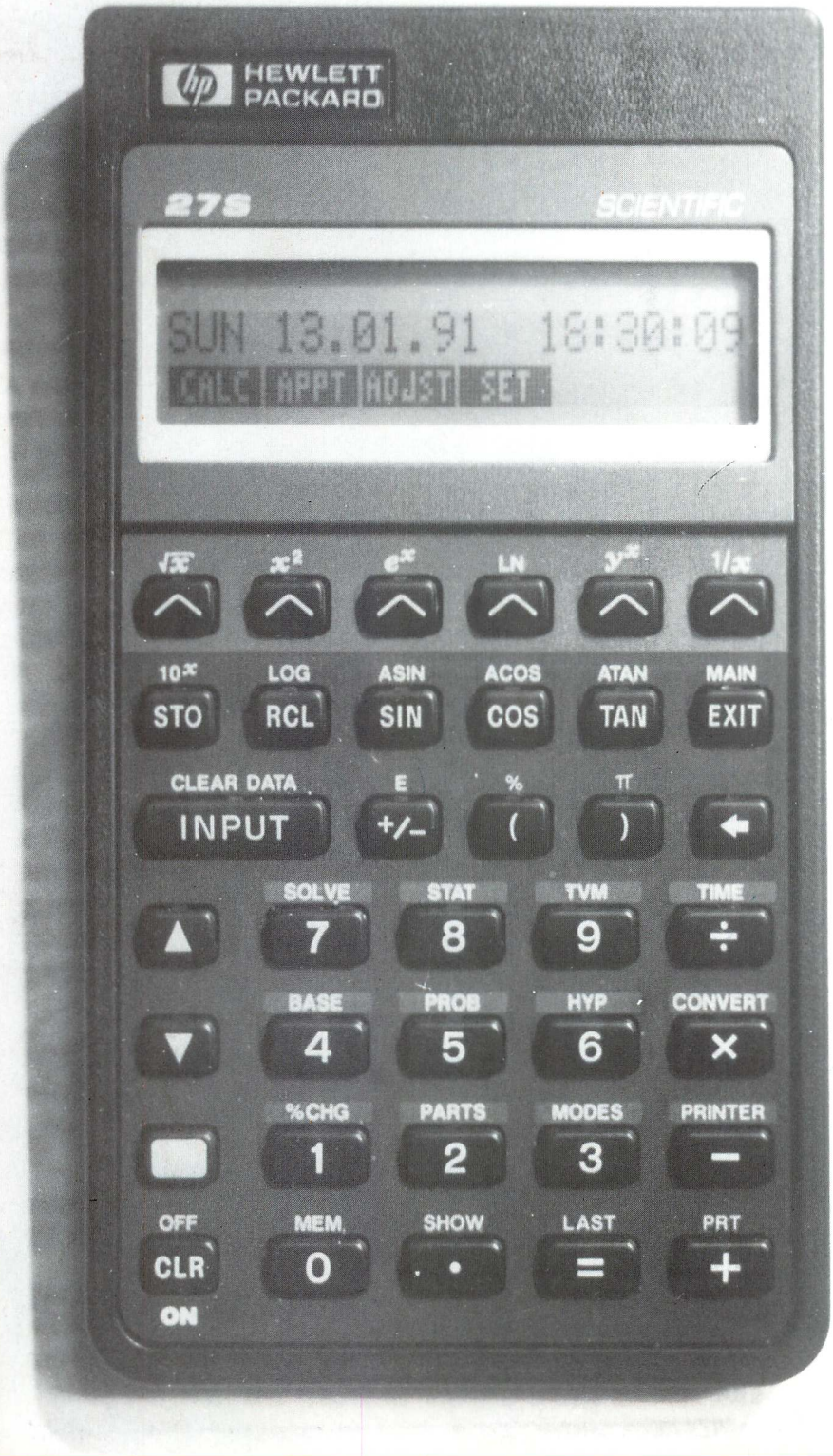
PRISMA

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

September-Dezember 1990 Nr. 5/6 **D 2856 F**



SERVICELEISTUNGEN

BEST OF PRISMA
Schutzgebühr: 30,- DM

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

Inhaltsverzeichnis PRISMA
Schutzgebühr: 3,- DM in Briefmarken

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

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

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

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

Programme aus BEST OF PRISMA

a) Eine Kopie der Programme von BEST OF PRISMA auf Kassette erfordert das Beilegen einer Leerkassette und eines SAFU.

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

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

(069) 76 59 12

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

CLUBADRESSEN

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Inhalt

Informationsbedürfnisse

Ein wesentliches Einsatzgebiet der EDV ist das Abspeichern und Wiederfinden von beliebigen Informationen. Das Recherchieren und (schnelle) Wiederfinden einer gesuchten Information ist dabei überwiegend von größerer Bedeutung, als das eigentliche Abspeichern - obwohl natürlich auf keinen Fall etwas wiedergefunden werden kann, was vorher nicht gespeichert bzw. erfaßt worden ist.

So lapidar diese Beschreibung der Datenverwaltung auch ist, so treffend drückt sie aber das Grundprinzip für jegliche Datenbankanwendung aus. Auch wenn man genau weiß, daß die gesuchte Information im Computer abgespeichert ist, sind Suchen und Finden trotzdem nicht zwangsläufig miteinander verbunden. Ob ein gesuchter Begriff nämlich schnell bzw. überhaupt wiedergefunden werden kann, hängt wesentlich von der Datenbankorganisation ab. Und dort beginnt bereits die hohe Kunst des "Datenbanking", über die ohne Mühe ganze Bücher geschrieben werden können und auch geschrieben worden sind.

Unabhängig von diesen datenbanktheoretischen Betrachtungen sind in diesem Prisma gleich zwei Programme veröffentlicht, mit denen sich auf den Taschencomputern HP48SX und HP71B Daten und Informationen erfassen, suchen, verwalten und hoffentlich auch finden lassen. Wem übrigens im einen oder anderen Fall das Abtippen eines umfangreichen Programmlistings zu mühsam und vor allem auch zu fehlerträchtig ist, der kann über die zuständige Clubbibliothek (siehe "Serviceleistungen") gegen Einsendung eines Datenträgers mit Freiumschlag das oder die entsprechenden Programme fertig zugesendet bekommen.

Ebenfalls zur Datenbankthematik gehört ein Kurzbericht über die Nantucket Clipper-Entwickler-Konferenz Dezember 1990 in Köln. Clipper ist ein dBASE-ähnlicher Compiler zur Programmierung von Datenbankanwendungen. Dort wurde Clipper-Programmierern die Möglichkeit geboten, sich umfassend über verschiedenste Fachgebiete der Datenbankentwicklung zu informieren.

Auch wir - die CCD-Mitglieder, die PRISMA-Leser - haben "Informationsbedürfnisse", und die Redaktion versucht diesem Bedarf weitgehend gerecht zu werden. Für den Jahresschluß 1990 haben wir uns deshalb zu einer starken Doppelnummer entschieden. Zum Zeitpunkt der Drucklegung dieses Heftes wird das Jahr 1991 bereits begonnen haben und wir wünschen allen Mitgliedern einen guten Start in das Neue Jahr.

Alf-Norman Tietze
(Chefredakteur)

Übrigens: Die PRISMA-Redaktion sucht noch einen zusätzlichen Mitarbeiter aus dem Frankfurter Raum für den Bereich HP71. Wer sich mit diesem Taschencomputer etwas auskennt und Spaß daran hätte, Artikel darüber aufzubereiten, der möchte sich bitte bei Alf-Norman Tietze (069/346240), Martin Meyer (06196/23150) oder Dieter Wolf (069/765912) melden.

Einladung zur
Mitgliederversammlung

Liebe Clubfreunde,
hiermit laden wir zur ordentlichen Jahreshauptversammlung (Mitgliederversammlung) des Jahres 1991 ein.

Zeit: 6. April 1991, 11:00 Uhr

Ort: Frankfurter Saal im Intercity-Restaurant, Hauptbahnhof Frankfurt/M

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. Neuwahlen des Beirates
8. Haushaltsplan 1991
9. PRISMA
10. Anträge
11. Verschiedenes

Wie in jedem Jahr, so findet auch 1991 wieder eine ordentliche Mitgliederversammlung statt. Wegen der anstehenden Feiertage, aber auch wegen des Termins der CEBIT (13. - 20.3.) hat der Vorstand den Termin auf den 6. April bestimmt, um möglichst allen Clubmitgliedern die Teilnahme zu ermöglichen.

Dieses Jahr stehen wieder Beirats-Wahlen an; der Beirat soll nach der Satzung den Vorstand beraten und bei Entscheidungen von erheblicher Tragweite vom Vorstand vorher gehört werden. Die Beiratsmitglieder werden deshalb auch zu den Vorstandssitzungen eingeladen.

Der Vorstand hofft, daß wegen dieser wichtigen Wahl und damit eine breite und anregende Diskussion über die Arbeit unseres Clubs stattfinden kann, auf eine rege Beteiligung an der Mitgliederversammlung.

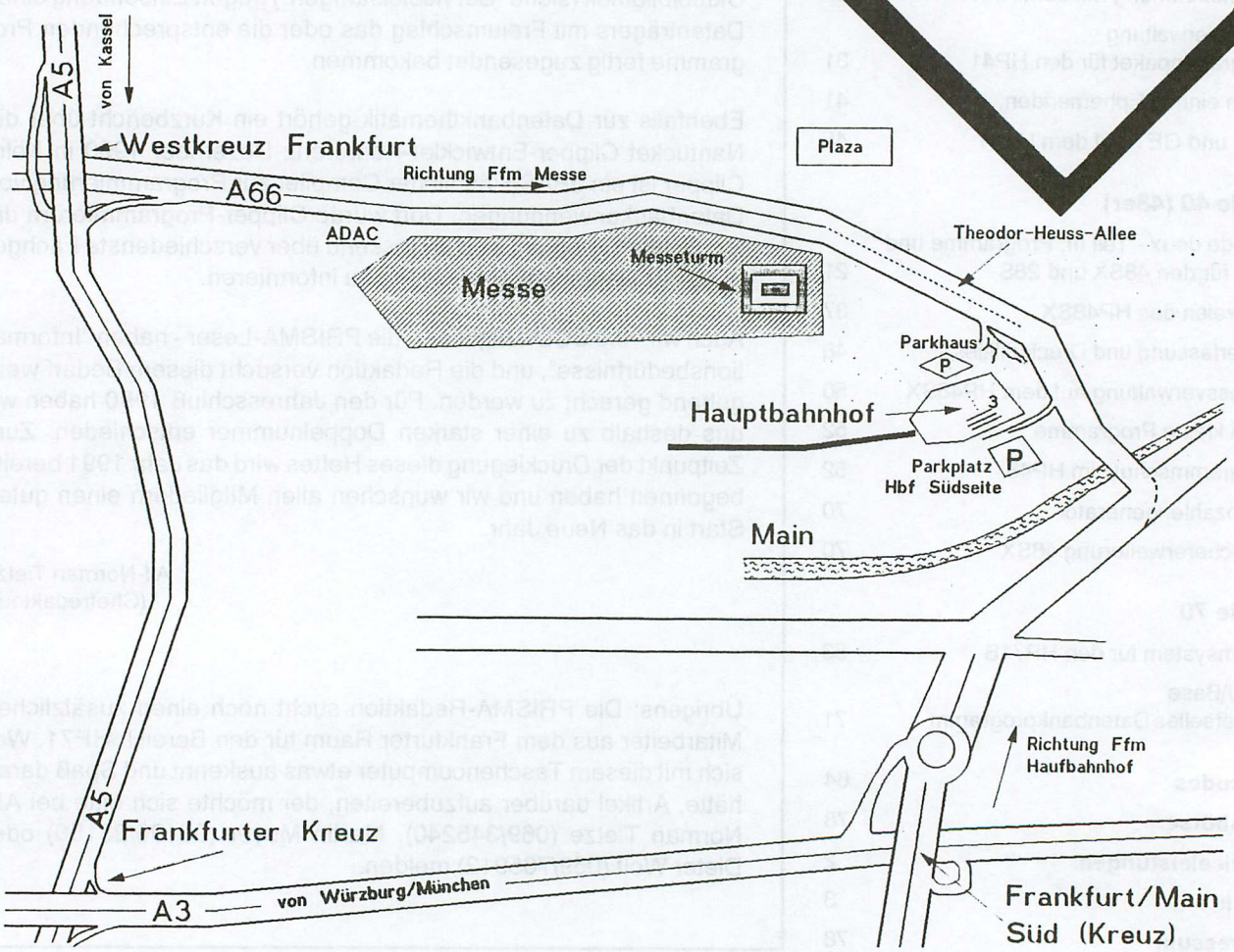
Für den Vorstand.
Gerhard Link
1. Vorsitzender

Wie komme ich hin?

