die nächsten Hefte

TI-58/59 Emulation

Fadenpendel

Auswertung der Fragebogenaktion

ATARI-Shareware

IP - Ephemeriden

Vieles im Umbruch

Nicht nur in der DDR erscheinen neue Sterne am Himmel, auch von Hewlett Packard wird 10 Jahre nach der Vorstellung des HP-41 sein Nachfolger präsentiert.

Die Bezeichnung des Neulings ist die Mischung aus 41 und 28, von beiden wurden die herausragenden Eigenschaften in einen Topf geworfen und kräftig gewürzt.

Mit diesem Gerät will Hewlett Packard wieder einmal seine führende Stellung auf dem Gebiet der intelligenten Taschenrechner festigen, billige Kopien der HP-Taschenrechner gibt es ja wie Sand am Meer.

Billig war aber selten preiswert, jede Leistung hat seinen Preis, auch wenn so mancher Student bei HPs Preisen immer wieder stöhnt.

Mit seiner neuen Taschenrechnergeneration nach der Serie 10 ist HP eigentlich mit seinem Preisniveau auf eträglichere Ebenen herabgestiegen, dafür wurde erstaunliche Intelligenz und Durchdachtheit geboten, ein Taschenrechner besteht ja nicht nur aus vielen Tasten und einem dünnen Begleitheftchen, wie dies bei so manchem Fernostprodukt heute noch der Fall ist.

Jetzt wird so mancher denken er kann seinen 'alten' einmotten, so schnell wird der HP41, seit nun schon fast 10 Jahren das Zugpferd des CCD, nicht von der Bildfläche verschwinden, er ist und bleibt nämlich der letzte HP-IL Taschencontroller seiner Art, HP hat anscheinend diese erste und einzige intelligente serielle Verbindung zwischen Computern begraben, eigentlich sehr schade, es muß sich aber auch alles bezahlt machen...

Ach ja, ich hätte beinahe den letzten existierenden 'großen Bruder' des HP41 vergessen, ich meine MARYs II, dieser hat ja neben einer seriellen V24 auch eine IL-Schnittstelle, ob man diesen nicht als Verbindungsrechner gebrauchen kann, vielleicht kein uninteressanter Gedanke, vor allem um Daten vom HP41 bekommen zu können...

Gut, soweit zu dieser Neuigkeit, es stehen ja bei der kommenden Mitgliederversammlung Wahlen auf der Tagesordnung, auch wenn so mancher das Wort Wahlen bald nicht mehr hören kann, in Europa ist eben das Wahlfieber ausgebrochen.

Wir wählen einen neuen Vorstand, der mit Sicherheit mit vielen neuen Aspekten konfrontiert werden wird, das Jahr 1990 bringt ja politisch und technisch viel Neuland.

Ich rufe hiermit zu zahlreichem Erscheinen bei der Mitgliederversammlung auf, damit der neue Vorstand ein Vorstand aller Mitglieder sein kann, neue Gesichter bringen auch frischen Wind in den CCD, es sind also alle aufgefordert sich Gedanken über die Besetzung bzw. die Kandidaten für einen neu zu formierenden Kopf des CCD zu machen.

Aktivität ist also angesagt, das ist die Kraft, die den CCD seit seinem Bestehen auszeichnet und die Grundlage für alles darstellt, womit wir alle die 3 Buchstaben assoziieren, die uns verbinden.

Martin Meyer

(Redaktion)

Kurzprotokoll

der Vorstandssitzung des CCD

am Samstag, den 27. Januar 1990, 11 Uhr in der Geschäftsstelle, Schwalbacher Str. 50, 6000 Frankfurt 1

Beginn: 11.15 Uhr
Ende: 16.45 Uhr

Anwesend:
Dr. Wolfgang Fritz, Dieter Wolf, Werner Dworak, Alf-Norman Tietze.

Als Vertreter des Beirats:
Peter Kemmerling, Martin Meyer, Alexander Wolf.

Entschuldigt:
Erich Klee, Manfred Hammer

Tagesordnung:

TOP 1: Festlegung der Tagesordnung
TOP 2: Genehmigung des Protokolls der Sitzung vom 7. Oktober 1989
TOP 3: Berichte aus den Fachgruppen
- a) Mitgliederstand
- b) MS-DOS
- c) ATARI-Gruppe
- d) CP/M-Auflösung
- e) Örtliche Treffs

TOP 4: Finanzbericht
- a) Stand der Finanzen
- b) Haushaltsplan 1990
- c) Clubinterner Wettbewerb

TOP 5: Mitgliederversammlung 1990
- a) Einladung, Ort und Tagesordnung
- b) Vorstandsbericht auf der MV
- c) sonstiges Organisatorisches
- d) Kandidaten

TOP 6: PRISMA-Herstellung
TOP 7: PRISMA Redaktion
TOP 8: Festlegung der Termine 1990
TOP 9: Verschiedenes

Zu TOP 1:
Festlegung der Tagesordnung

Die Tagesordnung wird genehmigt.

Zu TOP 2:
Genehmigung des Protokolls der Sitzung vom 7. Oktober 1989

Wird genehmigt.

Zu TOP 3. a-e:
Berichte aus den Fachgruppen

Die Zahl der Mitglieder der MS-DOS Gruppe ist weiterhin leicht ansteigend, viele ehemalige CP/M-Gruppen-Mitglieder sind jetzt (zumindest vorläufig) zu ihr gestoßen.

Dieter Wolf schildert die Kostensituation der 3 Gruppierungen innerhalb des CCD.

MS-DOS
Die Hauptleistung des Clubs für die Mitglieder der MS-DOS Gruppe ist die Herausgabe der MS-DOS Diskette mit einem Kostenaufwand von ca. 30000 DM, dem steht gegenüber ein Beitragsaufkommen von ca. 45000 DM, d.h. die Gruppe trägt sich gut selbst.

Taschencomputer
Die Hauptleistung des Clubs für die Mitglieder des Taschencomputerbereichs ist die Herausgabe von PRISMA mit einem Gesamtaufwand von ca. 100000 DM, dem steht gegenüber ein Beitragsaufkommen von ca. 88000 DM.

ATARI
Die Hauptleistung des Clubs für die Mitglieder der Atari-Gruppe ist die Herausgabe der Disketten und ist mit einem Kostenaufwand von ca. 4100 DM verbunden, dem stehen gegenüber ein Beitragsaufkommen von ca. 4500 DM, d.h. auch diese kleine Gruppe trägt sich finanziell.

Die Auflösung der CP/M-Gruppe lief vollkommen problemlos ab. Zur Einstellung der organisatorischen Unterstützung des CCD bezüglich der Mitgliedschaft in dem amerikanischen Verein FOG mit dem Bezug der Zeitschriften FOGHORN und/ oder FOGLIGHT gab es, so Werner Dworak, keine Reaktionen.

Die Nachricht von Jürgen Schramm aus Köln wurde durchdiskutiert. Bezüglich der dort aufgestellte Bitte um finanzielle Unterstützung bei der Raummiete verweist Dieter Wolf darauf, daß diese Unterstützung ja bisher schon immer auf Antrag gewährt worden ist. Grundsätzlich ist der Vorstand zur Unterstützung bereit.

Für den Ablauf von örtlichen Treffen ist nach Meinung des Vorstandes die Benennung eines Verantwortlichen am Ort notwendig. Daher beschließt der Vorstand:

Bei Einladungen zu örtlichen Treffen ist im PRISMA aufzuführen: : "Verantwortlich und Ansprechpartner ist ..."

Zu TOP 4:
Finanzbericht

a) Stand der Finanzen
Der Finanzbericht wurde entgegengenommen. (Er wird auf der Mitgliederversammlung ausführlich besprochen werden.)

b) Haushaltsplan 1990
Dieter Wolf legt seinen Haushaltsvorschlag für 1990 mit einem Volumen von DM 140000 vor, der mit geringfügigen Änderungen einstimmig angenommen wird.

c) Clubinterner Wettbewerb
Es gibt aufgrund des clubinternen Wettbewerbs nur 19 Eintritte.

Zu TOP 5:
Vorbereitung Mitgliederversammlung

Ort: Frankfurt. Die bisherigen Räume stehen diesmal nicht zur Verfügung. Dieter Wolf legt Räumlichkeiten fest, möglicherweise im Gewerkschaftshaus. Termin ist Samstag, der 21.04.1990.

Zu TOP 6:
PRISMA-Herstellung

Dieter Wolf berichtet, daß Formeln noch nicht mit eigenen Mitteln gesetzt werden können, die Versuche mit DTP-Programmen sind erfolgversprechend gewesen. Es wird beschlossen:

Pagemaker, Chiwriter oder Large-Windows und zusätzlich GoScript-Plus werden für die PRISMA-Herstellung angeschafft, der finanzielle Aufwand entspricht etwa den Satzkosten von zwei PRISMA-Ausgaben.

Zu TOP 7:
PRISMA Redaktion

Alf-Norman Tietze: Matthias Rabe wird als zuständig für LEX-Files aufgeführt. HP bringt als Nachfolger des 41er einen neuen Rechner hp 48 sx heraus.

Zu TOP 8:
Festlegung der Termine 1990

Vorgesehen für 1990 sind die folgenden Termine: für die CCD- Hauptversammlung: Samstag, 21. April 1990; für die CCD- Vorstandssitzungen: 10.3.90, 20.4.90, 7.7.90, 6.10.90 und 8.12.90. Bei Bedarf werden zusätzliche Termine eingeschoben.

Zu TOP 9:
Verschiedenes

Die Fragebogenauswertung hat Peter Kemmerling durchgeführt, es gab einen Rückfluß von 166 Fragebogen. Die Auswertung und Interpretation erscheint im PRISMA.

PRISMA soll auch an die wissenschaftlichen Bibliotheken der DDR gegeben werden.

Die Anfragen aus der DDR aufgrund der Anzeigen sind nach wie vor sehr zahlreich. Wolfgang Fritz schlägt vor, den Interessenten aus der DDR, denen in der Regel ein Beitritt auf der Basis DM nicht möglich ist, die Mitgliedschaft auf der Basis Mark der DDR zu ermöglichen (1:1). Hierüber kann nur die Mitgliederversammlung entscheiden. Der Vorstand wird prüfen, ob er nicht für Interessenten aus der DDR bis zu einer zu erwartenden Währungsunion einen assoziierten Status (keine vollwertige Mitgliedschaft) schaffen kann auf der Basis 1:1, welche den Bezug von PRISMA und der Disketten ermöglicht. Über diesen Weg kann der finanzielle Verlust begrenzt werden.

Wolfgang Fritz (125)
-Vorsitzender-

Einladung zur
Mitgliederversammlung

Am 21. April 1990 um 11 Uhr findet im Hauptbahnhof Frankfurt/Main, Intercity-Restaurant, Frankfurter Saal, unsere jährliche Mitgliederversammlung statt.

Tagesordnung:

1. Begrüßung durch den Vorstand
2. Feststellung der Beschlußfähigkeit und andere Formalitäten
3. Bericht des Vorstandes
4. Bericht des Beirats
5. Bericht der Kassenprüfer
6. Entlastung des Vorstands
7. Neuwahl
   a) des Vorstands
   b) eines Kassenprüfers
   c) gegebenenfalls Nachwahlen
8. Haushaltsplan 1990
9. PRISMA
10. Anträge
11. Verschiedenes

Liebe Mitglieder,

erneut ist Frankfurt als zentral gelegene Stadt, an der auch unsere Geschäftsstelle ist, unser Tagungsort.

Die Amtszeit des Vorstands läuft aus, es sind fünf Vorstandsmitglieder für die nächsten drei Jahre zu wählen. Außerdem ist einer der beiden Kassenprüferpositionen neu zu besetzen für die nächsten drei Jahre. Andere Wahlen stehen nicht an: Beiratswahlen sind erst wieder 1991 fällig. Lediglich, wenn zum Beispiel aus dem Kreis des Beirats jemand in den Vorstand wechselt, sollten dort gegebenenfalls Nachwahlen durchgeführt werden.

Welche Entwicklung der CCD nimmt, darüber wird berichtet werden. Interessanter ist, welche Ideen die Mitglieder für die Weiterentwicklung unseres Clubs einbringen werden; gute Ideen werden der Vorstand und der Beirat gerne aufgreifen.

Anträge an die Mitgliederversammlung sollten möglichst bald an den Vorstand gerichtet werden. Die Satzung sagt dazu aus (§ 13 Abs. 2): "Jedes Mitglied kann bis spätestens eine Woche vor der Mitgliederversammlung beim Vorstand schriftliche eine Ergänzung der Tagesordnung beantragen. ... Anträge auf Ergänzung der Tagesordnung, die in der Mitgliederversammlung gestellt werden, beschließt die Versammlung."

In diesem Jahr wird erneut im Anschluß an die Mitgliederversammlung ein MS-DOS-Workshop stattfinden. Es geht diesmal um Microsoft WORD 5.0. Das Haupteinsatzgebiet von PCs ist wohl die Textverarbeitung, und WORD ist eines der sehr guten und sehr verbreiteten Textverarbeitungsprogramme. Es besitzt eine solche Fülle von Möglichkeiten, daß viele Anwender diese gar nicht kennen und daher nicht anwenden. An PCs, die wir aufstellen, sollen aber auch Anfänger, die dieses oder andere Textverarbeitungsprogramme nicht kennen, einen Eindruck von den Möglichkeiten eines solchen Programmes erhalten.

Die persönliche Kontaktnahme mit anderen Mitgliedern ist eine der schönen Möglichkeiten unseres Clubs. Kommt daher zum Treffen, wir laden alle Mitglieder herzlich ein.

Mit freundlichen Grüßen

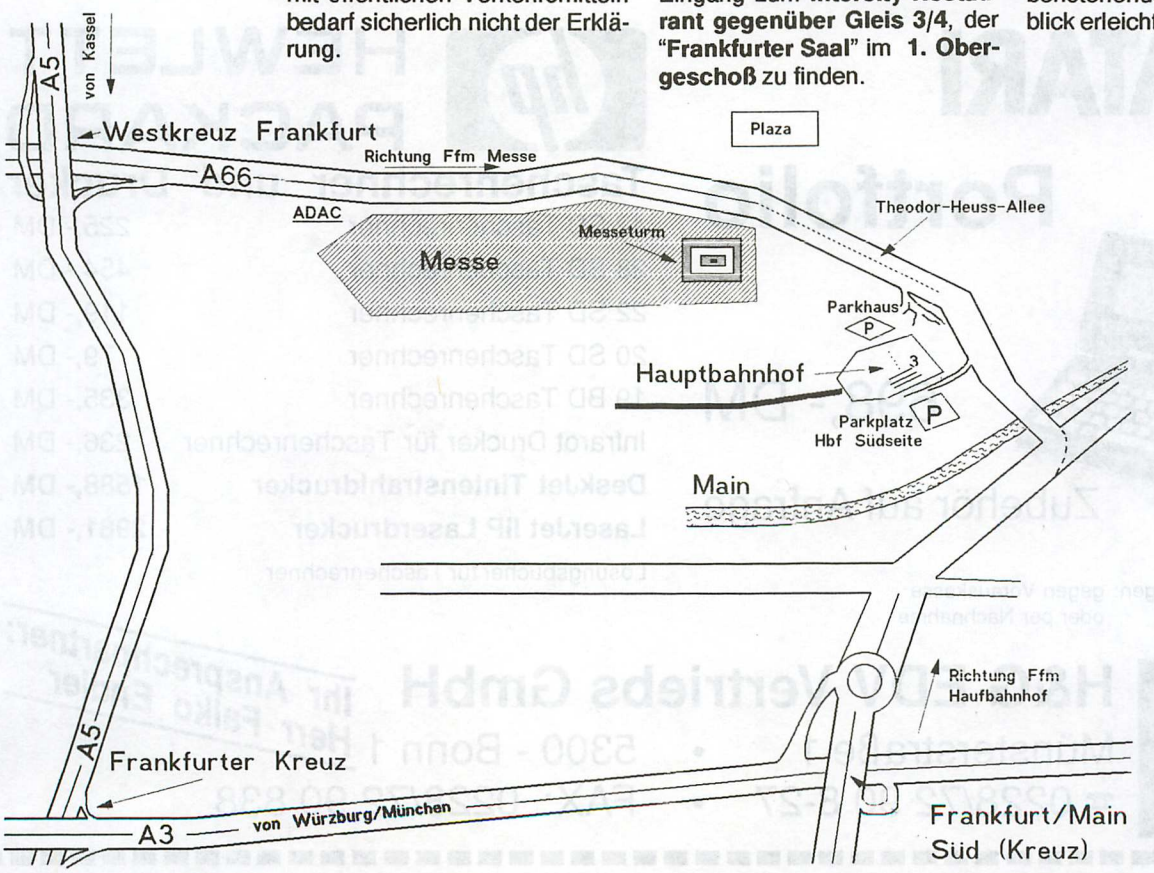
Wolfgang Fritz
1. Vorsitzender

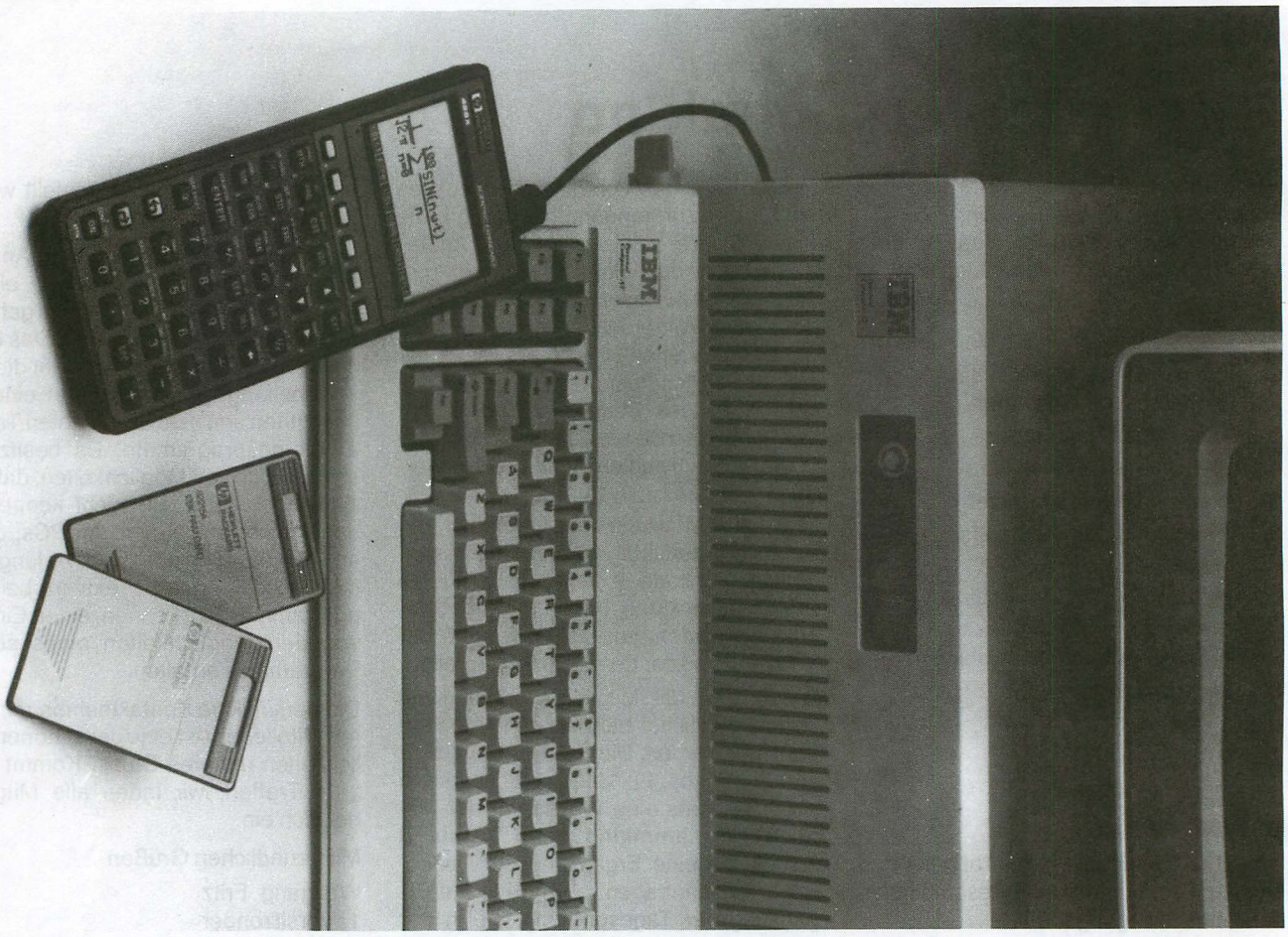
Wie komme ich hin?

Die Anfahrt zum Hauptbahnhof mit öffentlichen Verkehrsmitteln bedarf sicherlich nicht der Erklärung.

Im Hauptbahnhof selbst ist der Eingang zum **Intercity-Restaurant gegenüber Gleis 3/4**, der **"Frankfurter Saal"** im **1. Obergeschoß** zu finden.

Für die Autofahrer soll die nebenstehende Skizze den Überblick erleichtern.





Der HP-48SX

das neue Taschenrechner-Flaggschiff von HP

von Matthias Rabe

Ein paar Tage vor Redaktionsschluß dieser Prisma-Ausgabe hatte ich Gelegenheit, den neuen Wunderrechner von HP für einige Stunden intensiv zu begutachten. Wegen der Kürze der Zeit soll und kann dieser Artikel kein ausführlicher Testbericht sein. Vielmehr will ich nur meine ersten Eindrücke schildern. Um es vorwegzunehmen: der HP-48SX ist in erster Linie ein Taschenrechner, kein Computer wie der HP71 oder HP75.

Ausgeliefert wird der HP-48 in einer gepolsterten Nylontasche mit Klettverschluß und einem etwa 850-seitigen, zweibändigen Handbuch. Er sieht etwa so aus, wie die anderen Taschenrechner von HP (HP-17, HP-27, HP-32, HP-42 etc.), hat jedoch ein größeres Display und zwei Tastenreihen mehr, also insgesamt 49 Tasten. Auf der Unterseite befinden sich zwei Klappen. Unter der einen ist das Batteriefach für drei Mignonzellen (Typ AAA), unter der anderen verstecken sich zwei Modulschächte für RAM- und ROM-Module, die bidirektionale Infrarotschnittstelle und der Anschluß für eine serielle RS-232-Schnittstelle.

Das interne RAM ist 32kB groß, wovon etwa 30kB dem Benutzer für Programme und Daten zur Verfügung stehen. Das Betriebssystem ist in 256 kB ROM untergebracht. Das ist das gewaltigste, was es je für einen Taschenrechner gab. Dementsprechend stellt der 48'er mit seinen mehr als 2100 intergierten Funktionen auch alles bisher dagewesene in den Schatten. Der Speicher kann mit zwei Modulen noch um maximal 256 kB erweitert werden, wobei RAM und ROM beliebig gemischt werden können. Bereits erhältlich sind RAM-Erweiterungen mit 32kB und 128kB und eine Gleichungslöser-Programmkarte, die eine Sammlung von mehr als 300 Formeln, zu einigen Formeln auch erklärende Grafiken und außerdem noch das Periodensystem der Elemente, das grafisch dargestellt werden kann, enthält (Bild 1).

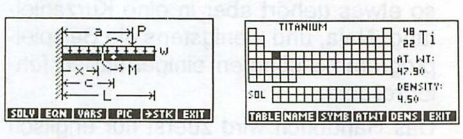


Bild 1

Andere Programmkarten sind bereits angekündigt, wie zB. eine HP-41-Emulation (schon wieder) und ein Programmpaket für's Vermessungswesen.

Die Speichererweiterungen haben etwa Scheckkartenformat und werden übereinander in den Rechner geschoben. Die RAM-Karten haben eine leicht auswechselbare Batterie und können somit auch außerhalb des Rechners ihre Daten jahrelang behalten. Sie eignen sich auch zum Datenaustausch zwischen zwei HP-48 oder als Massenspeicher (wenn sie nicht so teuer wären). An ihnen ist ein kleiner Schalter angebracht, mit dem sich ein Schreibschutz schalten läßt. Somit können Daten in einer RAM-Karte so sicher aufgehoben sein, wie in einer ROM-Karte.

Die Infrarotschnittstelle dient nicht nur zum Ausdruck von Daten auf dem HP82240 Infrarot-Drucker. Da sie bidirektional ist, lassen sich mit ihr auch Daten zwischen zwei HP-48 austauschen. Für die serielle Schnittstelle wird ein Schnittstellenmodul benötigt, welches mit Software für den IBM-PC oder den Apple Macintosh ausgeliefert wird. Damit ist es möglich, Den PC als Terminal für den HP-48 zu benutzen, Daten auf dem Massenspeicher des PC zu sichern, auf dem PC Programme für den HP-48 zu schreiben und auch auf dem am PC angeschlossenen Drucker auszugeben. Durch die Möglichkeit,den PC als Terminal zu benutzen, wirkt der HP-48 wie ein auf dem PC laufendes Programm.

Einen vorhandenen PC als Massenspeicher zu benutzen, ist übrigens die einzige von HP vorgesehene Methode, um Programme zu sichern - es gibt keine anderen Massenspeicher für den HP-48. Zu diesem Schritt hat sich HP bereits vor etwa drei Jahren entschlossen und bekannt gegeben, daß keine HP-IL-Geräte mehr entwickelt würden, da durch den Preisverfall der Personalcomputer es bereits billiger geworden ist, sich einen PC anzuschaffen, als ein IL-Diskettenlaufwerk.

Die RS-232 des HP-48 unterstützt übrigens kein Hardware-Handshake. Das sollte in der Regel aber kein Handikap sein, da jeder Computer seine serielle Schnittstelle auch mit Software-Handshake betreiben kann. Das ist alles nur eine Frage der Software. Es gibt höchstens ein paar Peripheriegeräte, die auf Hardware-Handshake bestehen. Ich habe da z.B. so einen EPROM-Brenner. Aber wozu sollte ich mit einem Taschenrechner EPROMs brennen, wenn ich sowieso einen PC haben muß, der das viel besser kann? Die RS-232 läßt sich auf die üblichen Baud-Raten zwischen 1200 und 9600 einstellen. Als Software-Handshake-Protokoll ist nur das Receiver-Protokoll möglich (XON/-XOFF), aber das Transmitter-Protokoll findet wegen einiger Unzulänglichkeiten sowieso kaum noch Verwendung. Um Übertragungsfehler, für die besonders die Infrarotschnittstelle anfällig ist, zu vermeiden, muß ein Datenübertragungsprotokoll benutzt werden, das Übertragungsfehler erkennt und korrigiert bzw. die Neuübertragung der fehlerhaften Daten veranlasst. Derer gibt es jede Menge. Der HP-48 benutzt Kermit, ein komfortables, weit verbreitetes Protokoll. Leider wird das noch populärere (aber schlechtere, da einfachere) XModem-Protokoll, welches zB. die CCD-Mailbox benutzt, nicht unterstützt. Aber das könnte programmiert werden, da es möglich ist, beliebige Daten, ohne jedes Protokoll zu senden und zu empfangen. Mit einem entsprechenden Programm ist es also durchaus möglich, mit dem HP-48 die CCD-Mailbox zu benutzen, zumal das achtzeilige Display schon einen recht guten Überblick über die empfangenen Daten erlaubt, womit wir schon beim nächsten Thema wären.

Das LCD erlaubt mit seinen 131 * 64 Pixel die Darstellung von recht guten Grafiken oder Text in acht Zeilen mit je zweiundzwanzig Zeichen. Im Normalfall wird in der untersten Zeile die Belegung der sechs Funktionstasten dargestellt. Darüber befinden sich die Eingabezeile und die obersten Stackelemente. Auf dem "unbegrenzt" mitwachsendem Stack können alle, dem HP-48 zur Verfügung stehenden Datentypen, also Fließkommazahlen, komplexe Zahlen, Integerzahlen, Strings, Vektoren, Matrizen, Grafiken, Listen und Programme stehen. Auch Programme werden auf dem Stack eingegeben. Es ist möglich, andere, auf dem Stack liegende Objekte durch Tastendruck in ein Programm zu integrieren, unabhängig vom Datentyp.

Ist das Programm fertig, so kann es auf dem Stack ausgeführt (danach ist es weg) oder in einer Variablen gespeichert werden.

Variablen können einen nahezu beliebigen, bis zu 127 Zeichen langen Namen haben und werden in einer Baumstruktur gespeichert, ähnlich dem Dateisystem von MS/DOS-Rechnern. Dadurch kann man Ordnung halten, indem man zusammen gehörende Programme und Daten in Verzeichnissen zusammenfaßt. Sonst könnte in den maximal 288kB RAM schnell das große Chaos ausbrechen. Auch können mehrere Variablen den gleichen Namen haben, sofern sie sich in verschiedenen Verzeichnissen befinden. Soll auf eine Variable zugegriffen werden, so wird erst das aktuelle Verzeichnis durchsucht. Wird kein Eintrag des angegebenen Namens gefunden, so wird im darüberliegenden Vaterverzeichnis gesucht usw. bis zum HOME genannten Wurzelverzeichnis. So können im HOME-Verzeichnis alle Daten und Program-

me abgelegt werden, auf die von überall aus zugegriffen werden soll.

Sollen Daten übertragen werden, ob zum Datenaustausch zwischen zwei HP-48 oder zur Datensicherung auf einem PC, so kann eine einzelne Variable angegeben werden, ein ganzes Verzeichnis (auch das HOME-Verzeichnis), welches dann mit allen enthaltenen Variablen und Unterverzeichnissen übertragen wird, oder der gesammte Speicher, wobei dann auch Tastenzuweisungen und Alarme (hört, hört!) kopiert werden. Werden derartige Daten empfangen, so werden Variablen bzw. Verzeichnisse entsprechend ins aktuelle Verzeichnis integriert. Ein kompletter Speicherauszug überschreibt logischer Weise alles.

Programme können nicht nur, wie oben beschrieben, im Binärmodus übertragen werden, sondern auch im sogenannten ASCII-Modus, sodaß es möglich ist, Programme mit dem PC zu ändern bzw. auszudrucken.

Programmiert wird der HP-48, so wie der HP-28, in einer RPL genannten Sprache. RPL steht für Reverse Polish Lisp, was im Klartext bedeutet, daß es sich um eine Art FORTH handelt und die Datenstrukturen von LISP gestohlen ("Unbedeutende Künstler leihen, große Künstler stehlen." - Igor Stravinsky) sind. In RPL sind einige tolle Sachen realisiert, wie lokale Variablen, alle möglichen Kontrollstrukturen, wie zB. IFERR...THEN...ELSE...ENDERR und vieles mehr. Mir fehlt da noch völlig der Überblick. Es lassen sich auch tolle interaktive Programme erstellen, mit Menüs und allem Drum und Dran. Die Programmierbarkeit der Grafik läßt kaum noch Wünsche offen: es lassen sich einzelne Punkte zeichnen, Linien, Kreise, Elipsen etc., wobei die virtuelle Zeichenfläche von der physikalischen völlig unabhängig ist.

Selbstgeschriebene Programme können in ein benutzerdefinierbares Menü integriert werden.

Die Tastatur hat zwei shift-Tasten, linksshift und rechts-shift genannt und eine Alpha-Taste, die, ähnlich der USER-Taste, auf die alphanumerische Belegung der Tastatur umschaltet. Das Alphabet ist wie beim HP-41 sortiert angeordnet. Mit diesen Umschaltmöglichkeiten ist die Tastatur sechsfach belegt. Jede der damit möglichen 276 Tasten(-kombinationen) läßt sich mit einem beliebigen Objekt belegen, welches, wenn die Taste im USER-Modus gedrückt wird, ausgewertet wird. Auswerten heißt bei Programmen und Funktionen ausführen, bei Zahlen etc., daß sie auf dem Stack landen.

Die Uhr des HP-48 läßt sich auf das deutsche Anzeigeformat einstellen. Die Uhrzeit kann ständig in der obersten Zeile des Displays angezeigt werden.

Es lassen sich Appointment- (englisch: Verabredung) und Kontrollalarme setzen. Appointments lassen den Rechner zur programmierten Zeit piepen und einen Text anzeigen, Kontrollalarme dienen zur Ausführung von Programmen.

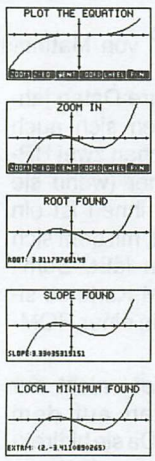


Bild 2

Viele der Möglichkeiten des HP-48 sind bereits vom HP-28 bekannt. So kann der 48'er symbolisch rechnen, Funktionsgrafen darstellen, diese Grafiken zoomen und in ihnen nach Nullstellen, Extrema und Schnittpunkten von zwei Funktionen suchen und Steigungen und Integrale bestimmen.

Der HP-48 verfügt über acht verschiedene Darstellungsarten für Grafiken: Funktionsdiagramme, Balkendiagramme, Histogramme, Verteilungsdiagramme, Kegelschnitte, Polarkoordinaten, parametrische Darstellungen und Wahrheitstabellen (Bild 3)

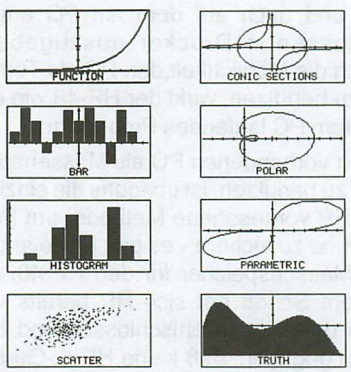


Bild 3

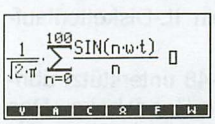


Bild 4

Neu ist der EquationWriter, der es ermöglicht, Gleichungen symbolisch einzugeben.

Eine der tollsten Sachen ist, daß Werten eine Einheit zugeordnet werden kann, die der HP-48 bei Bedarf automatisch umrechnet. So kann man zB. 6 Joule durch zwei Sekunden teilen und erhält als Ergebnis drei Watt oder von 1000 britischen Gallonen 546 Liter subtrahieren, was vier Kubikmeter ergibt. Außerdem achtet der Rechner darauf, daß nicht "Äpfel und Birnen" addiert werden. Der HP-48 kennt 148 Einheiten, zusätzliche lassen sich definieren.