Die Anfahrt zum Hauptbahnhof mit öffentlichen Verkehrsmitteln bedarf sicherlich keiner 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 untenstehende Skizze die Orientierung erleichtern.



3 Jahre Atari-Gruppe

von Werner Müller

Seit etwa September 1987 gibt es eine Atari-Gruppe im CCD. Im folgenden einn Überblick über die Disketten, die den Atari-Gruppenmitgliedern zugesendet wurden. Ich habe nur die wesentlichen Dateien dort aufgenommen, um einen Überblick zu ermöglichen. So wurde z.B. die auf jeder Diskette vorhandene Datei CCD_INFO.* weggelassen. In dieser Datei gibt es neuesten Klatsch aus der Atari-Welt sowie ein Überblick über die Diskette.

Zur Atari-Gruppe zunächst einige wenige statistische Angaben: Sie hat 38 Mitglieder. Bisher sind 23 Disketten versendet worden. Diese Disketten enthielten 1436 Dateien in 284 Ordnern. Diese Dateien belegen einen Platz von 16003000 Bytes.

Wie entstehen die Disketten?

Die Disketten leben im Prinzip von Einsendungen der Atari-Gruppen-Mitglieder. Die Einsendungen werden auf die nächsten Infos kopiert.

Da die Gruppe zur Zeit noch klein ist, können unmöglich die Disketten von den Mitgliedern allein gefüllt werden. Alle Atari-Gruppenmitglieder sind an den neuesten, funktionsfähigen Programmen interessiert, die auf Grund ihres Vertriebsweges durch Kopieren von Disketten weitergegeben werden dürfen. Es werden daher alle erhaltbaren Diskettenserien für den Atari nach Brauchbarem durchsucht und Fachzeitschriften sorgfältig gelesen. Da ich eine Sammelleidenschaft für Texteditoren besitze, sind diese Programme überdurchschnittlich auf den Disketten vertreten. Von anderen Programmen versuche ich immer nur eins (für mich das beste) weiterzugeben. Auch hier gibt es leider Überschneidungen.

Zusätzlich werden Programm-Demoversionen, von denen man etwas lernen kann, weitergegeben (falls der Umfang im Rahmen bleibt).

Die Maxime "0 Bytes free" kann nicht auf jeder Diskette erreicht werden.

Für jedes Programm auf den Disketten gibt es mindestens eine Hardwarekonfiguration, auf der die Programme korrekt ablaufen, nämlich meine (Atari ST 520+, 1 Mb, SM124, Vortex HDplus20, NecFloppy (2DD)). Dieser Satz gilt nicht für alle Programme auf den verschiedensten PD-Serien. Ob Eure Hardware kompatibel zu meiner ist, müßt Ihr dann herausfinden.

Computerviren gibt es auch für den Atari. Hierzu werden die neusten, frei kopierbaren Virenkiller-Programme auf den CCD-Disketten verbreitet. Der Bootsektor der CCD-Disketten wird mit dem Sakrotan V4.14 Bootsektor-Schutzprogramm beschrieben. Bei dem Booten sollte "Kein Virus im Bootsektor" auf dem Monitor erscheinen. Leider kann die Diskette dann nicht ohne Modifikation von einem MS-DOS Rechner gelesen werden. In meinem Rechner habe ich ständig das Programm MORTIMER laufen, das mich warnen würde, falls ein Linkvirus aktiv würde. Die Programme werden mit dem VDU (Virus Distruction Utility) von Richard Karmarkers auf Linkviren getestet. Mehr geht leider nicht.

Von den Programmen mit englischsprachiger Dokumentation werden (mit der Zeit) von den Atari-Gruppen-Mitgliedern deutsche Dokumentationen angefertigt, so daß die nur deutschsprachigen Mitglieder auch diese Programme nutzen können.

Die Mitarbeit an den Disketten ist freiwillig.

Programme, die auf den CCD-Disketten weiterverbreitet wurden, erscheinen zum Teil in neueren Versionen. Diese neuen Versionen werden regelmäig in der Datei UPDATE.DOC beschrieben. Diese können von CCD-Atari-Gruppen Mitgliedern gegen Einsendung einer Diskette mit SAFU bei mir angefordert werden.

Programme, die Shareware sind (Ich weise immer darauf hin), dürfen erst dann im Routinebetrieb verwendet werden, wenn der Shareware Beitrag an die Autoren entrichtet wurde. Gegen das Ausprobieren und gegen einen gelegentlichen Einsatz der Programme ist nichts einzuwenden. Es werden nur solche Shareware Programme weiterverbreitet, die meiner Meinung nach den Sharewarepreis rechtfertigen.

Wie erhalte ich die CCD-Disketten?

Man entscheidet sich einfach dazu, zahlendes Mitglied der Atari-Gruppe zu werden. Die Disketten sind keine PD-Diskettenserie. Sie entsprechen vielmehr einer Art Mailbox auf Diskette, zu der jedes Mitglied per Diskette Zugriff hat. Wer mir Beiträge schickt, erhält seine Diskette mit der nächsten CCD-Diskette zurück. Alte Disketten können von neuen Atari-Gruppenmitgliedern auf Anfrage erhalten werden (Preis je nach Umfang).

Im folgenden findet Ihr das Inhaltsverzeichnis der Atari- Disketten.

Werner Müller, Schallstr. 6, 5000 Köln 41
E-Mail: MBK1:W.MUELLER

P.S. Was heißt eigentlich SAFU mit Diskette?

Dies scheint nicht allen CCD-Mitgliedern klar zu sein. Deshalb im folgenden eine kurze Erklärung:

SAFU bedeutet Selbst-Addressierter-Frei-Umschlag: Am besten 1.70 in Briefmarken auf einen wattierten Umschlag (DIN A5) kleben, die eigene Anschrift darauf schreiben und eine Diskette (3 1/2 Zoll) einlegen. Dann diesen Umschlag in einen größeren Umschlag stecken, meine Adresse darauf schreiben, ausreichend frankieren und ab in den Briefkasten. (Natürlich muß auch irgendwo euer Wunsch zu finden sein, entweder auf der Diskette als Textdatei oder auf einem Zettel). Ihr erhaltet dann möglichst bald die gewünschte Diskette zurück (hängt von meiner Freizeit ab).

September 1987:

Diskette: 1	(Einsteiger Diskette)
\CCD_INFO.001	
CLI.PD	Kommando-Interpreter
DESKTOP.INV	Vorbereitete DESKTOP.INF Datei
DOAUTO	vorbereiteter Auto-Ordner
EDIMAX	Ein erster Editor
MALEN	einfaches Zeichenprogramm
MAXIDISK	Eine resetfeste Ramdisk
MAXIDISK.INV	Vorbereitete MAXIDISK.INF Datei
PATIENCE	Ein Kartenspiel
(Übergang CPM auf Atari)	
\CCD_INFO.002	
CPMZ80.TOS	Der eigentliche Emulator
CPM_PRO	Ein Program für CPM80
DBASE.TST	Testbericht DBASE 2 für Atari
EIN_TEST	Testbericht über Beckertext
EIN___SP.IEL	Stagger
GFABASRO.PRG	Der GFA-Basic 2.0 RunOnly Interpreter

Diskette 2:	
\CCD_INFO.003	
ACCESS.DOC	Dokumentation zu den Accessories
ANALOG.AC	Eine Uhr
AUTOOFF.AC	Ein Bildschirmschoner
CALENDAR.AC	Ein Kalender
DIRPRINT.AC	Druckt den Diskettenінhalt aus
DISKSORT	Diskettenverwaltungsprogramm
JOPPICH	Einige Spezialprogramme
KERMIT_P.LUS	Das Kermit Programm
PROCALC.AC	Ein Rechner
PRTDIR.AC	Druckt Diskетteninhalt aus
RAMFREI.AC	Wieviel Speicher ist noch frei?
UNITERM	Das Uniterm-Programm (V 1.7)
WATCH.AC	Eine Uhr
\CCD_INFO.004	
MOLEKUEL	Das Molekül-Programm

Diskette: 3	
\CCD_INFO.005	
EDIMAX.55	Edimax in der neuen Version
LITTLE_3.05	Ein super Malprogramm
UNITERM	Dokumentation zu Uniterm
XMDM	Xmodem Filetransferprogramm
\CCD_INFO.006	
ANFANG.TXT	Wie geht man mit dem Atari um?
EXPERT.PRG	Ein Aufkleberdruckprogramm
KOPIERER	Alle möglichen Disketten-Kopierprogramme
LABYRINT	Ein Spiel

Diskette: 4	
\CCD_INFO.007	
DISKMON	Ein Disketten Monitor
FSELECT	Eine neue Fileselektor Box
LITTLEPA.INT	Little Painter update Bericht

LOADER	Gehört auf jede Festplatte!!!
MANFRED Life	Programm von CCD-Mitglied Manfred...
OMIKRON.TST	Omikron Basic Test
OMIK_TIP.S	und Tips
\CCD_INFO.008	Nur Programme keine Quellkodes!!!
FRACTAL1	Fraktale in Basic
FRACTAL2	Fraktale in Maschinensprache

Diskette: 5
\CCD_INFO.009

HYPERFMT.FLD	Hyperformat (viel Platz auf Diskette)
KERNE_2	Krieg der Kerne (Spiel)
KLEINE_P.ROG	Verschiedenes (Quickmaus, Rat-Trap, Viruskiller u.a.)
OMIKRON	Etwas in Omikron Basic
UPDATE.DOC	Von welchen Programmen gibt es neue Versionen?

\CCD_INFO.010

ARC.TTP	Archive Programm
ASH195.PRG	Shell für Archiv Programm
EMACS.ARC	Mirco-Emacs
GULAM.ARC	Gulām-Shell

Die Programme dieser Diskette sind gearkt. Ergebnis des Umkopierens von einseitigen auf doppelseitige Disketten.

Diskette: 6
\CCD_INFO.011

ARC	Ein File Komprimierer
EMACS.REF	Micro-Emacs Editorreferenzkarte
GOEDICKE	Kritik und Tips von Goedicke
MAXIIDISK.4MB	Update resetfeste Ramdisk
OMIKRON.30	Neue OmikronBasic Version
ST_NEWS.DOC	Eine Diskettenzeitschrift
VIRUS.DOC	Was ist ein Computervirus?
WATOR	Ein Simulationsprogramm Hai/Fisch

\CCD_INFO.012

SCHECK	Schecks schreiben
ST_CALC	Tabellenkalkulationsprogramm

Diskette: 7
\CCD_INFO.013

ARC_195	Einfache Shell für Komprimierprogramm
HDUTIL	Harddiskutility Programm
MEMFIL14	Ein Filedump Programm
OMIKRON.NEW	Was kann Omikron 3.0 (nicht dokumentierte Befehle)
SAGROTAN.403	Ein super Viruskiller (Boot/Link)
SHELL	Shell-Programm als Accesory
VDU32.DOC	Der "Beste" Viruskiller (Test) !!!

\CCD_INFO.014

ALLE_DIS.KS	Ein Überblick der CCDDISKs1-12
ANTIVIR	Antiviren Programm
CFG_ANAL.YSE	1st Word Plus Treiber Analysator
FORTH.ARC	Ein Forth-Compiler
LANDER	Ein Mondlande Spiel

1989: Ab hier doppelseitige Disketten

Diskette: 8
CCDINFO.015

FILEBOX	Neue Fileselektor Box
KALEID.PRG	kleines Progamm (hübsch)
KRITIKEN	(1st_address, MasterText, ST-DIGITAL)
LUPEN	Bildschirm Vergrößerer
TRANSIST.OR	Schaltplanzeichenprogramm
UPDATE.DOC	Neue Programmversionen
WT_TOOLS	Die Weller tools für GFA-Basic

Diskette: 9
CCDINFO.016

BIT_3_1	Sehr gutes Diskkopierprogamm
CARPET	Ein 3D-Funktionsplotter
CONVERT	Wandelt Bildformate um (CT-Disks, Minix-ST, Turbo-ST)
KRITIKEN	Omikron-Programm in Source
MCWRITER	Ein PD-Diskettenversand
PD_EXPRE.SS	Terminkalender v. den CT-Disks
TERMIN	Weller-Tools für die Harddisk
WT_HD	

Diskette: 10
CCDINFO.017

ADAMZ	CCD-Pascal Programme von Adamkiewicz
FLOECK	Test v. Thorsten Floeck (crunch)
GFABASRO.PRG	RunOnly Interpreter Vers. 3.0
GREIFER	Pattern Such und Ersetzt Progr.
HARDCOPY	Bildschirmausdruck in allen Variationen
IFS	Fraktale einmal anders
PLOTTER.2:1	Funktionsplotter (plottet alles.)
POSTERPR.INT	Bildschirm als Poster

Diskette: 11
CCD_INFO.018
\MODULA-2 System aus München

Diskette: 12
CCD_INFO.019

DATADISK	Diskettenkatalogsystem (D)
RDV5.RDU	Autobootfähige Ramdisk
SBASE	Einfaches Datenbanksystem
STCAT40	Diskettenkatalogprogramm (e)
ST_CLUB	Berichte und Prg's vom ST-Club
TURTLE30	Festplattensicherungsprg.