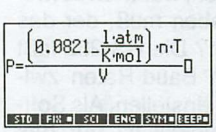


Bild 5

Kein Licht ohne Schatten?

Es fällt schwer, angesichts des überwältigenden Leistungsumfanges des HP-48SX in so kurzer Zeit etwas zu meckern zu finden. Etwas enttäuscht bin ich über die Geschwindigkeit des Rechners. Sie dürfte etwa im selben Bereich liegen, wie beim HP-28. Ich hätte mir einen Geschwindigkeitsschub wie zwischen dem HP-41 und dem HP-28 gewünscht. Besonders störend macht sich die Langsamkeit beim EquationWriter bemerkbar, aber auch das Scrollen geschieht recht gemütlich. Der Kontrast des Displays könnte auch etwas größer sein. Wer schon mal den Apple-Laptop gesehen hat, weis, daß es auch besser geht. Nicht, daß das Display besonders schlecht ist, aber es ist halt nicht "State Of The Art", wie der Rest des Rechners. Außerdem fehlen der RS-232 noch geringere Übertragungsraten, wie zB. 300 Baud für langsame Akustikkoppler.

Daß die Funktions- und Cursortasten im Alphamodus auch mit Zeichen belegt sind ist recht lästig. Beliebige Funktionen kann man bei HP wohl immer noch nicht symbolisch integrieren. Mathematica ist noch nicht überflüssig. Schade.

Die INPUT-Funktion kann leider nur durch die ENTER-Taste beendet werden. Damit ist es nicht möglich, zB. einen Texteditor zu schreiben, weil dann einige Tasten benötigt werden, um Programmfunktionen direkt auf Tastendruck auszuführen (Cursorsteuerung etc.). Aber wo ein Wille ist, ist auch ein Weg. Ein Weg wäre, eine entsprechende Funktion in Assembler zu schreiben. Einen Assembler gibt es zwar nicht, aber das ist kein grundsätzliches Problem. Viel schwerer wiegt jedoch die Tatsache, daß nicht damit zu rechnen ist, daß HP die Dokumentation des Betriebssystems veröffentlicht. Ob es diese Dokumentation wohl irgendwann auf NOMAS-Basis (NOt MAnufactor Supported) wie beim HP-75 geben wird? Ich würde in Freudentränen ausbrechen.

Ansonsten habe ich nur noch am Handbuch was zu nörgeln: es ist zwar mit 851 Seiten recht umfangreich, jedoch, ob der Komplexität des Gerätes manchmal etwas oberflächlich und was am schlimmsten ist: es fehlt ein Referenzhandbuch wie beim HP-71. Zwar gibt es eine Auflistung aller Befehle mit einer kurzen Beschreibung und einigen Querverweisen, so etwas gehört aber in eine Kurzanleitung. Naja, und wenigstens die Beispielprogramme könnten einigermaßen fehlerfrei sein.

Das Handbuch wird zuerst nur englisch ausgeliefert. Ab Juni sollen diese dann kostenlos gegen deutsche umgetauscht werden können.

Und nun das Wichtigste: der HP-48SX kostet 892,62 DM, das Schnittstellenkabel mit der Software für den IBM-PC oder Macintosh 287,28 DM, die 32kB RAM-Karte 229,14 DM, 128kB 723,90 DM und die Gleichungslöser Programmkarte 286,14 DM (alle Preise inclusive Mehrwertsteuer).

C- Programme - ein Überblick

Zweiter Teil

Functions, Pointer und I/O von C

Wie bereits früher erläutert, weist der Sprachkern von C keinerlei Routinen für Ein- und Ausgabe auf. Dies ist ein ganz wesentlicher Unterschied zu FORTRAN, BASIC und Pascal. Das I/O in C wird generell über Functions abgewickelt, die in Bibliotheken enthalten sind. Diese Bibliotheken variieren von Compiler zu Compiler und Betriebsystem zu Betriebsystem. Wir werden daher nur solche Functions nennen, die allgemeinerweise vorhanden sind. Um dieses Functionskonzept des I/O besser zu verstehen, muß zunächst der Begriff und der Umgang mit C- Functions erläutert werden.

1. C- Functions

Eine C-Function ist vergleichbar einer Procedure in Pascal oder den FUNCTION- Unterprogrammen und SUBROUTINES in FORTRAN. Teilweise gibt es jedoch nicht unbeträchtliche Unterschiede. Eine Function soll eine definierte Leistung erbringen. Normalerweise übernimmt sie Parameter vom rufenden Programm und liefert normalerweise einen einzigen Wert zurück. Standardmäßig ist der Return-Value vom Typ int. Soll er das nicht sein, so muß das explizit angegeben werden ! Ein Beispiel (Listing 1):

In diesem Beispiel geht alles wie erwartet vor sich: Das Hauptprogramm ruft eine Function iadd auf, übergibt zwei Parameter, iadd rechnet und liefert das Ergebnis als Return- Wert zurück. Tatsächlich passiert hier Subtileres: Die Werte der Parameter i und j werden als Kopie auf den Stack gelegt und die Function holt diese Werte vom Stack. Dies nennt man call by value. Das ist das übliche Verfahren in C und t.w. auch anderen Programmiersprachen.

2. Pointer

Das umgekehrte Verfahren ist call by reference. Dieses Verfahren wendet z.B. generell FORTRAN an und kann auch in C oft nutzbringend eingesetzt werden. Dabei werden nicht die Parameterwerte auf den Stack geschoben, sondern die Adressen der Werte, also die Adressen der Speicherstellen der Werte. Ehe die Vor- und Nachteile beider Verfahren erläutert werden, dazu ein Beispiel (Listing 2).

Wie bereits bekannt, ist & der Adressoperator und der Aufruf iadd im Hauptprogramm übergibt nun die Adressen der Parameter i und j an die Function. Die Function selbst erwartet nun Adressen, sogenannte Pointer. Die Deklaration

```
int * pi
```

sagt, daß pi einen Pointer (eine Adresse) auf einen Integerwert darstellt. Zum Verarbeiten selbst muß dann mit *pi und *pj gearbeitet werden, keinesfalls mit pi und pj, denn damit würden in diesem Beispiel die Adressen selbst addiert werden (was in anderen Fällen durchaus beabsichtigt sein kann). Beim Verarbeiten liefern *pi und *pj die Werte, die auf den Adressen pi und pj liegen. Man kann benennungsmässig Pointer generell mit p anfangen lassen, muß man aber nicht. Es hat sich bei mir bewährt, ähnlich FORTRAN, gewisse Namensregeln auch in C einzuhalten. Vorschlag:

- int und long beginnen mit i,j,k,l,m,n
- Pointer beginnen mit p
- Character beginnen mit c
- float und double beginnen mit a .. z, außer c,i,j,k,l,m,n und p

Es ist sehr praktisch, mitten in einem Programm der Variablen gleich ihren Typ ansehen zu können, ohne erst an den Anfang (dort Deklarationen) zu gehen.

Es ist nun so, daß die Bibliotheksfunktionen als Übergaben teilweise Werte und teilweise Adressen (also Pointer) erwarten. Beispiele:

double sin(double x):
x ist Wert vom Typ double

Listing 1 (b12.c)

```
#include <stdio.h>

/*******************************************
* Function- Deklaration, muß hier nicht sein,
*  sollte aber immer vorgenommen werden
*******************************************/
int iadd(int i, int j);

/*******************************************
*  Hauptprogramm
*******************************************/
main()
{
  int i,j,k;

  printf("\ngib i und j: ");
  scanf("%d %d",&i,&j);

  k= iadd(i,j);

  printf("\nErgebnis: k= %d\n",k);
}

/*******************************************
*  Function iadd
*******************************************/

int iadd(int i, int j)
{
int k;

k= i + j;

return(k);
}
```

Listing 2 (b13.c)

```
#include <stdio.h>

/*******************************************
* Function- Deklaration, muß hier nicht sein,
*  sollte aber immer vorgenommen werden
*******************************************/
int iadd(int *pi, int *pj);

/*******************************************
*  Hauptprogramm
*******************************************/
main()
{
  int i,j,k;

  printf("\ngib i und j: ");
  scanf("%d %d",&i,&j);

  k= iadd(&i,&j);

  printf("\nErgebnis: k= %d\n",k);
}

/*******************************************
*  Function iadd
*******************************************/

int iadd(int *pi, int *pj)
{
int k;

k= *pi + *pj;

return(k);
}
```

int mkdir(char *path):
path ist ein Pointer auf char

Da ist höllisch aufzupassen: Beim Schreiben von C-Programmen muß eigentlich immer das Handbuch der Run-Time- Library des jeweiligen Compilers griffbereit liegen, da diese Aufrufe absolut exakt erfolgen müssen. Falsche Übergaben werden bei DOS und TOS fast immer mit Festfahren des Betriebsystems quittiert. Denn Parameter, die eigentlich normale Variable sind und nun irrtümlicherweise als Pointer mißbraucht werden, können überall hinzeigen, oft auf die Speicherstellen des Betriebsystems ! Betriebsysteme wie OS/2 und UNIX fangen solche Unstimmigkeiten wenigstens ab und stoppen das Programm, ohne selbst im Nirwana zu verschwinden.

Bei eigenen Funktionen ist es jedermann freigestellt, ob er by value oder by reference übergibt. Bei C selbst wird so vorgegangen, daß Skalare normalerweise by value übergeben werden, aber Vektoren werden immer per reference auf den Stack geschoben. Da sind wir gleich bei den Vor- und Nachteilen beider Verfahren:

Call by value:

Vorteil:

- es werden Kopien der Skalare an die Funktion geliefert, dort existieren sie lokal weiter und können dort verändert werden, ohne daß das rufende Programm davon beeinflußt wird.

Nachteil:

- braucht z.B. bei double-Werten länger als call by reference
- Vektoren können so nicht übergeben werden, sonst müßten ja bei einem Vektor mit 10000 Elementen 10000 Werte auf den Stack geschoben werden !

Call by reference:

Vorteil:

- durch die Adressübergabe geht das bei double-Werten schneller als mit call by value. Für Vektoren die einzig sinnvolle Methode.

Nachteil:

- Übergebene Werte sind nicht lokal, sondern rufendes und gerufenes Programm können gleichermaßen zugreifen. Wird dann verblüffend, wenn das rufende Programm innerhalb einer Schleife eine Function ruft, der Schleifenzähler übergeben wird und die Function selbst den Schleifenzähler verändert !

Um auch den Aufwand einzugehen: Dies hängt sehr davon ab, ob sog. Near-Calls oder Far-Calls getätigt werden. Bei C unter DOS und OS/2 ist das durch die Segmentstruktur der Intel- Prozessoren eine Frage des gewählten Memomy-Modells. Bei dem standardmäßig vorgegebenen Small- Memory- Modell werden 16- Bit Adressen, beim Large-Memory- Modell dagegen 32-Bit Adressen übergeben. Auf die Memory- Modelle kommen wir in einer späteren Folge zurück.

Zahlenbeispiele in Tabelle 1.

Wie ersichtlich, ist die Übergabe by reference meist schneller (weil weniger Bits auf den Stack geschoben werden müssen) als call by value. Für kleine, kompakte Programme, was C-Programme oft sind, stört das absolut nicht. Bei sehr großen Programmen, die viele Unterprogramm-Aufrufe tätigen und fast ausschließlich mit double-Werten operieren (typisch für FORTRAN-Programme), kann das maßgeblichen Einfluß auf die Laufzeit haben. Da FORTRAN-Programme ohnehin immer per reference rufen und mit 32-Bit Adressen arbeiten, sind sie im Falle

Typ der Variablen	Typ des Aufrufs	call by value	call by reference
int	near	16 Bit	16 Bit
int	far	16 Bit	32 Bit
long und float	near	32 Bit	16 Bit
long und float	far	32 Bit	32 Bit
double	near	64 Bit	16 Bit
double	far	64 Bit	32 Bit

Listing 3 (b14.c)

```
#include <stdio.h>
#include <direct.h>     /* für mkdir */

main()
{
  int iret;
  char cpath[80];

  printf("\ngib Pfad: ");
  scanf("%s",cpath);

  iret= mkdir(cpath);
  if(iret == 0)
    printf("\nDirectory erzeugt !");
  else
    printf("\nDirectory nicht erzeugt !");

}
```

Listing 4 (b15.c)

```
#include <stdio.h>
#include <direct.h>     /* für mkdir */

main()
{
  /* schön falsch ! */
  int iret;
  char *pc;

  printf("\ngib Pfad: ");
  scanf("%s",pc);

  iret= mkdir(pc);
  if(iret == 0)
    printf("\nDirectory erzeugt !");
  else
    printf("\nDirectory nicht erzeugt !");

}
```

Listing 5 (b16.c)

```
#include <stdio.h>
#include <direct.h>     /* für mkdir */

main()
{
  int iret;
  char *pc;

  pc= "harry";

  iret= mkdir(pc);
  if(iret == 0)
    printf("\nDirectory harry erzeugt !");
  else
    printf("\nDirectory harry nicht erzeugt !");

}
```

Listing 6 (b17.c)

```
#include <stdio.h>
#include <direct.h>     /* für mkdir */

main()
{
  int iret;
  char cpuffer[80], *pc;

  pc= cpuffer;  /* oder pc= &cpuffer[0] */

  printf("\ngib Pfad: ");
  scanf("%s",pc);

  iret= mkdir(pc);
```

```
  if(iret == 0)
    printf("\nDirectory erzeugt !");
  else
    printf("\nDirectory nicht erzeugt !");

}
```

Listing 7 (b18.c)

```
#include <stdio.h>

main()
{
  int i;

  printf("\ngib Taste, dann Entertaste : ");
  i= getchar();
  printf("\ndas war die Taste:");
  putchar(i);
}
```

Listing 8 (b19.c)

```
#include <stdio.h>

main()
{
  int i1,i2,i3,i4;

  puts("Los geht's: ");

  i1= 'H';
  i2= 105;  /* ASCII kleines i, dezimal */
  i3= 0x21; /* ASCII !, hexadezimal, durch 0x
                          kenntlich gemacht */
  i4= '\n'; /* Zeilenvorschub, das ist e i n
                                    Zeichen */
  putchar(i1);
  putchar(i2);
  putchar(i3);
  putchar(i4);
}
```

Listing 9 (b20.c)

```
#include <stdio.h>
#include <conio.h>

main()
{
  int i;

  puts("Los geht's: ");
  while( (i= getch()) != 'x')
    putchar(i);
}
```

Listing 10 (b21.c)

```
#include <stdio.h>

main()
{
  /* dieses Programm führt ein DOS- TYPE aus */
  FILE *f1;
  int i;
  char cfname[80];

  printf("Filenamen angeben: ");
  scanf("%s",cfname);
  printf("\n");

  f1= fopen(cfname,"rb");
  while( (i= fgetc(f1)) != EOF)
    putchar(i);
  fclose(f1);
}
```

obengenannter Programmtypen fast immer schneller als gleichwertige C-Programme, die double per value übergeben (wer es nicht glaubt, probiert es aus). Hier ist call by reference auch für C-Programme sinnvoll.

Diese Vergleiche werden ganz bewußt gebracht, um Hintergründe zu beleuchten, die bei anderen Quellen meist unter den Tisch fallen. Die Aussagen mancher Autoren, "C-Programme sind immer schneller als alles andere" sind unhaltbar. C-Programme können sehr schnell sein, aber nur dann, wenn derartige Feinheiten beachtet werden. Die Vergleiche C und FORTRAN sind aus meiner Sicht deshalb sinnvoll, weil meines Erachtens diese beiden Sprachen viel miteinander gemeinsam haben.

Ehe wir weiter auf Functions eingehen, muß zumindest andeutungsweise der Typ des Vektors in C erläutert werden:

Ein Vektor, auch Array genannt, schreibt sich in C z.B. wie folgt:

```
int    ivektor[1000];
double dvektor[1000];
char   cvektor[1000];
```

Das entspricht bis dahin genau dem aus BASIC und FORTRAN bekannten INTEGER IVEKTOR(1000) usw. Aber eine Unterscheidung ist sehr wichtig: In C beginnen Vektoren immer mit der Zählung 0, d.h. der Vektor ivektor[1000] läuft von ivektor[0] bis ivektor[999]. Der erste Elementindex ist also immer 0. Das ist nicht veränderbar wie in anderen Sprachen.

Es besteht eine direkte Verbindung zwischen Vektoren und Pointern:

Sei

```
int ivektor[1000]
```

ein Vektor

und

```
int*pi
```

ein Pointer auf int

dann kann mit

```
pi = &ivektor[0]
```

dem Pointer pi die Adresse des ersten Vektorelementes von ivektor zugeordnet werden. Dann ist

```
*pi = ivektor[0]
```

Und weiter gilt:

```
*(pi +j)= ivektor[j]
```

Wie ersichtlich, kann nun auf ein Vektorelement j von ivektor entweder mit ivektor[j] oder genauso gut mit *(pi +j) (d.h. liefere Wert der Speicherstelle pi mit dem Offset j) zugegriffen werden. Also können Vektoren und Pointer äquivalent sein: Und genau das sind sie auch, denn der Zugriff auf ivektor[j] wird auf Maschinenebene ohnehin mit *(pi +j) abgewickelt !

Daraus resultierend, sind die Zuordnungen

```
pi = &ivektor[0]
```

und

```
pi = ivektor
```

völlig gleichbedeutend. Ein Vektor ist immer von Haus aus ein Pointer ! Und aus diesem Grund werden in C, FORTRAN und anderen Sprachen Vektoren mit call by reference übergeben und so beginnt in C die Indexzählung konsequent bei 0.

Ob man, weil oft übersichtlicher, im Programm bei der Vektorschreibweise bleibt, oder mit zugeordnetem Pointer und Offset arbeitet, ist von der Sache her belanglos. Vielleicht als griffige Unterscheidung: Vektoren legt man meist statisch an, während Arrays, auf die mit Pointern zugegriffen wird, meist dynamisch angefordert werden. Das mit der dynamischen Speicheranforderung kommt in einer späteren Folge.

Ein typischer Einsatzfall für die Vektorschreibweise in C sind Character-Vektoren. Wie schon gesagt, gibt es in C nicht explizit den Typ des Strings. Das wird mit Charactervektoren erledigt.

Wir wollen ein neues Directory anlegen. Dazu werden wir die Bibliotheks-Function

```
int mkdir(char *path)
```

nutzen (Listing 3):

Was fällt auf ? Zunächst halten wir 80 Bytes für unseren Directorynamen vor, wovon 79 Bytes nutzbar sind, denn C hängt immer als Begrenzung ein \0- Zeichen, das ASCII-NULL- Zeichen an. Bei scanf darf nicht &cpath geschrieben werden, denn cpath ist ja ein Vektor und mithin bereits ein Pointer, eine Adresse. Dasselbe gilt für mkdir.

Was darf man nicht ? So etwas wie Listing 4.

Hier haben wir den Fall, daß der Pointer pc nicht initialisiert

ist: er zeigt irgendwo hin ! System wird abstürzen .. ausprobieren.

Merke: ein Pointer muß immer vor Gebrauch initialisiert werden, z.B. wie b16.c (Listing 5) oder wie b17.c (Listing 6).

3. Einfache I/O- Functions

Das sollen hier Functions sein, die mit der Console arbeiten. Zwei davon, die universellsten, kennen wir bereits:

- printf() : formatierte Ausgabe auf Console
- scanf() : formatierte Eingabe auf Console

Dazu gehört das Header- File stdio.h .

Diese Funktionen sind derart leistungsfähig (und daher auch relativ langsam), daß deren genaues Handling im Run- Time- Library Handbuch nachgeschlagen werden muß.

Einfachere Funktionen sind z.B.:

- int putchar(int) : Schreibe ein Zeichen auf Console
- int getchar(void) : Lese ein Zeichen von Console, Entertaste
- char * gets(char *cstring) : Lese String von Console, kann aber genauso mit scanf erledigt werden

Auch hier stdio.h .
Die Function- Deklaration "int getchar(void)" bedeutet: Function getchar übernimmt keine Parameter und liefert einen Wert vom Typ int zurück.

Beispiel b18.c (Listing 7).

Warum eigentlich "i= getchar()" mit i vom Typ int und nicht i vom Typ char ? Es ist in C einfach so, daß einzelne Zeichen als int behandelt werden, was dann Beispiel b19.c (Listing 8) erlaubt.

Hier wurde auch gleich die Function puts() verwendet, die dann angebracht ist, wenn lediglich Strings ohne alle Formatierung ausgegeben werden sollen.

Sehr praktisch, aber nicht auf allen Systemen verfügbar (aber jedenfalls DOS, OS/2 und TOS) sind

- int getch(void) : liefert Zeichen direkt von Console,ohne Echo
- int getche(void) : wie vor, mit Echo.

Beispiel b20.c (Listing 9).

Mit derartigen Routinen getch() und putchar() lesen z.B. UNIX- Systeme Terminals aus, verarbeiten und senden an Terminal zurück (wobei genauer statt putchar z.B. fputc() benutzt wird).

Noch eine Feinheit: Die Funktionen putchar() und getchar() sind sog. Makros. Dazu gibt es auch fgetchar() und fputchar(), die genauso arbeiten, aber als echte Functions implementiert sind. Makros brauchen mehr Platz, da die Sourcen eingesetzt werden, sind aber schneller als Functions. Makros können gewisse Seiteneffekte haben, so daß, wer ganz sicher gehen will, generell die äquivalenten Functions einsetzt.

Mit den genannten Routinen printf(), scanf(), puts(), getchar(), putchar() und getch() können die meisten Interaktionen an der Console bedient werden. Es gibt noch weitaus mehr Routinen, die aber für den Anfang nicht nötig sind.

4. File I/O- Functions

Die hier aufgeführten Routinen gehören zur Klasse der sog. Stream- I/O Functionen. Sie arbeiten gepuffert, schreiben und lesen also in interne Puffer. Im Gegensatz dazu gibt es die sog. Low- Level I/O Funktionen, die für normale C-Programme nicht nötig sind und darum hier ausgelassen werden.

Was bedeutet Stream I/O ? Man versteht darunter völlig strukturlose Files, eben einen Byte- Stream. In C gibt es keine Records wie in FORTRAN. Alle Dateien sind a priori ungeblockt; eine Blockung kann nur der Programmierer explizit vorsehen. Auch Steuerzeichen wie CR und/oder LF haben zunächst keinerlei Bedeutung.

Hier die häufig genutzten Routinen:

- fprintf() : wie printf, aber in File
- fscanf() : wie scanf, aber von File
- fputc() : wie putchar/ fputchar, aber

Listing 11 (b22.c)

```
#include <stdio.h>

main()
{
  /* dieses Programm führt ein primitives
  *  DOS COPY aus */
  FILE *f1, *f2;
  int i;
  char cfname1[80], cfname2[80];

  printf("\nEingabe-File angeben: ");
  scanf("%s",cfname1);
  printf("\nAusgabe-File angeben: ");
  scanf("%s",cfname2);

  f1= fopen(cfname1,"rb");
  f2= fopen(cfname2,"w+b");
  while( (i= fgetc(f1)) != EOF)
    fputc(i,f2);
  fclose(f1);
  fclose(f2);
}
```

Listing 12 (b23.c)

```
#include <stdio.h>
#define MAXBYTE 32000

int ipuffer[MAXBYTE];

main()
{
  /* dieses Programm führt ein primitives
  *  DOS COPY aus, Pufferung in Vektor,
  *  maximale Filelänge 32000 Bytes */

  FILE *f1, *f2;
  int i,j;
  extern int ipuffer[];

  char cfname1[80], cfname2[80];

  printf("\nEingabe-File angeben: ");
  scanf("%s",cfname1);
  printf("\nAusgabe-File angeben: ");
  scanf("%s",cfname2);

  f1= fopen(cfname1,"rb");
  f2= fopen(cfname2,"w+b");

  /* Puffer beschreiben */
  j= 0;
  while( (ipuffer[j]= fgetc(f1)) != EOF)
    j++;

  /* Puffer auslesen */
  for(i= 0; i j; i++)
    fputc(ipuffer[i],f2);

  fclose(f1);
  fclose(f2);
}
```

Listing 13 (b24.c)

```
#include <stdio.h>
#include <process.h>      /* für exit() */

main(int iargc,char *cargv[])
  {
  /* dieses Programm führt ein primitives
  *  DOS COPY aus, mit Commandline */

  FILE *f1, *f2;
  int i;

  if(iargc != 3)
    {
```

```
    printf("\nGebrauch: b24 quelle ziel..
                                    stop");
    exit(1);
    }

  if((f1= fopen(cargv[1],"rb")) == NULL)
    {
    printf("\nkann quelle nicht öffnen..
                                    stop");
    exit(1);
    }

  if((f2= fopen(cargv[2],"w+b")) == NULL)
    {
    printf("\nkann quelle nicht öffnen..
                                    stop");
    exit(1);
    }

  while( (i= fgetc(f1)) != EOF)
    fputc(i,f2);
  fclose(f1);
  fclose(f2);
  }
```

Listing 14 (b25.c)

```
#include <stdio.h>
#include <process.h>      /* für exit()    */
#include <ctype.h>        /* für toupper() */

main(int iargc,char *cargv[])
{
  /* dieses Programm führt ein primitives
   *  DOS COPY aus, mit Commandline
   *  Zielfile enthält Groß- Buchstaben */

  FILE *f1, *f2;
  int i,j;

  if(iargc != 3)
    {
    printf("\nGebrauch: b24 quelle ziel..
                                    stop");
    exit(1);
    }

  if((f1= fopen(cargv[1],"rb")) == NULL)
    {
    printf("\nkann quelle nicht öffnen..
                                    stop");
    exit(1);
    }

  if((f2= fopen(cargv[2],"w+b")) == NULL)
    {
    printf("\nkann quelle nicht öffnen..
                                    stop");
    exit(1);
    }

  while( (i= fgetc(f1)) != EOF)
    {
    j= toupper(i);
    fputc(j,f2);
    }
  fclose(f1);
  fclose(f2);
  }
```

Listing 15 (b26.c (altbacken))

```
#include <stdio.h>
#include <conio.h>

main()
  {
  int i;

  printf("\nweiter (j/n) ? ");
```

in File

- fgetc() : wie getchar/ fgetchar, aber von File
- fopen() : File öffnen
- fclose() : File schließen

Dazu Beispiel b21.c (Listing 10).

Jede Menge Neues. Zunächst muß ein Pointer von Typ FILE definiert werden, der sog. Filehandle. Die Function fopen() liefert ihn an, wenn das File existiert. Mit fgetc() werden die Bytes nacheinander in der while- Schleife aus dem File, auf das der Filepointer f1 zeigt, ausgelesen und mit putchar auf den Bildschirm geschrieben. Das Abbruchkriterium ist EOF, End-of-File. Diese Konstante ist in stdio.h definiert, darum braucht man sich nicht zu kümmern. Ordentlicherweise schließt man ein File, wenn es nicht mehr gebraucht wird (aber man muß es nicht unbedingt). Erläuterung bedarf die Sequenz "rb" in fopen. Sie bedeutet: Read binary. Das File wird also zum Lesen eröffnet. Binary bedeutet: alle Zeichen ohne Konvertierung übernehmen. Normalerweise wird die Folge CR/LF, also ASCII 13 und 10 in ein einzelnes Newline Zeichen \n übersetzt bzw. später rückgewandelt: sog. Textmode. Das ist hier nicht nötig, also binary.

Andere denkbare Modi sind:

- r+b, r+ : öffnen zum Lesen/Schreiben, binär oder Textmode
- w+r, w+ : öffnen neues File zum Lesen/ Schreiben, binär oder Textmode

Es gibt weitere Modi, die in den entsprechenden Compiler-Handbüchern nachzuschlagen sind.

Bei einem richtigen Programm würde man oben zusätzlich einbauen:

```
if(f1 == NULL)
  {
  printf("\n .. kann
    File nicht öffnen
                ..stop");
  exit(1);
  }
```

NULL ist ebenfalls in stdio.h definiert. Wir können sofort ein weiteres Beispiel entwerfen, das ein primitives Copy-Kommando von DOS, OS/2 oder UNIX nachbildet (b22.c (Listing 11)).

Man nimmt zunächst an, daß immer abwechselnd ein Zeichen gelesen und weggeschrieben wird. Tatsächlich wird aus einem internen Puffer gelesen und in einen zweiten internen Puffer geschrieben. Dieser zweite Puffer wird erst auf Platte geschrieben, wenn er voll ist. Somit arbeitet das Programm relativ rasch. Man kann natürlich auch streng nach dem EVA- Prinzip (Eingabe- Verarbeitung- Ausgabe) vorgehen und würde zunächst File 1 vollständig in einen Vektor einlesen und dann diesen Vektor in File 2 geben. Das kann je nach Compiler und Betriebsystem schneller oder langsamer als obiges Programm b22.c vor sich gehen (b23.c (Listing 12)).

Anhand dieses Programms können gleich weitere neue Features betrachtet werden: Zunächst wird ein Vektor ipuffer definiert und zwar als

```
ipuffer[MAXBYTE]
```

Dabei ist MAXBYTE ein Platzhalter, der gleich am Fileanfang mit

```
#define MAXBYTE 32000
```

(ohne Semikolon !)

festgelegt wird. Mit der #define Anweisung wird der Präprozessor des C-Compilers angewiesen, überall, wo MAXBYTE steht, 32000 einzusetzen. Das ist also ein Textersatz. Mit #define können auch ganze Makros eingesetzt werden. Diese Makro-Technik, d.h. mit #define komplexere C-Anweisungen zu definieren, sollte der Beginner möglichst nicht vornehmen, da dabei durch eventuelle Seiteneffekte schwer lokalisierbare Fehler auftreten können. Besser sind dann echte Functions (langsamer, aber sicher im Gebrauch). #define MAXBYTE 32000 entspricht etwa der FORTRAN PARAMETER Anweisung.

Bemerkenswerter ist jedoch, daß "int ipuffer[MAXBYTE];" vor dem eigentlichen Hauptprogramm definiert ist. Dadurch wird ipuffer zu einem externen Array: Alle Programmeinheiten, d.h. Hauptprogramm und Functions, können darauf zugreifen, wenn in ihnen ipuffer mit der

Speicherklasse "extern" definiert ist. Das Pedant dazu sind die berühmt- berüchtigten (aber extrem effizienten) COMMON-Blöcke in FORTRAN. Da hier aber nur das Hauptprogramm ipuffer nutzt, ist doch vordergründig dieser Aufwand gar nicht nötig.. oder doch ? Die eigentliche ausführliche Erklärung dessen erfolgt in einer späteren Folge.

Hier nur soviel: Würde, wie man gemeinhin annehmen würde, im Hauptprogramm definiert sein: int ipuffer [32000] (oder mit MAXBYTE), so wäre der Speicherbedarf dafür 64000 Bytes. ipuffer wäre kein externes Array, sondern ein Array der Speicherklasse auto: DOS und OS/2 lassen hier nur 32 KB zu ! Außerhalb als Klasse extern sind 64 KB zulässig. Tatsächlich gibt es Möglichkeiten, auch Arrays mit mehr als 64 KB anzufordern (hier nur der Tip für ganz ungeduldige Leser: int huge ipuffer[200000], bei MS-C mit Switch /AL compilieren). Das hängt mit der Segmentstruktur der Intel-Prozessoren zusammen. Aber wir müssen diese Diskussion wirklich auf später verschieben.

Unser Beispiel- Programm entspricht noch in keiner Weise dem echten Copy-Programm aus DOS und OS/2 bzw. dem cp von UNIX. Dort ist ja bekanntlich der Aufruf:

```
copy quelle ziel
```

Es müßte nun eine Möglichkeiten, die Files quelle und ziel direkt von der Commandline zu übernehmen. C wäre keine Sprache für Systemdienst-Programmierung, wenn Beispiel b24.c (Listing 13) nicht ginge.

So ist das bereits ein semiprofessionelles Copy-Programm. Was jetzt noch fehlt, ist die Behandlung von "Wildcards", z.B. *.*. Aber diesen Punkt überlassen wir den Betriebsystem- Entwicklern.

Wie ersichtlich, startet das Hauptprogramm main statt mit main() nun mit

```
main(int iargc,
     char *cargv[])
```

Dabei zählt iargc die Anzahl der übergebenen Argumente und cargv ist ein Pointer auf einen Vektor von Pointern. Wenn nun zwei Argumente, hier quelle und ziel, übergeben werden, so ist iargc= 3, denn laut Konvention ist

cargv[0]	der Programmname, hier b24
cargv[1]	das erste eigentliche Argument, hier quelle
cargv[2]	das zweite eigentliche Argument, hier ziel.

Das ist eine Konvention, die für alle Betriebsysteme gilt. Die Betrachtung enthüllt aber eine weitere Neuigkeit, die sehr von anderen Programmiersprachen abweicht: C-Programme wie main, aber auch C-Functions (ein typischer Fall ist printf) können eine variable Anzahl von Argumenten übernehmen ! Alles andere sind bekannte Elemente, aber das Programm ist schon etwas kompakter in typischem C-Stil, wie

```
if((f1= fopen
    (cargv[1],"rb"))
              == NULL)
```

zeigt.

5. Functionen für Character- Handling

Diese Funktionen führen Tests auf Bytes aus oder konvertieren sie. Sehr praktisch sind z.B. :

- int tolower(int i) :
  konvertiere i in Klein- Buchstaben
- int toupper(int i) :
  konvertiere i in Groß- Buchstaben

Damit können wir sofort ein kleines Beispiel- Programm erstellen, das alle Buchstaben in einem File in Groß-Buchstaben konvertiert und in ein zweites File schreibt (b25.c (Listing 14)).

Ein anderer, gern genutzter, Einsatzfall für toupper und tolower sind Abfragen (b26.c und b27.c (Listing 15)).

Man kann nun auf die gute Idee kommen, statt

```
i= getch();
if(toupper(i) == 'J')
```

gleich kompakter zu schreiben:

```
if(toupper(getch())
              == 'J');
```

Zumindest bei OS/2 ist dann die Überraschung groß, denn das Programm arbeitet nicht wie erwartet ! Daran sieht

```
  i= getch();

  if(i== 'j' || i== 'J')
    printf("\n..es ging weiter");
  else
    printf("\n..es ging nicht weiter");

  }
```

Listing 15 (b27.c (schon besser))

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

main()
  {
  int i;
  printf("\nweiter (j/n) ? ");

  i= getch();
  if(toupper(i) == 'J')
    printf("\n..es ging weiter");
  else
    printf("\n..es ging nicht weiter");

  }
```

Listing 16 (b28.c)

```
#include <stdio.h>

main()
  {
  char *pstring;

  pstring= "Hallo"; /* Initialisieren ! */

  printf("\n%s",pstring);
  printf("\nnun maximal 5 Zeichen eingeben: ----
\b\b\b\b\b");
  scanf("%s",pstring);
  printf("\nAntwort: %s",pstring);
  }
```

Listing 17 (b29.c)

```
#include <stdio.h>
#include <string.h

main()
  {
  char cstring1[11], cstring2[6];

  printf("\nmaximal 5 Zeichen für String 1
                  eingeben: ----\b\b\b\b\b");
  scanf("%s",cstring1);
  printf("\nmaximal 5 Zeichen für String 2
                  eingeben: ----\b\b\b\b\b");
  scanf("%s",cstring2);

  strcat(cstring1,cstring2);

  printf("\nAntwort: %s",cstring1);
  }
```

Listing 18 (b30.c)

```
#include <stdio.h>
#include <string.h>

main()
  {
  char cstring1[11], cstring2[6];

  printf("\nmaximal 5 Zeichen für String 1
                  eingeben: ----\b\b\b\b\b");
```

```
  scanf("%s",cstring1);
  printf("\nmaximal 5 Zeichen für String 2
                  eingeben: ----\b\b\b\b\b");
  scanf("%s",cstring2);

  printf("\nAntwort: %s",strcat(cstring1,
                                cstring2));
}
```

Listing 19

```
PROGRAM STRING

CHARACTER*5 CSTR1, CSTR2

WRITE(*,'('' maximal 5 Zeichen für String 1
                              eingeben: '')')
READ(*,'(A5)') CSTR1
CSTR2 = CSTR1
WRITE(*,'('' Antwort: '',A5)') CSTR2

STOP
END
```

Listing 20 (b31.c)

```
#include <stdio.h>
#include <string.h>

main()
  {
  char cstring1[6], cstring2[6];

  printf("\nmaximal 5 Zeichen für String 1
                  eingeben: ----\b\b\b\b\b");
  scanf("%s",cstring1);

  strcpy(cstring2,cstring1);

  printf("\nAntwort: %s",cstring2);
  }
```

Listing 21 (b32.c)

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

main()
  {
  int ibyte;
  char cstring[6];
  char *pbyte;

  printf("\nmaximal 5 Zeichen fuer String
                  eingeben: ----\b\b\b\b\b");
  scanf("%s",cstring);
  printf("\nnach welchem Zeichen soll gesucht
                                  werden: ");
  ibyte= getche();

  pbyte= strchr(cstring,ibyte);

  if(pbyte != NULL)
    printf("\ngesuchtes Zeichen %c gefunden",
                                    *pbyte);
  else
    printf("\ngesuchtes Zeichen %c nicht
                              gefunden",ibyte);

  }
```

Listing 22 (b33.c)

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
```

man, daß es nichts bringt, alle möglichen Functions zu verschachteln, lieber auf mehrere Zeilen verteilen, was ohnehin übersichtlicher ist.

Wer etwas in der Header-Datei ctype.h stöbert, wird feststellen, daß toupper dort als Makro definiert ist. Derartige Makros können mitunter dann nicht beabsichtigte Effekte hervorrufen, da ihr Sourcecode direkt in die Source des Anwenders eingesetzt wird.

Dem Beginner ist also dringend zu raten, Makros, aber auch Functions zunächst isoliert und ungeschachtelt, wie in b27.c gezeigt, einzubauen.

Wenn das Programm dann sauber läuft, kann man in einem nächsten Schritt zu der kompakteren Form, wie oben beschrieben, übergehen.

Ich möchte trotz alledem empfehlen, nach der Regel vorzugehen: Ein Kommando pro Zeile. Auf das Laufzeitverhalten hat dies nur in seltenen Fällen negativen Einfluß. Die Vorteile sind: Klare, übersichtliche Sourcen, die im Zweifelsfall auch mit einem Debugger leichter zu bearbeiten sind. Auch die Portierbarkeit kann so nur gewinnen.

Weitere, praktische Functions zum Thema Character-Handling sind u.a.:

- int isalnum(int) :
  Test, ob alphanumerischer Character
- int isalpha(int) :
  Test, ob Buchstabe
- int isascii(int) :
  Test, ob ASCII-Character
- int isdigit(int) :
  Test, ob Ziffer
- int islower(int) :
  Test, ob Klein-Buchstabe
- int isupper(int) :
  Test, ob Groß-Buchstabe
- int isspace(int) :
  Test, ob sog. White-Space-Character

Was ist ein White- Space-Character ? Dazu gehören das Leerzeichen, \n, Formfeed \f, Carriage Return \r und die Tabulatoren \t und \v. Für alle o.g. Functions ist das Headerfile ctype.h einzufügen.

6. String- Functionen

Auch auf die Gefahr hin, mich zu wiederholen: In C gibt es nicht den eigentlichen Typ des Strings wie in FORTRAN, BASIC und Pascal. Strings werden in C mit Character-Vektoren bzw. über Pointer abgebildet. Grobe Merkregel für den Anfang:

- Character- Arrays für sozusagen variable Strings, also Strings, die sich im Programmlauf verändern können
- Pointer für festgelegte Strings

Also:

```
char cpuffer[10];
scanf("%s",cpuffer);
```

aber

```
char *pstring;
pstring= "Hallo";
```

Der Unterschied ist: Character-Arrays brauchen nicht zu initialisiert werden, da eine feste Anzahl von Bytes reserviert wird. Character-Pointer müssen immer initialisiert werden. Wenn ein Character-Pointer einmal initialisiert ist, kann er auch verändert werden (b28.c (Listing 16)).

Hier haben wir den Pointer pstring zunächst mit 6 Bytes, d.h. "Hallo" und \0, initialisiert. Man kann im nächsten Schritt mehr als 6 Bytes hineingeben, aber sicherheitshalber bleibt man bei der initialisierten Anzahl. Nebenbei haben wir hier die Escape- Folge \b eingesetzt, d.h. backspace. Im obigen Beispiel geht der Cursor also 5 Zeichen zurück.

Es gibt nun eine Reihe von String- Handling Funktionen. Sehen wir uns die Funktion strcat an: Sie verkettet zwei Strings (b29.c (Listing 17)).

Die Funktion strcat hängt String 2 an String 1 an und ersetzt das \0 in String 1. Das Ergebnis steht in String 1 und wird mit \0 beendet. Wir erkennen, daß es keinen Stringoperator "Verketten" // wie in FORTRAN 77 oder & oder + wie in BASIC gibt.

Die Funktion strcat kann wie oben beschrieben eingesetzt werden:

```
strcat(cstring1,
       cstring2);
```

Der eigentliche Return- Value der Function strcat ist ein Pointer, der auf den modifizierten String cstring1 zeigt.

Mithin kann das Beispiel auch wie b30.c (Listing 18) lauten.

Die nächste String- Funktion ist die Kopierfunktion strcpy. Ein direktes Umspeichern wie in FORTRAN geht in C nicht (Listing 19).

In C muß dazu die Funktion strcpy herangezogen werden (b31.c (Listing 20)).

Wie in b31.c kann auch hier direkt strcpy in printf eingesetzt werden:

```
printf("\nAntwort :
 %s",strcpy(cstring2,
          cstring1));
```

Die Function Declaration ist also:

```
char *strcpy(char
   *cziel, const char
           *cquelle);
```

Natürlich muß darauf geachtet werden, daß cziel mindestens genauso groß ist wie cquelle.

Eine weitere interessante String- Function ist strchr. Deklaration:

```
char *strchr(const
    char *cstring,
       int ibyte);
```

Die Funktion sucht innerhalb des Strings cstring nach dem ersten Auftreten des Zeichens ibyte (das zunächst als char konvertiert wird). Sie liefert einen Pointer auf das gefundene Zeichen zurück oder NULL, wenn es nicht gefunden wurde. Beispiel b32.c (Listing 21).

Die letzte Sektion der Ausgabe ist recht interessant: Mit %c wird das Zeichen ausgegeben, obwohl ibyte eigentlich einen int- Wert repräsentiert. Wir werden das Programm etwas erweitern und lassen uns das Zeichen, die ASCII-Dezimaldarstellung und die Hexadezimaldarstellung auswerfen (b33.c (Listing 22)).

Es wird höchste Zeit, Näheres über Formate für printf, scanf und ihre Verwandten fprintf, fscanf usw. zu erfahren: Wir bringen nur einen Auszug, denn die vollständige Behandlung ist aufwendig und in den Compilermanuals nachzulesen. Formate werden immer mit % eingeleitet:

c	: ein einzelnes Zeichen
d, i	: signed int
D, l, ld, li	: signed long int
f	: float
e, E	: float mit Exponent
g, G	: entspricht e oder f, was besser paßt
lf, le, lE, lg, lG	: wie eben, aber für double
s	: String
x, X	: unsigned hexadezimal int
u	: unsigned int
U	: unsigned long int

Es fällt auf, daß z.B. für long int mehrere Formate äquivalent verwendet werden können. Das hängt mit Kompatibilitätsüberlegungen zu älteren C-Compilern zusammen.

Die Typen für float und double brauchen noch etwas Erklärung. e bzw. g sagt: Exponenten klein schreiben, E und G bedeutet großschreiben. Zu Gleitkommazahlen gibt man allgemein (muß man aber nicht) eine (Mindest-) Feldweite und eine Genauigkeit an, also z.B. %6.2f: Mindestfeldweite 6 Stellen, 2 Nachkommastellen. Beispiel b34.c (Listing 23) zeigt einige Formate für Gleitkomma- Ausgaben.

Dieses Programm produziert die in Tabelle 2 dargestellte Ausgabe.

Die Gleitkomma- Formate bedeuten:

%+G :	ausgeben in möglichst kompakter Darstellung, mit führendem +
%8.0f:	Zahl mindestens 8 Zeichen weit ausgeben, keine Nachkommastellen
%8.5f:	Zahl mindestens 8 Zeichen weit ausgeben, 5 Nachkommastellen
%f:	Zahl ausgeben ohne nähere Spezifikation
%G:	ausgeben in möglichst kompakter Darstellung, ggf führendes -

```
main()
  {
  int ibyte;
  char cstring[6];
  char *pbyte;

  printf("\nmaximal 5 Zeichen fuer String
                eingeben: ----\b\b\b\b\b");
  scanf("%s",cstring);
  printf("\nnach welchem Zeichen soll
                        gesucht werden: ");
  ibyte= getche();

  pbyte= strchr(cstring,ibyte);

  if(pbyte != NULL)
    printf("\ngesuchtes Zeichen %c %i %x
           gefunden",*pbyte,*pbyte,*pbyte);
  else
    printf("\ngesuchtes Zeichen %c %i %x
         nicht gefunden",ibyte,ibyte,ibyte);

  }
```

Listing 23 (b34.c)

```
#include <stdio.h>

main()
{
  float pi=   3.14159;
  float em= - 2.71828;

  double dr=  3.7689e20;
  double ds=  0.123456789e-40;
  double dt= -0.987654321e201;

  printf("\npi= %+G pi= %8.0f pi= %8.5f
                    pi= %f\n",pi,pi,pi,pi);
  printf("\nem= %G em= %8.2f em= %8.5f
                    em= %f\n",em,em,em,em);

  printf("\ndr= %G dr= %8.2le dr= %8.5lE
                  dr= %20.13lE\n",dr,dr,dr,dr);
  printf("\nds= %G ds= %8.2le ds= %8.5lE
                  ds= %20.13lE\n",ds,ds,ds,ds);
  printf("\ndt= %G dt= %8.2le dt= %8.5lE
                    dt= %lf",dt,dt,dt,dt);

}
```

Der Kenner mehrerer Programmiersprachen merkt, daß das C-Formatiersystem weitestgehend von FORTRAN 77 übernommen wurde, mit einer Ausnahme: Die bei F77 eingegebene Feldweite hat bindenden Charakter, wenn die Zahl länger ist, wird sie nicht gedruckt, sondern eine Sternchen- Reihe. Bei C ist die Feldweite eine Mindestangabe. Beides hat Vor- und Nachteile. Die Verwendung von %f oder %lf kann unerwünschte Ergebnisse produzieren, vgl. die Ausgabe von dt mit %lf. Die Übereinstimmung der Formatiersysteme von C und F77 ist kein Zufall, denn Kernighan und Ritchie (die C- Erfinder) waren an der Entwicklung der FORTRAN-Präprozessoren ratfor und efl für UNIX- Systeme beteiligt.

Auch bei %i, %li und %s können Feldweiten angegeben werden, z.B. %8i. Ähnliche Überlegungen gelten für das Einlesen mit scanf, fscanf etc. Es lohnt sich, mit Formaten zu experimentieren, um die Möglichkeiten kennenzulernen.

In der dritten Folge werden wir auf Arrays und Pointer und die diversen Speichermodelle eingehen.

Dr.-Ing. Frank Rieg

```
pi= +3.14159 pi=       3 pi=  3.14159  pi=  3.141590
em= -2.71828 em=   -2.72 em=  -2.71828 em=  -2.718280
dr= 3.7689E+020 dr= 3.77e+020 dr= 3.76890E+020 dr= 3.7689000000000E+020
ds= 1.23457E-041 ds= 1.23e-041 ds= 1.23457E-041 ds= 1.2345678900000E-041
dt= -9.87654E+200 dt= -9.88e+200 dt= -9.87654E+200 dt= -987654321000.... (etliche Nullen)
```

Multiplot

Plotten auf dem ThinkJet von Ulrich Laag

Das Programm Multiplot dient zur Ausgabe von Funktionsgrafen auf dem ThinkJet-Drucker. Es können mehrere Funktionen gleichzeitig gezeichnet werden. Die Längen der X- und Y-Achse und deren Maßstab sind frei wählbar.

Ich habe mich bemüht, das Programm so zu schreiben, daß es möglichst schnell arbeitet. Die Druckzeit wird aber im Wesentlichen von den gewählten Achsenlängen und der Anzahl der Funktionen bestimmt.

Zur Bedienung:

"Anzahl der f(x)"

Es ist die Anzahl der zu plottenden Funktionen einzugeben. Dann End-Line drücken.

"f(n)="

Eingabe der n-ten Funktion über das Tastenfeld. Es müssen alle Rechenzeichen mit eingegeben werden, z.B. 6*X-3 oder 9*sin(X^2) aber nicht 3X^3-2X+6 !

Die Tasten 'UP' (#50) oder 'DOWN' (#51) schließen die Eingabe einer Funktion ab. Man sieht dann die nächste oder vorherige Funktion. Bei der letzten Funktion schließt 'DOWN' die Eingabe aller Funktionen ab.

"Neue Funktionen ? (J)"

Nach einem Ausdruck zeigt der Rechner diesen Text. Möchte man jetzt mit den gleichen Funktionen einen neuen Ausdruck machen, wird einfach irgendeine Taste außer der 'J' (Ja)-Taste gedrückt.

Die anderen Abfragen beziehen sich alle auf das Ausgabeformat der Achsen und erklären sich eigentlich von selbst. Die Bilder 1,2 und 3 verdeutlichen ihre Bedeutung sicherlich ausreichend.

Einige Erklärungen zum Programm:

Die Diagramme werden Zeilenweise in Y-Richtung aufgebaut. Die Zeile wird in einem String erzeugt und muß somit immer aus einer durch acht teilbaren Anzahl von Punkten bestehen. Ein auf eine eins gesetztes Bit im String entspricht einem gedruckten Punkt in der Ausgabezeile. Die Punkte sind in jeder Zeile von links nach rechts, mit Null beginnend, durchnummeriert. In der benutzerdefinierten Funktion 'FNB$' (Zeilen 110 ... 130) wird der N-te Punkt der Zeile C$ auf eine eins gesetzt. Ungültige Angabenfür N werden ignoriert.

Die mathematischen Funktionen werden als Strings in dem Feld F$ abgespeichert (Zeilen 150 ... 270).

Im Unterprogramm 'PAR' (Zeile 490 ff) werden die Parameter der X- und Y-Achse abgefragt. Die für die Grafikausgabe umgerechneten Werte übergibt das Unterprogramm 'PAR' an die Felder P1(n) für die Y- und P2(n) für die X-Achse. In den Zeilen 290 .. 340 werden aus diesen Werten einige Basisdruckzeilen erzeugt (siehe auch Erklärung der Variablen).

Die Zeile 350 druckt 'Y->' in Breitschrift und die Programmzeilen 360 und 370 bereiten den Drucker auf die Ausgabe von Grafikzeilen vor. Mit der Schleife A4 (Zeile 380) beginnt dann die Erzeugung der Grafikzeilen.

Zeilen 390 und 400: Y-Achse.

Zeile 410 : Grafikzeile mit Rasterpunkten.

Zeilen 420 ... 450: Berechnung der Funktionswerte und Eintragung in die Basiszeilen.

In der Zeile 460 wird dann die komplette Grafikzeile an den Drucker gesendet. Ist der Ausdruck beendet, wird der ThinkJet wieder in den Normalmodus geschaltet und ein akustisches Signal ausgegeben (Zeilen 470 ... 480).

Wichtige Variablen und ihre Bedeutung:

P3 = Länge der Y-Achse in Byte.

G$(1) = Leerstring mit Punkt an der Stelle der X-Achse.

G$(2) = Leerstring mit Punkt an der Stelle der X-Achse und mit Punkt an der Stelle unter der X-Achse und mit Rasterpunkten.

G$(3) = String mit der Länge der Y-Achse gefüllt mit Punkten (Y-Achse).

G$(4) = Übergabestring. Er enthält alle zum Ausdruck der Reihe benötigen Punkte.

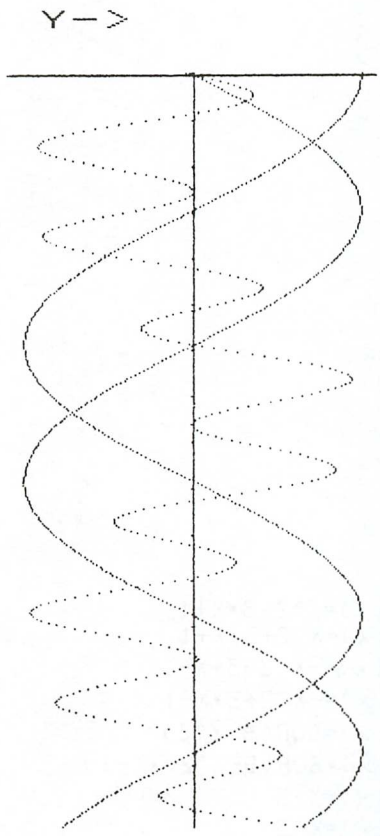
G2$ = Beginn der Grafikzeile mit Angabe der Länge.

N = Position des Punktes im String. Die DEF FNB$ setzt an der Stelle N in der Grafikzeile einen Punkt.
Beispiele: N=13; im String C$ wird im 2 Byte das Bit mit der Wertigkeit 2^2 auf 1 gesetzt.
N=16; Byte 3, Bit 2^7.

```
f1(x)=SIN(X)
f2(x)=COS(X)
f3(x)=SIN(2.3*X)*COS(3.5*X)

Y min,max          : -1.1   1.1
Y-Pos. d. X-Achse:   0
Y-Rasterteilung    :  2

X min,max          :  0   500
X-Pos. d. Y-Achse:   0
X-Rasterteilung    :  30
```



	(1)	(2)	(3)	(4)	(5)	(6)
P	Länge der Achse in cm	min. Wert	max. Wert	Position der Achse	Rasterteilung	
P1	Länge der Y-Achse in Punkten	Y min. Wert	delta Y pro Punkt	Y-Position der X-Achse in Punkten	Y-Rasterteilung in Punkten	Y-Position des ersten Rasterpunktes
P2	Länge der X-Achse in Punkten	X min. Wert	delta X pro Punkt	X-Position der Y-Achse in Punkten	X-Rasterteilung in Punkten	X-Position des ersten Rasterpunktes

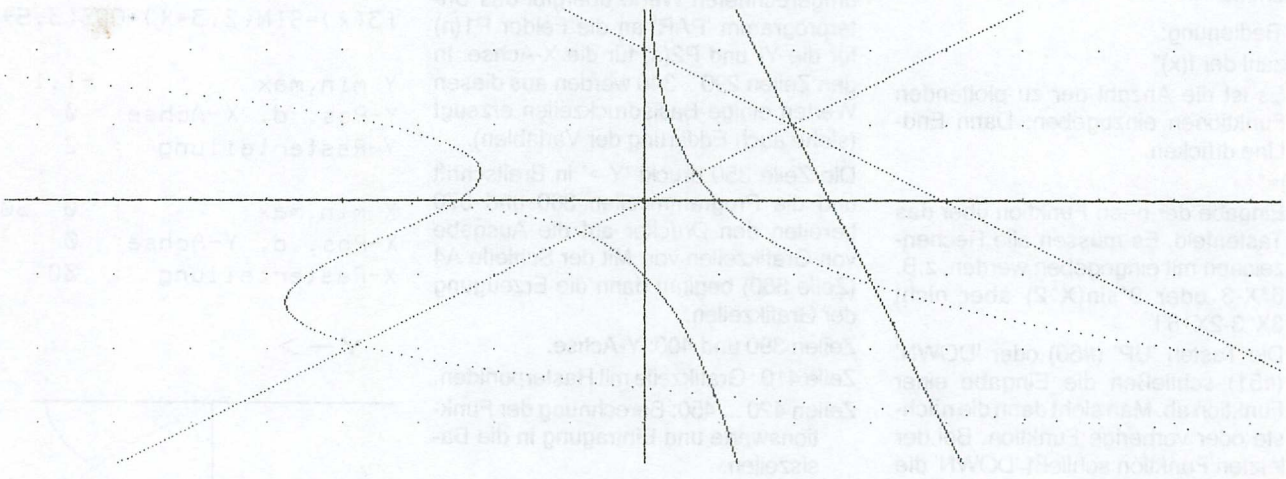
A = Anzahl der mathematischen Funktionen.
F$(n) = mathematische Funktionen als String.

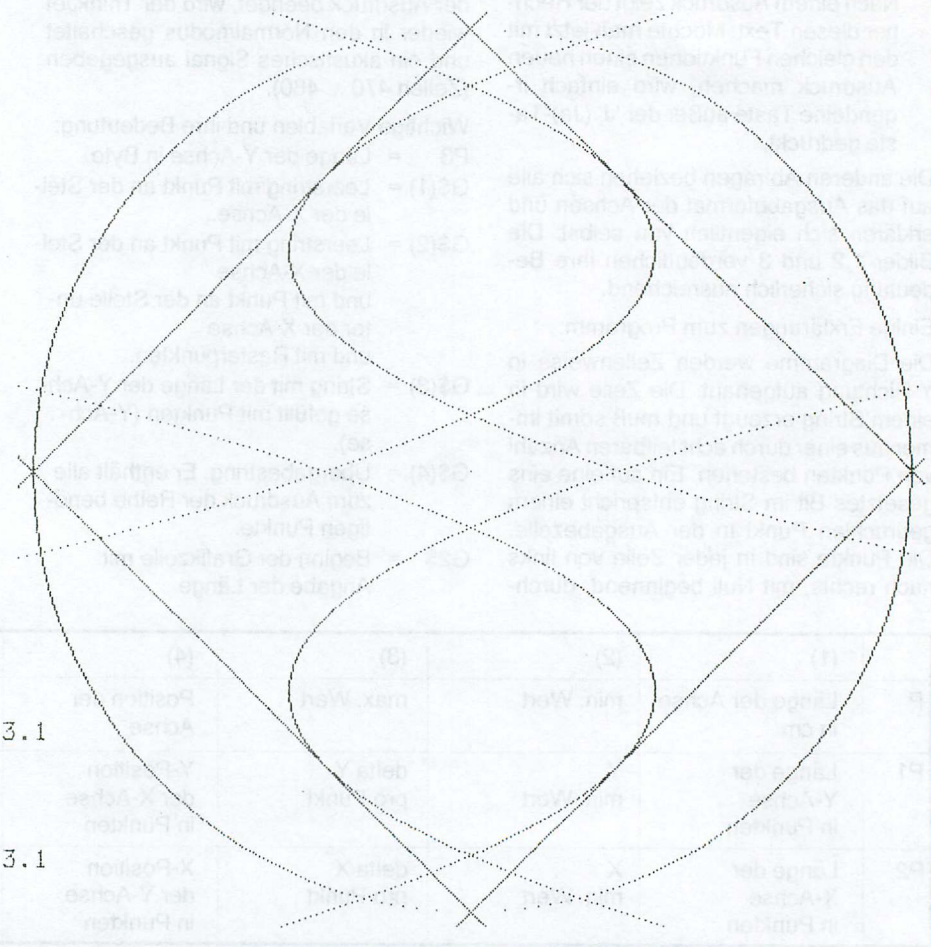
Ich hoffe, die Erklärung war einigermaßen verständlich. Über einen Hinweis, wie die Ausführungsgeschwindigkeit gesteigert werden kann, würde ich mich sehr freuen.

```
f1(x)=X^3-1.5*X^2-1.5*X-2.5
f2(x)=EXP(X)
f3(x)=LN(X)
f4(x)=-2*X
f5(x)=2*X+4
f6(x)=X/2+2.5

Y min,max        : -8.5  8.5
Y-Pos. d. X-Achse:  0
Y-Rasterteilung  :  1

X min,max        : -2.5  3.5
X-Pos. d. Y-Achse:  0
X-Rasterteilung  :  1
```





```
f1(x)=X^2-3*X+1
f2(x)=X^2+3*X+1
f3(x)=-X^2-3*X-1
f4(x)=-X^2+3*X-1
f5(x)=SQR(9-X^2)
f6(x)=SQR(9-X^2)*(-1)
f7(x)=X+3
f8(x)=X-3
f9(x)=-X+3
f10(x)=-X-3

Y min,max        : -3.1  3.1
Y-Pos. d. X-Achse:  10
Y-Rasterteilung  :  10

X min,max        : -3.1  3.1
X-Pos. d. Y-Achse:  10
X-Rasterteilung  :  10
```

```
  1 ! MULTIPLOT von Ulrich Laag , CCD(1703) , LEX-File: STRINGLX , CUSTUTIL
100 OPTION BASE 1 @ STD @ DESTROY ALL @ DEFAULT OFF
110 DEF FNB$[80](C$,N) @ IF N<0 OR N>P1(1)-1 THEN 130
120 N1=IP(N/8)+1 @ N2=7-MOD(N,8) @ C$=SBIT$(C$,N1,N2,1)
130 FNB$=C$ @ END DEF
140 PRINTER IS :HP2225B @ GOTO 170
150 DISP "Neue Funktionen ? (J)"
160 IF KEYWAIT$#"J" THEN GOTO 250
170 INPUT "Anzahl der f(x)";A
180 DESTROY F$ @ DIM F$(A)[75]
190 FOR A1=1 TO A @ C0$=CHR$(27)&"(" @ C1$=CHR$(27)&")"
200 INLINE C0$&"f"&STR$(A1)&"(X)= "&C1$&F$(A1),1,7,"#50#51",A2
210 F$(A1)=DISP$ @ ON A2 GOTO 220,240
220 IF A1>1 THEN A1=A1-1
230 GOTO 200
240 NEXT A1
250 FOR A3=1 TO A
260 PRINT "f"&STR$(A3)&"(x)="&F$(A3) @ NEXT A3
270 PRINT ""
280 CALL PAR("Y","X",P1()) @ CALL PAR("X","Y",P2())
290 P3=P1(1)/8 @ DIM G$(4)[P3] @ G2$=CHR$(27)&"*b"&STR$(P3)&"W"
300 G$(1)=RPT$(CHR$(0),P3) @ G$(1)=FNB$(G$(1),P1(4))
310 G$(2)=G$(1) @ G$(2)=FNB$(G$(2),P1(4)-1)
320 FOR A5=P1(6) TO P1(1)-1 STEP P1(5)
330 G$(2)=FNB$(G$(2),A5) @ NEXT A5
340 G$(3)=RPT$(CHR$(255),P3)
350 PRINT CHR$(10)&CHR$(27)&"&k1S Y-)"&CHR$(27)&"&k0S"
360 PRINT "";"" @ ENDLINE ""
370 PRINT CHR$(27)&"*r640S"&CHR$(27)&"*rA"; @ X=P2(2)-P2(3)
380 FOR A4=-1 TO P2(1)
390 IF A4=P2(4) THEN G$(4)=G$(3) @ GOTO 460
400 IF A4=P2(4)-1 THEN G$(4)=G$(2) @ GOTO 420
410 IF MOD(A4,P2(5))-P2(6)=0 THEN G$(4)=G$(2) ELSE G$(4)=G$(1)
420 FOR A6=1 TO A
430 ON ERROR GOTO 450
440 N=IP((VAL(F$(A6))-P1(2))/P1(3)+.5) @ G$(4)=FNB$(G$(4),N)
450 NEXT A6
460 PRINT G2$;G$(4); @ X=X+P2(3) @ NEXT A4
470 PRINT CHR$(27)&"*rB"; @ ENDLINE @ PRINT CHR$(10)&CHR$(10)
480 BEEP 1000 @ GOTO 150
490 SUB PAR(P1$,P2$,P())
500 DISP "Länge "&P1$&"-Achse (cm):";
510 INPUT P(1) @ P(1)=IP(ABS(P(1))/.21)*8
520 IF P1$="Y" AND P(1)>640 THEN P(1)=640
530 T$=P1$&" min,max        : " @ DISP T$;
540 INPUT P(2),P(3) @ IF P(2)>=P(3) THEN BEEP 300 @ GOTO 530
550 PRINT T$;P(2);P(3)
560 T$=P1$&"-Pos. d. "&P2$&"-Achse: " @ DISP T$;
570 INPUT "","0";P(4) @ PRINT T$;P(4)
580 T$=P1$&"-Rasterteilung  : " @ DISP T$;
590 INPUT P(5) @ PRINT T$;P(5) @ PRINT ""
600 P(3)=(P(3)-P(2))/P(1) @ P(4)=IP((P(4)-P(2))/P(3)+.5)
610 P(5)=IP(P(5)/P(3)+.5) @ P(6)=MOD(IP(-P(2)/P(3)+.5),P(5))
620 END SUB
```

Ulrich Laag (1703)
Dönberger Straße 80
5600 Wuppertal 1

MEMLOST

von Dr. Karlheinz Lutz

BASIC 1440

Wer noch nie einen Systemabsturz mit seiner HP-71-Anlage und den angeschlossenen Geräten erlebt hat, ist entweder ein ganz großer Meister oder ein blutiger Anfänger. Wen aber sein kleiner Freund mit der unschuldigen Meldung *Memory Lost* schon öfters beglückt hat, der weis, wie schmerzlich der Verlust aller Daten in seinem Computer ist und wieviel Mühe und Zeit es kostet, ihm wieder alles beizubringen, was er schon einmal gelernt hatte.

Das Programm *MEMLOST* hilft beim Wiedergewinnen seiner alten Kraft und stellt sicher, daß an alles Nötige gedacht wird. Freilich muß man auch ein wenig Vorarbeit leisten, z.B. den KEY-File-Namen wissen und alle Programme, die man geladen haben will, in die Zeile 1000 DATA .... vorher eingetragen haben. Dann bootet aber das Programm wie bei einem großen Computer.

Programmbeschreibung

Das Programm prüft an Hand des Datums, ob das Gedächtnis tatsächlich verloren ist, und stoppt, wenn das Datum nicht gleich 00/01/01 ist (dann ist nämlich nicht das ganze Memory leer - Ausnahme am 1.1.2000 - an diesem Tag mit RUN,40 starten).

Nacheinander sind *Datum* und *Uhrzeit* einzugeben, es folgt die Frage, ob *EXAKT* gesetzt und ein Wert für *AF* eingegeben werden soll (diesen Wert sollte man sich notiert haben).

In Zeile 130 kann ein String für *STARTUP* eingegeben werden, der bei jedem Einschalten des HP-71 aktiviert wird. Im vorliegenden Beispiel wird folgendes veranlaßt:

- Aufruf eines SUB-Programmes (im LIBrary-File LIB6) zur Tastaturzuweisung (KB)
- Zuweisung des von HP empfohlenen ENDLINE-Strings
- Aufruf des SUB-Programmes A4 in LIB6 (DIN A 4 Format).

Im weiteren wird nach dem KEY-File-Namen gefragt, falls ein Null-String eingegeben wird, überspringt der Rechner diesen Abschnitt, sonst wird geprüft, ob unter dem angegebenen Namen ein File im Massenspeicher gefunden wird und ob dieser den richtigen File-Typ besitzt. Wenn nicht, wird die Eingabeaufforderung wiederholt. Im zutreffenden Fall wird der Schlüssel-File direkt zu KEYS in den HP-71 geladen.

Darauf folgend werden die in Zeile 1000 DATA .... angegebenen Files vom Massenmedium in den HP-71-Speicher übertragen und gesichert (wenn nicht das S in der INPUT-Zeile gelöscht wurde). Nach dem Abarbeiten der in der DATA-Zeile befindlichen Files wird gefragt, ob noch weitere Files geladen werden sollen.

Zuletzt ist beispielhaft an einige Programme gedacht, die man vielleicht laufen lassen möchte, wie z.B. CMDSTK zur Änderung der Zahl gespeicherter Befehle im Commandostack (1 bis 16 ist möglich) oder zur Initialisierung einer Tastatur (diese Datei hat hier den Namen ESCAPE2). Dann verläßt der Rechner allerdings MEMLOST.

Anspringbar sind die LABEL:

- K mit CONT K oder RUN,K
  zum Laden eines KEY-Files, wenn keiner gespeichert ist
- F mit CONT F oder RUN,F
  zum Laden von Files aus dem Massenspeicher.

Wer MEMLOST auf einer Diskette bereitstellt, sich dazu den KEY-File-Namen merkt oder in Zeile 180 notiert, den STARTUP-File weis und die für ihn wichtigen Files in Zeile 1000 festhält - evtl. in den Zeilen 330 bis 350 statt ESCAPE2 seinen Programmnamen einsetzt - , der braucht einen Systemabsturz Memory Lost nicht mehr zu fürchten.