Diskette: 13
CCDINFO.020

DISAS	Ein Dissasembler
DISKDOUB	vollständige Version von Info 019 (da fehlte eine Datei)
EMACS.39	Micro-Emacs Update auf Version 3.9
HARDCOPY.PLS	Hardcopy Treiber für Deskjet (von Adamkiewicz)
PROFITEX.T	Ein "profi"-Textsystem (einfach)
QUICKINF	Desktop.inf ändern/neuladen
QUICKST	Bildschirmausgabe beschleunigen
RESTART	Kaltstart für Mega-ST
TEMPLMON.112	Maschinenspr. Monitor für Atari
VKILL2	Viruskiller in Version 2.0
VSCOPE	Virusschutzprogramm (resident)

Diskette: 14
CCDINFO.021 (siehe auch Prisma Heft: 5/1989)

AMCGDOS.DOC	AMCGDOS Original Dokumentation
ARCX.TTP	Entarcprogramm für OPUS.DOC
ASSIGN.SYS	GDOS
ATARIHD	Praxistip zur Atari-HD
GEMSYS	GDOS
OPUS	Das OPUS Progamm
OPUSDOCS.ARC	OPUS Dokumentation (gearct)
WIDERST	Farbecode für Widerstände (schwarz-Weiss)

Diskette: 15
CCDINFO.022

FONTKIT	Fontkit (PD-Version)
FONTS	Fonts zum Fontkit
FONTSEL.ACC	Accessory zum Laden von FONTS
GSEDT	Editor analog zum VAX-Editor
NEWPACK	Packer von Programmen
OK	Bericht OK-CLUB vom OMIKRON
PC_SPEED	Handbuch von PC-SPEED
QUICFIND	FILE-FINDER
QUICKDEX	Benchmark für Atari
QUICKINF	Update von Diskette 19
QUICKR_2	Starten v. Progammen
QUICKST	Update von Diskette 19
QUICLABL	Aufkleber herstellen
QVIEW13	Ansehen von Textdateien
SCHEIBE2	Scheibenkleister II (erste Erfahrungen)
THW	Teschnisches Hilfswerk (OMIKRON.BASIC-Programm)

1990:

Diskette: 16
CCDINFO.023 DFUE und Mailboxen

ATARIBOX	Log Files aus der Atari-Box
BBS	Ein Buletin Board System
HYPER.260	Hyperformat neuste Version
MEDIABOX	Log Files aus d. Mediabox Köln (Textfiles nicht Atari-spezifisch)
MINIBBS	
RAINBOW	Patchfiles von Atari für TOS 1.4
UPDATES.TXT	Neuste Programmversionen

Diskette: 17
CCDINFO.024

ARCX.TTP	Schon wieder geArcte Dateien
ASSIGN.SYS	Hilfsdatei für GDOS
AUTO	Neues AMCGDOS
COLOR.EM	Color-Emulatoren
DEUTSCH.TXT	Übersetzungen
*GEMINI	Gemini Shell (Nur TOS = 1.2!!)
GEMSYS	
LITTLE_4.32	Little Painter Version 4.32
MORTIMER.DOC	Bericht über Mortimer
NEODESK.DOC	Bericht über Neodesk
RIEDLING.DOC	Brief
TEST.DOC	Softwarekauf über Versand

* Diskette gegen Einsendung von SAFU mit DISK bei Werner Müller zu erhalten

Diskette: 18
CCDINFO.025
Turbo-Assembler mit Debugger

Diskette: 19
CCDINFO.026

1ST_KONV	Umwandlung ASCII-Wordplus
FCOPY_3	Update FCOPY 2.0 Programms
FORMULAR	Beschriften von Formularen
LABORANT	Programm für den Laboralltag
MARTIN	Martin Meyer's Hilfsprogramme
OMIKRON	Tips zu Omikron-Basic Strings
PRISMA	Prisma-Artikel in der Urfassung
WORDTAST.5_0	Hilfsprogramm für Wordplus

Diskette: 20
CCDINFO.027

CHEETAH1	File Kopierprogramm (schnell!) (Sektorweises kopieren)
DIFF315	File Vergleicher
HFCOPY37	Filekopierer
LHARC_60	File Komprimierungsprogramm
NSYSCOM2	Systemaufrufe anzeigen
POOLFIX3	Patch für TOS 1.4
STBLANK	Bildschirmschoner (Feuerwerk)
TESTS	Berichte diverse (Heureka, Sherlook, That's write)
ULTRA	Diskettenkomprimierer (incl. Update Omikron-Basic auf Version 3.03)
VKILLER	Viruskiller Version 3.13

Diskette: 21
CCDINFO.028

EASYGEM	Easygem.LIB Demo
SQL_LIB	SQL.LIB Demo
MORTIMER	Mortimer Demo
PFXPAK	Programmkomprimierer
LHARC	Filekomprimierer (Assembler)
TURBO.TXT	Turbo-Assembler Artikel (Prisma)

Diskette: 22
CCDINFO.029

CH_DISK.FLD	Disketten-Editor (Atari)
COMPRESS	Programme zum Ent- und Verpacken von Dateien
DL3	Direktory-Drucker
DLII024	Disketten-Editor (und Harddisk Reorganisierer)
HDX301	Harddisk Treiber (Atari)
MAXONPAS.TXT	Maxon_Pascal Turbo-Pascal kompatibel ??
POOLFIX4	Patch für Tos 1.4 (Nicht ATARI!)
TOSDATA	Welche Version im Speicher?
TOS_TEST.DOC	neue Zeitschrift TOS
WAS_NOCH.TXT	Was gibts noch an Programmen?

Diskette: 23
CCDINFO.030

BSS-BIT	Set Fast-Bit (TOS 1.4)
HYPERTEX.T	Hypertext System aus München
ICAP	Programmgenerator
ICAP/NET.DOC	
MEGATOOL.S	Formatierprogramme
MEM_TEST	Speichertest
NETCOPY	Kopieren über die Midi-Schnittstelle
QPRINT	Bildschirmausdruck nur Text
STETRIS	ein Spiel... (EOF)

Steuern über Centronics-Schnittstelle

Rechner: **ATARI-ST, alle PCs mit Centronics-Ausgang, HP75/71/41/(48) mit HP-IL Modul und IL-Konverter HP82166A bzw. GPIO-Interface HP82165A**

Zubehör: **Centronics-Relais-Interface UCR-80 (Fa. Auerwald GmbH ca. 99.-DM)**

Fast jeder Rechner besitzt einen Centronics-Ausgang - auch Parallelschnittstelle genannt - zum Ansteuern eines Druckers.
Diese Schnittstelle kann auch dazu verwendet werden, eine Relais-Bank anzusteuern. Mit einem 8-Bit Datenwort können dabei die 8 Datenleitungen des Centronics-Busses einzeln angesteuert werden, wie dies bereits in PRISMA 2/90 von Christoph Klug erläutert worden war.

Das Relais-Interface UCR-80 der Firma Auerwald GmbH (erhältlich über Conrad Electronic, Hirschau) ist der bequemste Weg, speicherprogrammierbare Steuerungen im Eigenbau zu erstellen. Es werden lediglich ein Druckerkabel und ein Netzgerät (10V/ max. 400mA) benötigt, um das Interface in Betrieb zu setzen.
Mit dem kleinen Programm in Bild 1 können alle 8 Relais einzeln gesetzt werden (EIN) und natürlich auch wieder gelöscht (AUS) werden. Mit einer Zusatzlogik aus AND-Verküpfungen, die man selbst aufbaut, können mit der Centronics-Schnittstelle maximal 255 Kanäle adressiert werden.
Aber auch mit 8 Kanälen, wie sie das UCR-Interface zur Verfügung stellt, kann man recht komplexe Programmsteuerungen, Zeitschalter usw. entwerfen; sogar eine komplette Waschmaschinensteuerung wäre möglich. Die Relais sind für 8 Ampere ausgelegt.

Falls der Leser einen ATARI mit Echtzeituhr besitzt, so könnte er Zeitschalter mit absoluter oder relativer Zeitvorgabe realisieren.

Bei Zeitschaltern mit mehreren Stunden oder sogar Tagen Betriebsdauer sind natürlich batteriebetriebene Rechner wie der HP41 oder der HP71 wegen des niedrigen Stromverbrauchs im Vorteil.

Steuerungen mit dem HP-IL Konverter

Um eine Centronics-Schnittstelle vom HP41/HP71 aus bedienen zu können benötigt man entweder den sogenannten IL-Converter HP82166A oder das etwas teurere GPIO-Interface HP82165A, wobei letzteres aus einem Konverter und Netzteil aufgebaut ist.

Beide Geräte sind nicht mehr im HP-Vertriebsprogramm, können also nur noch als Restbestände bei Händlern oder gebraucht über z.B. die Clubbörse bezogen werden.
Beiden meisten HP Taschenrechneranwender dürften jedoch die entsprechenden IL-Einsteckmodule zu den Taschenrechnern bereits vorhanden sein, da vermutlich die meisten mindestens ein IL.-Gerät (z.B. Drucker oder Kassettenlaufwerk) betreiben.

Die erforderliche Gerätekonfiguration für die beiden Taschenrechner HP41/71 bis zur Centronics-Schnittstelle ist in Bild 2 zusammengestellt. Übrigends ist diese Konfiguration auch erforderlich, wenn man einen Centronics-Drucker anschließen will. Dies ist bei heutigen Druckern bis auf z.B. den DESKJET die einzige Schnittstelle. Näheres wurde bereits von P. Ehrenberg [1] in PRISMA beschrieben.

In den Bildern 3a, b, c ist die Beschaltung des IL-Konverters als unidirektionales Centronics-Interface nach HP-Manual [2] wiedergegeben. In diesem Anwendungsfall müssen nicht alle 34 Konverterpins angeschlossen sein. Man benötigt für den Konverter eine stabilisierte 5V-Versorgung, die mit 100mA belastbar sein muß (z.B. L200, LM7805 o.ä.). Die Konverter-Pins 3 und 31 werden an +5V angeschlossen, die Pins 1, 7 und 11 an GND (Masse), siehe Bild 3a.
Die übrigen 12 Konverter Pins mit den Bezeichnungen nach Bild 3c:DA0, ... DA7, DAVO, RDYI, GND, DCLO gehen über ein 12-adriges Kabel (Länge ca. 20cm) zum 36-poligen Centronics-Stecker, wie es Bild 3b darstellt. Dabei bedeuten die Zahlen innerhalb der Steckerumrandung die entsprechende Centronics Pinnummer; außerhalb stehen die Konverter-Pinnummern mit den Bezeichnungen nach 3c.
Am Konverter selbst sollte nicht gelötet werden.
Die Konverter-Pins werden über eine doppelreihige Norm-Buchsenleiste (2.54mm Rastermaß, 64-polig) angeschlossen. Konverter, Netzgerät und Buchsenleiste werden am besten in ein kleines Kunststoffgehäuse oder auch Holzgehäuse eingebaut, so daß das Centronics-Kabel fest am Gehäuse verbunden ist.

Testprogramme zur Installation

Wenn der IL-Konverter komplett aufgebaut ist, dann verbinden Sie alle Geräte entsprechend Bild 2 und prüfen mit den Programmen nach Bild 4 bzw. Bild 5 die Grundfunktionen des Relaisinterfaces für den HP41 bzw. den HP71 als Steuerrechner durch.