```
10 ! MEMLOST  Anlauf nach 'Memory Lost'  26.11.1989
15 ! Dr. Karlheinz Lutz,Bayernstr.39, 8501 Rückersdorf bei Nürnberg
20 DELAY INF
30 D0$=DATE$ @ IF D0$#"00/01/01" THEN DISP "Memory vorhanden " @ STOP
40 D1$="jjjj/mm/tt" @ ON ERROR GOSUB 'ERR'
50 INPUT "SETDATE ",D1$;D1$ @ SETDATE D1$
60 ON ERROR GOSUB 'ERR' @ INPUT "SETTIME ",'hh:mm:ss';D2$ @ SETTIME D2$
70 DISP "EXACT setzen ? J/N "
80 K$=UPRC$(KEY$) @ IF K$="J" THEN EXACT ELSE IF K$="N" THEN 90 ELSE 80
90 DISP "AF setzen ? J/N "
100 K$=UPRC$(KEY$) @ IF K$="J" THEN 110 ELSE IF K$="N" THEN 120 ELSE 100
110 INPUT "AF= ",STR$(AF);A
120 DIM D3$[95]
130 INPUT "STARTUP : ","CALL KB@ENDLINE ' '&CHR$(10)&CHR$(13)@CALL A4";D3$
140 STARTUP D3$
150 DISP "Warnton aus ? J/N "
160 K$=UPRC$(KEY$) @ IF K$="J" THEN BEEP OFF ELSE IF K$="N" THEN BEEP ON ELSE 16
0
170 'K': ON ERROR GOSUB 'ERR'
180 INPUT "KEY-Name laden ","xKEY";Y$ @ IF Y$="" THEN 'F'
190 COPY Y$&":TAPE" TO KEYS
200 'F': DIM F$(10)[10],S$[1],T$[5] @ T$=":TAPE"
210 FOR I=1 TO 10 @ READ F$(I) @ NEXT I
220 FOR I=1 TO 10
230 IF F$(I)="" THEN 270
240 ON ERROR GOSUB 'ERR' @ DELAY INF @ INPUT "lade ",F$(I)&",S";F$(I),S$
```

```
250 COPY T$ TO F$(I) @ IF UPRC$(S$)="S" THEN SECURE F$(I)
260 NEXT I
270 DISP "weitere Files laden ? J/N "
280 K$=UPRC$(KEY$) @ IF K$="J" THEN 240 ELSE IF K$="N" THEN 290 ELSE 280
290 DISP "soll CMDSTK laufen ? J/N "
300 K$=UPRC$(KEY$) @ IF K$="J" THEN 310 ELSE IF K$="N" THEN 330 ELSE 300
310 INPUT "speichere x Befehle ","16";X
320 COPY :TAPE TO CMDSTK @ CALL CMDSTK(X) @ PURGE CMDSTK
330 DISP "soll ESCAPE2 laufen ? J/N "
340 K$=UPRC$(KEY$) @ IF K$="J" THEN 350 ELSE IF K$="N" THEN 'E' ELSE 340
350 RUN ESCAPE2:TAPE
360 'E': DESTROY ALL @ DISP "Ende " @ END
370 'ERR': IF ERRL=50 AND ERRN=11 THEN 40
380 IF ERRL=60 AND ERRN=11 THEN 60
390 IF ERRL=190 AND (ERRN=255022 OR ERRN=63) THEN 'K'
400 IF ERRL=250 AND ERRN=255022 THEN DISP ERRM$ @ GOTO 270
410 DISP "Fehler ";ERRL;ERRM$ @ RETURN
1000 DATA LIB6,E6,FINDLEX,DISKCAT,,,,,,,,,,,,,,
```

Dr. Karlheinz Lutz
Bayernstraße 39
D-8501 Rückersdorf

ZEICHENsatz-Programm

PDIV - Polynomdivision

BASIC 729 Bytes

Dieses Programm entspricht im wesentlichen dem von Günther Bereths (1/85), benutzt aber Sonderzeichen, die durch Aufruf von ZEICHEN zur Verfügung stehen. Dadurch wird eine recht ansprechende Ausgabe der Lösungspolynome erreicht.

Zeichen

naturwissenschaftliche Sonderzeichen

BASIC 1224 Bytes

Dieses Programm stellt einige interessante Sonderzeichen für das Display zur Verfügung. Integralzeichen, Index- und Exponentenzahlen und diverse Mathematikzeichen sind mit dabei!

EDLEX-Lex-File

REM - Löscht REM-Zeilen

BASIC 299 Bytes

Diese kurze Routine löscht alle Bemerkungen in einem Programm, die durch '!' angegeben sind. Nach Ausführung dieser Routine liegt im Speicher ein zweiter File vor, der keine REM-Zeilen mehr enthalten sollte.

PDIV

```
 10 DESTROY ALL @ STD @ OPTION BASE 0 @ DIM L$[200] @ DELAY 0,0
 20 INPUT "Zhlergrad = ";A
 30 INPUT "Nennergrad = ";B
 40 FOR I=A TO 0 STEP -1 @ DISP "zx"&CHR$(I+149)&" = "; @ INPUT "";Z(I) @ NEXT I
 50 FOR I=B TO 0 STEP -1 @ DISP "nx"&CHR$(I+149)&" = "; @ INPUT "";N(I) @ NEXT I
 60 FOR I=A-B TO 0 STEP -1
 70 C(I)=Z(B+I)/N(B)
 80 FOR J=B TO 0 STEP -1
 90 Z(J+I)=Z(J+I)-C(I)*N(J)
100 NEXT J @ NEXT I
110 FOR I=A-B TO 0 STEP -1
120 IF C(I)=0 THEN 160
130 IF C(I)>0 THEN A$="+" ELSE A$=""
140 IF I=1 THEN L$=L$&A$&STR$(C(I))&"x" @ GOTO 160
150 IF I=0 THEN L$=L$&A$&STR$(C(I)) ELSE L$=L$&A$&STR$(C(I))&"x"&CHR$(I+149)
160 NEXT I
170 FOR I=B-1 TO 0 STEP -1
180 IF Z(I)=0 THEN 220
190 IF Z(I)>0 THEN A$="+" ELSE A$=""
200 IF I=1 THEN R$=R$&A$&STR$(Z(I))&"x" @ GOTO 220
210 IF I=0 THEN R$=R$&A$&STR$(Z(I)) ELSE R$=R$&A$&STR$(Z(I))&"x"&CHR$(I+149)
220 NEXT I
230 DELAY 8,8
240 DISP L$; @ IF R$#"" THEN DISP " Rest: ";R$ ELSE DISP
250 DISP "  C¥ontinue   E¥nd"
260 A$=KEYWAIT$ @ IF A$="C" THEN 10 ELSE IF A$="E" THEN 270 ELSE 260
270 DELAY 0,0 @ STD @ DISP "Done: PDIV"
```

REM

```
 10 DIM Z$[96] @ INPUT "File ? ";F$ @ DISP "working..." @ I=0
 20 COPY F$ TO F$&"1" @ F$=F$&"1"
 30 TRANSFORM F$ INTO TEXT
 40 ASSIGN #1 TO F$ @ RESTORE #1,1
 50 WHILE I<FILESZR(F$)
 60 READ #1,I;Z$
 70 IF POS(Z$,' ! ') THEN Z$=Z$[1,POS(Z$," ! ")-1]
 80 IF LEN(Z$)<=5 THEN Z$=""
 90 IF LEN(Z$) THEN REPLACE #1,I;Z$ ELSE DELETE #1,I @ GOTO 110
100 I=I+1
110 END WHILE
120 OFF ERROR @ ASSIGN #1 TO *
130 TRANSFORM F$ INTO BASIC
140 DELAY 0 @ DISP "Done: REM"
```

Dennis Föh
Hermann-Hanker-Straße 17
3400Göttingen

ZEICHEN

```
 10 CALL ZEICHEN

============================================================
 20 SUB ZEICHEN
 30 DELAY 0,0 @ DISP "working..."
 40 CHARSET ""
 50 CHARSET CHR$(255)&CHR$(255)&CHR$(195)&CHR$(129)&CHR$(129)&CHR$(0)
 60 CHARSET CHARSET$&CHR$(63)&CHR$(4)&CHR$(4)&CHR$(60)&CHR$(0)&CHR$(252)
 70 CHARSET CHARSET$&CHR$(36)&CHR$(36)&CHR$(60)&CHR$(0)&CHR$(129)&CHR$(129)
 80 CHARSET CHARSET$&CHR$(195)&CHR$(255)&CHR$(255)
 90 CHARSET CHARSET$&CHR$(223)&CHR$(209)&CHR$(145)&CHR$(145)&CHR$(243)&CHR$(128)
100 CHARSET CHARSET$&CHR$(255)&CHR$(193)&CHR$(195)&CHR$(131)&CHR$(255)&CHR$(128)
110 CHARSET CHARSET$&CHR$(145)&CHR$(145)&CHR$(255)&CHR$(145)&CHR$(145)&CHR$(128)
120 CHARSET CHARSET$&CHR$(129)&CHR$(129)&CHR$(255)&CHR$(1)&CHR$(1)
130 CHARSET CHARSET$&CHR$(64)&CHR$(128)&CHR$(126)&CHR$(1)&CHR$(2)&CHR$(0)
140 CHARSET CHARSET$&CHR$(0)&CHR$(0)&CHR$(2)&CHR$(7)&CHR$(2)&CHR$(0)
150 CHARSET CHARSET$&CHR$(0)&CHR$(0)&CHR$(2)&CHR$(2)&CHR$(2)&CHR$(0)
160 CHARSET CHARSET$&CHR$(0)&CHR$(112)&CHR$(136)&CHR$(136)&CHR$(112)&CHR$(0)
170 CHARSET CHARSET$&CHR$(0)&CHR$(32)&CHR$(16)&CHR$(8)&CHR$(248)&CHR$(0)
180 CHARSET CHARSET$&CHR$(0)&CHR$(144)&CHR$(200)&CHR$(168)&CHR$(144)&CHR$(0)
190 CHARSET CHARSET$&CHR$(0)&CHR$(136)&CHR$(168)&CHR$(168)&CHR$(112)&CHR$(0)
200 CHARSET CHARSET$&CHR$(0)&CHR$(56)&CHR$(32)&CHR$(32)&CHR$(248)&CHR$(0)
210 CHARSET CHARSET$&CHR$(0)&CHR$(184)&CHR$(168)&CHR$(168)&CHR$(72)&CHR$(0)
220 CHARSET CHARSET$&CHR$(0)&CHR$(112)&CHR$(168)&CHR$(168)&CHR$(64)&CHR$(0)
230 CHARSET CHARSET$&CHR$(0)&CHR$(136)&CHR$(72)&CHR$(40)&CHR$(24)&CHR$(0)
240 CHARSET CHARSET$&CHR$(0)&CHR$(80)&CHR$(168)&CHR$(168)&CHR$(80)&CHR$(0)
250 CHARSET CHARSET$&CHR$(0)&CHR$(16)&CHR$(168)&CHR$(168)&CHR$(112)&CHR$(0)
260 CHARSET CHARSET$&CHR$(0)&CHR$(14)&CHR$(17)&CHR$(17)&CHR$(14)&CHR$(0)
270 CHARSET CHARSET$&CHR$(0)&CHR$(4)&CHR$(2)&CHR$(1)&CHR$(31)&CHR$(0)
280 CHARSET CHARSET$&CHR$(0)&CHR$(18)&CHR$(25)&CHR$(21)&CHR$(18)&CHR$(0)
290 CHARSET CHARSET$&CHR$(0)&CHR$(17)&CHR$(21)&CHR$(21)&CHR$(10)&CHR$(0)
300 CHARSET CHARSET$&CHR$(0)&CHR$(7)&CHR$(4)&CHR$(4)&CHR$(31)&CHR$(0)
310 CHARSET CHARSET$&CHR$(0)&CHR$(23)&CHR$(21)&CHR$(21)&CHR$(9)&CHR$(0)
320 CHARSET CHARSET$&CHR$(0)&CHR$(14)&CHR$(21)&CHR$(21)&CHR$(8)&CHR$(0)
330 CHARSET CHARSET$&CHR$(0)&CHR$(17)&CHR$(9)&CHR$(5)&CHR$(3)&CHR$(0)
340 CHARSET CHARSET$&CHR$(0)&CHR$(10)&CHR$(21)&CHR$(21)&CHR$(10)&CHR$(0)
350 CHARSET CHARSET$&CHR$(0)&CHR$(2)&CHR$(21)&CHR$(21)&CHR$(14)&CHR$(0)
360 CHARSET CHARSET$&CHR$(0)&CHR$(36)&CHR$(18)&CHR$(36)&CHR$(72)&CHR$(36)
380 CHARSET CHARSET$&CHR$(84)&CHR$(82)&CHR$(81)&CHR$(82)&CHR$(84)&CHR$(0)
390 CHARSET CHARSET$&CHR$(68)&CHR$(68)&CHR$(95)&CHR$(68)&CHR$(68)&CHR$(0)
400 CHARSET CHARSET$&CHR$(8)&CHR$(8)&CHR$(48)&CHR$(255)&CHR$(1)&CHR$(1)
410 DELAY 0 @ DISP "Done: ZEICHEN"
```

Codierungstheorie

von Michael Schilli

HP-41 CV, 665 Zeilen

1. Allgemeines

Die Codierungstheorie findet immer dann Anwendung, wenn es gilt, einzelne Symbole eines Quell-Zeichenvorrats Q (z.B. {A,B,...G}) eindeutig auf Elemente eines Ziel-Zeichenvorrats Z (z.B. {0,1}) abzubilden. Hierbei kann, wie im (in Klammen angegebenen) Beispiel vorgegeben, der abzubildende Zeichenvorrat Q ruhig mehr Elemente enthalten als der Ziel-Zeichenvorrat Z: in diesem Fall wird jedem Element aus Q eine Kombination aus mehreren Zeichen der Vorratsmenge von Z zugeordnet (beispielsweise wie in Abb.1).

Ein typisches Anwendungsbeispiel aus der Praxis ist hierfür die Anpassung einer Nachrichtenquelle, die n verschiedene Symbole mit den Auftrittswahrscheinlichkeiten p(Q1), p(Q2), ... , p(Qn) aussenden kann. Dies kann an einem Binärkanal, beispielsweise eine elektrische Leitung, die mit den digitalen Signalen "0" und "1" betrieben wird und die ausgesandten Symbole der Quelle über eine weite Strecke hinweg übertragen kann, geschehen. Jedem abgegebenen Symbol aus Q wird nun eine bestimmte Folge von Nullen und Einsen zugeordnet und diese dann in den Kanal eingespeist. Die Art und Weise dieser Zuordnung, d.h. die Abbildungsvorschrift, kann nun sehr unterschiedlich realisiert sein: Neben der Zuweisung durch Durchnummerieren mit Dualzahlen (wie in Abb. 1 geschehen) existieren mehrere einschlägige Verfahren, von denen an dieser Stelle zwei (das Shannon-Fano und das Huffman Verfahren) vorgestellt werden sollen. Beide Methoden berücksichtigen zusätzlich die Auftrittswahrscheinlichkeiten der Quellsymbole - es ist nämlich wichtig zu wissen, daß es für eine effiziente Signalübertragung entscheidend ist, die mittlere Zahl der zu übertragenden Bits je übermittelten Quellsignal möglichst klein zu halten. Aus diesem Grund ordnet man häufig auftretenden Störsignalen sehr kurze Codewörter zu, und, nur falls notwendig längere. Hierbei muß jedoch beachtet werden, daß kein auftretendes Codewort mit dem Anfang des eines anderen übereinstimmen darf - sonst könnte ein nachgeschalteter Decodierer die lükkenlos gesendeten Wörter nicht mehr eindeutig trennen. Die Güte dieses Codes spiegelt sich im Wert der relativen Redundanz (Überfluß, Abhängigkeit)

$$r = \frac{E-H}{E}$$

wieder (gute Codierung: $r \rightarrow 0$), wobei E der Informationsgehalt des Codes, d.h. die mittlere Anzahl an Übertragungsbits pro Sendesymbol: $\overline{m}$) und

$$H = \sum_i \left[p(Qi) * ld \frac{1}{p(Qi)} \right]$$

ld( ): logarithmus digitales, Basis 2

die Entropie der Quelle (Unordnung, wird maximal bei Gleichverteilung der Symbole) repräsentieren.

Die Auftrittswahrscheinlichkeiten der Quellsymbole seien in unserem Beispiel wie folgt vorgegeben:

p(A) = 0.41
p(B) = 0.16
p(C) = 0.15
p(D) = 0.13
p(E) = 0.11
p(F) = 0.03
p(G) = 0.01

In Abbildung 1 sind die sich hieraus ergebenden Kennwerte für eine "numerische" Codierung aufgelistet.

Bei der in Abb. 2 wiedergegebenen Codierung wurde das Shannon-Fano-Verfahren angewandt. Aufgrund der Tatsache, daß das dominante Symbol A nur zwei Bits zur Übermittlung benötigt, sinkt die mittlere Codewortlänge $\overline{m}$ deutlich und damit auch die relative Redundanz r.

Durch die Anwendung des Huffman-Verfahrens gewinnt man den bestmöglichen Code, d. h. den Code mit der geringsten Redundanz (Abb 3.).

Im folgenden sollen die zwei gängigsten Verfahren zur Quellencodierung explizit erklärt werden.

2. Das Shannon-Fano-Verfahren

1) Ordnen der Quellensymbole nach fallender Wahrscheinlichkeit (Abb. 4.0).
2) Unterteilung der entstandenen Symbolreihe in zwei zusammenhängende Untergruppen dergestalt, daß die Summen der Symbolwahrscheinlichkeiten beider Teilgruppen in etwa übereinstimmen[1] : Trennstrich zwischen AB und CDEFG; p(AB)=0,57, p(CDEFG)=0,43.
3) Alle Symbole, die in der 1. Teilgruppe zugehören, erhalten den (Teil-) Code "0", jene, die sich in der 2. Untergruppe befinden, das Symbol "1" zugeordnet.
4) Fortsetzung der Punkte 2) und 3) mit allen so entstandenen Untergruppen - bis am Ende jedes "Astes" des "Teilungsbaums" nur noch zwei Teilgruppen übrig sind (Abb. 4.2 bis 4.4).

		4.0	4.1	4.2	4.3	4.4
0.41	A	0	0			
0.16	B	0	1			
0.15	C	1	0	0		
0.13	D	1	0	1		
0.11	E	1	1	0		
0.03	F	1	1	1	0	
0.01	G	1	1	1	1	

Abb. 4 (Shannon-Fano-Verfahren)

3. Das Huffman-Verfahren

1) Ordnen de Quellensymbole nach fallender Wahrscheinlichkeit und Zuordnung der Bezeichnungen a1, a2,..., an (Abb. 5).
2) Die beiden Symbole in der Reihe werden unter einem Namen (a67 in Abb. 5.1) zusammengefaßt, die Einzelwahrscheinlichkeiten addiert und das Ganze in einem neuem Schaubild entsprechend fallender Wahrscheinlichkeiten einsortiert (Abb. 5.1).

A: 000
B: 001
C: 010
D: 011
E: 100
F: 101
G: 110

$\overline{m}$ = 3.00
H = 2.31208
r = 22.93 %

Abb. 1 ("numerierende" Codierung)

A: 00
B: 01
C: 100
D: 101
E: 110
F: 1110
G: 1111

$\overline{m}$ = 2.47
H = 2.31208 (const.)
r = 6.39 %

Abb. 2 (Codierung nach Shannon-Fano)

A: 0
B: 111
C: 110
D: 100
E: 1011
F: 10101
G: 10100

$\overline{m}$ = 2.37
H = 2.31208 (const.)
r = 2.44 %

Abb. 3 (Codierung nach Huffman)

```
      EINGABE:      ANZEIGE:       KOMMENTAR:

      XEQ"CODE"     "P1="          Programmstart

      0.41 R/S      "P2="          Eingabe von p1
      0.16 R/S      "P3="          Eingabe von p2
      0.15 R/S      "P4="          Eingabe von p3
      0.13 R/S      "P5="          Eingabe von p4
      0.11 R/S      "P6="          Eingabe von p5
      0.03 R/S      "P7="          Eingabe von p6
      0.01 R/S      "P8="          Eingabe von p7

      0 R/S         "SH HUF"       Abbruch der Eingaberoutine
```

Ablauf 1

```
SF 00           ...               Teilung mitverfolgen -> SF 00
XEQ"A"          ...               Start des  Shannon-Fano-Verfahrens
                                  durch drücken der Taste "A".
                "T: 1-2 3-7"      1.Teilung: AB-CDEFG
                "T: 1-1 2-2"      2.Teilung: A-B     (Untergruppe 1.Art)
                "T: 3-4 5-7"      3.Teilung: CD-EFG  (Untergruppe 1.Art)
                "T: 3-3 4-4"      4.Teilung: C-D     (Untergruppe 2.Art)
                "T: 5-5 6-7"      5.Teilung: E-FG    (Untergruppe 2.Art)
                "T: 6-6 7-7"      6.Teilung: F-G     (Untergruppe 3.Art)

                "C1=00"           Code von A: 00
R/S             "C2=01"           Code von B: 01
R/S             "C3=100"          Code von C: 100
R/S             "C4=101"          Code von D: 101
R/S             "C5=110"          Code von E: 110
R/S             "C6=1110"         Code von F: 1110
R/S             "C7=1111"         Code von G: 1111

R/S             "M/=2,4700"       m̄ = 2,47     (mittlere Codewortlänge)
R/S             "H=2,311208"      H = 2.31208  (Quellenentropie)
R/S             "RR=6,39 %"       r = 6,39 %   (relative Redundanz)
```

Ablauf 2

```
SF 00                             ... wir wollen wieder mitverfolgen.
XEQ"B"          ...               Start des Huffman-Verfahrens durch
                                  drücken der Taste "B"
                "1.SCHRITT"       1.Teilungsschritt
                "1: a1: 0,41"     (siehe Abb.5.1)
                "2: a2: 0,16"     (      -"-      )
                "3: a3: 0,15"     (      -"-      )
                "4: a4: 0,13"     (      -"-      )
                "5: a5: 0,11"     (      -"-      )
                "6: a67: 0,04"    (      -"-      )

                "2.SCHRITT"       2.Teilungsschritt
                "1: a1: 0,41"     (siehe Abb.5.2)
                "2: a2: 0,16"     (      -"-      )
                "3: a3: 0,15"     (      -"-      )
                "4: a567: 0,15"   (      -"-      )
                "5: a4: 0,13"     (      -"-      )

                "3.SCHRITT"       3.Teilungsschritt
                ...               (siehe Abb.5.3)

                ...               ...

                "5.SCHRITT"       5.Teilungsschritt

                "1: a234567: 0,59"
                "2: a1: 0,41"     (siehe Abb.5.5)


R/S             "C1=0"            Code von A: 0
R/S             "C2=111"          Code von B: 111
R/S             "C3=110"          Code von C: 110
R/S             "C4=100"          Code von D: 100
R/S             "C5=1011"         Code von E: 1011
R/S             "C6=10101"        Code von F: 10101
R/S             "C7=10100"        Code von G: 10100

R/S             "M/=2,3700"       m̄ = 2,37     (mittlere Codewortlänge)
R/S             "H=2,31208"       H = 2,31208  (Quellenentropie)
R/S             "RR=2,44 %"       r = 2,44 %   (relative Redundanz)
```

Ablauf 3

3) Schritt 2) wird solange wiederholt, bis in der (dann) letzten Tabelle (Abb. 5.5) nur noch zwei verschiedene Symbole übrigbleiben.

   Hierauf folgt die eigentliche Codierung, beginnend mit der letzten Tabelle:

4) Den beiden untersten Symbolen werden die (Teil-) Codes "1" und "0" zugeordnet (in eckigen Klammern in Abb. 5.50).

5) Wiederholung von Schritt 4) bis alle Tabellen abgearbeitet sind. Neue (Teil-) Codes werden einfach rechts an die bereits vorhandenen angehängt (fett gedruckt).

4. Programmablauf auf dem HP-41

Es ist zu beachten, daß die Symbolwahrscheinlichkeiten in absteigender Reihenfolge zu geschehen hat: Eine zusätzliche Sortierroutine in dieses eh schon riesige Programm einzubauen hielt ich - auch aus Geschwindigkeitserwägungen heraus - nicht für sinnvoll (siehe Ablauf 1).

Mit dem Eintippen von "0" wird die Eingaberoutine verlassen und nach der Art der gewünschten Codierung gefragt - in bester Menümanier: XEQ A startet die Shannon-Fano-Codierung, XEQ B das Huffman-Verfahren. Möchte man hingegen aus den eingegebenen Wahrscheinlichkeiten lediglich die Quellentropie errechnen lassen, so ist die Taste "H" (XEQ"H") zu drücken, um die entsprechende Routine zu starten.

Setzt man vor dem Start einer Codierung das Flag 00, so kann man die einzelnen Gruppenteilungen im Display verfolgen (siehe Ablauf 2).

Mit einem weiteren R/S kann die gesamte Ausgabe wiederholt werden. Nach dieser Demonstration der Shannon-Fano-Routine nun zum Huffman-Verfahren, welches, wie auch das vorherige Programm, zu jedem beliebigen Zeitpunkt gestartet werden kann- es reicht, wenn die Wahrscheinlichkeiten einmal eingegeben worden sind (siehe Ablauf 3).

---

[1] "In etwa übereinstimmen" ist freilich ein etwas schwammiger Begriff, aber *so* in der Definition des Verfahrens vorhanden. Er besagt jedoch nichts anderes, als daß die Differenz der Symbolwahrscheinlichkeits-Summen beider Teilgruppen einen minimalen Wert annehmen sollte. Hier ist dieses Verfahren also nicht eindeutig: Es sind durchaus Wahrscheinlichkeitsverteilungen denkbar, bei denen man sich ebenso legitim für die eine wie für die andere Trennung der Gruppen entscheiden könnte.

a1	0.41	[0]	a1	0.41	[0]	a1	0.41	[0]
a2	0.16	[111]	a2	0.16	[111]	a2	0.16	[111]
a3	0.15	[110]	a3	0.15	[110]	a3	0.15	[110]
a4	0.13	[100]	a4	0.13	[100]	→ a567	0.15	[101]
a5	0.11	[1011]	a5	0.11	[1011]	a4	0.13	[100]
a6	0.03	[10101]	→ a67	0.04	[1010]			
a7	0.01	[10100]						
	5.0			*5.1*			*5.2*	
a1	0.41	[0]	a1	0.41	[0]	→ a2-7	0.59	[1]
→ a4567	0.28	[10]	a23	0.31	[11]	a1	0.41	[0]
a2	0.16	[111]	→ a4567	0.28	[10]			
a3	0.15	[110]						
	5.3			*5.4*			*5.5*	

Abb. 5 (Huffman-Verfahren)

```
001*LBL "CODE"
002 20
003 STO 05
004 1
005 STO 10
006 STO 00
007 STO 12
008 -
009 STO 13
010 10
011 +
012 XEQ"SIZ"
013 FC?C 25
014 PROMPT
015 1 E3
016 STO 11
017 CF 29
018*LBL 00
019 "P"
020 FIX 0
021 ARCL 00
022 "}="
023 PROMPT
024 RCL 10
025 X<Y?
026 GTO 27
027 X<>Y
028 STO 10
029 RCL 13
030 RCL 00
031 +
032 X<>Y
033 STO IND Y
034 RCL 12
035 ST+ 00
036 X<>Y
037 X!=0?
038 GTO 00
039*LBL 01
040 RCL 12
041 ST+ X
042 ST- 00
043 RCL 00
044 RCL 13
045 +
046 RCL 11
047 /
048 RCL 05
049 +
050 STO 01
051 SF 27
052 "SH HUF"
053 PROMPT
054*LBL 14
055 RCL 00
056 RCL X
057 RCL 11
058 /
059 +
060 RCL 01
061 +
062 STO 06
063 RCL 12
064*LBL 02
065 STO IND Y
066 ISG Y
067 GTO 02
068 RTN
069*LBL B
070 RCL 00
071 4
072 *
073 RCL 05
074 +
075 RCL 12
076 -
077 XEQ"SIZ"
078 FC?C 25
079 PROMPT
080 3
081 STO 19
082 2
083 STO 18
084 XEQ 14
085 X<>Y
086 INT
087 RCL X
088 RCL 00
089 +
090 RCL 12
091 -
092 RCL 11
093 /
094 +
095 STO 03
096 STO 02
097 RCL 00
098 RCL X
099 RCL 11
100 /
101 +
102 +
103 STO 04
104 RCL 12
105*LBL 15
106 STO IND 02
107 ST+ X
108 ISG 02
109 GTO 15
110 RCL 01
111 STO 02
112 RCL 00
113 RCL 19
114 *
115 STO 07
116*LBL 16
117 RCL 02
118 RCL 07
119 +
120 RCL IND 02
121 STO IND Y
122 ISG 02
123 GTO 16
124 RCL 00
125 STO 13
126*LBL 20
127 RCL 13
128 RCL 04
129 INT
130 +
131 RCL 12
132 -
133 RCL X
134 DSE X
135 RCL IND X
136 RCL IND Z
137 +
138 STO 14
139 RDN
140 RCL 00
141 ST- Y
142 ST- Z
143 RDN
144 RCL IND X
145 STO 16
146 RCL IND Z
147 STO 17
148 +
149 STO 15
150 RCL 04
151 RCL 00
152 RCL 13
153 -
154 RCL 18
155 +
156 RCL 11
157 /
158 -
159 STO 02
160*LBL 17
```

```
161 RCL 14
162 RCL IND 02
163 X<Y?
164 GTO 18
165 ISG 02
166 GTO 17
167*LBL 18
168 RCL 14
169 RCL 15
170*LBL 19
171 RCL 02
172 RCL 00
173 -
174 X<>Y
175 X<> IND Y
176 X<> Z
177 X<> IND 02
178 RCL Z
179 ISG 02
180 GTO 19
181 RCL 02
182 RCL 00
183 -
184 X<>Y
185 STO IND Y
186 X<> Z
187 STO IND 02
188 RCL 12
189 ST- 13
190 FS? 00
191 XEQ 29
192 SF 05
193 RCL 16
194 XEQ 21
195 RCL 17
196 CF 05
197 XEQ 21
198 RCL 13
199 RCL 12
200 X!=Y?
201 GTO 20
202 SF 06
203 GTO 10
204*LBL 21
205 RCL 06
206 STO 10
207 X<>Y
208 RCL 18
209*LBL 22
210 RCL Y
211 X<>Y
212 MOD
213 ST- Y
214 X<>Y
215 LASTX
216 /
217 LASTX
218 RCL Z
219 X=0?
220 GTO 23
221 X<>Y
222 ST* IND 10
223 FS? 05
224 ISG IND 10
225 STO X
226 X<>Y
227*LBL 23
228 X<> Z
229 X<>Y
230 ISG 10
231 GTO 22
232 RTN
233*LBL 29
234 RCL 13
235 RCL 12
236 X=Y?
237 RTN
238 RCL 00
239 RCL 11
240 /
241 RCL 12
242 +
243 STO 14
244 2
245 STO 08
246 RCL 00
247 RCL 13
248 -
249 FIX 0
250 CLA
251 ARCL X
252 "}. SCHRITT"
253 AVIEW
254 RCL 03
255 STO 09
256 RCL 13
257 RCL 11
258 /
259 RCL 12
260 +
261 STO 02
262*LBL 30
263 CLA
264 ARCL 02
265 "}: a"
266 RCL 14
267 STO 07
268 RCL IND 09
269 RCL 08
270*LBL 31
271 RCL Y
272 X<>Y
273 MOD
274 ST- Y
275 X<>Y
276 LASTX
277 /
278 X<>Y
279 X!=0?
280 ARCL 07
281 X<>Y
282 X=0?
283 GTO 32
284 LASTX
285 ISG 07
286 GTO 31
287*LBL 32
288 "}: "
289 FIX 2
290 RCL 09
291 RCL 00
292 +
293 RCL IND X
294 RND
295 X=0?
296 FIX 5
297 ARCL IND Y
298 AVIEW
299 ISG 09
300 STO X
301 FIX 0
302 ISG 02
303 GTO 30
304 RTN
305*LBL A
306 RCL 05
307 RCL 00
308 2,5
309 *
310 +
311 INT
312 XEQ"SIZ"
313 FC?C 25
314 PROMPT
315 CF 06
316 RCL 05
317 RCL 12
318 -
319 STO 13
320 XEQ 14
321 X<>Y
322 STO 17
323 CLX
324 STO 16
325 RCL 00
326 RCL 11
327 /
328 RCL 12
329 +
330 XEQ"PUSH"
331*LBL 03
332 XEQ"POP"
333 X=0?
334 GTO 10
335 INT
336 LASTX
337 FRC
338 RCL 11
339 *
340 STO 04
341 X<>Y
342 STO 03
343 X<>Y
344 RCL 13
345 +
346 RCL 11
347 /
348 +
349 RCL 13
350 +
351 STO 02
352 STO 15
353 CLX
354 STO 07
355 STO 08
356*LBL 06
357 RCL IND 02
358 ST+ 08
359 ISG 02
360 GTO 06
361 RCL 15
362 STO 02
363 RCL 12
364 STO 14
365*LBL 07
366 RCL IND 02
367 STO 09
368 ST- 08
369 ST+ 07
370 RCL 08
371 RCL 07
372 -
373 ABS
374 RCL 14
375 X<Y?
376 GTO 08
377 X<>Y
378 STO 14
379 ISG 02
380 GTO 07
381*LBL 08
382 RCL 02
383 INT
384 RCL 05
385 -
386 RCL X
387 RCL 11
388 /
389 RCL 03
390 +
391 X<>Y
392 RCL 04
393 RCL 11
394 /
395 RCL 12
396 +
397 +
398 X<>Y
399 STO 03
400 X<>Y
401 STO 04
402 FC? 00
403 GTO 28
404 X<>Y
405 INT
406 LASTX
407 FRC
408 RCL 11
```