Literatur:

[1] P. Ehrenberg, PRISMA 84.4.19, "Peripherie".
[2] HP82166A-Technical Manual HP-IL Converter; Nov. 1981, HP-Nr. 82166-90002.
[3] G. Friedeman, "Control the world with HP-IL", Heldermann-Verlag, Berlin
ISBN 0-9612174-9-9
[4] Bedienungsanleitung UCR-80 Centronics-Relaisinterface; Auerswald GmbH & CoKG, D3302 Cremmlingen


```
0 'TEST-PROGRAMM ZUR RELAIS-AUSWAHL
1 DEFINT "A"
2-Start
3 INPUT "WELCHER KANAL? ";A
4 IF A>255 THEN END
5 LPRINT CHR$(A);
6 GOTO "START"
7 END
```

Bild 1

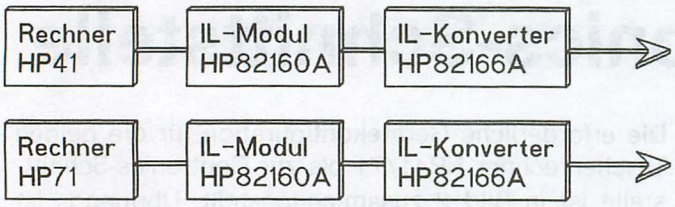


Bild 2: Konfiguration für HP-Taschenrechner

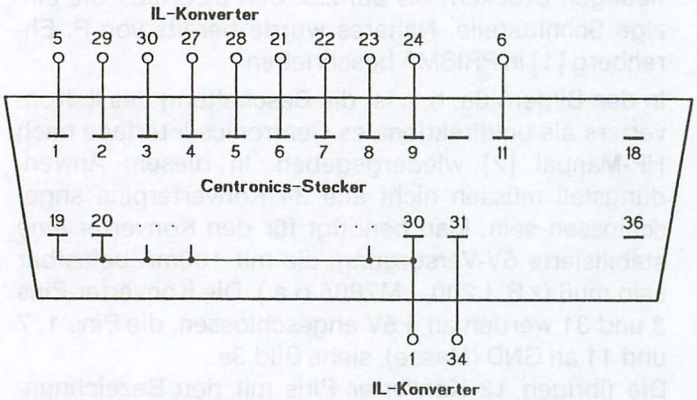


Bild 3b: Centronics Kabelbeschaltung zum Konverter

```
01LBL "COUNT"
02 MANIO
03 SF 17
04 SF 21
05LBL 01
06 "KANAL=?"
07 PROMPT
08 ACCHR
09 GTO 01
10 END
```

Bild 4

```
10 DESTROY ALL
20 'START':
30 INPUT 'KANAL-NR=?';A
40 IF A>255 THEN END
50 PRINT CHR$(A);
60 GOTO 'START'
70 END
```

Bild 5

Peter Jochen
Heilbronner Straße 240
7410 Reutlingen

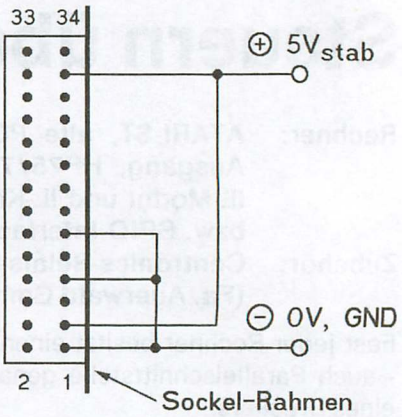


Bild 3a: Spannungsversorgung des Konverters

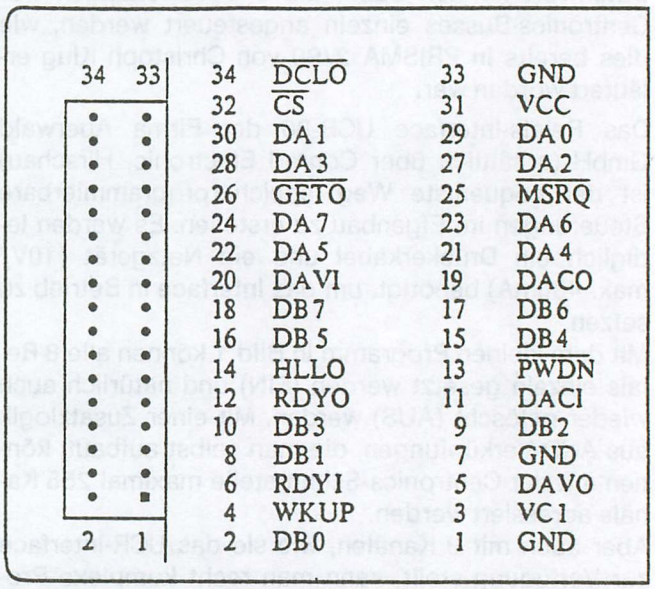


Bild 3c: Pin-Bezeichnungen des Konverters nach HP-Manual

HP DeskJet 500

Der DeskJet 500 ist voll abwärtskompatibel zum DeskJet Plus, d.h. 300 dpi wie seine LaserJet-Brüder in gestochen scharfer schwarzer Schrift bis zum letzten Tropfen. Er ist mit bis zu 3 Seiten pro Minute genauso schnell wie sein Vorgänger und ebenso leise.

Die eingebauten Schriften wurden um die Letter-Gothic ergänzt, CG Times wurde in 12 Punkt als Proportionalschrift fest eingebaut.

Es gibt vier neue Schriftkassetten mit "schönen" Schriften:

- Global Text Scholbook in 8 und 10 Punkt (ähnlich Times Roman), sowie CG Triumvirate in 10 und 14 Punkt (ähnklich Helvetica).
- Garamond Collection in 8, 10, 12 und 14 Punkt.
- Dom Casual in 12, 14, 18 und 24 Punkt für Überschriften, die "gestyled" aussehen sollen.
- Brush in 12, 18 und 24 Punkt, nur fett. Diese Schrift sieht wie eine Schreibschrift aus.

Bei den angegebenen Schriftgrößen muß man die Fähigkeit des DeskJet erwähnen, die normalen Punktgrößen zusätzlich auch zu halbieren. Es stehen also immer doppelt soviele Schriftgrößen zur Verfügung.

Eine absolute Neuheit für die DeskJet's ist die Möglichkeit zum Kerning, also zum Unterschneiden von Buchstaben. Die Abbildung zeigt diesen Effekt.

Der Vergleich der ersten beiden Worte zeigt das Prinzip, das "o" wird etwas unter das "T" geschoben, das Wort wirkt viel

Total
Total
Feste Teiler

ausgeglichener. So richtig zum Tragen kommt das Kerning bei kursiven Schriften, da bei diesen durch die Neigung der Buchstaben sonst die horizontalen Abstände unnatürlich groß werden. Dies gilt *verstärkt*, wenn nur Versalien verwendet werden.

Am erfreulichsten ist, daß der DeskJet 500 um einige Hundert Mark billiger geworden ist. Preisangebote unter DM 1500.- sind nicht selten.

mm

Der HP27S
dienstbarer Geist für alle Tage

Wieviele Menschen gibt es eigentlich, die zwar einen Taschenrechner haben wollen, mit UPN aber partout nicht klarkommen oder es vielleicht garnicht wollen. Bei den weiblichen Benutzern von Taschenrechnern ist die Akzeptanzschwelle bei dem Wort UPN besonders niedrig.
Dieser Einstellung hat Hewlett Packard Rechnung getragen, indem eine Reihe von Taschenrechnern kreiert hat, die nach dem altbekannten Rechenprinzip AOS (Algebraisches Operations-System) arbeiten, d.h. schön brav Klammern aufmachen und wieder zumachen (oder auch nicht).
Von Nippons Taschenrechnerherstellern gibt es ja bereits eine Unzahl verschiedener Taschenrechner auch für den täglichen Gebrauch. Diese haben alle eins gemeinsam: Sie sind schlecht zu bedienen (zu kleine Tasten, wabbelig etc.), die Befehle sind wahllos über das Tastenfeld gestreut, sodaß man immer zu einer Expidition gezwungen ist, wenn man mal etwas anderes als die vier Grundrechenarten benötigt. Das Display zeigt ziemlich lieblos ein paar Zahlen an, das ist es dann schon meistens, eine Back-Space Taste sucht man so gut wie überall vergebens. Ein Tippfehler, und man kann alles wieder von vorne eingeben...

Vor einigen Monaten begab es sich, daß sich bei meiner besseren Hälfte der Wunsch nach einem dienstbaren Rechenknecht kundtat. UPN, wat iss'n dat, bloß nicht etwas neues lernen müssen schloß schon einmal alle normalen technisch-wissenschaftlichen Taschenrechner von Hewlett Packard aus.

Mein Blick in den Katalog der Taschenrechner von HP blieb dann an dem größten der sogenannten Alternativrechner hängen, dem HP27S. Er erfüllte alle Grundbedingungen, die für die Auswahl gestellt worden waren:

- Einfachste Handhabung
- übersichtliches Tastenfeld mit gut treffbaren und drucksicheren Tasten
- großes, übersichtliches Display mit alphanumerischer Anzeige, möglichst mehr als eine Zeile
- mehr als einen Speicher zum Abspeichern von Zwischenwerten
- Statistikfunktionen
- Prozentrechnung
- verschiedene Zahlenbasen
- mathematische Funktionen inklusive hyperbolischer Trigonometriefunktionen
- Zeitfunktionen inklusive Uhr
- einstellbare Zahlendarstellungen: Zahl der Nachkommastellen, Punkte zwischen den Tausenderstellen, wissenschaftliches Format der Exponentendarstellung, d.h. nur 1 E3, 1 E6, 1 E9 usw.
- Konstant Memory, d.h. nach dem (selbsttätigen) Ausschalten sollen alle Daten und Eingaben erhalten bleiben, um sofort nach dem Wiedereinschalten weiterarbeiten zu können.
- Bei Bedarf soll die Möglichkeit bestehen einen Drucker oder ähnliches ansprechen zu können.
- Gut geschriebenes und didaktisch durchdachtes deutsches Handbuch, kein Handzettel in 12 Sprachen.

So, nach diesen einleitenden Worten wollen wir uns langsam dem Ziel unseres Interesses widmen, dem Rechner selbst.

Bild 3 zeigt den Rechner, hier kann man anhand der beschriebenen Punkte schon mal einen groben Überblick gewinnen. Man kann leider die Menüs nicht erkennen, die mit der SHIFT-Taste (5) zu erreichen sind: Sie liegen über den Tasten 1 (=%CH) bis ÷ (=TIME) und sind auf dem Tastenfeld hellgrau hinterlegt.
Das ist auch schon einer der Clous des HP27S: Er hat trotz seiner Vielzahl von eingebauten Funktionen nur wenige Bezeichnungen auf dem Tastenfeld, die Befehle sind zu inhaltlichen Gruppen zusammengefaßt, die über den Aufruf der Menüs erreichbar sind.

Ruft man nun ein Menü auf, so wird die untere der beiden Displayzeilen durch Softkeys belegt, wie dies in Bild 1 zu sehen ist, hier wurde das CONVERT-Menü (Taste SHIFT *) aufgerufen.

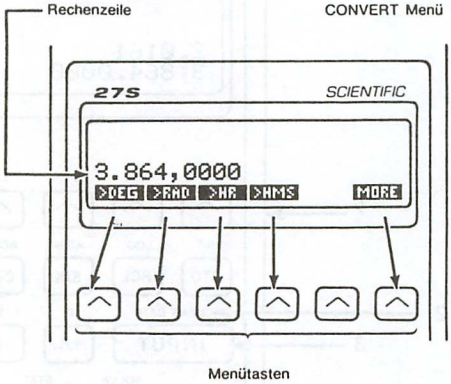


Bild 1: CONVERT-Menü

Durch Drücken der entsprechenden Taste unter der inversen Bezeichnung ruft man die entsprechende Funktion auf, die sich in der Regel auf die darüber befindliche Eingabezeile auswirkt.
Wenn ein Menü mehr als eine Reihe Softkeys benötigt, so ist die rechte Taste einfach mit der Anweisung MORE belegt, womit man in die zweite oder eventuell auch dritte Ebene herabsteigen kann. Mit der Taste EXIT kommt man immer wieder aus den Menüs heraus, wenn dies nicht durch die Funktion selbst gegeben ist.
Apropos Display: Der HP27S hat in guter alter HP-Manie einen Stack; richtig gelesen, der Rechner merkt sich die letzten vier Rechenergebnisse in einer Art Historik-Speicher mit vier Ebenen. Mit den Pfeiltasten kann man vorwärts und rückwärts blättern und sich

das Ergebnis aussuchen, mit dem man weiterrechnen will.
Bild 2 zeigt die Art und Weise, wie weitere Eingaben behandelt werden. Das Rechenzeichen ist immer zu sehen; wird die Reihe der Zahlen zu lang für die Displayzeile, so wird die Zeile nach links hinaus geschoben. Der Rechner berechnet immer soviel, wie gerade mathematisch möglich ist.

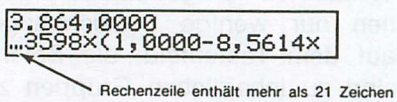
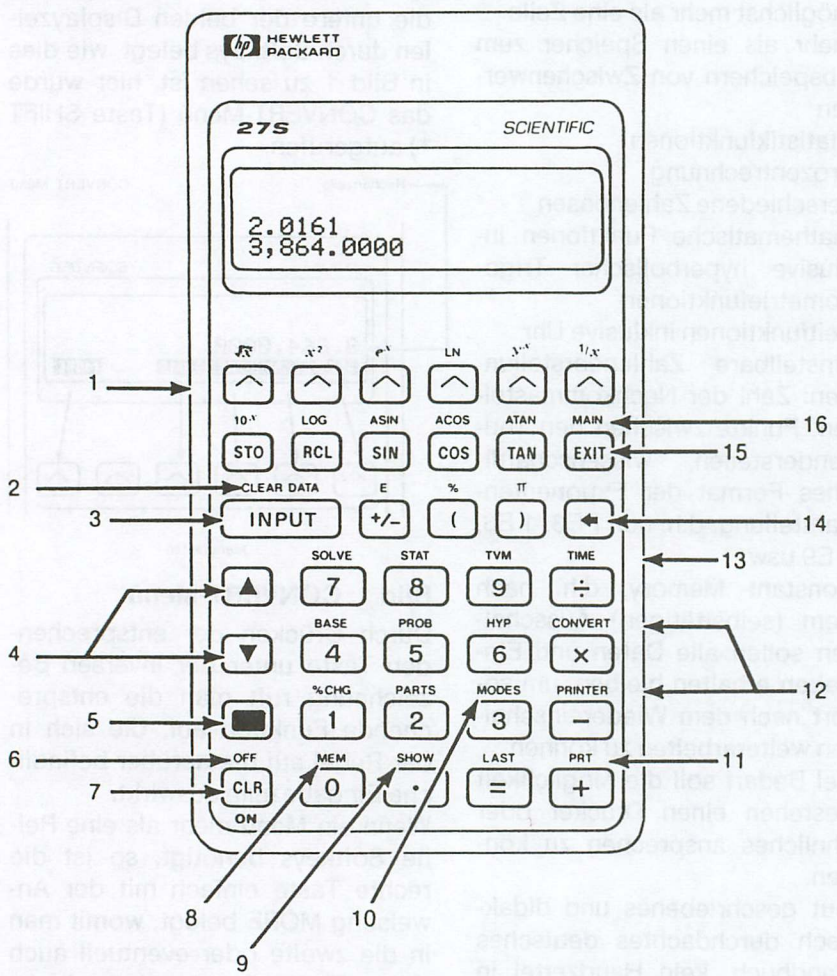


Bild 2: Normale Berechnungen

Das Display zeigt außer den zwei Zeilen mit Zahlen bzw. eine Zeile mit Zahlen und die untere mit Softkeys noch in einer obersten Zeile den Status an: Der Status der Umschalttaste (SHIFT), ein Zeichen gibt es für den Druckbetrieb, ein Zeichen zeigt die Aktivität eines Alarms an, ein Indikator signalisiert zu schwache Batterien und der letzte den Winkelmodus für die trigonometrischen Funktionen.
Meldungen und Fehler werden im Klartext angezeigt. Die etwa 40 verschiedenen Fehlermeldungen sind hervorragend erklärt.



1. Menütasten
2. Löschen von Teilen des Speicherbereichs
3. Dateneingabe
4. Durchsehen von Listen
5. Umschalttaste
6. Ausschalten des Rechners
7. Einschalten des Rechners; Löschen der Rechenzeile
8. Verfügbarer Speicher (MEMory)
9. Anzeige der vollen Genauigkeit
10. Anzeigeformat; Modi
11. Drucken der Rechenzeile (PRinT)
12. Steuerungs- und numerische Menüs
13. Applikationen
14. Löschen einzelner Zeichen
15. Vorangehendes Menü
16. Hauptanzeige (MAIN)

Bild 3: HP27S Tastenfeld

Es werden also auch die Ursachen genannt, man steht also nicht immer vor der sinnigen Meldung "DATA ERROR".

Speicheraufteilung

Dem Anwender stehen 10 Speicherregister zum direkten Zugriff zur Verfügung, diese sind mit den Tasten STO und RCL 0-9 erreichbar. Dabei sind auch die vier Grundrechenarten erlaubt, man kann also direkt in ein Speicherregister hinein multiplizieren oder addieren. Dazu drückt man nach STO einfach eines der vier Grundrechenzeichen: STO X steht dann z.B. im Display, es fehlt dann nur noch die Angabe des entsprechenden Registers.
Der HP27S verfügt über insgesamt 8 kByte Speicher, von dem etwa 1300 Byte für interne Variablen verbraucht werden, der Rest steht für Listen, Alarme und Gleichungen zur Verfügung.
Mit der Funktion MEM (SHIFT Taste 0) kann man sich jederzeit den noch verfügbaren Speicher ansehen, er wird in Anzahl Bytes und in % angegeben, man erhält also sofort einen Überblick über den Stand der Dinge.

Displaymodi

Man kann, wie schon erwähnt, die Darstellungsweise der Zahlen im Display mannigfaltig beeinflussen:

Bild 4: Menü MODES

Wie in Bild 4 zu sehen ist stehen so gut wie alle Anzeigeformate zur Verfügung.
Drückt man die Taste MORE, so werden einem in der zweiten Ebene folgende Einstellmöglichkeiten geboten:

D/R schaltet zwischen DEG und RAD für trigonometrische Funktionen um, GRAD gibt es nicht.

BEEP gibt die Möglichkeit den Piepser ganz abzuschalten oder nur für Alarme freizugeben; sonst wird bei jeder Fehleingabe oder Fehler-

meldung gepiepst, in einer Klausur eventuell sehr lästig und störend.

PRNT sorgt dafür, daß die Ausdruckgeschwindigkeit über die Infrarotschnittstelle gebremst wird, um einem Infrarotdrucker mit Batteriebetrieb noch genügend Zeit zum Ausdruck zu lassen.

Die maximale Rechengenauigkeit von 12 Stellen nach dem Komma kann man sich mit der Funktion SHOW (Taste .) anschauen, wenn es denn unbedingt nötig ist.

Nach dem Drücken des Softkeys FIX erscheint die Aufforderung "TYPE #DIGITS (0-11);"
"PRESS [INPUT]"
Hieran sieht man in etwa, wie die Benutzerführung gemacht ist, beinahe idiotensicher, solange man English for runaways beherrscht.

In einigen Menüs wird die Eingabe von ALPHA-Zeichen für die Namensgebung z.B. einer Liste nötig. Hier werden die Softkeys mit den Buchstabengruppen belegt, durch Drücken einer solchen kommt man dann zur Auswahl des entsprechenden Buchstabens eine Menüebene tiefer, ein sehr zeitraubendes und umständliches Verfahren. Es war aber die einzige Möglichkeit die Buchstaben nicht auf der Tastatur unter zu bringen. Sonderzeichen wie Winkel oder Grad oder auch ein Summenzeichen Σ für spezielle Beschriftungen sind zugänglich, Umlaute ebenso. Es gibt aber nur Großbuchstaben.

Numerische Funktionen

Wie in Bild 5 zu sehen ist befinden sich eine große Anzahl normaler numerischer Funktionen unter einem Menüeintrag verborgen, sie würden die Übersichtlichkeit wegen ihrer Vielzahl nicht gerade fördern.
Ich möchte nur kurz auf die einzelnen Befehle eingehen, da ich nicht die mathematischen Grundlagen und Randbedingungen abhandeln kann.

PROB Menü

Dieses Menü berechnet Kombinationen, Permutationen, Fakultäten (N!) und erzeugt außerdem eine Reihe von Zufallszahlen (RAN#).

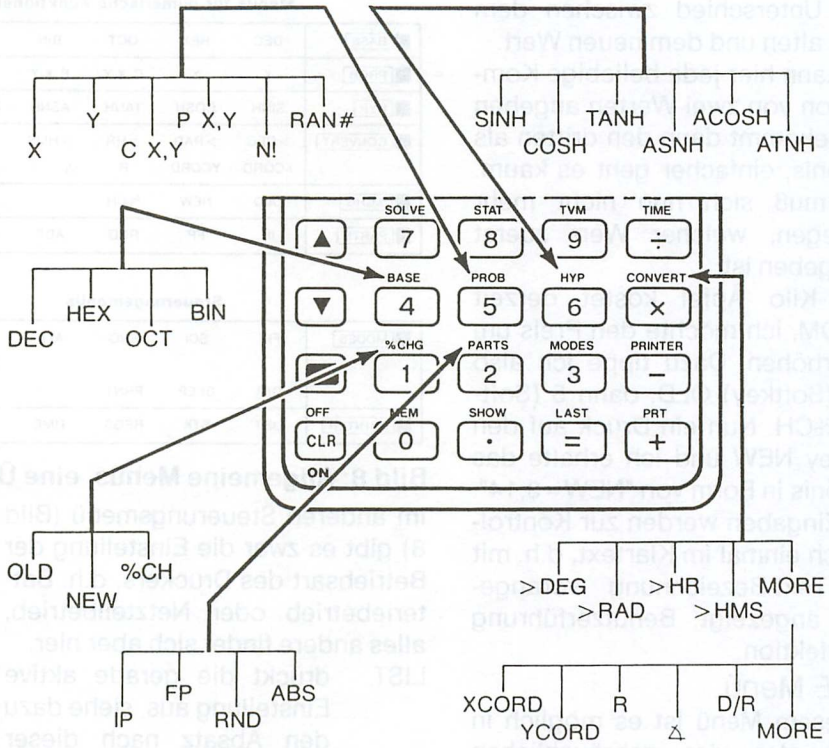


Bild 5: Menüs für numerische Funktionen

Das *HYP* Menü

ermöglicht die Berechnung von SINH, COSH und TANH sowie deren Umkehrfunktionen ASINH, ACOSH und ATANH.

CONVERT Menü

>DEG wandelt die angegebene Zahl im Bogenmaß nach Grad um.

>RAD wandelt die angegebene Zahl im Grad Maß nach Bogenmaß um.

>HR konvertiert die angegebene Zahl im Stunden-Minutenformat in dezimales Stundenformat um.

>HMS wandelt die Zahl im dezimalen Stundenformat in Stunden - Minutenformat um.

XCORD speichert die X-Koordinate

YCORD speichert die Y-Koordinate

R berechnet zu den vorher eingegebenen X/Y-Koordinaten den Radius.

∢ berechnet dann auch den Winkel zwischen den beiden Geraden.

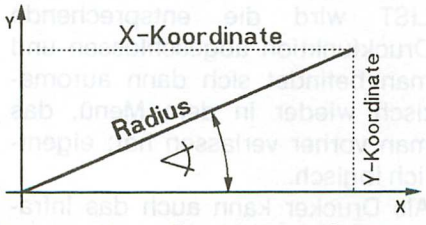


Bild 6: Koordinatentransform.

Natürlich kann man auch den Winkel und den Radius eingeben und sich daraus die X/Y-Koordinaten berechnen lassen.
Im Display und dem vierzeiligen Stack stehen nach den Berechnungen nicht nur die Zahlen, es steht auch im Klartext davor, was es ist: "RADIUS=10.000" oder "XCOORD=12.1234". Tippt man nun ein Rechenzeichen ein, so verschwindet der Text und es bleibt nur noch die Zahl stehen.

D/R kommt auch in diesem Menü vor, damit man jederzeit den Winkelmodus ändern kann.

PARTS Menü

IP schneidet den Nachkommateil einer Dezimalzahl ab.

FP eleminiert den Vorkommateil einer Dezimalzahl.