```
409 *
410 "T: "
411 FIX 0
412 ARCL Y
413 "}-"
414 ARCL X
415 RCL 04
416 INT
417 LASTX
418 FRC
419 RCL 11
420 *
421 "}"
422 ARCL Y
423 "}-"
424 ARCL X
425 AVIEW
426*LBL 28
427 RCL 04
428 RCL 06
429 INT
430 RCL 12
431 -
432 RCL X
433 RCL 11
434 /
435 +
436 +
437 STO 02
438 2
439 STO 09
440*LBL 04
441 RCL IND 02
442 RCL 09
443 *
444 RCL 12
445 +
446 STO IND 02
447 ISG 02
448 GTO 04
449 RCL 04
450 ISG 04
451 XEQ"PUSH"
452 RCL 03
453 RCL 06
454 INT
455 RCL 12
456 -
457 RCL X
458 RCL 11
459 /
460 +
461 +
462 STO 02
463*LBL 05
464 RCL IND 02
465 ST+ IND 02
466 ISG 02
467 GTO 05
468 RCL 03
469 ISG 03
470 XEQ"PUSH"
471 GTO 03
472*LBL"PUSH"
473 RCL 12
474 ST+ 16
475 CLX
476 RCL 16
477 RCL 17
478 +
479 X<>Y
480 STO IND Y
481 RTN
482*LBL"POP"
483 RCL 16
484 X=0?
485 RTN
486 RCL 16
487 RCL 17
488 +
489 RCL IND X
490 RCL 12
491 ST- 16
492 X<>Y
493 RTN
494*LBL 10
495 "0"
496 ASTO 18
497 "1"
498 ASTO 19
499 RCL 06
500 STO 02
501 2
502 STO 08
503 LN
504 STO 09
505*LBL 09
506 "C"
507 RCL 02
508 RCL 06
509 INT
510 -
511 RCL 12
512 +
513 FIX 0
514 ARCL X
515 "}="
516 FS? 06
517 GTO 24
518 RCL IND 02
519 STO 03
520 RCL 08
521 RCL Y
522 LN
523 RCL 09
524 /
525 INT
526 RCL 12
527 +
528 Y^X
529 X>Y?
530 DSE L
531 LASTX
532 RCL 08
533 X<>Y
534 Y^X
535 STO 04
536 ST- 03
537*LBL 11
538 RCL 08
539 ST/ 04
540 RCL 03
541 RCL 04
542 X<=Y?
543 ST- 03
544 X<=Y?
545 ARCL 19
546 X>Y?
547 ARCL 18
548 RCL 12
549 RCL 04
550 X>Y?
551 GTO 11
552 GTO 25
553*LBL 24
554 RCL 08
555 RCL IND 02
556*LBL 26
557 RCL 12
558 X=Y?
559 GTO 25
560 RDN
561 X<>Y
562 RCL Y
563 X<>Y
564 MOD
565 X=0?
566 ARCL 18
567 X!=0?
568 ARCL 19
569 ST- Y
570 X<>Y
571 LASTX
572 /
573 LASTX
574 X<>Y
575 GTO 26
576*LBL 25
577 PROMPT
578 ISG 02
579 GTO 09
580 RCL 06
581 STO 02
582 CLX
583 STO 18
584*LBL 12
585 RCL IND 02
586 RCL 08
587 RCL Y
588 LN
589 RCL 09
590 /
591 INT
592 RCL 12
593 +
594 Y^X
595 X>Y?
596 DSE L
597 LASTX
598 RCL 02
599 RCL 00
600 -
601 X<>Y
602 RCL IND Y
603 *
604 ST+ 18
605 ISG 02
606 GTO 12
607 "M/="
608 FIX 4
609 ARCL 18
610 PROMPT
611*LBL H
612 RCL 01
613 STO 02
614 CLX
615 STO 19
616*LBL 13
617 RCL IND 02
618 1/X
619 LN
620 RCL 09
621 /
622 RCL IND 02
623 *
624 ST+ 19
625 ISG 02
626 GTO 13
627 "H="
628 FIX 5
629 ARCL 19
630 PROMPT
631 RCL 18
632 RCL 19
633 -
634 RCL 18
635 /
636 "RR="
637 100
638 *
639 FIX 2
640 ARCL X
641 "} %"
642 PROMPT
643 GTO 10
644*LBL"SIZ"
645 SF 25
646 RCL IND X
647 FS? 25
648 X<>Y
649 RCL 12
650 +
651 "SIZE >= "
652 FIX 0
653 ARCL X
654 RTN
```

Fortsetzung Seite 36

Steuern 1990

von August Schorr

693 Zeilen, 1562 Bytes, 224 Regs., SIZE 023, HP41CV, X-F, PRINTER

Das Programm ermittelt auf der Grundlage des ab 1.1.1990 geänderten Steuerrechtes

- das *zu versteuernde Einkommen* (ZVE),
- die *Einkommensteuer* (EST) und *Kirchensteuer* (KIST) für das festgestellte ZVE,
- über eine gesonderte Abfrage die Jahres- oder Monats-*Lohnsteuer* (LST) und die *Lohn-KIST* für die Steuerklassen 1 bis 4.

Die besonderen Bestimmungen für rentenversicherungsfreie Beschäftigte (z.B. Beamte) sind bei der EST (für Ehepaare auch im "MIX") und bei der LST ("*Beamtentabelle*") berücksichtigt.

EINKOMMENSTEUER

Das Programm folgt, wie schon das in PRISMA 4/88 für die Vorjahre, weitgehend dem "Schema zur Selbstberechnung" auf Seite 15/16 der "Anleitung zur Einkommensteuererklärung" für 1989 und ist für alle Steuerfälle geeignet, die über das SCHEMA gerechnet werden können.

Es berechnet nach Eingabe der dort geforderten Grunddaten die aufgrund von Einkunftsarten, Familienstand, Lebensalter, Kinderzahl, Rentenversicherungsstatus usw. anzurechnenden Frei-, Pauschal- und Entlastungsbeträge bis hin zum tatsächlich zu versteuernden Einkommem (ZVE) und druckt anschließend die EST und KIST aus.

In den folgenden Programmerläuterungen wird zur Erklärung der Begriffe auf Zeilennummern der bisherigen Einkommensteuererklärung und ihrer Anlagen verwiesen. Diese Zeilennummern mögen in den ja erst Ende 1990 vorliegenden Formularen etwas geändert sein. Da aber das Steuer*system* grundsätzlich unverändert blieb, haben sich auch die Begriffsinhalte nicht geändert und die "Anleitung zur Einkommensteuererklärung" kann weiterhin hilfreich sein.

Beispiele für verwendete Abkürzungen aus der EST-Erklärung für 1989:

ERKL 71 - 74 = EST-Erklärung Zeile 71 bis 74

SCH 7 = Schema zur Selbstberechnung Zeile 7

N 3 = Anlage N zur EST-Erklärung Zeile 3

ANL 12 RO = Anleitung zur EST-Erklärung Seite 12 rechts oben usw.

Grundsätzliches zum Ablauf

Die Eingabeaufforderungen ("Abfragen") erscheinen zunächst in der Anzeige und werden dann zusammen mit der Eingabe ausgedruckt. Wenn in einer Zeile keine *Kennziffer* (KZ) oder kein *Wert* einzugeben ist, muß eine *Null* eingegeben werden.

Alle *Abzugsbeträge* (Werbungskosten, Sonderausgaben usw.) sind *positiv* - ohne Minuszeichen - einzugeben.

Immer die *tatsächlichen* Aufwendungen eingeben, mögliche höhere Pauschbeträge oder auch Kürzungen (z.B. bei den Bausparbeiträgen) berücksichtigt das Programm.

Der folgende Musterausdruck ist manipuliert, um alle im Programm möglichen Druckzeilen erklären zu können. Die abgefragten KZ (=Flags) steuern die weiteren Abfragen und unterdrücken die jeweils nicht benötigten. So hat der kürzeste Ablauf 6 Zeilen weniger als der Musterausdruck.

Start: XEQ "EST90", nach den Eingaben weiter mit R/S. Neustart nach vollständigem Durchlauf mit R/S, nach Abbruch mit RTN, R/S.
Vor-Abfrage ohne Ausdruck: EST=1/ LST=0; Wahl zwischen EST- und LST-Programm. Für den Musterausdruck war die "1" einzugeben.

1 SPLITT.=3/N=0;
zusammen veranlagte Ehepaare = 3, Alleinstehende = 0.
Bei "3" werden nun zuerst die Daten des Ehemannes abgefragt;

2 RV-FREI=2
(das N=0 schenke ich mir künftig, es trifft für alle KZ-Abfragen zu).
2 = der in ANL 12 RO bzw. N 31 - 36 aufgeführte Personenkreis: Aktive/ pensionierte Beamte, weiterbeschäftigte Altersrentner mit Werkpension, Vorstandsmitglieder von AG's u.A.;

3 LOHN?
Eingabe laut N 3 und/oder KSO 55 (Abgeordnete).
Bei Lohn=0 entfallen die folgenden Abfragen 4 und 5;

4 DARIN VERS.BEZ?
Die in Ziffer 3 enthaltenen Pensionen laut N 22 und oder KSO 56;

5 WERB.K.?
Summe der anrechenbaren Werbungskosten aus N 37 - 63;

6 RENTE?
Bruttobetrag der Rente(n) laut KSO 40.
Bei Rente=0 unterbleibt die folgende Abfrage;

7 STPFL.%? = Ertragsanteil der Rente (KSO 41) in % eingeben.
Bei mehreren Renten mit unterschiedlichen Ertragsanteilsätzen: Vorher *Summe* der Ertragsanteile aller Renten *in DM* errechnen und ihren Prozentsatz vom Gesamtrentenbrutto ermitteln. Dieser Prozentsatz kann mit beliebigen Dezimalstellen eingegeben werden. Ausgedruckt wird (wegen FIX 0) ein gerundeter Satz, gerechnet wird mit den Dezimalen.

8 EINK.KAP.V.?
Zinsen, Dividenden usw. laut KSO 4 - 18 Spalten 30 bzw. 31 nach Abzug der DM 100 *übersteigenden* Werbungskosten in Zeile 22. Die anrechenbaren Körperschafts- und Kapitalertragsteuern (Spalte 34 + 35) müssen nach Durchlauf *manuell* mit der festgestellten EST verrechnet werden.

9 ANDERE EINKÜNFTE?
Dazu zählen Gewinne (oder Verluste) aus:
- Land- und Forstwirtschaft laut Anlage L
- selbständiger Arbeit (GSE)
- Gewerbebetrieb (GSE)
- Vermietung und Verpachtung (V und FW)
- weiteren Einkünften (KSO 44 - 53).

Ergibt die *Summe* dieser Einkünfte einen *Verlust*, ist dieser *mit CHS als Minusbetrag* einzugeben (der Freibetrag für selbständige Arbeit entfällt ab 1990, der für Landwirte = siehe Ziffer 14).

10 HAT EHEFRAU EINK.?
Bei JA=4 werden die Abfragen 2 - 9 wiederholt, *bei RV-FREI?* jedoch mit der *KZ 7*.

11 Ausdruck: BRUTTO-EINKOMMEN: Summe der Beträge aus den Ziffern 3, 6, 8 und 9.

12 EHEM>64=1

13 FRAU>64=5;
JA= wenn der betreffende Ehegatte im Steuerjahr älter als 64 Jahre war/wurde. Letztere Abfrage wird ausgelassen, wenn die Ehefrau kein Einkommen hatte.

14 FREIBETRAG LANDWIRTE=
Hier ist zutreffendenfalls der Betrag von maximal DM 2000, bei Splitting bis 4000, einzugeben.

15 Ausdruck: GESAMTBETRAG DER EINKÜNFTE= Nach Berücksichtigung der an die Einkunftsarten gebundenen Abzugsbeträge (SCH 22). Der Betrag wird auch im EST-Bescheid unter derselben Bezeichnung aufgeführt.

16 SONDERAUSGABEN?
Laut ERKL 74 - 83

```
EINK.ST.90

SPLITT.=3/N=0              3

EINK.EHEM.
RV-FREI=2/N=0              2

LOHN?                  34000
DARIN VERS.BEZ?        20000
WERB.K.?                2400
RENTE?                 24000
STPFL.%?                  25
EINK.KAP.V.?             600
AND.EINK.?+-            -2300

HAT EHEFR.EINK?J=4/N=0 4

EINK.EHEFR.
RV-FREI=7/N=0              0

LOHN?                  18000
DARIN VERS.BEZ?            0
WERB.K.?                3600
RENTE?                 14000
STPFL.%?                  28
EINK.KAP.V.?            4000
AND.EINK.?+-            3500

BRUTTO-EINK.=          95800

EHEM>64=1/N=0              1
FRAU>64=5/N=0              0

FRBTR.LANDW.?              0

GES.EINKUENFTE=        51400

SOND.AUSG.?             1200
VORS.AUFW.?            14000
BAUSPK.?                6000

ANGER.VORS.A.=         11180

AGW.BEL.?                900
KINDERZAHL?              0,5
STKL.2=2/N=0               0
SO.ABZ.?                   0

STPFL.EINK.=           36600

EST=                 5114,00
8,0% KIST=            397,12
```

```
LST
MON=12/JAHR=0             12
STKL.3 OD. 4/N=0           0
RV-FREI=2/N=0              2
LOHN?                6934,65
KINDERZAHL?              0,5
STKL.2=2/N=0               0
LST=                 1749,41
8,0% KIST=            138,95
```

17 VORSORGEAUFWENDUNGEN=
Sonderausgaben laut ERKL 62 - 63 / 67 - 71 (die früher erforderliche Eingabe des AG-Anteils zur RV entfällt).

18 Beiträge an BAUSPARKASSEN=
Die tatsächlichen Beiträge sind einzugeben, die 50 prozentige Kürzung erfolgt im Programm.

19 Ausdruck: ANGERECHNETE VORSORGEAUFWENDUNGEN = aus eigener Beitragsleistung oder, wenn höher, die Vorsorgepauschale.

20 AUSSERGEW.BELASTUNGEN=
laut ERKL 86 - 117 (kompliziert, vorher klären) ANL 7 RO bis 9.

21 KINDERZAHL=
Laut LST-Karte bzw. ANL 3 LU;

22 STKL.2=2;
steuert den Haushaltsfreibetrag für die Steuerklasse 2.
Die Abfrage unterbleibt bei Ziffer 1 SPLITT=3 oder bei SPLITT=0, wenn die Kinderzahl=0 ist.

23 SONSTIGE ABZÜGE=
Verluste aus Vorjahren laut ERKL 84 - 85 und andere Abzüge, die direkt vom ZVE gekürzt werden dürfen.

24 Ausdruck: STEUERPFL.EINKOMMEN = ZVE;

25 Ausdruck: EST = Einkommensteuer;

26 Ausdruck: 8% KIST = Kirchensteuer.
Der Prozentsatz kann in Zeile 675 korrigiert werden.
Hinweis: Aufgrund unterschiedlicher Rundungsvorschriften der Bundesländer kann sich eine Differenz bis zu 99 Pfg. ergeben.

Anmerkung:
Man kann auch, bei bekanntem ZVE direkt in die EST-Formel springen.
Beispiel: Wie hoch ist die Grenzbelastung der letzten 100 DM mit EST ?
Man gibt das um 108 DM (wegen der 54 DM Rundung) verminderte ZVE in das X-Register, setzt bei Splitting das Flag 03 und geht mit XEQ B in das Programm.
Die Differenz der beiden EST-Ausdrucke, dividiert durch 1,08, ist die prozentuale Belastung der letzten 100 DM.
Reizvoll ist es nun, die Zahlen einiger Vorjahre über dieses Programm zu rechnen (aber bitte nicht gleich den Gewinn bei einem fürstlichen McDonald's-Mahl verprassen!).

LOHNSTEUER

Start und Wiederholung wie bei der EST. Bei der Vorabfrage EST=1/LST=0 ist die "0" richtig.

1 MON=12/JAHR=0;
im Beispiel ist die 12 eingegeben.

2 STKL.3 OD. 4/N=0;
bei den Steuerklassen 3 oder 4 ist die entsprechende Ziffer einzugeben, bei 1 und 2 = 0.

3 RV-FREI=2/N=0;
die "2" ruft die "Beamtentabelle" auf (siehe bei EST Ziffer 2).

4 LOHN=
Jahres- oder Monatsbruttolohn;

5 KINDERZAHL=
laut LST-Karte;

6 STKL.2=2/N=0;
für Haushaltsfreibetrag der Steuerklasse 2. Die Abfrage entfällt, wenn in Ziffer 2 die "3" oder "4" zutrifft; ferner, wenn dort zwar die "0" gilt, die Kinderzahl aber = 0 ist.

7 Ausdruck der Lohnsteuer;

8 Ausdruck der Kirchensteuer.
Rundungsvorbehalt und Prozentsatzkorrektur wie in Ziffer 26 der EST.

```
01♦LBL "EST90"
02 ADV
03 CLX
04 CLRG
05 X<>F
06 FIX 0
07 CF 29
08 CF 10
09 "EST=1/LST=0"
10 PROMPT
11 X=0?
12 GTO D
13 "EINK.ST.90"
14 PRA
15 ADV
16 "SPLITT.=3/N=0"
17 PROMPT
18 XEQ 15
19 SF IND X
20 ADV
21 "EINK.EHEM."
22 FS? 03
23 PRA
24 "RV-FREI=2/N=0"
25 PROMPT
26 XEQ 15
27 SF IND X
28♦LBL A
29 ADV
30 "LOHN?"
31 PROMPT
32 XEQ 15
33 X=0?
34 GTO 07
35 STO 15
36 STO 17
37 STO 22
38 STO 21
39 "DARIN VERS.BEZ?"
40 PROMPT
41 XEQ 15
42 ST- 15
43 STO 16
44 40
45 %
46 INT
47 4000
48 X>Y?
49 X<>Y
50 ST- 16
51 ST- 17
52 ST- 21
53 RCL 17
54 STO 18
55 "WERB.K.?"
56 PROMPT
57 XEQ 15
58 2000
59 X<=Y?
60 X<>Y
61 ST- 18
62 0
63 RCL 18
64 X<Y?
65 X<>Y
66 STO 18
67♦LBL 07
68 "RENTE?"
69 PROMPT
70 XEQ 15
71 X=0?
72 GTO 08
73 STO 19
74 ST+ 22
75 "STPFL.%?"
76 PROMPT
77 XEQ 15
78 RCL 19
79 X<>Y
80 %
81 INT
82 STO 19
83 200
84 X>Y?
85 X<>Y
86 ST- 19
87♦LBL 08
88 "EINK.KAP.V.?"
```

```
 89 PROMPT
 90 XEQ 15
 91 ENTER↑
 92 ENTER↑
 93 .003003
 94 FS? 09
 95 REGSWAP
 96 RDN
 97 STO 00
 98 ST+ 20
 99 ST+ 22
100 700
101 X>Y?
102 X<>Y
103 STO 01
104 .003003
105 FS? 09
106 REGSWAP
107 "AND.EINK.?+-"
108 PROMPT
109 XEQ 15
110 ST+ 20
111 ST+ 22
112 FC? 03
113 GTO C
114 FS? 04
115 GTO C
116 ADV
117 "HAT EHEFR.EINK?"
118 "J=4/N=0"
119 PROMPT
120 XEQ 15
121 X>0?
122 GTO 19
123♦LBL C
124 FS? 04
125 XEQ 18
126 FC? 03
127 GTO 20
128 RCL 00
129 RCL 03
130 X>Y?
131 SF 08
132 +
133 1400
134 X>Y?
135 X<>Y
136 RCL 01
137 RCL 04
138 +
139 -
140 FS? 08
141 ST- 11
142 FC? 08
143 ST- 20
144 RCL 04
145 ST- 11
146 CF 08
147 CF 09
148♦LBL 20
149 RCL 01
150 ST- 20
151 RCL 13
152 RCL 22
153 +
154 ADV
155 "BRUTTO-EINK.="
156 XEQ 15
157 ADV
158 FS? 03
159 "EHEM>64=1/N=0"
160 FC? 03
161 "STPF>64=1/N=0"
162 PROMPT
163 XEQ 15
164 SF IND X
165 X=0?
166 GTO 01
167♦LBL 00
168 RCL 15
169 RCL 20
170 X<0?
171 CLX
172 +
173 40
174 %
175 RND
176 3720
177 X<Y?
178 X<>Y
179 RDN
180 ST- 20
181 ST- 21
182 CLX
183 FS? 05
184 GTO 04
185♦LBL 01
186 FC? 03
187 GTO 04
188 FC? 04
189 GTO 04
190 "FRAU>64=5/N=0"
191 PROMPT
192 XEQ 15
193 SF IND X
194 X=0?
195 GTO 04
196 XEQ 18
197 GTO 00
198♦LBL 04
199 FS? 05
200 XEQ 18
201 RCL 18
202 RCL 09
203 +
204 RCL 19
205 RCL 10
206 +
207 +
208 RCL 20
209 RCL 11
210 +
211 +
212 STO 20
213 ADV
214 "FRBTR.LANDW.?"
215 PROMPT
216 XEQ 15
217 ST- 20
218 RCL 20
219 ADV
220 "GES.EINKUENFTE="
221 XEQ 15
222 ADV
223 "SOND.AUSG.?"
224 PROMPT
225 XEQ 15
226 108
227 FS? 03
228 ST+ X
229 X>Y?
230 X<>Y
231 RDN
232 ST- 20
233 "VORS.AUFW.?"
234 PROMPT
235 XEQ 15
236 STO 00
237 RCL 15
238 RCL 06
239 +
240 .12
241 *
242 INT
243 4000
244 FS? 03
245 ST+ X
246 X<>Y
247 -
248 X<0?
249 0
250 RCL 00
251 X>Y?
252 X<>Y
253 ST- 00
254 STO 01
255 "BAUSPK.?"
256 PROMPT
257 XEQ 15
258 2
259 /
260 RND
261 RCL 00
262 +
263 STO 00
264 2340
265 FS? 03
266 ST+ X
267 STO 02
268 X>Y?
269 X<>Y
270 ST- 00
271 ST+ 01
272 RCL 00
273 X<=0?
274 GTO 02
275 2
276 /
277 INT
278 2
279 ST/ 02
280 RDN
281 RCL 02
282 X>Y?
283 X<>Y
284 ST+ 01
285♦LBL 02
286 0
287 STO 00
288 STO 16
289 2000
290 STO 19
291 ST+ X
292 STO 10
293 2340
294 STO 02
295 FS? 03
296 GTO 03
297 FS? 02
298 GTO 09
299 RCL 21
300 ENTER↑
301 ENTER↑
302 GTO 05
303♦LBL 03
304 2
305 ST* 02
306 ST* 10
307 FC? 02
308 GTO 14
309 FS? 07
310 GTO 10
311♦LBL 14
312 FS? 02
313 GTO 12
314 FS? 07
315 GTO 11
316 RCL 21
317 RCL 12
318 +
319 ENTER↑
320 ENTER↑
321 GTO 05
322♦LBL 09
323 RCL 21
324 18
325 %
326 RND
327 RCL 19
328 X>Y?
329 X<>Y
330 STO 16
331 GTO 17
332♦LBL 10
333 RCL 21
334 RCL 12
335 +
336 18
337 %
338 RND
339 RCL 10
340 2
341 /
342 X>Y?
343 X<>Y
344 STO 16
345 GTO 17
346♦LBL 12
347 RCL 21
348 STO 04
349 RCL 12
350 STO 03
351 GTO 13
352♦LBL 11
353 RCL 21
354 STO 03
355 RCL 12
356 STO 04
357♦LBL 13
358 RCL 04
359 18
360 %
361 RND
362 ENTER↑
363 ENTER↑
364 RCL 10
365 2
366 /
367 X>Y?
368 X<>Y
369 XEQ 16
370 STO 00
371 RDN
372 RDN
373 RCL 19
374 X>Y?
375 X<>Y
376 .18
377 /
378 RND
379 RCL 03
380 +
381 RCL 03
382 RCL 04
383 +
384 X<>Y
385♦LBL 05
386 .18
387 *
388 RND
389 STO 07
390 X<>Y
391 12
392 %
393 INT
394 RCL 10
395 X<>Y
396 -
397 X<0?
398 0
```

```
399 RCL 07
400 X>Y?
401 X<>Y
402 ST- 07
403 STO 16
404 RCL 07
405 RCL 02
406 X>Y?
407 X<>Y
408 ST- 07
409 ST+ 16
410 RCL 07
411 X<=0?
412 GTO 17
413 2
414 /
415 INT
416 2
417 ST/ 02
418 RDN
419 RCL 02
420 X>Y?
421 X<>Y
422 RCL 16
423 +
424 STO 16
425*LBL 17
426 RCL 16
427 XEQ 16
428 RCL 01
429 RCL 00
430 X>Y?
431 X<>Y
432 RDN
433 X<=Y?
434 RDN
435 ST- 20
436 FS? 10
437 RTN
438 ADV
439 "ANGER.VORS.A.="
440 XEQ 15
441 ADV
442 CF 05
443 "AGW.BEL.?"
444 PROMPT
445 XEQ 15
446 ST- 20
447*LBL E
448 FIX 1
449 "KINDERZAHL?"
450 PROMPT
451 XEQ 15
452 STO 00
453 CF 01
454 CF 02
455 X≠0?
456 SF 01
457 FIX 0
458 1512
459 FC? 05
460 ST+ X
461 *
462 ST- 20
463 FC? 03
464 FC? 01
465 GTO 25
466 FS? 05
467 GTO 25
468 "STKL.2=2/N=0"
469 PROMPT
470 XEQ 15
471 SF IND X
472 0
473 5616
474 FC? 02
475 RDN
476 ST- 20
477 FS? 10
478 RTN
479*LBL 25
480 FS? 10
481 RTN
482 "SO.ABZ.?"
483 PROMPT
484 XEQ 15
485 ST- 20
486 CLX
487 RCL 20
488 ADV
489 "STPFL.EINK.="
490 XEQ 15
491 ADV
492 GTO B
493 RTN
494*LBL 19
495 6.015008
496 REGSWAP
497 SF 04
498 SF 09
499 ADV
500 "EINK.EHEFR."
501 PRA
502 "RV-FREI=7/N=0"
503 PROMPT
504 XEQ 15
505 SF IND X
506 GTO A
507*LBL 16
508 ENTER↑
509 ENTER↑
510 54
511 MOD
512 -
513 RTN
514*LBL 18
515 6.015008
516 REGSWAP
517 RTN
518*LBL 15
519 ALENG
520 ARCL Y
521 ALENG
522 ENTER↑
523 X<> Z
524 AROT
525 -
526 X<>Y
527 CHS
528 24
529 +
530 STO L
531*LBL 06
532 "├ "
533 DSE L
534 GTO 06
535 RDN
536 AROT
537 PRA
538 CLA
539 RDN
540 RTN
541*LBL D
542 ADV
543 SF 10
544 "LST"
545 PRA
546 "MON=12/JAHR=0"
547 PROMPT
548 XEQ 15
549 X≠0?
550 SF 06
551 "STKL.3 OD. 4/N="
552 "├0"
553 PROMPT
554 XEQ 15
555 SF IND X
556 FS?C 04
557 SF 05
558 "RV-FREI=2/N=0"
559 PROMPT
560 XEQ 15
561 SF IND X
562 "LOHN?"
563 PROMPT
564 FIX 2
565 XEQ 15
566 1
567 12
568 FS? 06
569 X<>Y
570 RDN
571 *
572 INT
573 ST+ 20
574 STO 21
575 2000
576 100
577 FS? 03
578 ST+ X
579 +
580 ST- 20
581 XEQ 02
582 XEQ E
583 RCL 20
584*LBL B
585 FS? 03
586 2
587 FS? 03
588 /
589 ENTER↑
590 ENTER↑
591 54
592 MOD
593 -
594 5617
595 X>Y?
596 GTO 21
597 RDN
598 8154
599 X>Y?
600 GTO 22
601 RDN
602 120042
603 X>Y?
604 GTO 23
605 RDN
606 .53
607 *
608 22842
609 -
610 INT
611 GTO 24
612*LBL 21
613 0
614 GTO 24
615*LBL 22
616 RDN
617 .19
618 *
619 INT
620 1067
621 -
622 GTO 24
623*LBL 23
624 RDN
625 8100
626 -
627 1 E4
628 /
629 ENTER↑
630 ENTER↑
631 151.94
632 *
633 1 E3
634 *
635 INT
636 1 E3
637 /
638 1900
639 +
640 *
641 INT
642 472
643 +
644*LBL 24
645 FS?C 03
646 ST+ X
647 1
648 12
649 FS? 06
650 X<>Y
651 RDN
652 /
653 1 E2
654 *
655 INT
656 1 E2
657 /
658 FIX 2
659 "EST="
660 FS? 10
661 "LST="
662 XEQ 15
663 RCL 00
664 300
665 *
666 1
667 12
668 FS? 06
669 X<>Y
670 RDN
671 /
672 -
673 FIX 1
674 CLA
675 8
676 ARCL X
677 "├% KIST="
678 %
679 FIX 2
680 X<0?
681 0
682 XEQ 15
683 END
```

August Schorr
Wildschwanbrook 42 D
2000 Hamburg 73

M-Code ASRCH

mit dem Demo "UCAT"

von Klaus Huppertz

132 Zeilen, 267 Bytes, HP-41 CX, Assembler.

Nach "XRADR" (1/89) und "FINDAD" (von Bernhard Saalfeld, 5/88) kommt nun eine wirklich universelle Funktion, die nicht nur zu **allen** Funktionen aus CAT 1, 2 und 3 die Adressen liefert, sondern auch den Typ und den XROM-Code, wenn vorhanden, zum Teil also eine Umkehrfunktion von "XRADR".
Alle Angaben werden im ALPHA-Register untergebracht. Das Schema folgt in der Programmbeschreibung (Grafik).

Der Typ wird wie folgt ausgegeben:

U USER-Code
M M-Code XROM-Funktionen
F Funktionen aus dem Mainframe, also Page 0..2

Programmbedienung:

Bei Ausführung über die Tastatur verhält sich die Funktion wie COPY, also nach Aufruf Eingabe von ALPHA <Funktionsname> ALPHA.
Bei Benutzung innerhalb eines User-CODE Programms wird der Funktionsname im ALPHA-Register erwartet. Zu beachten sind dabei zwei Dinge:

1. Die Funktion wird programmiert, indem man nach Aufruf zweimal die ALPHA-Taste **ohne** Eingabe eines Namens betätigt.
2. Sie läßt sich zwar über SST ausführen, meldet dann aber immer "NONEXISTENT"; sie funktioniert also nur einwandfrei bei **laufendem** Programm.

Programmbeschreibung:

Zeilen 8 bis 28:
Dieser Teil wird ausgeführt, wenn die Funktion von einem Programm aus aufgerufen wird (s.a. Zeile 6, 7).
Der Funktionsname im ALPHA-Register wird spiegelverkehrt und rechtsbündig ins Q-Register geschrieben. Dadurch werden die gleichen Voraussetzungen geschaffen, wie bei der Eingabe des Namens über ein ALPHA-Prompt, also XEQ "ASRCH" ALPHA <Funktionsname> ALPHA. Dies geschieht bei Aufrufen von "ASRCH" über die Tastatur automatisch und wird bewirkt durch die Kodierung der ersten beiden Buchstaben des Funktionsnamens. Ausschlaggebend für die Art des Promptes sind die zwei höchstwertigen Bits:
Die Kombination 1h,0h für die dritte Hexziffer der ersten beiden Buchstaben (244-Code) erlaubt die ALPHA-Eingabe oder das Ausführen ohne Eingabe. 1h,2h würde in jedem Fall eine ALPHA-Eingabe verlangen. Übrigens stehen alle Variationsmöglichkeiten mit der dritten Hexziffer (beim ersten Buchstaben ungleich 0), also 12 insgesamt, für verschiedene Prompts. Sie lassen sich problemlos ausprobieren, wenn man die Stackangaben meidet (sie funktionieren **nur** bei Betriebssystemfunktionen). Im Programm funktionieren die Prompts bei XROM-Funktionen generell nicht, weil der Parameter verloren geht. Deswegen muß bei programmierten "ASRCH" auch der gewünschte Funktionsname im ALPHA-Register stehen, um dann "zu Fuß" über den ersten Programmteil nach Q gebracht zu werden.

Der Prozess, der das bei den von Hand ausgeführten und den Betriebssystemfunktionen automatisch tut, nennt sich "partial key sequenzing". Denjenigen, die sich für dieses Thema interessieren, kann ich das Buch von Ken Emery "HP-41 MCode for Beginners", erhältlich z.B. über W&W, sehr empfehlen.

Zeilen 29 bis 36:
Das ist das Herzstück des Programms. Hier wird eine äußerst leistungsfähige Betriebssystemroutine aufgerufen, die einem geradezu alle benötigten Informationen auf dem Silbertablett serviert: "ASRCH (26C5h)".

Nach Ausführung stehen im CPU C-Register folgende Daten:

1. In den Nybblen 3 bis 0 die Anfangsadresse der gewünschten Funktion.
2. Bei CAT2-Funktionen in Nybblen 7 bis 4 der XROM-Code.

Weiterhin sind drei CPU-Flags gesetzt:

1. Flag2: Gesetzt für ROM- und gelöscht für RAM-Adressen.
2. Flag5: Gesetzt bei Betriebssystemfunktionen aus den Pages 0 bis 3, ansonsten gelöscht.
3. Flag9: Gesetzt für M-Code- und gelöscht für USER-Code-Funktionen.

CPU-C ist komplett gelöscht, wenn unter dem angegebenen Namen keine Funktion existiert.

Spätestens zu diesem Zeitpunkt wird klar, daß sich vor dieser Routine (26C5h) auch wirklich keine Funktion verstecken kann, die über einen Namen verfügt.

Das heißt: alle CAT 1, 2 und 3 Funktionen kann man über eine Routine abfragen !
An diesem Punkt war eigentlich nur noch die Frage offen, wie man die gelieferten Informationen ausgeben kann. Ich habe mich für das ALPHA-Register und folgendes Schema entschieden:

1..7	8	9..12
Filename	Space	Adress

13	14	15	16..19
Space	Typ	Space	XROM-Code

Alle Daten werden linksbündig geschrieben.

Zeilen 37 bis 59 schreiben den Funktionsnamen ins ALPHA-Register und füllen die restlichen Spalten bis Spalte 8 mit Spaces auf.

Zeilen 60 bis 101:
Hier wird die Funktionsadresse fertig gemacht. Dabei sind RAM- und ROM-Adressen zu unterscheiden: Die ROM-Adressen haben das Format pxxx, wobei p die Pagenummer und xxx die Adresse innerhalb der Page ist. Sie können sofort in ASCII-Zeichen umgewandelt werden. Etwas problematischer ist es bei den RAM-Adressen. Sie haben das Format yxxx, wobei y die **doppelte** Bytenummer innerhalb eines Registers darstellt und xxx die absolute Registeradresse: Zeilen 60 bis 72 besorgen also einen Byteshift nach rechts (Division durch 2), der sich **nur** auf das Nybble Nr. 3 bezieht.
Zeilen 80 bis 90 besorgt die Umwandlung in ASCII-Code, wobei zu beachten ist, daß die Ziffern 0 bis 9 als 30h bis 39h und die Buchstaben A bis F als 41h bis 46h dargestellt werden. Die restlichen Teile dieses Blocks stehen neben dem Listing erklärt.

Zeilen 104 bis 123 fügen einen Buchstaben hinzu, um den Typ der Funktion kennzuzeichnen.
Hier wird die Statusinformation der CPU-Flags 2, 5 und 9 genutzt.

Zeilen 126 bis 134:
Handelt es sich um eine ROM-Adresse (egal welcher Art), werden Vorbereitungen getroffen, um über den Block Zeilen 75 bis 103 den XROM-Code anzuhängen. Die Bedeutung dieses Codes ist für die Betriebssystemfunktionen aus den Pages 0 bis 2 recht zweifelhaft. Wer das CCD-Modul besitzt, könnte damit versuchen, die entsprechenden Funktionen aufzurufen. Das Ergebnis fällt aber in der Regel anders aus als erwartet. Das hängt wohl damit zusammen, daß die ersten 3 Pages anders organisiert sind als die Nachfolgenden.
Das Hauptanwendungsgebiet dieser Funktion liegt sicher in der schnellen Ausgabe **einzelner** Adressen oder XROM-Codes.