RND rundet die Zahl auf die angezeigte Anzahl von Stellen, die gerade im Display eingestellt ist. Intern wird ansonsten immer mit der vollen Genauigkeit gerechnet.

ABS bildet den Absolutbetrag einer Zahl, d.h. das Vorzeichen geht verloren.

%CH Menü

OLD speichert den "alten" Wert.

NEW speichert den "neuen" Wert.

%CH berechnet den prozentualen

Unterschied zwischen dem alten und dem neuen Wert.

Man kann hier jede beliebige Kombination von zwei Werten angeben und bekommt dann den dritten als Ergebnis, einfacher geht es kaum. Man muß sich nun nicht mehr überlegen, welcher Wert zuerst einzugeben ist:

Mein Kilo Äpfel kostet derzeit 2,99 DM, ich möchte den Preis um 5% erhöhen. Dazu tippe ich also 2,99 (Softkey) OLD, dann 5 (Softkey) %CH. Nun ein Druck auf den Softkey NEW und ich erhalte das Ergebnis in Form von "NEW=3,14". Alle Eingaben werden zur Kontrolle noch einmal im Klartext, d.h. mit der Text-Bezeichnung vorangestellt angezeigt, Benutzerführung in Perfektion.

BASE Menü

In diesem Menü ist es möglich in jedem der vier gebräuchlichen Zahlensysteme Zahlen einzugeben, diese zu verrechnen und anschließend wieder dasselbe zu verlassen.

Intern wird immer ganz normal gerechnet, außer daß der Nachkommateil fehlt.

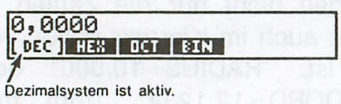


Bild 7: BASE-Menü

Beim Umschalten auf den HEX-Modus wird die Softkeyleiste mit ABCDEF belegt, damit man diese direkt eingeben kann. Mit EXIT geht's dann zurück in das vorhergehende Menü.

Die obere Displayzeile reicht natürlich für die Darstellung einer kompletten Binärzahl nicht mehr aus. Ist die Zahl zu lang, so wird sie links aus dem Display geschoben. Hier hilft nur die Funktion SHOW (SHIFT . Taste), die Zahl wird dann zweizeilig angezeigt.

Die Taste ± erzeugt das Zweierkomplement der Zahl, die Funktion Y^x ist die einzige, die man auch noch zu Berechnungen hinzuziehen kann.

PRINTER Menü

Hinter diesem Menü verbirgt sich alles, was sich auf die Verwendung des Infrarotdruckers bezieht, die einzige Ausnahme ist PRNT.

Menüs für numerische Funktionen

BASE	DEC	HEX	OCT	BIN		
PROB	X	Y	C X,Y	P X,Y	N!	RAN#
HYP	SINH	COSH	TANH	ASNH	ACOSH	ATNH
CONVERT	>DEG	>RAD	>HR	>HMS		MORE
	XCORD	YCORD	R	Δ	D/R	MORE
%CHG	OLD	NEW	%CH			
PARTS	IP	FP	RND	ABS		

Steuerungsmenüs

MODES	FIX	SCI	ENG	ALL	./,	MORE
	D/R	BEEP	PRNT			MORE
PRINTER	LIST	STK	REGS	TIME	MSG	TRACE

Bild 8: Allgemeine Menüs, eine Übersicht

Im anderen Steuerungsmenü (Bild 8) gibt es zwar die Einstellung der Betriebsart des Druckers, d.h. Batteriebetrieb oder Netzteilbetrieb, alles andere findet sich aber hier.

LIST druckt die gerade aktive Einstellung aus, siehe dazu den Absatz nach dieser Befehlsliste.

STK druckt den History-Speicher, d.h. den 4-zeiligen Stack aus.

REGS druckt den Inhalt der Speicherregister 0-9.

TIME druckt das aktuelle Datum und die aktuelle Uhrzeit.

MSG druckt eine beliebige Meldung zum Kommentieren von Druckerlistings.

TRACE protokolliert ab sofort alle Rechenaktionen auf dem Drucker mit, auch das Wiederausschalten desselben.

Das PRINTER-Menü kann von anderen Menüs aus aktiviert werden, ohne diese zu unterbrechen. Dies ist für die Funktion des LIST-Befehls notwendig.

Befinde ich mich also im TIME-Menü, so werden die Alarme ausgedruckt. Befinde ich mich dagegen im Statistik Menü, so wird die gerade aktuell bearbeitete Liste ausgedruckt, damit man sie sich übersichtlicher auf Papier ansehen kann.

Nach dem Drücken des Softkeys LIST wird die entsprechende Druckfunktion abgeschlossen und man befindet sich dann automatisch wieder in dem Menü, das man vorher verlassen hat; eigentlich logisch...

Als Drucker kann auch das Infrarotinterface [PRISMA 3/89] dienen.

Statistik

Dieses Menü ermöglicht die statistischen Berechnung mit einer oder auch zwei Variablen, wobei diese in Form von Listen vorliegen müssen, die sehr komfortabel editiert werden können. Ebenso können Listen für die spätere Verwendung unter einem Namen abgespeichert werden. Die Namenswahl ist beliebig, allerdings auf 5 Buchstaben beschränkt.

Für den späteren Aufruf werden diese dann in einer Liste von Softkeys zum Aufrufen zur Verfügung gestellt, wenn man dies wünscht, Eintippen ist also überflüssig.

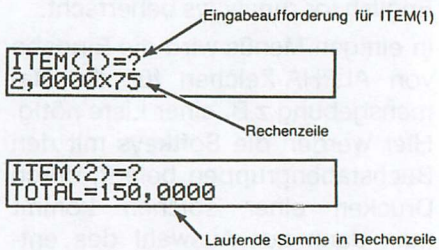


Bild 9: Eingabe einer Liste

Wie man in Bild 10 sehen kann, ist das Statistik-Menü etwas umfangreicher. Aber keine Angst, die Ingenieure von HP haben alles dafür getan, daß dem Anwender das Arbeiten mit den Statistik-Funktionen so einfach wie möglich gemacht wird.

In Bild 9 sieht man, wie das Display aussieht, wenn man SHIFT STAT drückt. Es erscheint der Wert des letzten eingegebenen Wertes der zuletzt bearbeiteten Liste, ITEM ist einfach der englische Ausdruck für Position in einer Liste.

In unserem Bild 9 ist noch keine Liste vorhanden, die Position 1 der Liste ist also leer, deshalb das Fragezeichen hinter dem "=".

Ich kann jetzt erst einmal den Wert berechnen, der an dieser Stelle der Liste eingefügt werden soll; sobald ich die Taste INPUT betätige wird der aktuell in der Eingabezeile stehende Wert als neuer Wert der Liste übernommen, es erscheint "ITEM(2)=?". Es wird auch immer sofort die Gesamtsumme der Liste ermittelt und in Zeile 2

des Displays mit "TOTAL=xxx" angezeigt.
Durch Drücken der Taste EXIT wird das Statistik-Menü wieder angezeigt, dies ist die erste Zeile unterhalb von SHIFT STAT in Bild 10.

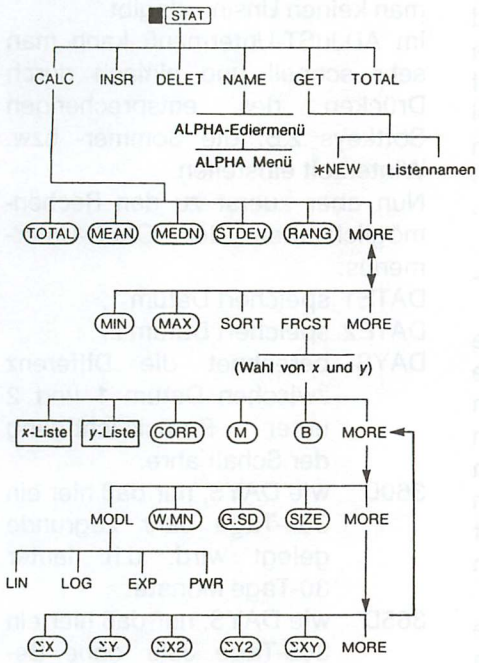


Bild 10: Statistik Menüstruktur

Hier kurz die Erklärungen für die Befehle der ersten Softkeys des Statistik-Menüs:

INSR fügt ein neues Listenelement vor das aktuelle ein.

DELET löscht das gerade angezeigte Element der Liste.

NAME gibt der gerade in Bearbeitung befindlichen Liste einen Namen, mit dem man später die Liste jederzeit wieder zur Bearbeitung aufrufen kann.

GET belegt die Softkeys mit den Namen aller im Speicher befindlichen Listen. Damit kann man durch einfachen Tastendruck dieselbe zum Bearbeiten holen.

TOTAL ermittelt die Summe aller Listenelemente, wie dies bei der Eingabe eines neuen Listenelements sowieso schon geschieht.

Die Anzahl der im Speicher ablegbaren Listen ist nur durch den Speicher begrenzt, dies waren maximal 7 kByte. Ist die Liste der Namen größer als 5, so wird einfach der rechte Softkey mit "MORE" belegt und die Liste der Namen in der zweiten Ebene fortgesetzt, so einfach ist das.
Zum Löschen der aktuellen Liste tippt man einfach SHIFT INPUT, also CLEAR DATA. Es erfolgt die Frage "CLEAR THE LIST", womit nur der Inhalt der Liste gemeint ist.
Der Name derselben bleibt also erhalten. Mit den Softkeys "YES" bzw. "NO" wählt man das Gewünschte, worauf die Frage "ALSO CLEAR LIST NAME" wieder mit der Softkey-Belegung "YES" und "NO" der Beantwortung harrt.
So stellt man sich eine idiotensichere Benutzerführung vor, man benötigt fast nirgends das Handbuch zu Rate zu ziehen.
Die Betätigung des ersten Softkeys im STAT-Menü erschließt uns den Weg zu den weiteren statistischen Berechnungsmöglichkeiten, wie in Bild 10 zu sehen ist.
Man kann jederzeit die Liste mit den Pfeil-Tasten rauf bzw. runterlaufen lassen, um sich andere Elemente anzusehen.

CALC-Softkey

Hier stehen dem Anwender verschiedene Möglichkeiten zur Berechnungen aus der aktuellen Liste zur Verfügung:

TOTAL berechnet wieder die Gesamtsumme,

MEAN berechnet den arithmetischen Mittelwert,

MEDN berechnet den Median,

STDEV berechnet die Standardabweichung,

RANG berechnet die Differenz zwischen dem kleinsten (=MIN) und dem größten (=MAX) Wert,

MIN ermittelt den kleinsten Wert,

MAX ermittelt den größten Wert der aktuellen Liste.

SORT Sortiert die aktuelle Liste in aufsteigender Reihenfolge, absteigende Reihenfolge ist nicht möglich.

Kurvenanpassung

Durch das Drücken des Softkeys FRCST werden statistische Berechnungen mit zwei Listen möglich, dazu zählt auch eine Kurvenanpassung mit vier verschiedenen Typen, dazu aber gleich.
Erst die normalen Funktionen:

CORR berechnet den Korrelationskoeffizient,

W. MN berechnet den gewogenen Mittelwert der X-Werte, wobei die Y-Werte als Gewichte oder Häufigkeiten dienen.

G. SD berechnet die Standardabweichung einer Gruppe von Zahlen (X-Werte), welche mit der spezifizierten ganzzahligen Häufigkeit (Y-Werte) auftreten.

SIZE zeigt die Anzahl der Elemente der Listen.

ΣX berechnet die Summe (TOTAL) der X-Werte,

ΣY berechnet die Summe (TOTAL) der Y-Werte,

ΣX2 berechnet die Summe der quadrierten X-Werte,

ΣY2 berechnet die Summe der quadrierten Y-Werte,

ΣXY berechnet die Summe der Produkte der X- und Y-Werte.

Mit dem Softkey MODL kann man eine der vier Kurvenanpassungsmodelle wählen:

Linear	$y=m*x+b$
Exponentiell	$y=b*e^{m*x}$
Logarithmisch	$y=b+m*\ln(x)$
Potenz	$y=b*x^m$

Der Softkey M berechnet die Konstante M, der Softkey B desgleichen die Konstante B.

Das SOLVE-Menü

Der HP27S gibt uns noch ein weiteres hervorragendes Werkzeug in die Hand, wenn es darum geht, immer wiederkehrende Formeln mit geringfügig geänderten Werten auszurechnen.
Man muß hier nur ein einziges Mal die Formel eingeben und dann die Werte der Variablen, um ein Ergebnis zu erhalten. Will man jetzt das Ergebnis für die Änderung einer der Variablen haben, so tippt man nur noch den neuen Wert der zu ändernden Variable ein, fertig. Man erspart sich das nochmalige Eintippen aller alten Werte, diese sind noch gespeichert.
Zur Verdeutlichung dieser recht mächtigen und dennoch einfach zu handhabenden Funktion ziehe ich das Beispiel aus dem Handbuch heran, wo es um die Gleichung des freien Falls geht:
*S=V0*TIME-0,5*G*TIME^2*

S ist der zurückgelegte Weg eines Körpers im freien Fall, V0 ist die Anfangsgeschwindigkeit, G ist die Konstante der Erdbeschleunigung und TIME ist die Zeit, die der freie Fall dauern soll.
Zwei Fragen gilt es zu beantworten:

1. Wie tief fällt ein Körper in 5 Sekunden?
2. Wie lange kann es dauern, bis der Körper 500m zurückgelegt hat?

Wir tippen SHIFT SOLVE ein, dann den Softkey NEW. Nun geben wir die Gleichung, genauso, wie sie oben geschrieben steht, ein und drücken INPUT:

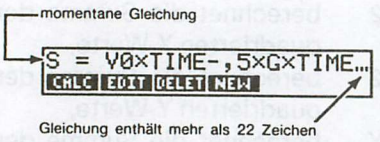


Bild 11: Gleichung freier Fall

Drücken wir jetzt den Softkey CALC, so analysiert der HP27S die Gleichung auf syntaktische Korrektheit, es könnte ja ein Klammerfehler drin sein oder sonst ein mathematischer Unsinn.
Bild 12 zeigt das Ergebnis:

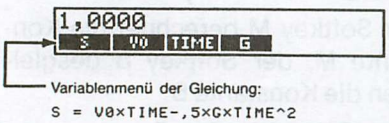


Bild 12: Analysierte Gleichung

Wir sehen nun, daß die Softkeys mit den einzelnen Variablen belegt sind. Wir geben jetzt also alle Daten bis auf die gesuchte ein, ein Druck auf die gesuchte Variable bringt uns das gewünschte Ergebnis.
Der Clou an der Sache ist der, daß es dabei garnicht darauf ankommt, daß die gesuchte Variable die auf der linken Seite der Gleichung sein muß, es kann jede beliebige sein!
Ergo, man spart sich das Umstellen der Gleichung, was meist doch nur Fehler produziert.
Gehen wir also an die Lösung der ersten Frage:

Tasten	Anzeige	Kommentar
9,8067 G	G=9,8067	Konstante speichern
0 V0	V0=0,0000	Anfangsgeschw.
5 TIME	TIME=5,0000	Zeitdauer
S	S=-122,5838	zurückgelegte Strecke

Für die zweite Frage brauchen nun die Konstante G und die Anfangsgeschwindigkeit nicht mehr neu eingegeben werden:

Tasten	Anzeige	Kommentar
500 ±	S=-500,0000	zurückzulegende Strecke von 500m
TIME	TIME=10,0981	

Für die Ermittlung der Zeit wird ein iterativer Prozess benutzt, da die Variable TIME ja zweimal auf der rechten Seite vorkommt. Bei diesem Annäherungsverfahren kann man dem Rechner zuschauen, die Zwischenergebnisse werden laufend im Display angezeigt. Richtig befriedigend jemanden arbeiten zu sehen.
Man kann soviele Gleichungen wie der Speicher faßt eingeben. Diese werden in einer Liste, die mit den Cursortasten durchrollt werden kann, zugänglich gemacht. Man kann den Gleichungen natürlich auch Namen verpassen, damit man diese leichter identifizieren kann.
In einer Gleichung können alle dem HP27S bekannten Funktionen und Rechenoperationen auftreten, logische Verknüpfungen mit AND, OR, XOR und NOT sind ebenso zugelassen wie Funktionen aus dem Zeitmenü TIME, zu dem ich gleich komme. Auch bedingte Ausdrücke mit IF sind möglich, d.h. es lassen sich in Gleichungen auch logische Fälle mit einarbeiten, eine Quelle der unerschöpflichen Möglichkeiten.
So ganz nebenbei lassen sich auch Nullstellen mit dem Löser suchen, das nur der Vollständigkeit halber.

TIME-Menü

In dieses Menü will ich nicht in's letzte Detail hinabsteigen, das würde hier zu weit führen. In Bild 13 kann man sich die Menüstruktur des TIME-Menüs anschauen, ich werde kurz in der folgenden Tabelle einzelne interessante Befehle herausgreifen.
Schaltet man in das TIME-Menü, so wird ähnlich wie beim HP41CX die laufende Uhrzeit mit dem aktuellen Datum und dem Wochentag im Klartext im Display angezeigt.
Grundsätzliche Funktionen dieses Menüs sind die Einstellung der internen Uhr mittels des Untermenüs SET. Man kann sich dieses anschauen, wenn man der rechten Linie in Bild 13 folgt. Die mittleren 3 Softkeys dienen der Einstellung der Datums- sowie Zeitformate der beiden Kontinente.
Die Help-Taste informiert über den gerade gültigen Syntax der Zeit und der Datumseinstellung, damit man keinen Unsinn eingibt.
Im ADJuST-Untermenü kann man sehr schnell und *einfach durch* Drücken des entsprechenden Softkeys z.B. die Sommer- bzw. Winterzeit einstellen.
Nun aber zuerst zu den Rechenmöglichkeiten des CALC-Untermenüs:

DATE1 speichert Datum 1,
DATE2 speichert Datum 2,
DAYS berechnet die Differenz zwischen Datum 1 und 2 unter Berücksichtigung der Schaltjahre.
360D wie DAYS, nur daß hier ein 360-Tage Jahr zugrunde gelegt wird, d.h. lauter 30-Tage Monate.
365D wie DAYS, nur daß hier ein 365-Tage Jahr ohne Berücksichtigung von Schaltjahren zugrunde gelegt wird.
TODAY zeigt das aktuelle Datum an und legt es im Stack ab, damit es für die Eingabe in DATE1 oder DATE2 verwendet werden kann.

Die Reihenfolge der Eingaben bei den Datumsberechnungen ist beliebig, es gilt die Regel, daß nach zwei Eingaben die dritte berechnet werden kann. Ich kann also fragen, wie viele Tage ich schon diese Welt belaste, indem ich für DATE1 meinen Geburtstag und für DATE2 das aktuelle Datum (TODAY) eingeben und dann den Softkey DAYS drücke.
Eine andere Möglichkeit wäre die Frage, welcher Tag in 100 Tagen ist: TODAY drücken für das aktuelle Datum, DATE1 zur Eingabe desselben; nun die gewünschte Differenz von 100 Tagen mit DAYS eingeben und ohne Eingabe DATE2 drücken. Ergebnis: "DATE2= 13.03.1991 WED".
Na, an welchem Tag habe ich das wohl geschrieben ?
Auch in diesem Menü dient die Funktion CLEAR DATA der Löschung von Variablen, hier DATE1, DATE2 und DAYS.

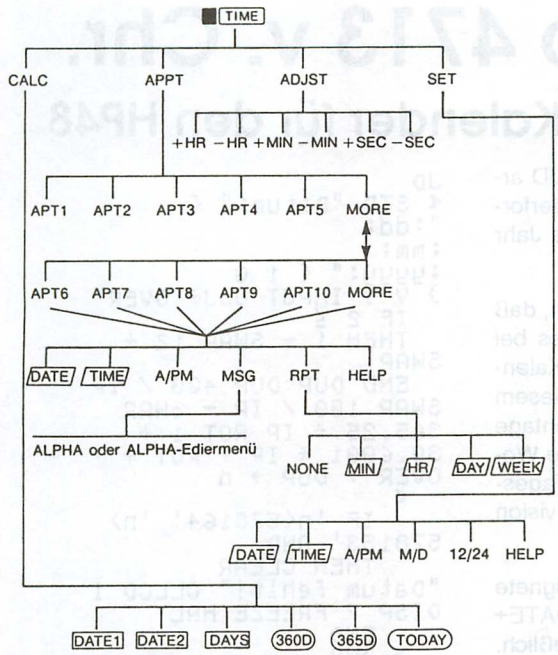


Bild 13: TIME-Menüstruktur

Die Uhrzeit anzeigen und ein paar Daten berechnen kann ja jeder, so ein kleiner Terminkalender kann aber durchaus manchmal wertvolle Dienste leisten. Auch hier kann uns der HP27S weiterhelfen.

Bis zu 10 Alarme lassen sich gleichzeitig im Rechner speichern. Die Menüstruktur ist unter dem Softkey APPT zu erreichen.

Zu jedem Alarm wird Datum und Uhrzeit eingegeben, eine Meldung ist optional, d.h. man kann eine Textmeldung eingeben oder auch nicht. Ein Alarm ohne Meldung zeigt einfach Datum und Uhrzeit an, "gepiepst" wird in jedem Fall.

Wer jeden Morgen um die gleiche Uhrzeit geweckt werden will, der kann eine beliebige Wiederholzeit eingeben, die von 1 Minute bis zu 999999999999 Wochen...

So, mehr hat die Uhr hier nicht zu bieten, das dürfte für den Anfang auch reichen. Wenden wir uns also unserem letzten Kapitel zu, dem

TVM-Menü

TVM heißt neudeutsch *T*ime *V*alue of *M*oney oder Zeitwert des Geldes.

Wat is'n dat, werden jetzt wieder viele, wohl zu Recht, fragen. Ganz einfach ausgedrückt ist hiermit die Möglichkeit gemeint, den Wert des Geldes über einen bestimmten Zeitraum ermitteln zu lassen bzw. sich z.B. einen Tilgungsplan aufstellen zu lassen. Natürlich tut's auch das Sparbuch.

Klingt das wieder furchtbar monströs, vor allem, wenn man an die undurchsichtigen Computerausdrucke vieler Geldhaie oder auch "Vermögensberater" denkt. Wie wäre es denn, wenn man ganz einfach zu Hause und ohne Programmieren einmal nachrechnen lassen kann, ob das so schön vorformulierte denn überhaupt der Realität entsprechen kann.

Keine Angst, der HP27S steht auch bei diesem Thema mit seiner Softkeystruktur gerade Ungeübten ohne große Vorkenntnisse zu Diensten.

Bild 14 zeigt die Struktur des TVM-Menüs als Übersicht.

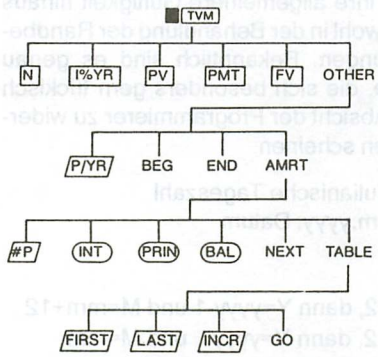


Bild 14: TVM-Menüstruktur

Die nachfolgende Tabelle ist dafür gedacht keine offenen Fragen über die Eingabemöglichkeiten offen zu lassen, ohne in das letzte Detail vorzustoßen. Dafür muß dann schon das Handbuch herhalten.

Ich werde nach der Tabelle für die Allgemeinheit ein Beispiel exerzieren, das dürfte einfacher für die meisten sein, als jeden Befehl der Tabelle zu durchstöbern.

- N speichert oder berechnet die Anzahl der vorzunehmenden Zahlungen; die Einheit ist beliebig.
- I%YR speichert oder berechnet den nominalen jährlichen Zinssatz in Prozent.
- PV Speichert oder berechnet den Barwert einer Reihe zukünftiger Zahlungen.
- PMT speichert oder berechnet den Betrag der wiederkehrenden periodischen Zahlung.
- FV speichert oder berechnet den Endwert unter Berücksichtigung der Verzinsung früherer Zahlungen.
- P/YR speichert die gewünschte Anzahl der Zahlungenm pro Jahr (1...999).
- BEG setzt den BEGinn-Modus, der bei einer vorschüssigen Zahlungsweise verwendet wird.
- END setzt den END-Modus, der bei einer nachschüssigen Zahlungsweise verwendet wird.
- AMRT zeigt das Menü zur Berechnung eines Tilgungsplanes.