```
0001   CEA3   Ø88   "H"            Programmname;
0002   CEA4   ØØ3   "C"            darin verschlüsselt:
0003   CEA5   Ø12   "R"            Aufforderung zur Eingabe eines alphanumerischen
0004   CEA6   Ø13   "S"            Strings (z.B.: wie bei COPY)
0005   CEA'   1Ø1   "A"
0006   CEA8   2CC   ?FSET 13       Läuft gerade ein User-Code Programm?
0007   CEA9   ØB3   JNC CEBF +16   wenn nicht: Sprung
0008   CEAA   31C   R= 1           aktiven Pointer auf 1 setzen
0009   CEAB   Ø2E   B=Ø ALL        CPU-B löschen
0010   CEAC   13Ø   LDI S&X
0011   CEAD   ØØ6                  Schleifenzähler (maximal 7 Zeichen darf ein Name
0012   CEAE   ØA6   A<>C S&X       haben) nach CPU-A S&X
0013   CEAF   178   READ 5(M)      Namen aus Alpharegister nach CPU-C
0014   CEBØ   2E6   ?C≠Ø S&X       Sind noch Zeichen des Namens vorhanden?
0015   CEB1   Ø4B   JNC CEBA +Ø9   wenn nicht: Sprung
0016   CEB2   ØEA   C<>B R<        ansonsten: Zeichen in umgekehrter Reihenfolge an
                                   die Zeichenkette in CPU-B anhängen
0017   CEB3   3CE   RSHFC ALL      nächstes Zeichen in Position rücken
0018   CEB4   3CE   RSHFC ALL
0019   CEB5   ØEE   C<>B ALL       Zeichenkette in CPU-B um zwei Nybble nach links
0020   CEB6   37C   RCR 12         rücken, um Platz für das nächste ASCII-Zeichen
0021   CEB7   ØEE   C<>B ALL       zu schaffen
0022   CEB8   1A6   A=A-1 S&X      Schleifenzähler dekrementieren und Sprung an den
0023   CEB9   3BB   JNC CEBØ -Ø9   Anfang der Schleife bis maximal 7 Zeichen übertragen
0024   CEBA   Ø46   C=Ø S&X        Statusregister anwählen
0025   CEBB   27Ø   RAM SLCT
0026   CEBC   ØEE   C<>B ALL       umgedrehte Zeichenkette nach CPU-C
0027   CEBD   23C   RCR 2          die letzte überflüssige Rotation rückgängig machen
0028   CEBE   268   WRIT 9(Q)      und im Statusregister Q ablegen
0029   CEBF   278   READ 9(Q)      hier ist der Einstieg für die Funktion, wenn sie
                                   über die Tastatur aufgerufen wird
0030   CECØ   1D8   C<>M ALL       Zeichenkette nach CPU-M; Einstiegsbedingung für
0031   CEC1   315                  das Herzstück des Programms
0032   CEC2   Ø98   ?NCXQ 26C5     "ASRCH" , danach habe ich das Programm benannt
0033   CEC3   2EE   ?C≠Ø ALL       Funktion gefunden?
0034   CEC4   381                  wenn nicht:
0035   CEC5   ØØA   ?NCGO Ø2EØ     Fehlermeldung
0036   CEC6   1D8   C<>M ALL       Ergebnis für späteren Gebrauch in Sicherheit bringen
0037   CEC7   345
0038   CEC8   Ø4Ø   ?NCXY 1ØD1     Alpharegister löschen
0039   CEC9   13Ø   LDI S&X
0040   CECA   ØØ7                  diesmal einen Schleifenzähler für die Ausgabe des
0041   CECB   ØE6   C<>B S&X       Funktionsnamens nach CPU-B
0042   CECC   278   READ 9(Q)      Funktionsname nach CPU-C
0043   CECD   39C   R= Ø           aktiven Pointer auf Null setzen für späteren
                                   Datentransport nach CPU-G
0044   CECE   2E6   ?C≠Ø S&X       wenn kein Zeichen mehr vorhanden, Sprung aus der
0045   CECF   Ø5B   JNC CEDA +ØB   Schleife
0046   CEDØ   Ø58   G=C  R;+       Anfangsbedingung für "APNDNW"
0047   CED1   3CE   RSHFC ALL      nächstes Zeichen in Position rücken
0048   CED2   3CE   RSHFC ALL
0049   CED3   268   WRIT 9(Q)      Rest der Zeichenkette zurückspeichern
0050   CED4   Ø51                  "APNDNW": Zeichen im Alpharegister anhängen
0051   CED5   ØB4   ?NCXQ 2D14
0052   CED6   ØE6   C<>B S&X       Schleifenzähler nach CPU-C
0053   CED7   266   C=C-1 S&X      und dekrementieren, wenn max. 8 Zeichen übertragen
0054   CED8   39B   JNC CECB -ØD   sind, dann wird die Schleife verlassen und kein
0055   CED9   Ø2B   JNC CEDE +Ø5   Space mehr angehängt
0056   CEDA   13Ø   LDI S&X
0057   CEDB   Ø2Ø                  Space
```

```
0058  CEDC  Ø58  G=C  R;+       Anfangsbedingung und Rücksprung um
0059  CEDD  3BB  JNC CED4 -Ø9   Space anzuhängen
0060  CEDE  198  C=M ALL        Ergebnis aus "ASRCH" nach CPU-C
0061  CEDF  2ØC  ?FSET 2        wenn es sich um eine ROM-Adresse handelt:
0062  CEEØ  Ø5F  JC CEEB +ØB    dann Sprung zum nächsten Test
0063  CEE1  1ØE  A=C ALL        Kopie für die Korrektur der Bytezahl nach CPU-A
0064  CEE2  Ø5C  R= 4           Nybble links von der Bytezahl anwählen
0065  CEE3  Ø42  C=Ø  R         und löschen
0066  CEE4  1FA  C=C+C M        3er Bitshift nach links
0067  CEE5  1FA  C=C+C M
0068  CEE6  1FA  C=C+C M        und anschließende Rotation um ein Nybble nach rechts
0069  CEE7  33C  RCR 1          also ein Bitshift nach rechts
0070  CEE8  Ø1C  R= 3           die korrigierte Bytezahl anwählen
0071  CEE9  ØA2  A<>C  R        und in die Funktionsadresse einsetzen
0072  CEEA  ØAE  A<>C ALL       das Ergebnis nach CPU-C
0073  CEEB  Ø3C  RCR 3          Rotation, sodaß die links stehende Bytezahl im
                                Exponenten von CPU-C steht
0074  CEEC  1Ø4  CLRF 8         ermöglicht die folgende Schleife zweimal zu durchlaufen
0075  CEED  1D8  C<>M ALL       Ergebnis nach CPU-M
0076  CEEE  13Ø  LDI S&X
0077  CEEF  ØØ3                 Schleifenzähler für vierstellige Adresse
0078  CEFØ  ØE6  C<>B S&X       nach CPU-B S&X
0079  CEF1  198  C=M ALL        Ergebnis nach CPU-C
0080  CEF2  31C  R= 1           Pointer auf Nybble links von der Bytezahl stellen
0081  CEF3  ØDØ  LD R 3         Zahl zum ASCII-Code ergänzen (für Ziffern 0..9)
0082  CEF4  3DC  R=R+1          Pointer wieder auf die alte Stelle setzen
0083  CEF5  1Ø2  A=C  R         Ziffer nach CPU-A
0084  CEF6  Ø56  C=Ø XS
0085  CEF7  2AØ  SETDEC         Dezimalmodus, um einen möglichen Übertrag auf die
                                zweite Stelle von rechts festzustellen (von 9 auf A)
0086  CEF8  226  C=C+1 S&X      Test
0087  CEF9  266  C=C-1 S&X
0088  CEFA  362  ?A≠C  R        hat sich nun die zweite Stelle durch einen Übertrag
0089  CEFB  Ø13  JNC CEFD +Ø2   verändert? wenn nicht, wird die Korrektur der
0090  CEFC  226  C=C+1 S&X      ersten Stelle übersprungen
0091  CEFD  26Ø  SETHEX         für die nachfolgenden Unterprogramme Hex-Modus
0092  CEFE  39C  R= Ø           Pointer auf Null setzen um fertiges ASCII-Zeichen
0093  CEFF  Ø58  G=C  R;+       nach CPU-G zu bringen; Einstieg für:
0094  CFØØ  Ø51                 "APNDNW"
0095  CFØ1  ØB4  ?NCXQ 2D14     Zeichen im Alpharegister anhängen
0096  CFØ2  198  C=M ALL        Ergebnis wieder nach CPU-C um
0097  CFØ3  2FC  RCR 13         die nächste Ziffer an die richtige Stelle zu rücken
0098  CFØ4  1D8  C<>M ALL       und abspeichern nach CPU-M
0099  CFØ5  ØE6  C<>B S&X       Schleifenzähler nach CPU-C holen
0100  CFØ6  266  C=C-1 S&X      und dekrementieren; wenn alle Zeichen übertragen
0101  CFØ7  34B  JNC CEFØ -17   dann kein Rücksprung
0102  CFØ8  1ØC  ?FSET 8        wurde der Schleifenblock zum zweiten mal abgearbeitet:
0103  CFØ9  1Ø7  JC CF29 +2Ø    dann Sprung
0104  CFØA  13Ø  LDI S&X
0105  CFØB  Ø2Ø                 Space
0106  CFØC  Ø58  G=C  R;+       Anfangsbedingung für:
0107  CFØD  Ø51                 "APNDNW"
0108  CFØE  ØB4  ?NCXQ 2D14     Space im Alpharegister anhängen
0109  CFØF  Ø8C  ?FSET 5        wenn es sich nicht um eine Routine aus dem 12K-Betriebs-
0110  CF1Ø  Ø23  JNC CF14 +Ø4   system handelt, dann Sprung
0111  CF11  13Ø  LDI S&X
0112  CF12  Ø46  "F"            für Mainframe
0113  CF13  Ø43  JNC CF1B +Ø8   nächsten Test überspringen
0114  CF14  24C  ?FSET 9        wenn es sich nicht um eine M-Code-XROM-Funktion
0115  CF15  Ø23  JNC CF19 +Ø4   handelt, dann Sprung
```

```
0116  0F16  13Ø  LDI S&X
0117  0F17  Ø4D  "M"          für M-Code
0118  0F18  Ø1B  JNC 0F1B +Ø3  nächsten Fall überspringen
0119  0F19  13Ø  LDI S&X
0120  0F1A  Ø55  "U"          für Usercode
0121  0F1B  Ø58  G=C  R;+      Kennbuchstaben nach CPU-G
0122  0F1C  Ø51                und mit Hilfe von "APNDNW" im
0123  0F1D  Ø84  ?NCXQ 2D14    Alpharegister anhängen
0124  0F1E  2ØC  ?FSET 2       wenn es sich um eine RAM-Funktion handelt,
0125  0F1F  Ø53  JNC 0F29 +ØA  dann Sprung zum Programmende
0126  0F2Ø  13Ø  LDI S&X
0127  0F21  Ø2Ø                ansonsten noch ein Space zur Trennung
0128  0F22  Ø58  G=C  R;+      vom Kennbuchstaben anhängen
0129  0F23  Ø51
0130  0F24  ØB4  ?NCXQ 2D14    mit "APNDNW"
0131  0F25  198  C=M ALL       Ergebnis aus "ASRCH" nach CPU-C um die XROM-
0132  0F26  13C  RCR 8         Nummer anzuhängen
0133  0F27  1Ø8  SETF 8        Kennzeichnung dafür, daß der letzte Schleifenblock
                               zum zweiten mal durchlaufen wird
0134  0F28  22B  JNC CEED -3B  Sprung zum Schleifenanfang
0135  0F29  2CC  ?FSET 13      wenn Usercode-Programm läuft,
0136  0F2A  36Ø  C RTN         dann Funktionsende
0137  0F2B  2C9
0138  0F2C  Ø42  ?NCGO 1ØB2    "AVIEW": Anzeige der fertigen Zeile
```

Um aber zu zeigen, wie gut man mit dieser Funktion "spielen" kann, z.B.: Erstellen von frei zusammengestellten Funktionslistings für die Dokumentation eines Programms, habe ich folgendes Beispielprogramm gemacht:

Es nennt sich "UCAT" und enthält unter numerischen Marken folgende Funktionen:

LBL 01: Eröffnen oder Anwählen eines ASCII-Files
02: Neuerstellen oder Einfügen von einem oder mehrerer Funktionsnamen
03: Ausgabe eines Funktionslistings ab der momentanen Pointerposition
04: Verrücken einzelner Funktionsnamen unter Angabe von alter und neuer Positionsnummer
09: Vergrößern des Files um ein Register
10: FIX 2-Einstellung

Nummer 9 und 10 sind für die Bedienung von Hand überflüssig, werden aber vom Programm benötigt.

Ein Beispiel:

Das Programm wird über XEQ "UCAT" oder falls der Programmzeiger bereits richtig steht, über XEQ 01, aufgerufen. Es erscheint die Frage: "FNAME?" (3). Daraufhin wird ein Filename eingegeben. Existiert unter diesem Namen noch kein File, wird er neu eröffnet, ansonsten wird er nur zum Arbeitsfile gemacht (6-16).

Dann wird der Filetyp überprüft (17-34). Wurde versehentlich der Name eines Programm- oder Datenfiles angegeben, erfolgt erneut die Aufforderung "FNAME?". Sie wird mit anderem Namen beantwortet und fortgefahren. Ergibt die Prüfung, daß es sich um einen ASCII-File handelt, zeigt der Rechner den Integerwert des momentanen Pointers an (35-41) außer, wenn er Null ist (42, 43), also in der Regel bei Neueröffnung des Files.

Sonst wird, falls kein Eingriff von Hand erfolgt, oder es sich um einen ASCII-File mit anderem Inhalt handelt (!), die Position des letzten Eintrags markiert, der aus praktischen Gründen immer ein Space darstellt (kein Funktionsname).

Will man an die Liste neue Funktionen anhängen, fährt man mit R/S fort.

Zum Einfügen gibt man die Einfügeposition an und drückt dann R/S. Die Positionsnummern stehen übrigens im Drukkerlisting ganz rechts (114-124).

Nun schaltet das Programm den ALPHA-Modus ein und erwartet nach dem Anhalten die Eingabe eines Funktionsnamens (44-57). Dann wird mit R/S fortgefahren (58-61).

Je nachdem, ob mit RESZFL der File vergrößert werden mußte (125-132) oder nicht, erscheint beim nächsten Anhalten ein leeres ALPHA-Register oder der letzte eingegebene Name. Das irritiert vielleicht etwas, aber eine einheitliche Anzeige hätte das ohnehin schon reichlich lange Programm noch länger gemacht. Zum Beenden des Programms wird einmal R/S ohne Eingabe gedrückt.

Es erfolgt die komplette Ausgabe der Funktionstabelle über die ALPHA-Anzeige oder, wenn angeschlossen, über den Drucker (62-94).

Die Funktionstabelle läßt sich nun beliebig mit den Funktionen XEQ 01 bis 05 in die gewünschte Form bringen.

UFF ! Das war's ! Bis auf weiteres und happy programming.

```
01*LBL "UCAT"
02*LBL 01
03 "FNAME?"
04 AON
05 STOP
06 "├      "
07 ASTO X
08 AOFF
09  E
10 SF 25
11 CRFLAS
12 FS? 25
13 " "
14 FS?C 25
15 INSREC
16 RDN
17 1.9
18*LBL 13
19 RCL X
20 XROM 25,52
21 R↑
22 ASTO X
23 R↑
24 X=Y?
25 GTO 14
26 R↑
27 ISG X
28 GTO 13
29*LBL 14
30 RDN
31 "AS"
32 ASTO X
33 X≠Y?
34 GTO 01
35 RCLPT
36*LBL 02
37 FIX 0
38 INT
39 "P= "
40 ARCL X
41 "├ ?"
42 X≠0?
```

```
 43 PROMPT
 44 CLA
 45 AON
 46*LBL 15
 47 SEEKPT
 48 XROM 25,49
 49 7
 50 -
 51 X<0?
 52 XEQ 09
 53 RDN
 54  E
 55 +
 56 CF 23
 57 STOP
 58 FS? 23
 59 INSREC
 60 FS?C 23
 61 GTO 15
 62 AOFF
 63 ADV
 64 "  FNC ADR. T "
 65 "├ XR"
 66 AVIEW
 67 CLX
 68 SEEKPT
 69 CLD
 70 FC? 55
 71 SF 21
 72*LBL 03
 73 SF 25
 74 GETREC
 75 FC?C 25
 76 GTO 10
 77 SF 25
 78 XROM 17,16
 79 FS? 55
 80 XEQ 19
 81 FC? 55
 82 AVIEW
 83 FC?C 25
 84 CLD
 85 ISG X
 86 ""
 87 GTO 03
 88*LBL 10
 89 FIX 2
 90 FC? 55
 91 CF 21
 92  E
 93 -
 94 RTN
 95*LBL 04
 96 "START↑ZIEL"
 97 PROMPT
 98 X<>Y
 99 SEEKPT
100 GETREC
```

```
101 SEEKPT
102 DELREC
103 X<>Y
104 SEEKPT
105 INSREC
106 RTN
107*LBL 05
108 "P?"
109 PROMPT
110 CLA
111 SEEKPT
112 DELREC
113 RTN
114*LBL 19
115 ALENG
116 15
117 -
118 X<0?
119 "├      "
120 "├  "
121 RDN
122 ARCL X
123 PRA
124 RTN
125*LBL 09
126 RDN
127  E
128 CLA
129 FLSIZE
130 +
131 XROM 25,..
132 END
```

```
 20 EMDIRX
 48 ASROOM
 78 ASRCH
131 RESZFL
```

Klaus Huppertz
Nievelsteinstraße 30
4050 Mönchengladbach 3

LR-Zerlegung

von Michael Schilli

168 Zeilen, 240 Bytes, SIZE 010 + N², HP-41C

Jede Matrix M, die nach dem Gauß-Verfahren mit natürlicher Pivotfolge behandelt werden kann, läßt sich in der Form

$$\underset{\sim}{M} = \underset{\sim}{L} * \underset{\sim}{R}$$

mit $\underset{\sim}{L} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ * & & 1 \end{pmatrix}$, $\underset{\sim}{R} = \begin{pmatrix} * & & * \\ & \ddots & \\ 0 & & * \end{pmatrix}$

schreiben.
Die obere Dreiecksmatrix R erhält man aus dem gaußschen Algorithmus mit natürlicher Pivotwahl; die untere Dreiecksmatrix L aus den Quotienten aus dem jeweiligen ersten relevanten Element einer Zeile und dem entsprechenden Pivotelement.
Z.B.:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{a_{21}^{(1)}}{a_{11}^{(1)}} & 1 & 0 \\ \frac{a_{31}^{(1)}}{a_{21}^{(1)}} & \frac{a_{32}^{(2)}}{a_{22}^{(2)}} & 1 \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{bmatrix}$$

```
 01*LBL "LR"
CLX  STO 09  10  STO 01
STO 05  "DIM ?"  PROMPT
STO 00  STO 02  STO 03
1  +  STO 04  CF 29  "a"
ASTO 06  ":"  ASTO 07
"= ?"  ASTO 08

 22*LBL 00
RCL 00  STO 02

 25*LBL 01
FIX 0  CLA  ARCL 06
RCL 04  RCL 03  -
ARCL X  ARCL 07  RCL 04
RCL 02  -  ARCL X
ARCL 08  PROMPT
STO IND 05  ISG 05  ADV
DSE 02  GTO 01  DSE 03
GTO 00  TONE 9  RCL 00
1  -  STO 08

 52*LBL 02
RCL 09  1  +  RCL 00
DSE X  1 E3  /  STO Z  +
STO 06  STO 02  CLX
RCL 09  +  STO 07
STO 03

 69*LBL 03
RCL 03  STO 07  RCL 06
RCL 00  *  RCL 01  +  +
ENTER↑  RCL IND X
RCL 09  RCL 00  *
RCL 09  +  RCL 01  +
RCL IND X  X<>Y  RDN  /
STO IND Y  STO 04
ISG 07  X<>Y  INT  ISG X
ADV  STO 05

 99*LBL 04
RCL IND 05  RCL 01
RCL 07  +  RCL 09
RCL 00  *  +  RCL IND X
RCL 04  *  X<>Y  RDN  -
STO IND 05  ISG 05  ADV
ISG 07  GTO 04  ISG 06
GTO 03  1  ST+ 09
DSE 08  GTO 02

125*LBL 07
SF 21  RCL 00  STO 02
STO 03  1  +  STO 04
RCL 01  STO 05

135*LBL 05
RCL 00  STO 02

138*LBL 06
FIX 0  RCL 02  RCL 03  -
"R"  X>0?  "L"  RCL 04
RCL 03  -  ARCL X  "├:"
RCL 04  RCL 02  -
ARCL X  "├="  FIX 5
ARCL IND 05  AVIEW
ISG 05  ADV  DSE 02
GTO 06  DSE 03  GTO 05
BEEP  GTO 07  END
```

Michael Schilli
Daucherstraße 2
8900 Augsburg

Fortsetzung von Seite 27
Codierungstheorie

```
655*LBL 27
656 "FEHLER: P"
657 FIX 0
658 RCL 00
659 RCL 12
660 +
661 ARCL X
662 "}>P"
663 ARCL 00
664 PROMPT
665 END
```

"+" = "}"

Michael Schilli
Daucherstraße 2
8900 Augsburg

Geschwindigkeitsmessung

Mit Speicherung des Meßzeitpunktes

von Dr. G. Heilmann

98 Zeilen, 217 Bytes, 31 Regs., HP-41 CX, TIME

Das Programm mißt die Geschwindigkeiten von Objekten über eine feste Meßstrecke in km/h. Zeitpunkt des Endes jeder Einzelmessung und die mittlere Geschwindigkeit im Zeitintervall der Messung werden im ASCII-File Q abgelegt. Die Meßzeit sollte für ausreichende Genauigkeit 3 sec nicht unterschreiten. Etwa 45 Messungen können nacheinander erfaßt werden. Überlauf von Q führt zur Meldung: END OF FL. Dann, oder nach Abschluß einer Meßreihe gibt

XEQ E

bei angeschlossenem Drucker eine Liste nach Abb.1 aus.

Messung vom 16.10.1989

Zeit	km/h
14:41	64.5
14:41	57.3
14:46	51.3
14:47	62.5
14:48	70.7
14:52	45.3
14:54	55.7
14:54	56.5
14:55	61.4
14:56	48.0
14:57	77.2
14:58	67.2
15:00	80.6
15:02	77.4
15:02	72.5
15:03	60.9
15:05	54.5
15:06	59.1
15:06	69.1
15:07	51.8
15:09	67.7
15:11	56.0
V(Mittel)	62.1

Abb. 1

1. XEQ SX

   startet das Programm und weist SX der Taste B zu

1.1 KM?

   fordert zur Eingabe der Länge der Meßstrecke in KM auf

1.2 R/S

   es erklingt ein kurzes Signal und die eingegebene Länge der Meßstrekke erscheint im X-Register.

2. Der Rechner ist meßbereit.

   0.0

2.1 Messung starten:

   R/S

2.2 Messung beenden:

   R/S

Nach etwa 0,5s erscheint die gemessene Geschwindigkeit in km/h ohne Benennung mit einer Nachkommastelle. Nach weiteren 0,5s ertönt ein Signal. Der Rechner ist wieder meßbereit. Nach 2.

3. In einer Meßreihe kann man während einer Messung:

3.1 den Rechner ausschalten und vor dem Ende der erwarteten Meßzeit wieder einschalten,

3.3 beliebig das Tastenfeld benutzen (die Stoppuhr muß unberührt bleiben, der Zeiger im erweiterten Speicher und der Inhalt von R00, die Länge der Meßstrecke in km, müssen vor dem Ende der Meßzeit wiederhergestellt werden),

3.4 Bereitschaft zur Beendigung der Messung (bei 3.3 nach Rückkehr ins Programm SX) bringt

   XEQ C

   mit Anzeige der Meßstrecke. Weiter nach 2.

4. In einer Meßreihe kann die Länge der Meßstrecke geändert werden:

4.1 XEQ A

   und nach 1.1

4.2 die Länge der Meßstrecke von Hand in R00 eingegeben werden.

5. Das Programm HG ist zur graphischen Aufbereitung der gemessenen Daten für eine Überwachung des innerörtlichen Verkehrs gedacht und benutzt eine Idee von Ronald Gordon, PPC-Club (Vol.7,Nr.9,S.17), abgedruckt in S. Dearing 'Tips, Tricks, Routinen für Taschenrechner der Serie HP-41', Heldermann, Berlin 1984, S.138.

6. Fehlerbetrachtung:

Das Programm SX ist auf schnellstmögliche erneute Meßbereitschaft ausgelegt und verzichtet deßhalb auf zusätzliche Ausgaben (z.B. Einheiten). Die Zeitmessung verursacht außer durch die persönlichen Gleichung des Beobachters einen systematischen Fehler von ≤0,1s. Der Meßfehler für die Zeit wird also zwischen

Histogramm

km/h
bis
50
55
60
65
70
75
80
85
90
95
und darüber

Abb. 2

0,2s und 0,4s liegen. Für Geschwindigkeitsmessungen mit weniger als 10% Fehler sollte die Meßstrecke so gewählt werden, daß bei der Geschwindigkeit, die am wichtigsten erscheint, die Meßstrecke in mindestens 5s zurückgelegt wird.

7. Schlußbemerkung.

Wer keinen Lesestift besitzt, der möge große Buchstaben für die kleinen eingeben.

Programmteil E:

Die Ergebnisse können auch ohne Drucker durch geeignete Änderungen dieses Programmteiles vom Rechner ausgegeben werden, etwa wie im Beispielprogramm, das am bequemsten als Ersatz ab Zeile 042 von SX eingegeben wird.

```
01*LBL "SX"
02 SF 27
03 7
04 PSIZE
05 "SX"
06 12
07 PASN
08 "Q"
09 70
10 SF 25
11 PURFL
12 CRFLAS
13*LBL A
14 FIX 1
15 STOPSW
16 "KM?"
17 PROMPT
18 STO 00
19*LBL C
20 VIEW 00
21*LBL 00
22 CLX
23 SETSW
24 TONE P
25 STOP
26 RUNSW
```

```
27 STOP
28 STOPSW
29 TIME
30 CLA
31 ATIME
32 APPREC
33 RCL 00
34 RCLSW
35 HR
36 /
37 VIEW X
38 CLA
39 ARCL X
40 APPREC
41 GTO 00
42*LBL E
43 "Q"
44 CLX
45 SEEKPTA
46 ADV
47 CLA
48 " Messung vom "
49 FIX 6
50 DATE
51 ADATE
52 PRA
53 XEQ 03
54 " Zeit"
55 ACA
56 14
57 SKPCHR
58 "km/h"
59 ACA
60 ADV
61 XEQ 03
62 ^REG 01
63 CL^
64 SF 25
65 FIX 1
66*LBL 01
67 GETREC
68 FC? 25
69 GTO 02
70 ACA
71 GETREC
72 18
73 ALENG
74 -
75 SKPCHR
76 ACA
77 ADV
78 ANUM
79 ^+
80 GTO 01
81*LBL 02
82 XEQ 03
83 "V(Mittel)"
84 ACA
85 8
86 SKPCHR
87 MEAN
88 ACX
89 ADV
90*LBL 03
91 SF 00
92*LBL 04
93 "------------"
94 ACA
95 FS?C 00
96 GTO 04
97 ADV
98 END
```

```
01*LBL "HG"
02 11
03 PSIZE
04 SF 25
05 CF 29
06 FIX 0
07 CLX
08 "Q"
09 SEEKPTA
10  E2
11 1/X
12 CLRGX
13*LBL 00
14 GETREC
15 FC? 25
16 GTO 01
17 GETREC
18 9
19 ANUM
20 50
21 -
22 5
23 /
24 INT
25 X>Y?
26 X<>Y
27 X<=0?
28 CLX
29 ENTER^
30 SIGN
31 +
32 ISG IND X
33 ""
34 RCL IND X
35 RCL 00
36 X<=Y?
37 X<>Y
38 STO 00
39 GTO 00
40*LBL 01
41 SF 12
42 "Histogramm"
43 ACA
44 PRBUF
45 "----------"
46 ACA
47 PRBUF
48 CF 12
49 "km/h"
50 ACA
51 PRBUF
52 "bis"
53 ACA
54 PRBUF
55 10
56*LBL 02
57 RCL IND X
58 RCL 00
59 X<=Y?
60 X<>Y
61 STO 00
62 RCL Z
63 DSE X
64 GTO 02
65 1.01
66 154
67 RCL 00
68 /
69*LBL 03
70 ST* IND Y
71 ISG Y
72 GTO 03
73 1.01
74 STO 00
75*LBL 04
76 45
77 RCL 00
78 INT
79 5
80 *
81 +
82 CLA
83 ARCL X
84 ACA
85 RCL IND 00
86 RCL X
87 7
88 X<=Y?
89 SF 00
90 MOD
91 X<>Y
92 LASTX
93 /
94 LASTX
95 FS?C 00
96 ST+ Z
97 SIGN
98 -
99 INT
100 X<=0?
101 GTO 06
102 31
103*LBL 05
104 ACCHR
105 DSE Y
106 GTO 05
107 RDN
108*LBL 06
109 X<>Y
110 127
111*LBL 07
112 ACCOL
113 DSE Y
114 GTO 07
115 PRBUF
116 ISG 00
117 GTO 04
118 "und darüber"
119 ACA
120 PRBUF
121 END
```

```
01*LBL E
02 SF 21
03 SF 25
04 FIX 1
05 "Q"
06 CLX
07 SEEKPTA
08*LBL 01
09 GETREC
10 FC? 25
11 RTN
12 ASTO X
13 GETREC
14 ANUM
15 X<>Y
16 CLA
17 ARCL X
18 X<>Y
19 "}  "
20 ARCL X
21 AVIEW
22 GTO 01
23 END
```

Dr. G. Heilmann
5408 Seelbach

Einfaches Navigationsprogramm DE

mit Anlauforten in ASCII-Files

von Dr. Heilmann

455 Zeilen, 777 Bytes, 111 Regs., HP-41CX, optional MM

Wie schon in der Überschrift erwähnt besteht die Besonderheit des Programmes darin, daß die Namen und Koordinaten von Orten auf der Erde in ASCII-Files abgelegt werden. Diese Orte (z.B. die einer Reise) und deren Koordinaten können dann der Reihe nach oder in beliebiger Reihenfolge abgefragt werden. Das Programm errechnet für Orte A und B, deren geographische Koordinaten nacheinander eingegeben worden sind:

a) Die Großkreisentfernung (der Großkreis ist kürzeste Verbindung zweier Punkte auf einer Kugel) und die Kurswinkel (beim Durchlaufen eines Großkreises ändert sich der Kurswinkel) von A in Richtung B in A und umgekehrt.

b) Die Entfernung auf der Loxodrome (die Loxodrome ist Verbindung auf der man auf einer Kugel mit festem Kurs von A nach B gelangt) und die Kurswinkel von A in Richtung B in A und umgekehrt.

Konventionen:

Die Entfernungsangaben können wahlweise in Kilometer (km) -Flag 03 gelöscht- oder in Seemeilen (sm) -Flag 03 gesetzt- erfolgen.

Die Kurswinkel werden in dezimalen Grad von Norden über Osten von 0° - 360° eingeben.

Geographische Koordinaten bei der Eingabe in Grad, Minuten, Sekunden. Breite von Süden nach Norden -90° < B < 90°, Länge von Osten nach Westen -180° < L < 180°. Singularität (B=90) führt zur Fehlermeldung NONEXISTENT.

Eine ausgewählte Anzahl von Orten lassen sich mit ihren Koordinaten durch das Hilfsprogramm O abspeichern.

Anwendung:

1.1. Das Programm O starten

XEQ O ANZ.-ORTE?

Es werden die ASCII-Files O und K angelegt.

Anzahl der Orte für die Grundaustattung des HP-41-CX mit erweitertem Speicher gelöscht 106, bei zwei X-Memory 511.

1.2. ORT?

Ortsnamen eingeben. Nur die ersten vier Zeichen werden gespeichert.

R/S

1.3.a. Geographische Breite
ENTER
geographische Länge.

(Bogensekunden bleiben unberücksichtigt!)

1.3.b Meldung NO ROOM, nicht genügend Speicherplatz für K.

Mit CAT4 überprüfen, wieviel Platz O belegt. Dann die gleiche Anzahl Register für K reservieren. Dazu weniger Orte vorsehen oder andere Files löschen. Zurück nach 1.1.

1.4.a. R/S
wie 1.2.

1.4.b. R/S
(Zahl)

Eingabe beendet.

Das Hilfsprogramm P druckt die Inhalte der Files O und K aus.

2. XEQ P

Es werden abwechselnd die ersten vier Zeichen des eingegebenen Ortes und dessen Koordinaten in aufeinanderfolgenden Zeilen ausgedruckt.

3. Das Hauptprogramm DE starten.
XEQ DE
B,L,O c->a F00 gelöscht

Flag 00 gelöscht: die erste Ortsangabe fehlt.

Es gibt an dieser Stelle drei Möglichkeiten die Koordinaten eines Ortes einzugeben:

3.1.1. Aus den angegebenen Orten einen aussuchen:

XEQ B ORT?

Den Namen eines gespeicherten Ortes eingeben

3.1.2.a. R/S
nach 4.

3.1.2.b. R/S
NONEXISTENT
XEQ A
nach 3.

3.1.3. Die eingegebenen Orte der Reihe nach abfragen (c für cntinue)
XEQ a

Die ersten vier Zeichen des Ortsnamen erscheinen. Danach 4.

4. B.L,O
c->b
F00 gesetzt

Der nächste Ort kann eingegeben werden.

4.1.1. wie 3.1.1.

4.1.2.a. wie 3.1.2.a., aber nach 4.

4.1.2.b. wie 3.1.2.b., aber XEQ F nach 3.

4.1.3. wie 3.1.3., aber XEQ b nach 4.

5. Menü zur Abfrage der errechneten Größen mit Hinweis zur Fortsetzung zum nächsten Ort: (B,L,O)
B,L,O
e ∠
- ∠

e ∠und - ∠kennzeichnen die Entfernung, Kurswinkel hin und Kurswinkel zurück, wie eingangs beschrieben, in dieser Reihenfolge auf den Tasten C,D,E (Großkreis) und umgeschaltet c,d,e (Loxodrome).

Für den Übergang zum nächsten Ort nach 3. zurück.

```
01♦LBL "DE"          27 HR
02 XEQ 16            28 ENTER↑
03 PI                29 X<> 04
04 R-D               30 STO 00
05 STO 17            31 X<>Y
06 ENTER↑            32 RCL 14
07 +                 33 P-R
08 STO 18            34 X=0?
09 4                 35 SF 99
10 /                 36 X<> 06
11 STO 16            37 STO 02
12♦LBL A             38 X<>Y
13 SF 00             39 X<> 07
14♦LBL F             40 STO 03
15♦LBL 00            41 FS?C 00
16 "B,L 0    c-)b"   42 GTO 00
17 FS? 00            43 RCL 17
18 "B,L 0    c-)a"   44 RCL 01
19 PROMPT            45 RCL 05
20♦LBL 15            46 -
21 FC?C 22           47 X≠0?
22 GTO 00            48 GTO 17
23 HR                49 RCL 00
24 X<> 05            50 RCL 04
25 STO 01            51 -
26 X<>Y              52 X<0?
```

```
 53 SF 01
 54 STO 08
 55 STO 11
 56 CLX
 57 ENTER↑
 58 RCL 17
 59 FS?C 01
 60 X<>Y
 61 STO 09
 62 STO 12
 63 X<>Y
 64 STO 10
 65 STO 13
 66*LBL J
 67 "B↑L 0 e ∠ -∠"
 68 PROMPT
 69 GTO J
 70*LBL 16
 71 CLX
 72 X<>F
 73 CF 22
 74 CF 23
 75 SF 27
 76 CF 29
 77 20
 78 PSIZE
 79 SIGN
 80 STO 14
 81  E2
 82 STO 15
 83 RTN
 84*LBL c
 85 "d= "
 86 RCL 11
 87 GTO 10
 88*LBL C
 89 "e= "
 90 RCL 08
 91*LBL 10
 92 FS? 03
 93 60
 94 FC? 03
 95 111.2
 96 *
 97 FIX 0
 98 ARCL X
 99 FS? 03
100 "⊦ SM"
101 FC? 03
102 "⊦ KM"
103 AVIEW
104 RTN
105 GTO J
106*LBL d
107 "c= "
108 RCL 12
109 GTO 10
110*LBL D
111 "∠= "
112 RCL 09
113 GTO 10
114*LBL e
115 "-c="
116 RCL 13
117 GTO 10
118*LBL E
119 "-∠="
120 RCL 10
121*LBL 10
122 RCL 18
123 MOD
124 FIX 1
125 ARCL X
126 "⊦ GRAD"
127 AVIEW
128 RTN
129 GTO J
130*LBL 01
131 CHS
132 RCL 18
133 +
134 FC?C 01
135 SF 01
136 RTN
137*LBL 17
138 X<0?
139 SF 01
140 ABS
141 X>Y?
142 XEQ 01
143 STO 08
144 D-R
145 XEQ 10
146 XEQ 18
147 FC?C 01
148 -
149 STO 10
150 GTO J
151*LBL 10
152 FIX 1
153 RCL 00
154 RND
155 RCL 04
156 RND
157 X=Y?
158 GTO 02
159 RCL Z
160 RCL 00
161 RCL 04
162 RCL 16
163 ST- Z
164 -
165 R↑
166 SIGN
167 ST+ X
168 ST/ Z
169 /
170 TAN
171 X<>Y
172 TAN
173 /
174 RCL 14
175 -
176 LN1+X
177 /
178 STO 11
179 ATAN
180 X<0?
181 XEQ 03
182 RCL 17
183 X<>Y
184 +
185 XEQ 10
186 RCL 11
187 X↑2
188 RCL 14
189 +
190 SQRT
191 RCL 00
192 RCL 04
193 -
194 ABS
195 *
196 STO 11
197 RTN
198*LBL 02
199 RDN
200 COS
201 *
202 R-D
203 STO 11
204 RDN
205 RCL 16
206 +
207*LBL 10
208 LASTX
209 FS? 01
210 X<>Y
211 FS?C 02
212 X<>Y
213 STO 12
214 X<>Y
215 STO 13
216 RTN
217*LBL 03
218 SF 02
219 RCL 18
220 +
221 RTN
222*LBL 18
223 RCL 08
224 COS
225 RCL 02
226 RCL 06
227 *
228 *
229 RCL 03
230 RCL 07
231 *
232 +
233 STO 10
234 ACOS
235 STO 08
236 SIN
237 RCL 07
238 RCL 10
239 RCL 03
240 *
241 -
242 R↑
243 RCL 02
244 XEQ 10
245 FS? 01
246 -
247 STO 09
248 R↑
249 RCL 03
250 RCL 10
251 RCL 07
252 *
253 -
254 R↑
255 RCL 06
256*LBL 10
257 *
258 /
259 ACOS
260 RCL 18
261 X<>Y
262 RTN
263*LBL 19
264*LBL 04
265 "ORT?"
266 AON
267 PROMPT
268 AOFF
269 FC?C 23
270 GTO 04
271 "⊦..."
272 ASTO L
273 "...."
274 ARCL L
275 8
276 AROT
277 ASHF
278 RTN
279*LBL a
280 CLX
281 FS?C 04
282 RTN
283 GTO 10
284*LBL "P"
285 XEQ 16
286 SF 04
287 FIX 2
288 4
289  E3
290 /
291 CHS
292 STO 19
293*LBL b
294 4
295  E3
296 /
297 RCL 19
298 +
299*LBL 10
300 STO 19
301 SF 25
302 "O"
303 SEEKPTA
304 FC? 25
305 GTO a
306 GETREC
307 6
308 ALENG
309 X<=Y?
310 "⊦..."
311 ASTO L
312 "...."
313 ARCL L
314 8
315 AROT
316 ASHF
317 AVIEW
318 GTO 10
319*LBL B
320 "O"
321 CLX
322 SEEKPTA
323 XEQ 19
324*LBL 05
325 POSFL
326 X<0?
327 SF 99
328 ENTER↑
329 FRC
330  E3
331 ST* Y
332 X<>Y
333 4
334 MOD
335 X=0?
336 GTO 10
337 RDN
338 1/X
339 +
340 SEEKPT
341 GTO 05
342*LBL 10
343 "K"
344 R↑
345 SEEKPTA
346 GETREC
347 SF 01
348*LBL 06
349 ATOX
350 ENTER↑
351 SIGN
352 -
353 ATOX
354 RCL 15
355 X>Y?
356 SF 02
357 FS? 02
358 SIGN
359 -
360 RCL 15
361 /
362 +
363 FS?C 02
364 CHS
365 FS? 04
366 ACX
367 FS?C 01
368 GTO 06
369 FS? 04
370 GTO 10
371 SF 22
372 GTO 15
373*LBL 10
374 PRBUF
375 RTN
376*LBL "O"
377 XEQ 16
378 "ANZ. ORTE?"
379 PROMPT
380 STO 00
381 4
382 *
383 RCL X
384 252
385 /
386 INT
387 LASTX
388 FRC
389 X≠0?
390 SIGN
391 +
392 +
393 RCL 14
394 +
395 7
396 /
397 INT
398 LASTX
399 FRC
400 X≠0?
401 SIGN
402 +
403 SF 25
404 "O"
405 PURFL
406 CRFLAS
407 SF 25
408 "K"
409 PURFL
410 CRFLAS
411*LBL 07
412 SF 25
413 "O"
414 RCLPTA
415 XEQ 19
416 APPCHR
417 FC? 25
418 APPREC
419 SF 25
420 "K"
421 RCLPTA
422*LBL 08
423 "B,L?"
424 PROMPT
425 FC?C 22
426 GTO 08
427 X<>Y
428 CLA
429 SF 01
430*LBL 09
431 X<0?
432 SF 00
433 ABS
434 RCL 14
435 +
436 ENTER↑
437 INT
438 XTOA
439 -
440 RCL 15
441 *
442 LASTX
443 FS?C 00
444 SIGN
445 +
446 XTOA
447 X<>Y
448 FS?C 01
449 GTO 09
450 APPCHR
451 FC? 25
452 APPREC
453 DSE 00
454 GTO 07
455 END
```

Agent 0028

Licence to piep von Ralf Pfeiffer

Für Geheimdienste und Steuerfahnder beginnen schlechte Zeiten, denn die Programme CODE, DCD (DeCoDe) und KRYPT verschlüsseln jeden Text nach einer Methode, die der französische Kryptograph Blaise de Vigenere im 16. Jahrhundert erfand. Das Prinzip war einfach, aber wirkungsvoll, und ist so ähnlich aus der Mathematik bekannt: Man nimmt eine Zahl (die Nachricht) addiert eine zweite (den Schlüssel) hinzu und erhält eine neue Zahl, aus der man die erste nur mit dem Schlüssel erhält. Buchstaben ordnet man daher Zahlen zu, wie 1 für A und 26 für Z. Kommt man bei der Addition von Buchstaben über 26 hinaus, wird 26 abgezogen (Addition ohne Übertrag), z.B. D+Y = 4+25 = 29-26 = C. Umgekehrt wird beim Dechiffrieren verfahren, der Schlüssel wird jetzt zum Code abgezogen, falls Null oder eine negative Zahl übrig bleibt, wird 26 addiert, z.B. C-Y = 3-25 = -22+26 = D.

Normalerweise wählt man einen Schlüssel, der kürzer ist als der Klartext, und "streckt" ihn dann durch Wiederholung, z.B.:

Klartext:

ALLESNEUMACHTDERMAI

Schlüssel:

ROSEROSEROSEROSERØS

Code:

SAEJKCXZEPVMLSXWEPB

Je mehr Zeichen der Schlüssel (hier ROSE) hat, desto schwieriger ist der Code zu knacken. Hat der Schlüssel nur ein Zeichen so ergibt sich ein Sonderfall, eine Verschlüsselung die schon Julius Cäsar benutzte, und als Versatzverfahren bekannt ist.

KRYPT übernimmt die Addition bzw. die Subtraktion von Buchstaben. Eigentlich wäre das ganz einfach: Zwei Strings gleicher Länge und XOR verknüpft sie auf die gewünschte Art. Leider kommen dann Zeichen heraus, die nicht über die Tastatur eingegeben, oder im Display angezeigt werden können. Daher definiert der String in Zeile 2-3 die Zeichen, die in den Schlüssel, Klartext und Code enthalten sein dürfen, andere Zeichen werden falsch übersetzt. KRYPT erlaubt es, diesen String um beliebige Zeichen (z.B. Kleinbuchstaben) zu erweitern oder kürzen (z. B. Zahlen raus), oder die Reihenfolge zu ändern. Das Ver- und Entschlüsseln klappt jedoch nur wenn jeweils der gleiche String im Programm verwendet wurde.

Die Bedienung ist einfach: CODE verschlüsselt und DCD entschlüsselt. Beide erwarten in Ebene 1 den Schlüssel und Klartext bzw. den Code in Ebene 2. In Ebene 1 geben beide Programme dann das Ergebnis ihrer Bemühungen aus. KRYPT ist dabei "nur" ein Unterprogramm. Wer genaueres über Kryptologie wissen will, kann in SPEKTRUM DER WISSENSCHAFT 12/88 s.8-11 und 1/89 s.6-10 nachlesen.

Weil nur frische Nachrichten interessant sind,verwandelt MCOD (MorseCODer) beliebige Texte (auch verschlüsselte) sofort in Morse-Signale um. MDCD (MorseDeCoDer) entschlüsselt sie wieder und MKEY ist ein interaktives Programm, welches auf Tastendrücke wartet und sofort piepst. Übersetzt werden alle Zeichen, die der String in Zeile 5-7 enthält. Alle anderen Zeichen (auch SPACE) erzeugen bei MORSE eine kurze Sendepause, bei MKEY hält das Programm an. Zwei Zeichen haben eine besondere Bedeutung: * steht für "Anfangszeichen" und # für "Schlußzeichen", eine Konvention, die sich Btx-Benutzer leicht merken können, Nummer oder Name des anderen Teilnehmers beginnen ja ebenfalls mit * und enden mit #.

Im übrigen gelten die bekannten Regeln, Strich ("lang") dauert dreimal länger als Punkt ("kurz"). Zwischen den Elementen eines Buchstabens ist für die Dauer eines Punktes Pause, zwischen Buchstaben 2 Punkte Pause und zwischen Worten (z.B. durch SPACE getrennt) 5 Punkte. In MCOD und MKEY kann man das durch die Konstante in Zeile 1 (Zeit in Sekunden) ändern. Etwas schwierig ist die Herstellung des Strings in Zeile 2-4 in allen Programmen. Die Zeichen im String haben alle Dezimalwerte über 128, so daß der Ausdruck sofort lesbar ist. Dennoch gibt eine Tabelle diese Dezimalwerte im einzelnen an. Der String kann mit CHR zusammengesetzt werden, nach EDIT fügt man vor ihm ein Programmbegrenzungszeichen ein (dieses Zeichen habe ich bereits in PRISMA 88.4.37 beschrieben) und fährt hinter ihm mit dem Rest des Programmes fort. Wird ein Buchstabe in ein Morsezeichen übersetzt, suchen MCOD und MKEY dessen Position im String der Zeilen 5-7. Mit dieser Nummer holen sich die Programme das Zeichen, welches im String der Zeilen 2-4 die gleiche Position belegt. Dessen Zahlenwert ist eine Zahl zwischen 129 und 255.

Betrachtet man diese als Binärzahl, dann steht ganz links eine 1. Von hier sucht das Programm nach rechts die nächste 1, und liest dann alle folgende Bits (in umgekehrter Reihenfolge) als "kurz" (=0) oder "lang" (=1). Entsprechend umgekehrt verfährt MDCD.

Die Prozedur ist deshalb so aufwendig, weil ein Morsesignal eigentlich aus einer 6 Bit-Zahl besteht, die je Bit 3 Zustände kennt: Kurz, Lang und weder-noch. In den Programmen tauchen außerdem noch die Befehle MEM DROP auf. Normalerweise räumt der 28 von Zeit zu Zeit selbständig sein RAM auf, was zu einer kleinen Reaktionspause führt. Gleiches passiert bei MEM, und damit es während der Ausgabe eines Zeichens keine unerwartete Pause gibt, löst das Programm dieses RAM-Packing planvoll aus.

MCOD erwartet in Ebene 1 einen String, der dann in piepsen übersetzt wird. MKEY dagegen, erfordert keine Eingaben sondern lauert auf Tastendrücke, jedenfalls solange kein unbekanntes Zeichen betätigt wird. MCOD läuft in einer Endlosschleife und erwartet einen Punkt (.) für "kurz" und einen Minus (-) für "lang", jedes andere Zeichen bedeutet (einmal gedrückt) eine kurze Sendepause und das Ende des Zeichens oder (zweimal gedrückt) ein Leerzeichen (SPACE). MDCD läßt sich nur mit ON beenden.

An zwei weiteren BEEP-Funktionen arbeite ich noch:

Zu einem Telefonanrufbeantworter mit Fernabfrage gehört ein kleines Kästchen, mit dem man das treue Gerät durch das Telefon anpiepst, worauf es die Anrufe vorspielt. Das Startsignal könnte auch ein HP-28 geben. Zum anderen piepsen auch TELEFAX-Geräte (Fernkopierer) in das Postnetz, damit das Empfangsgerät eine Kopie des Orginals im Sendegerät erstellt. Der aktuelle G-3 Standard (9600 Baud) überfordert sicher den HP-28S aber auch bei dem älteren G-2 Standard zeigten sich die Marktführer im FAX-Business außerordentlich kooperationsunwillig. Da inzwischen auch die Post mit der standrechtlichen Entladung meiner Batterien gedroht hat, werde ich sicherlich erst in der April-PRISMA über meine Erfolge berichten können.

In den Kurzmeldungen in PRISMA 89.4.44 stand etwas über ein Programm, welches Polynome in algebraischen Objekten auflöst. So ähnlich arbeitet auch GLN welches Polynome bis 4. Grades in seine Linearfaktoren zerlegt. Das Programm erkennt nicht, ob das algebraische Objekt ein echtes Polynom ist (oder z.B. ein SIN(x) enthält), sondern versucht, über die Taylorreihe (also wie TAYLR) ein Näherungspolynom zu finden, und dessen Nullstellen zu bestimmen, die dann grobe Näherungen oder ganz unbrauchbar sind.

Die algebraischen Objekte dürfen nur ableitbare Funktionen verwenden, und außer x keine Namen enthalten. GLN erlaubt =, jedoch bringt der Rechner alles auf die Form '...=0' und entfernt das =0

aus dem Ergebnis. Das Programm erwartet in Ebene 1 ein beliebig sortiertes algebraisches Objekt, und ersetzt es durch das Produkt seiner (maximal 4 reellen und/oder komplexen) Linearfaktoren und (sofern nicht 1) dem Koeffizienten der höchsten Potenz. Die benötigten Unterprogramme GL2, GL3 und GL4 stehen in PRISMA 89.4.35 wobei GL3 und GL4 ausdrücklich nicht für Polynome mit komplexen Koeffizienten geeignet sind - komplexe Nullstellen finden die Programme natürlich. Keine algebraischen Objekte verarbeitet das Programm BINOM. Es findet lediglich die n reellen/komplexen Nullstellen einer Gleichung der Form

x exp n + a = 0

wobei n (in Ebene 2 eingegeben) ganzzahlig positiv und a (in Ebene 1) nicht Null sein sollte. Nach BINOM sind die Eingaben verschwunden und die ersten n Stackebenen mit den Lösungen gefüllt.

Die Programme MAXL und MINL suchen den größten bzw. den kleinsten Wert einer Liste, die entweder Strings oder reelle Zahlen enthält. Durch den Bug einiger (aller?) 28C Versionen können die Programme bei Strings mit gleichen Anfangsbuchstaben versagen.

Die Programme REV (REVerse), REVL (REVerse List) und REVA (REVerse Array) drehen die Reihenfolge im Stack, in Listen oder Arrays um. Für den Stack verlangt REV in Ebene 1 die Anzahl der umzukehrenden Stackebenen (also genau wie ROLL oder DROPN), REVL erwartet in Ebene 1 die Liste und REVA (wirkt nicht wie TRN) eine Matrix oder einen Vektor.

Listing →

Sam

beim HP-28S piept es

Für viele Leute gehört es zu den ersten Taten am HP-28S, Flag 51 zu setzen, sprich ihn seiner Stimme zu berauben. Das Gegenteil will das Programm "Sam". Denn wenn der Oszillator auch für musikalische Ambitionen entschieden zu wackelig schwingt und klingt, so reicht er für ein Morsesignal doch aus.

"Sam" läßt einen beliebigen String seinem geistigen Ahnen* nach erklingen. Es richtet sich nach dem international üblichen Code ohne deutsche Umlaute und interpretiert folgende Zeichen richtig:

0123456789 ABCDEFGHIJK
LMNOPQRSTUVWXYZ
.?/+-=<>

<> sind dabei die Klammern (); das Zeichen * wird als Trennung interpretiert, die Zeichen ,;: zum Speicher sparen als t. Alle anderen Zeichen werden nicht umgesetzt, erzeugen also eine kurze Pause.

Das Programm benutzt den String aus Stack 1, der bestehen bleibt, und die Liste ABC in der die Zeichencodes stehen.

Gebetempo und Tonfrequenz lassen sich über die BEEP-Funktion für Strich und Punkt verändern; durch Entfernen von DUP am Anfang steht der String nach Ablauf nicht mehr in Stack. Für einen zeitlich genauen Ablauf enpfiehlt sich vor dem Programmstart ein MEM Befehl (Speicherorganisation).

* Samuel Morse, 1791-1872, Erfinder des Morseapparates und -alphabets.

Karsten Reimann
Unter den Linden 4
3400 Göttingen

```
Sam
« → t « t DUP SIZE 1 SWAP FOR i t i i
  SUB NUM 41 - IF DUP DUP 0 > SWAP
  50 < AND THEN 'ABC' SWAP GET →STR
  DUP SIZE 1 SWAP FOR j IF DUP j j
  SUB STR→ 1 - THEN 660 1 8 / BEEP
  ELSE 660 1 24 / BEEP END NEXT DROP
  ELSE DROP END NEXT » »

ABC
{ 21112 2222 0 1212 121212 21121 22222
  12222 11222 11122 11112 11111 21111
  22111 22211 22221 0 0 2221 11211 1122
  112211 0 12 2111 2121 211 1 1121 221
  1111 11 1222 212 1211 22 21 222 1221
  2212 121 111 2 112 1112 122 2112 2122
  2211 }
```

DRBRIEF 2

Ergänzung zum Ausdruck eines Briefes

Im folgenden eine kleine Notiz zum Artikel von U. Laag in PRISMA 4/89, Seite 11.

Es geht auch ohne LEX-File und Escape-Zeichen. Einmal mit PLR gelistet oder gedruckt kann jedes Textfile später ohne weiteres mit Sonderzeichen ausgedruckt werden.

Dr. G. Heilmann
Obernhofer Straße 15
5408 Seelbach

```
010 ! PLR. Ersetzt Anzeige- durch Druckzeichen.
020 DESTROY ALL @ DIM A$[1],T$[8] @ INTEGER I,N
030 INPUT "Quellfile?";T$ @ ASSIGN #1 TO T$
040 CREATE TEXT AX @ ASSIGN #2 TO AX
050 RESTORE @ DESTROY B$ @ DIM B$[96] @ READ #1;B$ @ ON ERROR
                                                    GOTO 110
060 DATA ä,204,ö,206,ü,207,Ä,216,Ö,218,Ü,219,ß,222
070 READ A$,I
080 N=POS(B$,A$) @ IF N THEN B$=B$[1,N-1]&CHR$(I)&B$[N+1] @ GOTO 80
090 IF I#222 THEN GOTO 70
100 PRINT #2;B$ @ PRINT B$ @ ENDLINE @ GOTO 50
110 PURGE T$ @ RENAME AX TO T$ @ DESTROY ALL @ END
```

CODE

```
001 « 1 CF KRYPT
    » 31 Bytes
```

DCD

```
001 « 1 SF KRYPT
    » 31 Bytes
```

KRYPT

```
001 « DUP SIZE
    "ABCDEFGHIJKLMNOPQRS
TUVWXYZ 0123456789"
    DUP SIZE → a b c
005   « 1 a
        FOR n b OVER n
007 DUP SUB POS 1
          IF FS?
009       THEN NEG
          END 1 - SWAP
011     NEXT SIZE →ARRY
    OVER SIZE → d e
013   « ""
        DO SWAP 1 a
015 SUB LAST + OVER SIZE
    SUB SWAP ROT 1 a
017         FOR n OVER b
    DUP ROT n DUP SUB
019 POS d n GET + c MOD
    1 + DUP SUB +
021         NEXT SWAP
          UNTIL DROP
023 OVER SIZE NOT
          END 1 e
025       » SUB
      » SWAP DROP
027 » 315,5 Bytes
```

```
2: "BEI ARD UND ZDF REIH
ERN SIE AUF DIE ERSTEN S
ITZE"
1: "RTL PFUSCH"
CODE

1: "TYUQQXYIXVVJA3V5BXLP
WAZQ7OZID1XJP8U5Z9V0W6BI
YZJX"

"RTL PFUSCH" DCD

1: "BEI ARD UND ZDF REIH
ERN SIE AUF DIE ERSTEN S
ITZE"
```

MDCD

```
001 « CLLCD CR
    "▶ ¡≤×⊢≠↳∫ᵏ→«πΣμ‼≥∂
√α∃←¿ʲ»¢ƒ¥ⁱ ̄∠ÂÊÎ£õµÅPä ς
ᵥ` "
005 "ABCDEFGHIJKLMNOPQRS
TUVWXYZ0123456789.,-'?#*
=/"
    ""
009   DO 1
        DO KEY
011     UNTIL
          IF DUP
013       THEN "-" ROT
            IF POS
015         THEN LAST 1
    - ROT 2 * + SWAP NOT
017         END
          END
019     END DUP 1
        IF >
021     THEN 128 + CHR 3
    PICK 5 PICK ROT POS
023 DUP SUB
        ELSE DROP " "
025     END + DUP DUP 1
    DISP SIZE 23 MOD
027     IF NOT
        THEN PR1 DROP ""
029     END 1000 ,01
      UNTIL BEEP 0
031   END
    » 355 Bytes
```

MCOD

```
001 « 1 SWAP
    "▶ ¡≤×⊢≠↳∫ᵏ→«πΣμ‼≥∂
√α∃←¿ʲ»¢ƒ¥ⁱ ̄∠ÂÊÎ£õµÅPä ς
ᵥ` "
005 "ABCDEFGHIJKLMNOPQRS
TUVWXYZ0123456789.,-'?#*
=/"
    ROT 1 OVER SIZE
009   FOR n 3 DUPN n DUP
    SUB POS
011     IF DUP
        THEN DUP SUB NUM
013 128 - 1 OVER LN 2 LN
    / IP
015       START 1000
    OVER 2
017         IF MOD
            THEN 3
019         ELSE 1
            END 7 PICK *
021 BEEP 2 / IP
          NEXT 2
023     ELSE DROP 5
        END 6 PICK *
025 WAIT MEM DROP2
      NEXT 4 DROPN
027 » 310,5 Bytes
```

MKEY

```
001 « 2
    "▶ ¡≤×⊢≠↳∫ᵏ→«πΣμ‼≥∂
√α∃←¿ʲ»¢ƒ¥ⁱ ̄∠ÂÊÎ£õµÅPä ς
ᵥ` "
005 "ABCDEFGHIJKLMNOPQRS
TUVWXYZ0123456789.,-'1#*
=/"
      DO DUP2
009     DO KEY
        UNTIL
011     END POS
        IF DUP
013     THEN DUP SUB NUM
    128 - 1 OVER LN 2 LN
015 / IP 6 PICK WAIT
          START 1500
017 OVER 2
            IF MOD
019         THEN 3
            ELSE 1
021         END 6 PICK *
    BEEP 2 / IP
023       NEXT DROP 0
        ELSE DROP2 1
025     END
      UNTIL MEM DROP
027   END 3 DROPN
    » 294 Bytes
```

Zeichen im STRING in Zeile 002 - 004:

```
"134 145 149 137 130 148
 139 144 132 158 141 146
 135 133 143 150 155 138
 136 131 140 152 142 153
 157 147 191 190 188 184
 176 160 161 163 167 175
 234 243 225 222 204 170
 181 177 169"
```

REV

```
001 « 2 SWAP
      FOR i i ROLL
003   NEXT
    » 39 Bytes
```

REVL

```
001 « SIZE LAST 2 +
    LIST→
003   FOR i i ROLL
      NEXT →LIST
005 » 49 Bytes
```

REVA

```
001 « SIZE LAST ARRY→
    LIST→ 2
003   IF SAME
      THEN *
005   END 1 + 2 SWAP
      FOR i i ROLL
007   NEXT →ARRY
    » 71,5 Bytes
```

```
1: [[ 1 2 3 4 ]
    [ 5 6 7 8 ]]

REVA

1: [[ 8 7 6 5 ]
    [ 4 3 2 1 ]]

1: { 1 2 3 4 5 6 7 }

REVL

1: { 7 6 5 4 3 2 1 }
```

BINOM

```
001 « DUP ABS LN ROT → r
    n
003   « 0 > n 2 * OVER +
    2 -
005     FOR i r i π *
    R→C n / EXP RND IM
007 LAST DUP RE IFTE 2
        STEP
009   »
    » 121,5 Bytes
```

```
2: 4,00
1: -16,00

BINOM

4: 2,00
3: (0,00.2,00)
2: -2,00
1: (0,00.-2,00)
```

GLN

```
001 « 0 4
      FOR a "«→ X" OVER
003 →STR + STR→ 0 SWAP
    →NUM a FACT / SWAP
005 'X' ∂ '0*X' +
      NEXT DROP 4 1
007   FOR b
        IF STD
009     THEN LAST → t
          « 1 b
011         FOR c c ROLL
    t /
013         NEXT { NEG
    GL2 GL3 GL4 } b GET
015 EVAL t 1 b
            START 'X'
017 ROT RND - *
            NEXT
019       » ABORT
        END -1
021   STEP
    » 254 Bytes
```

```
'(X-1)*(SQ(X)-1)*X-
20*(X^2-1)'

GLN

1: '(X-1)*(X-5)*(X+1)*(
X+4)'
```

MINL

```
001 « LIST→ 2 SWAP
      START < LAST IFTE
003   NEXT
    » 37,5 Bytes
```

MAXL

```
001 « LIST→ 2 SWAP
      START > LAST IFTE
003   NEXT
    » 37,5 Bytes
```

```
1: { "X" "B" "ADAC"
"AA" }

MINL

1: "AA"

1: { 15 -2 29,3 7 }

MAXL

1: 29,3
```

Ralf Pfeiffer
Rubensstraße 5
5000 Köln 50

EST90

Zeile 1 von EST90 (1-5) CCD-Barcodes August Schorr

Zeile 2 von EST90 (6-9) CCD-Barcodes August Schorr

Zeile 3 von EST90 (10-13) CCD-Barcodes August Schorr

Zeile 4 von EST90 (14-16) CCD-Barcodes August Schorr

Zeile 5 von EST90 (17-17) CCD-Barcodes August Schorr

Zeile 6 von EST90 (18-21) CCD-Barcodes August Schorr

Zeile 7 von EST90 (22-24) CCD-Barcodes August Schorr

Zeile 8 von EST90 (25-26) CCD-Barcodes August Schorr

Zeile 9 von EST90 (27-31) CCD-Barcodes August Schorr

Zeile 10 von EST90 (32-38) CCD-Barcodes August Schorr

Zeile 11 von EST90 (39-39) CCD-Barcodes August Schorr

Zeile 12 von EST90 (40-44) CCD-Barcodes August Schorr

Zeile 13 von EST90 (45-52) CCD-Barcodes August Schorr

Zeile 14 von EST90 (53-55) CCD-Barcodes August Schorr

Zeile 15 von EST90 (56-61) CCD-Barcodes August Schorr

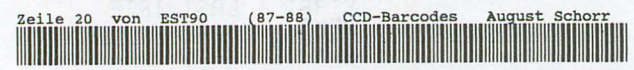
Zeile 16 von EST90 (62-68) CCD-Barcodes August Schorr

Zeile 17 von EST90 (69-74) CCD-Barcodes August Schorr

Zeile 18 von EST90 (75-77) CCD-Barcodes August Schorr

Zeile 19 von EST90 (78-86) CCD-Barcodes August Schorr

Zeile 20 von EST90 (87-88) CCD-Barcodes August Schorr



Zeile 21 von EST90 (89-93) CCD-Barcodes August Schorr

Zeile 22 von EST90 (94-100) CCD-Barcodes August Schorr

Zeile 23 von EST90 (101-105) CCD-Barcodes August Schorr

Zeile 24 von EST90 (106-107) CCD-Barcodes August Schorr

Zeile 25 von EST90 (108-112) CCD-Barcodes August Schorr

Zeile 26 von EST90 (113-117) CCD-Barcodes August Schorr

Zeile 27 von EST90 (118-118) CCD-Barcodes August Schorr

Zeile 28 von EST90 (119-121) CCD-Barcodes August Schorr

Zeile 29 von EST90 (122-127) CCD-Barcodes August Schorr

Zeile 30 von EST90 (128-134) CCD-Barcodes August Schorr

Zeile 31 von EST90 (135-143) CCD-Barcodes August Schorr

Zeile 32 von EST90 (144-151) CCD-Barcodes August Schorr

Zeile 33 von EST90 (152-155) CCD-Barcodes August Schorr

Zeile 34 von EST90 (156-159) CCD-Barcodes August Schorr

Zeile 35 von EST90 (160-160) CCD-Barcodes August Schorr

Zeile 36 von EST90 (161-161) CCD-Barcodes August Schorr

Zeile 37 von EST90 (162-168) CCD-Barcodes August Schorr

Zeile 38 von EST90 (169-176) CCD-Barcodes August Schorr

Zeile 39 von EST90 (177-185) CCD-Barcodes August Schorr

Zeile 40 von EST90 (186-190) CCD-Barcodes August Schorr

Zeile 41 von EST90 (191-192) CCD-Barcodes August Schorr

Zeile 42 von EST90 (193-199) CCD-Barcodes August Schorr

Zeile 43 von EST90 (200-208) CCD-Barcodes August Schorr

Zeile 44 von EST90 (209-214) CCD-Barcodes August Schorr

Zeile 45 von EST90 (215-217) CCD-Barcodes August Schorr

Zeile 46 von EST90 (218-220) CCD-Barcodes August Schorr

Zeile 47 von EST90 (221-223) CCD-Barcodes August Schorr

Zeile 48 von EST90 (224-225) CCD-Barcodes August Schorr

Zeile 49 von EST90 (226-232) CCD-Barcodes August Schorr

Zeile 50 von EST90 (233-234) CCD-Barcodes August Schorr

Zeile 51 von EST90 (235-243) CCD-Barcodes August Schorr

Zeile 52 von EST90 (244-251) CCD-Barcodes August Schorr

Zeile 53 von EST90 (252-255) CCD-Barcodes August Schorr

Zeile 54 von EST90 (256-264) CCD-Barcodes August Schorr

Zeile 55 von EST90 (265-272) CCD-Barcodes August Schorr

Zeile 56 von EST90 (273-283) CCD-Barcodes August Schorr

Zeile 57 von EST90 (284-290) CCD-Barcodes August Schorr

Zeile 58 von EST90 (291-297) CCD-Barcodes August Schorr

Zeile 59 von EST90 (298-305) CCD-Barcodes August Schorr

Zeile 60 von EST90 (306-312) CCD-Barcodes August Schorr

Zeile 61 von EST90 (313-321) CCD-Barcodes August Schorr

Zeile 62 von EST90 (322-330) CCD-Barcodes August Schorr

Zeile 63 von EST90 (331-338) CCD-Barcodes August Schorr

Zeile 64 von EST90 (339-347) CCD-Barcodes August Schorr

Zeile 65 von EST90 (348-358) CCD-Barcodes August Schorr

Zeile 66 von EST90 (359-369) CCD-Barcodes August Schorr

Zeile 67 von EST90 (370-378) CCD-Barcodes August Schorr

Zeile 68 von EST90 (379-389) CCD-Barcodes August Schorr

Zeile 69 von EST90 (390-401) CCD-Barcodes August Schorr

Zeile 70 von EST90 (402-410) CCD-Barcodes August Schorr

Zeile 71 von EST90 (411-420) CCD-Barcodes August Schorr

Zeile 72 von EST90 (421-427) CCD-Barcodes August Schorr

Zeile 73 von EST90 (428-438) CCD-Barcodes August Schorr

Zeile 74 von EST90 (439-439) CCD-Barcodes August Schorr

Zeile 75 von EST90 (440-443) CCD-Barcodes August Schorr

Zeile 76 von EST90 (444-447) CCD-Barcodes August Schorr

Zeile 77 von EST90 (448-449) CCD-Barcodes August Schorr

Zeile 78 von EST90 (450-456) CCD-Barcodes August Schorr

Zeile 79 von EST90 (457-462) CCD-Barcodes August Schorr

Zeile 80 von EST90 (463-468) CCD-Barcodes August Schorr

Zeile 81 von EST90 (469-469) CCD-Barcodes August Schorr

Zeile 82 von EST90 (470-473) CCD-Barcodes August Schorr

Zeile 83 von EST90 (474-481) CCD-Barcodes August Schorr

Zeile 84 von EST90 (482-484) CCD-Barcodes August Schorr

Zeile 85 von EST90 (485-488) CCD-Barcodes August Schorr

Zeile 86 von EST90 (489-493) CCD-Barcodes August Schorr

Zeile 87 von EST90 (494-496) CCD-Barcodes August Schorr

Zeile 88 von EST90 (497-499) CCD-Barcodes August Schorr

Zeile 89 von EST90 (500-501) CCD-Barcodes August Schorr

Zeile 90 von EST90 (502-505) CCD-Barcodes August Schorr

Zeile 91 von EST90 (506-514) CCD-Barcodes August Schorr

Zeile 92 von EST90 (515-518) CCD-Barcodes August Schorr

Zeile 93 von EST90 (519-526) CCD-Barcodes August Schorr

Zeile 94 von EST90 (527-533) CCD-Barcodes August Schorr

Zeile 95 von EST90 (534-542) CCD-Barcodes August Schorr

Zeile 96 von EST90 (543-545) CCD-Barcodes August Schorr

Zeile 97 von EST90 (546-549) CCD-Barcodes August Schorr

Zeile 98 von EST90 (550-550) CCD-Barcodes August Schorr

Zeile 99 von EST90 (551-554) CCD-Barcodes August Schorr

Zeile 100 von EST90 (555-557) CCD-Barcodes August Schorr

Zeile 101 von EST90 (558-561) CCD-Barcodes August Schorr

Zeile 102 von EST90 (562-565) CCD-Barcodes August Schorr

Zeile 103 von EST90 (566-573) CCD-Barcodes August Schorr

Zeile 104 von EST90 (574-577) CCD-Barcodes August Schorr

Zeile 105 von EST90 (578-583) CCD-Barcodes August Schorr

Zeile 106 von EST90 (584-591) CCD-Barcodes August Schorr

Zeile 107 von EST90 (592-597) CCD-Barcodes August Schorr

Zeile 108 von EST90 (598-603) CCD-Barcodes August Schorr

Zeile 109 von EST90 (604-608) CCD-Barcodes August Schorr

Zeile 110 von EST90 (609-615) CCD-Barcodes August Schorr

Zeile 111 von EST90 (616-622) CCD-Barcodes August Schorr

Zeile 112 von EST90 (623-628) CCD-Barcodes August Schorr

Zeile 113 von EST90 (629-634) CCD-Barcodes August Schorr

Zeile 114 von EST90 (635-641) CCD-Barcodes August Schorr

Zeile 115 von EST90 (642-648) CCD-Barcodes August Schorr

Zeile 116 von EST90 (649-655) CCD-Barcodes August Schorr

Zeile 117 von EST90 (656-658) CCD-Barcodes August Schorr

Zeile 118 von EST90 (659-665) CCD-Barcodes August Schorr

Zeile 119 von EST90 (666-672) CCD-Barcodes August Schorr

Zeile 120 von EST90 (673-678) CCD-Barcodes August Schorr

Zeile 121 von EST90 (679-678) CCD-Barcodes August Schorr

CODE

Zeile 1 von CODE (1-5) CCD-Barcodes Michael Schilli

Zeile 2 von CODE (6-13) CCD-Barcodes Michael Schilli

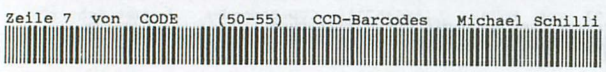
Zeile 3 von CODE (14-20) CCD-Barcodes Michael Schilli

Zeile 4 von CODE (21-28) CCD-Barcodes Michael Schilli

Zeile 5 von CODE (29-38) CCD-Barcodes Michael Schilli

Zeile 6 von CODE (39-49) CCD-Barcodes Michael Schilli

Zeile 7 von CODE (50-55) CCD-Barcodes Michael Schilli



Zeile 8 von CODE (56-66) CCD-Barcodes Michael Schilli

Zeile 9 von CODE (67-76) CCD-Barcodes Michael Schilli

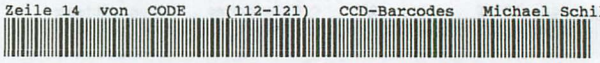
Zeile 10 von CODE (77-83) CCD-Barcodes Michael Schilli

Zeile 11 von CODE (84-92) CCD-Barcodes Michael Schilli

Zeile 12 von CODE (93-104) CCD-Barcodes Michael Schilli

Zeile 13 von CODE (105-111) CCD-Barcodes Michael Schilli

Zeile 14 von CODE (112-121) CCD-Barcodes Michael Schilli



Zeile 15 von CODE (122-129) CCD-Barcodes Michael Schilli

Zeile 16 von CODE (130-138) CCD-Barcodes Michael Schilli

Zeile 17 von CODE (139-146) CCD-Barcodes Michael Schilli

Zeile 18 von CODE (147-157) CCD-Barcodes Michael Schilli

Zeile 19 von CODE (158-165) CCD-Barcodes Michael Schilli

Zeile 20 von CODE (166-174) CCD-Barcodes Michael Schilli

Zeile 21 von CODE (175-180) CCD-Barcodes Michael Schilli

Zeile 22 von CODE (181-189) CCD-Barcodes Michael Schilli

Zeile 23 von CODE (190-195) CCD-Barcodes Michael Schilli

Zeile 24 von CODE (196-202) CCD-Barcodes Michael Schilli

Zeile 25 von CODE (203-210) CCD-Barcodes Michael Schilli

Zeile 26 von CODE (211-220) CCD-Barcodes Michael Schilli

Zeile 27 von CODE (221-227) CCD-Barcodes Michael Schilli

Zeile 28 von CODE (228-234) CCD-Barcodes Michael Schilli

Zeile 29 von CODE (235-247) CCD-Barcodes Michael Schilli

Zeile 30 von CODE (248-252) CCD-Barcodes Michael Schilli

Zeile 31 von CODE (253-261) CCD-Barcodes Michael Schilli

Zeile 32 von CODE (262-268) CCD-Barcodes Michael Schilli

Zeile 33 von CODE (269-277) CCD-Barcodes Michael Schilli


Zeile 34 von CODE (278-286) CCD-Barcodes Michael Schilli


Zeile 35 von CODE (287-292) CCD-Barcodes Michael Schilli


Zeile 36 von CODE (293-300) CCD-Barcodes Michael Schilli


Zeile 37 von CODE (301-308) CCD-Barcodes Michael Schilli


Zeile 38 von CODE (309-314) CCD-Barcodes Michael Schilli


Zeile 39 von CODE (315-323) CCD-Barcodes Michael Schilli


Zeile 40 von CODE (324-330) CCD-Barcodes Michael Schilli


Zeile 41 von CODE (331-338) CCD-Barcodes Michael Schilli


Zeile 42 von CODE (339-351) CCD-Barcodes Michael Schilli


Zeile 43 von CODE (352-360) CCD-Barcodes Michael Schilli


Zeile 44 von CODE (361-370) CCD-Barcodes Michael Schilli


Zeile 45 von CODE (371-380) CCD-Barcodes Michael Schilli


Zeile 46 von CODE (381-392) CCD-Barcodes Michael Schilli


Zeile 47 von CODE (393-403) CCD-Barcodes Michael Schilli


Zeile 48 von CODE (404-411) CCD-Barcodes Michael Schilli


Zeile 49 von CODE (412-420) CCD-Barcodes Michael Schilli


Zeile 50 von CODE (421-426) CCD-Barcodes Michael Schilli


Zeile 51 von CODE (427-438) CCD-Barcodes Michael Schilli


Zeile 52 von CODE (439-448) CCD-Barcodes Michael Schilli


Zeile 53 von CODE (449-454) CCD-Barcodes Michael Schilli


Zeile 54 von CODE (455-465) CCD-Barcodes Michael Schilli


Zeile 55 von CODE (466-470) CCD-Barcodes Michael Schilli


Zeile 56 von CODE (471-474) CCD-Barcodes Michael Schilli


Zeile 57 von CODE (475-482) CCD-Barcodes Michael Schilli


Zeile 58 von CODE (483-487) CCD-Barcodes Michael Schilli


Zeile 59 von CODE (488-496) CCD-Barcodes Michael Schilli


Zeile 60 von CODE (497-506) CCD-Barcodes Michael Schilli


Zeile 61 von CODE (507-515) CCD-Barcodes Michael Schilli


Zeile 62 von CODE (516-523) CCD-Barcodes Michael Schilli


Zeile 63 von CODE (524-535) CCD-Barcodes Michael Schilli


Zeile 64 von CODE (536-545) CCD-Barcodes Michael Schilli


Zeile 65 von CODE (546-553) CCD-Barcodes Michael Schilli


Zeile 66 von CODE (554-561) CCD-Barcodes Michael Schilli


Zeile 67 von CODE (562-570) CCD-Barcodes Michael Schilli


Zeile 68 von CODE (571-579) CCD-Barcodes Michael Schilli


Zeile 69 von CODE (580-588) CCD-Barcodes Michael Schilli


Zeile 70 von CODE (589-600) CCD-Barcodes Michael Schilli


Zeile 71 von CODE (601-607) CCD-Barcodes Michael Schilli


Zeile 72 von CODE (608-615) CCD-Barcodes Michael Schilli


Zeile 73 von CODE (616-625) CCD-Barcodes Michael Schilli

Zeile 74 von CODE (626-631) CCD-Barcodes Michael Schilli

Zeile 75 von CODE (632-637) CCD-Barcodes Michael Schilli

Zeile 76 von CODE (638-644) CCD-Barcodes Michael Schilli

Zeile 77 von CODE (645-648) CCD-Barcodes Michael Schilli

Zeile 78 von CODE (649-652) CCD-Barcodes Michael Schilli

Zeile 79 von CODE (653-656) CCD-Barcodes Michael Schilli

Zeile 80 von CODE (657-662) CCD-Barcodes Michael Schilli

Zeile 81 von CODE (663-665) CCD-Barcodes Michael Schilli

LR

Zeile 1 von LR (1-7) CCD-Barcodes Michael Schilli

Zeile 2 von LR (8-15) CCD-Barcodes Michael Schilli

Zeile 3 von LR (16-20) CCD-Barcodes Michael Schilli

Zeile 4 von LR (21-30) CCD-Barcodes Michael Schilli

Zeile 5 von LR (31-39) CCD-Barcodes Michael Schilli

Zeile 6 von LR (40-46) CCD-Barcodes Michael Schilli

Zeile 7 von LR (47-57) CCD-Barcodes Michael Schilli

Zeile 8 von LR (58-67) CCD-Barcodes Michael Schilli

Zeile 9 von LR (68-79) CCD-Barcodes Michael Schilli

Zeile 10 von LR (80-91) CCD-Barcodes Michael Schilli

Zeile 11 von LR (92-100) CCD-Barcodes Michael Schilli

Zeile 12 von LR (101-112) CCD-Barcodes Michael Schilli

Zeile 13 von LR (113-120) CCD-Barcodes Michael Schilli

Zeile 14 von LR (121-128) CCD-Barcodes Michael Schilli

Zeile 15 von LR (129-140) CCD-Barcodes Michael Schilli

Zeile 16 von LR (141-150) CCD-Barcodes Michael Schilli

Zeile 17 von LR (151-157) CCD-Barcodes Michael Schilli

Zeile 18 von LR (158-164) CCD-Barcodes Michael Schilli

Zeile 19 von LR (165-167) CCD-Barcodes Michael Schilli

DE

Zeile 1 von DE (1-5) CCD-Barcodes Dr.G.Heilmann

Zeile 2 von DE (6-14) CCD-Barcodes Dr.G.Heilmann

Zeile 3 von DE (15-16) CCD-Barcodes Dr.G.Heilmann

Zeile 4 von DE (17-18) CCD-Barcodes Dr.G.Heilmann

Zeile 5 von DE (19-22) CCD-Barcodes Dr.G.Heilmann

Zeile 6 von DE (23-33) CCD-Barcodes Dr.G.Heilmann

Zeile 7 von DE (34-42) CCD-Barcodes Dr.G.Heilmann

Zeile 8 von DE (43-51) CCD-Barcodes Dr.G.Heilmann

Zeile 9 von DE (52-61) CCD-Barcodes Dr.G.Heilmann

Zeile 10 von DE (62-67) CCD-Barcodes Dr.G.Heilmann

Zeile 11 von DE (68-71) CCD-Barcodes Dr.G.Heilmann

Zeile 12 von DE (72-78) CCD-Barcodes Dr.G.Heilmann

Zeile 13 von DE (79-85) CCD-Barcodes Dr.G.Heilmann

Zeile 14 von DE (86-92) CCD-Barcodes Dr.G.Heilmann

Zeile 15 von DE (93-98) CCD-Barcodes Dr.G.Heilmann

Zeile 16 von DE (99-102) CCD-Barcodes Dr.G.Heilmann

Zeile 17 von DE (103-107) CCD-Barcodes Dr.G.Heilmann

Zeile 18 von DE (108-114) CCD-Barcodes Dr.G.Heilmann

Zeile 19 von DE (115-119) CCD-Barcodes Dr.G.Heilmann

Zeile 20 von DE (120-126) CCD-Barcodes Dr.G.Heilmann

Zeile 21 von DE (127-132) CCD-Barcodes Dr.G.Heilmann

Zeile 22 von DE (133-141) CCD-Barcodes Dr.G.Heilmann

Zeile 23 von DE (142-147) CCD-Barcodes Dr.G.Heilmann

Zeile 24 von DE (148-157) CCD-Barcodes Dr.G.Heilmann

Zeile 25 von DE (158-166) CCD-Barcodes Dr.G.Heilmann

Zeile 26 von DE (167-177) CCD-Barcodes Dr.G.Heilmann

Zeile 27 von DE (178-185) CCD-Barcodes Dr.G.Heilmann

Zeile 28 von DE (186-198) CCD-Barcodes Dr.G.Heilmann

Zeile 29 von DE (199-209) CCD-Barcodes Dr.G.Heilmann

Zeile 30 von DE (210-219) CCD-Barcodes Dr.G.Heilmann

Zeile 31 von DE (220-231) CCD-Barcodes Dr.G.Heilmann

Zeile 32 von DE (232-244) CCD-Barcodes Dr.G.Heilmann

Zeile 33 von DE (245-254) CCD-Barcodes Dr.G.Heilmann

Zeile 34 von DE (255-265) CCD-Barcodes Dr.G.Heilmann

Zeile 35 von DE (266-271) CCD-Barcodes Dr.G.Heilmann

Zeile 36 von DE (272-275) CCD-Barcodes Dr.G.Heilmann

Zeile 37 von DE (276-284) CCD-Barcodes Dr.G.Heilmann

Zeile 38 von DE (285-288) CCD-Barcodes Dr.G.Heilmann

Zeile 39 von DE (289-294) CCD-Barcodes Dr.G.Heilmann

Zeile 40 von DE (295-302) CCD-Barcodes Dr.G.Heilmann

Zeile 41 von DE (303-308) CCD-Barcodes Dr.G.Heilmann

Zeile 42 von DE (309-313) CCD-Barcodes Dr.G.Heilmann

Zeile 43 von DE (314-321) CCD-Barcodes Dr.G.Heilmann

Zeile 44 von DE (322-329) CCD-Barcodes Dr.G.Heilmann

Zeile 45 von DE (330-339) CCD-Barcodes Dr.G.Heilmann

Zeile 46 von DE (340-347) CCD-Barcodes Dr.G.Heilmann

Zeile 47 von DE (348-356) CCD-Barcodes Dr.G.Heilmann

Zeile 48 von DE (357-365) CCD-Barcodes Dr.G.Heilmann

Zeile 49 von DE (366-372) CCD-Barcodes Dr.G.Heilmann

Zeile 50 von DE (373-376) CCD-Barcodes Dr.G.Heilmann

Zeile 51 von DE (377-381) CCD-Barcodes Dr.G.Heilmann

Zeile 52 von DE (382-391) CCD-Barcodes Dr.G.Heilmann

Zeile 53 von DE (392-402) CCD-Barcodes Dr.G.Heilmann

Zeile 54 von DE (403-409) CCD-Barcodes Dr.G.Heilmann

Zeile 55 von DE (410-415) CCD-Barcodes Dr.G.Heilmann

Zeile 56 von DE (416-421) CCD-Barcodes Dr.G.Heilmann

Zeile 57 von DE (422-430) CCD-Barcodes Dr.G.Heilmann

Zeile 58 von DE (431-441) CCD-Barcodes Dr.G.Heilmann

Zeile 59 von DE (442-449) CCD-Barcodes Dr.G.Heilmann

Zeile 60 von DE (450-453) CCD-Barcodes Dr.G.Heilmann

SX

Zeile 1 von SX (1-5) CCD-Barcodes Dr.G.Heilmann

Zeile 2 von SX (6-11) CCD-Barcodes Dr.G.Heilmann

Zeile 3 von SX (12-17) CCD-Barcodes Dr.G.Heilmann

Zeile 4 von SX (18-26) CCD-Barcodes Dr.G.Heilmann

Zeile 5 von SX (27-34) CCD-Barcodes Dr.G.Heilmann

Zeile 6 von SX (35-42) CCD-Barcodes Dr.G.Heilmann

Zeile 7 von SX (43-48) CCD-Barcodes Dr.G.Heilmann

Zeile 8 von SX (49-50) CCD-Barcodes Dr.G.Heilmann

Zeile 9 von SX (51-54) CCD-Barcodes Dr.G.Heilmann

Zeile 10 von SX (55-59) CCD-Barcodes Dr.G.Heilmann

Zeile 11 von SX (60-67) CCD-Barcodes Dr.G.Heilmann

Zeile 12 von SX (68-73) CCD-Barcodes Dr.G.Heilmann

Zeile 13 von SX (74-82) CCD-Barcodes Dr.G.Heilmann

Zeile 14 von SX (83-84) CCD-Barcodes Dr.G.Heilmann

Zeile 15 von SX (85-93) CCD-Barcodes Dr.G.Heilmann

Zeile 16 von SX (94-94) CCD-Barcodes Dr.G.Heilmann

Zeile 17 von SX (95-98) CCD-Barcodes Dr.G.Heilmann

HG

Zeile 1 von HG (1-5) CCD-Barcodes Dr.G.Heilmann

Zeile 2 von HG (6-12) CCD-Barcodes Dr.G.Heilmann

Zeile 3 von HG (13-20) CCD-Barcodes Dr.G.Heilmann

Zeile 4 von HG (21-32) CCD-Barcodes Dr.G.Heilmann

Zeile 5 von HG (33-41) CCD-Barcodes Dr.G.Heilmann

Zeile 6 von HG (42-43) CCD-Barcodes Dr.G.Heilmann

Zeile 7 von HG (44-45) CCD-Barcodes Dr.G.Heilmann

Verkaufe HP-41CX mit Kartenleser, IR-Modul, 2 x X-Memory, Financial I, Statistic I,Mathematic I zum Preis von DM 850.
Dr. Klaus Tebbe, Tel. 0208 / 42 05 05.

Verkaufe für HP-41 und HP-IL: RS 232 Schnittstelle HP 82164A, Extended I/O HP 82183A, HP-IL Devel-ROM HP 5061-5251, jeweils mit Originalhandbüchern und weiterer Literatur.
Heinz-Walter Schmidt, Tel. 02845 / 6204.

Wer ist Funkamateur und CCD-Mitglied ? Kontakt gesucht. Bin QRV im VHF-Bereich, besonders auf 145.55 MHz. Callsign DB8PP, name Armin, IARV loc: JN39WX. Armin Thesen, Bingerbrück.

Suche HP-71B. Heubner, Tel. 0511 / 85 30 11.

Suche HP-71 + IL-Modul + Speichermodul + CMI-Ramdisk mit RS-232-Anschluß.
Dr. Hanspeter Waldmann, Erlenweg 3, 7951 Dettingen, Tel 07354 / 8729.

Biete an: CCD-Modul + Handbuch 120.-, evtl. im Tausch mit Barcodeleser.
Ingo Schmale, Steinmeisterstraße 8, 4980 Bünde, Tel. 05223 / 149 57.

HP-71B System (Preise VB):

- HP-71 mit IL-Modul	DM 700
- Mathematik ROM	DM 160
- Curve Fit Rom	DM 160
- Statistik ROM	DM 160
- Forth/Assembler ROM	DM 160
- 96 kRam für Kartenleser-Port	DM 600
- Kassetenlaufwerk	DM 500

Thor Gehrmann, Tel 02339 / 3963 (abends).

HP VECTRA ES 12 (AT-kompatibel), 12 MHz, 640 k RAM, Diskettenlaufwerk 1.2 MB, Festplatte 20 MB, Schnittstellen: seriell, parallel, HP-HIL, HP-HIL-Maus, HP-DOS 3.2, Windows, HP-EGA Karte (256 k) und HP-EGA Monitor (14 Zoll, color). Preis VS.
Thor Gehrmann, Tel 02339 / 3963 (abends).

HP DeskJet plus mit Times Roman Cartridge (HP 22706 R), VB 2200 DM.
Thor Gehrmann, Tel 02339 / 3963.

HP 150 Komplettausstattung: **HP 150-2, 512 k**, Touch-Screen-Option, Hard-Disc 20 MB mit 3,5 Zoll Floppydrive, viel Software worunter **Memomaker**, Personal Card File und (original verpackt:) **Picture Perfect**; **ThinkJet-IB-Printer**. Zusammen DM 3700 VB.
Gysbert Hagemann, Alter Weg 1, D-6653 Blieskastel 2, Tel. 06842 / 3041 o. 3042 (geschäftszeitlich), 06842 / 2805 (privat).

Suche Barcodeleser für HP-41.
Oliver Bonn Tel. 06181 / 213 67

Speicher 256 KB RAM für HP DeskJet plus DM 349 ; original **Intel Above Board 286** Speichererweiterung extended und/oder expanded Memory für AT-Rechner. Ideal für z.B. MS-Word, Excel, Windows, AutoCad, 1-2-3 ..., bis 2 MB bestückbar, mit 512 KB bestückt DM 798, bzw. mit 1MB bestückt DM 929.
Toni Lerchenfeld, Tel 08323 / 7323, Fax: 08323 / 7717 oder Tel 0043 / 512 / 89 46 11.

Verkaufe: IL-Karte für IBM-PC (HP 82973) incl. IL-Link-Programm (HP 82477) DM 350.
G. Rowold, 04431 / 5999.

Suche für HP 82240A Printer und IR-Modul HP 82242A ein deutsches Handbuch oder eine Kopie.
Gunther Kemény, Am Schafgarten 3, 6724 Dudenhofen, Tel 06262 / 922 57 (Sa-So), 0631 / 254 36 (Mo-Fr).

Verkaufe: Digitalkassettenlaufwerk HP 82161A mit 10 Kassetten, DM 400 VB; Diskettengerät HP 9114B, DM 500 VB; HP-71B mit IL-Modul, DM 600 VB; Translator-Modul für HP-71B, DM 100; 2 Speichererweiterungsmodule für HP-71B (4 KB) je DM 40. Horst Ziegler, Tel. 040 / 790 56 72.

Verkaufe für HP-71: Forth-Assembler Modul + Buch "Programmieren in Forth" für DM 120; **Suche** deutsches Handbuch für Math-Modul.
Klaus Bendfeldt, Tel. 02751 / 2114.

Suche für HP-71:
- Speichererweiterung (Front port und Kartenleser-Schacht)
- Literatur und Erfahrungsaustausch (Clubmagazine ausländischer Clubs, Hardware-Erweiterungen, IL-Interface-Kit 82166, Mailbox und BTX mit HP-71).

Joachim Kühn, K.-Kreiten-Straße 5, 5300 Bonn 1,
Tel. 9228 / 65 00 89.

Suche Mitfahrgelegenheit von Bielefeld nach Frankfurt zur Mitgliederversammlung am 21.4. 90.
Matthias Rabe, Tel. 0521 / 32 44 74.

Verkaufe HP-71 (neueste Version) für DM 600, Diskettenlaufwerk (9114 A) mit W&W-Netzteil für DM 550 VB. Hp-28 Literatur: "HP Insights; Principles and Programming of the HP-28C/S" von W. Wickes für DM 60.
Jochen Haas, Tel. 0221 / 519870.

Zu verkaufen:

HP-41 Cx	DM 400
ThinkJet mit 3500 Blatt Papier, 2 Tintenpatronen und Acrylständer	DM 800
CT-Port-Extender	DM 50

Dr. Martin Hochenegger, Heidelberger Landstraße 97, 6100 Darmstadt, Tel. 06151 / 523 94.

Verkaufe:

IL-Drucker mit neuem AKKU	DM 350
IL-Laufwerk mit neuem Akku und 12 Magnetbändern	DM 450
IL-Modul	DM 100
CCD-Modul (Version A)	DM 100
IR-Modul	DM 100
Infrarot-Interface für PC/XT und AT	DM 150
2 X-Memory-Module	DM 200
Plotter-Modul	DM 75
Auto/Dup-Modul	DM 25
Barcode-Lesestift	DM 125
Netzteil	DM 20
Bücher:	
Best of PRISMA + Magnetband	DM 10
Barcodes mit dem HP-IL-System	DM 20
HP-41 in Industrie und Handwerk	DM 15

Oliver Rieck, Bonn, Tel. 0228 / 36 37 27 abends.

Verkaufe HP-71, Modell mit bedruckter Aluminium-Tastaturmaske, Ver# 1BBBB, mit Benutzerhandbuch und Tasche DM 470.
Matthias Kühn, Tel. 06106 / 719 07.

Zu verkaufen:

Time Modul 82182 AD	DM 50
Thermodrucker 82143 AD	DM 400
IL-Modul 82160 AD	DM 160
Quad Memory 82170 AD	50
Finanzpaket, deutsch	DM 30
Statistikpaket, deutsch	DM 40
Kassettenlaufwerk 82161 A, neuw.	DM 400
Translator Pac 41 für HP-71	DM 120
Kartenleser 82400A für HP-71	DM 150
GPIO-Interface für Serie 80	DM 250
IL-Development Modul	DM 120
9114B Floppy, gebr.	DM 780
2225B ThinkJet, gebr.	DM550
HP 75 D mit 8k Ram, Lesestift, Expansion Pod (64k Ramdisk, Modem), gebr.	DM 800

Alle Geräte und Teile, bis auf die gekennzeichneten, neu und originalverpackt.
Thomas Mareis, Cranachstraße 1, 800 München 40, Tel. 089 / 129 56 65.

Verkaufe HP 75 C, Preis VS.
Bernd Podbielski Tel. 0571 / 247 73.

Verkaufe komplett: HP-41 CX Turbo, Accu und Ladegerät, CCD-Modul, X-Memory, Mathe-Modul, Opr. Lesestift, Magnetkartenleser, 90 Magnetkarten, Programmsammlungen, Literatur, PRISMA 1/82-1/90, Best of PRISMA, PPC-Journale; Gesamtpreis DM 1600.
Andreas Eschman, Lahnstraße 2, 6096 Raunheim,
Tel 06142 / 466 42.

Subskriptionsangebot von W&W: Wer den **HP-48 SX** mit Hinweis auf den CCD und mit seiner Mitgliedsnummer bestellt, erhält zusätzlich den kleinsten jemals von HP gebauten Taschenrechner.

Impressum

Titel:
PRISMA

Herausgeber:
CCD - Computerclub Deutschland e.V.

Postfach 11 04 11
Schwalbacher Straße 50
6000 Frankfurt 1

Verantwortlicher Redakteur:
Alf-Norman Tietze (ant)

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

Herstellung:
CCD e.V.

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

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

Anzeigenpreise:
Es gilt unsere Anzeigenpreisliste 3 vom Juni 1987

Erscheinungsweise:
PRISMA erscheint jeden 2. Monat

Auflage:
2500

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

SERVICELEISTUNGEN

BEST OF PRISMA

Schutzgebühr: 30,- DM

Nachsendedienst PRISMA

Schutzgebühr: 5,- DM pro Heft für Jahrgänge 1982-1986
10,- DM pro Heft für Jahrgänge ab 1987

Inhaltsverzeichnis PRISMA

Schutzgebühr: 3,- DM in Briefmarken

Programmbibliothek HP71

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

MS-DOS Inhaltsverzeichnis

Kann durch das Einsenden einer formatierten 360 kB oder 1,2 MB 5,25"-Diskette oder einer formatierten 720 kB oder 1,44 MB 3,5"-Diskette angefordert werden.

ATARI-Inhaltsverzeichnis

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

UPLE

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

Programme aus BEST OF PRISMA

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

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

(069) 76 59 12

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

(SAFU = Selbst Adressierter Freiumschlag)

CLUBADRESSEN

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

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

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

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

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

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

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

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

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

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

Regionalgruppe Köln
Jochen Haas (2874), Roßstr. 27, 5000 Köln 30, (0221) 51 98 70

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

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

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

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

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

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

E-Technik
Werner Meschede (2670), Sorpestraße 4, 5788 Siedlingshausen
Grabau GR7 Interface
Holger von Stillfried (2641)
Am Langdiek 13, 2000 Hamburg 61

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

HP-71 Assembler (LEX-Files)
Matthias Rabe (2062), Teichsheide 13, 4800 Bielefeld

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

Naturwissenschaften
Thor Gehrmann (3423)
Hobeuken 18, 4322 Spockhövel 2, (02339) 39 63

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

Vermessungswesen
Ulrich Kulle (2719), Memeler Straße 26, 3000 Hannover 51,
(0511) 604 27 28

Programmbibliothek HP71
Henry Schimmer (786), Homburger Landstr. 63,
6000 Frankfurt 50

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

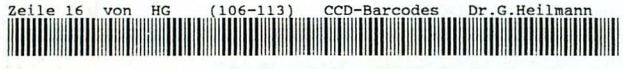
Postvertriebsstück
Gebühr bezahlt

D 2856 F

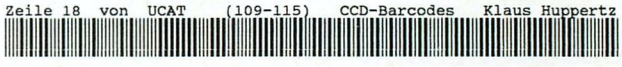
CCD – Computerclub Deutschland e.
Schwalbacherstraße 50
D-6000 Frankfurt am Main 1

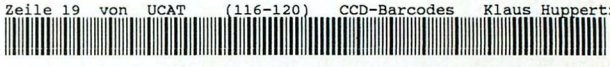
CCD
ISSN 0176-8735
PRISMA

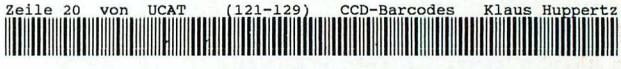
Januar / Februar 1990 Nr. 1



UCAT







Nächstes

Ortstreffen in Köln:

MS-DOS, ATARI , BTX und Datenfernübertragung stehen beim Treffen der PC-Anwender im Mittelpunkt. Termin: 16. Juni 1990, ab 13 Uhr im Vereinshaus der Humboldt-Siedlung, Frankfurter Straße 855, 5000 Köln 91 (Ostheim)

Verantwortlich und Ansprechpartner:

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

und

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

Regionalgruppe Köln

(Taschencomputer)

Die Regionalgruppe Köln trifft sich bis auf weiteres nicht mehr im Galerietreff Leverkusen. Der neue Treffpunkt wir in der nächsten PRISMA-Ausgabe veröffentlicht.

Infos von: Jochen Hass, Tel. 0221 / 519870.