Das Handbuch gibt ausführlich Beispiele zu folgenden Fällen:

Kredit vom Kreditnehmer-Standpunkt, Kredit vom Kreditgeber-Standpunkt, Leasingzahlungen zu Beginn jeder Periode, Konto-Einzahlungen am Ende jeder Periode, Darlehensberechnung am Beispiel eines Autokredits sowie Hypothekendarlehen mit und ohne Restschuld.

Man kann das Handbuch wie ein kleines 1x1 der Finanzmathematik benutzen.

Nun aber unser Beispiel:

Man zahlt 2.000.- DM auf ein Sparkonto mit 7,2% jährlichem Zins ein. Wie lange dauert es, bis ich mindestens 3.000.- DM erreicht habe ?

Tasten	Anzeige	Kommentar
1 P/YR	1 P/YR	1 Verzinsungsperiode/Jahr
7,2 I%YR	I%YR=7,20	jährlicher Zinssatz
2000± PV	PV=-2.000,00	Einzahlungsbetrag
0 PMT	PMT=0,00	keine periodischen Zahlungen
3000 FV	FV=3.000,00	erwarteter Endwert
N	N=5,83	ist die Anzahl der Jahre, bis 3.000.- DM erreicht sind.

Will man jetzt schnell wissen, was nach vollen 6 Jahren auf dem Konto ist, so muß man nur noch folgendes tun:

Tasten	Anzeige	Kommentar
6 N	N=6,00	Anzahl Jahre
FV	FV=3.035,28	Kontostand nach 6 Jahren Wartezeit

Unter dem Softkey AMRT verbirgt sich noch die interessante Möglichkeit, sich einen Tilgungsplan anzeigen bzw. ausdrucken zu lassen. Das ist das, was einem die

Bank vorlegt, wenn man einen Kredit für sein Haus aufgenommen hat und wissen will, zu welchem Zeitpunkt man wieviel Zinsen und wieviel Tilgung des Kredits zahlt.

#P speichert die Anzahl der Rückzahlungen für die Tilgungsperiode und startet die Berechnung des Tilgungsplans.

INT zeigt den Zinsanteil der geleisteten Zahlungen an.

PRIN zeigt den Tilgungsanteil der geleisteten Zahlungen an.

BAL zeigt die Restschuld am Ende der betrachteten Tilgungsperiode an.

NEXT berechnet die Tilgungspalnwerte der nächsten Zahlungsreihe.

TABLE verzweigt zum Untermenü zum Ausdrucken des Tilgungsplanes.

Na, neugierig geworden. Das Ganze ist eigentlich garnicht so schwer, wie es einem die Bankleute immer verkaufen wollen, die können auch bloß die vier Grundrechenarten.

Das Schlimme an diesen Tilgungsplänen ist die große Anzahl von Zahlen, jeder Schritt (Monat) wird aber im Klartext ausgewiesen, die Bezeichnungen sind leider in Englisch, das Handbuch übersetzt sie aber in's Deutsche.

Alle Beispiele sind ebenso wie das gesamte Handbuch in Deutsch, etwa ein Drittel des gesamten Handbuchs beschäftigt sich in eigenen Kapitel ausschließlich mit sehr praxisnahen Beispielen, ganz typisch Hewlett Packard.

Das Handbuch ist didaktisch excellent aufgebaut und gegliedert und bietet für absolute Laien mit seinen gut durchdachten, praxisnahen Beispielen zu allen Befehlen oder Funktionen. Mit über 250 Seiten kann man sich auch nicht über zu spartanische Aufmachung beklagen; an Grafiken zur Illustrierung der Beispiele oder Sachverhalte wurde nicht gespart.

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

Datum ab 4713 v. Chr.

im julianischen Kalender für den HP48

Die beiden Programme DAT und JD arbeiten nach den bekannten Kalenderformeln [1]. Zu beachten ist nur: Das Jahr 135 v.Chr. ist hier das Jahr -134.

Die Meldung "Datum fehlt" zeigt an, daß ein Datum gewählt wurde, welches bei Einführung des Gregorianischen Kalenders übersprungen wurde. Bei diesem Sprung aber zählte man die Wochentage einfach weiter. So ergeben sich die Wochentage aus der Julianischen Tageszahl einfach aus dem Rest der Division durch sieben.

Der für die Berechnung von JD geeignete eingebaute Befehl DDAYS ist wie DATE+ gültig bis zum 15.10.1582 einschließlich. Dies trifft übrigens auch für den HP41CX (HP41 Time Modul) zu.

Ein gewisser Wert dieser Programme über ihre allgemeinere Gültigkeit hinaus liegt wohl in der Behandlung der Randbedingungen. Bekanntlich sind es genau diese, die sich besonders gern tückisch der Absicht der Programmierer zu widersetzen scheinen.

JD: Julianische Tageszahl
dd.mm.yyyy: Datum

JD:

mm≤2, dann Y=yyyy-1 und M=mm+12
mm>2, dann Y=yyyy und M=mm
bis mit 04.10.1582: B = -2
ab mit 15.10.1582: B = IP(Y/400) - IP(Y/100)

JD = IP(365.25*Y) + IP(30.6001 * M+1) + D+B+1720996.5

Datum:

a=IP(JD+0.5)
a>2299161: c=a+1524
a≤2299161:b= IP((a - 187216.25) / 36524.25),
c=a+b-IP(b/4)+1525
b=IP((c-122.1)/365.25)
e=c-IP(365.25*b)
f=IP(c/30.6001)
dd=e-IP(30.6001*f)+FP(JD+0. 5)
mm= f-1-12*IP(f/14)
yyyy=b-4715-IP((mm+7)/10)

Literatur:

[1] O. Montenbruck, Grundlagen der Ephemeridenrechnung, München 1984

Dr. G. Heilmann
Obernhofer Straße 15
5408 Seelbach

```
JD
« STD "Datum:" {
":dd:
:mm:
:yyyy:" { 1 0
} V } INPUT OBJ→ OVER
  IF 2 ≤
  THEN 1 - SWAP 12 +
SWAP
  END DUP DUP 400 / IP
SWAP 100 / IP - SWAP
365.25 * IP ROT 1 +
30.6001 * IP + ROT +
OVER + DUP → n
  «
    IF 'n<578164' 'n>
578153' AND
    THEN CLEAR
"Datum fehlt!" CLLCD 1
DISP 7 FREEZE HALT
    END
  » DUP
  IF 578153 ≤
  THEN OVER 2 + -
  END 1720996.5 + DUP IP
7 / FP 7 * 1 TRNC CEIL →
x
  «
    CASE x 5 ==
      THEN "Sonntag"
      END x 6 ==
      THEN "Montag"
      END x 0 ==
      THEN "Dienstag"
      END x 1 ==
      THEN "Mittwoch"
      END x 2 ==
      THEN "Donnerstag"
      END x 3 ==
      THEN "Freitag"
      END x 4 ==
      THEN "Samstag"
      END
    END
  » SWAP 'JD' →TAG →STR
"
" + SWAP + CLLCD 3
DISP 7 FREEZE CLEAR
»

# 22189d
656

DAT
«
"Julianische
    Tageszahl:"
PROMPT 0 FIX .5 + DUP IP
DUP
  IF 2299161 <
  THEN 1524 +
  ELSE DUP 1867216.25 -
36524.25 / IP DUP ROT +
SWAP 4 / IP - 1525 +
  END 365.25 OVER 122.1
- OVER / IP DUP ROT * IP
NEG ROT + DUP 30.6001
SWAP OVER / IP SWAP OVER
DUP 1 - SWAP 14 / IP 12
* - 5 ROLL 4715 - OVER 7
+ 10 / IP - →STR SWAP
→STR SWAP + 5 ROLLD * IP
- SWAP FP + →STR SWAP +
"Datum: " SWAP + CLLCD 1
DISP 1 FREEZE CLEAR
»

# 708d
423
```

SORTIEREN

von Matthias Rabe

Die wichtigste Eigenschaft (nach der korrekten Funktion) eines Sortieralgorithmus' ist seine Geschwindigkeit, die in irgendeiner Form proportional zum Umfang der zu sortierenden Daten ist. Zur Beurteilung der Geschwindigkeit führe ich zuerst eine geeignete Notation ein, mit der sich der etwas nebulöse intuitive Ausdruck "... proportional zu ..." präzisieren läßt. Wer sich nur für die Quintessenz interessiert, kann den nun folgenden mathematischen Formalismus auch gerne überspringen.

Zuerst übernehmen wir die Konvention, daß $T_f(x)$ für die Anzahl der Operationen, die notwendig sind, um $f(x)$ auszuwerten, steht. Das 'T' steht für time (oder tempora), da die Anzahl der Operationen generell proportional zur Zeit ist, die für ihre Auswertung benötigt wird.

Als zweites benutzen wir die O-Notation, die es uns erlaubt, das Zeitverhalten einer Operation f so zu beschreiben:

$$T_f(x) = O(n)$$

oder: $T_f(x) = O(n^2)$,

wobei n die Größe von x ist.

Das 'O' steht für "order of at most", was präzise bedeutet: Wenn $g(n)$ eine Funktion ist, dann bedeutet:

$$g(n) = O(h(n)) \ ,$$

daß eine Konstante M existiert, mit

$$|g(n)| \leq M|h(n)|$$

für jedes positive n .

Dies beschreibt $g(n)$ nicht präzise, aber besagt, daß $g(n)$ beschränkt ist, durch eine Funktion proportional zu $h(n)$.

$|g(n)|$ ist der Absolutbetrag von $g(n)$. In unserem Fall ist $g(n)$ natürlich immer positiv, da es sich ja um die Ausführungszeit eines Programms handelt. Allerdings ist es eine in der Mathematik übliche Definition, und deshalb habe ich sie übernommen.

Als Beispiel nehmen wir an, daß $g(n)$ ein Polynom m-ten Grades sei:

$g(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_m n^m$.

Dann haben wir:

$$g(n) = O(n^m) \ .$$

Ein geeignetes M wäre in diesem Fall:

$$M = |a_1| + |a_2| + \ldots + |a_m| \ .$$

Die O-Notation muß allerdings mit Vorsicht angewandt werden. Für den Fall, daß

$f(n) = O(h(n))$ und $g(n) = O(h(n))$

impliziert dies nicht, daß auch

$$f(n) = g(n)$$

gilt. Insbesondere besagt dies nicht, daß z.B. bei

$$f(n) = O(n)$$

und $g(n) = O(n^2)$

für ein bestimmtes $n>1$ gilt, daß

$$f(n) < g(n) \ ,$$

nur weil n für $n>1$ kleiner als n^2 ist.

Was haben wir dann also von dieser nun mühsam erarbeiteten Notation? In erster Linie ist es interessant, das asymptotische Verhalten zu beschreiben. Sei z.B. im obigen Beispiel die Konstante M zur Funktion f 20 und zur Funktion g 2. Dann wird für kleine n die Funktion g schneller ausgewertet: ist $n = 5$, so benötigt die Funktion g nur 50 Operationen (Zeiteinheiten) (25^2) , während die Funktion f 100 Operationen (205) benötigt, also nur halb so schnell ist. Für größere n jedoch ist f sehr schnell viel besser als g: angewandt auf 20 benötigt f 400 Operationen, während g nun 800 Operationen ausführen muß. Für wachsendes n wird der Vorteil von f immer größer.

Wird zur Beurteilung eines Programms nur die O-Notation herangezogen, so spricht man von asymptotischer Analyse. In der Mathematik beschreibt die Asymptote das Verhalten einer Funktion, wenn sie sich "dem Unendlichen nähert". Asymptotische Analyse beschreibt in ähnlicher Weise das Verhalten eines Programms wenn es "ins Unendliche" geht, das bedeutet, wenn der Umfang der Daten immer größer und größer wird.

In vielen Fällen ist die asymptotische Analyse ausreichend. Generell ist die Zeit zur Ausführung eines Progrqamms proportional zur Anzahl auszuführender Operationen, unabhängig von der benutzten Implementierung. Ist die Ordnung eines Programms geringer, als die eines anderen, so wird es unter allen Implementierungen schneller laufen -für genügend große Eingaben-. Deshalb liefert die asymptotische Analyse Implementationsunabhängige Informationen.

Andererseits werden manchmal mehr Informationen benötigt. Insbesondere, wenn zwei Programme gleicher Ordnung verglichen werden sollen. Auch, wenn die Datenmengen nicht sehr umfangreich sind, ist es nicht sinnvoll, sich auf das asymptotische Verhalten zu fixieren und es ist wichtig, auch die Größe der Proportionalitätsfaktoren zu kennen. In diesen Fällen ist ein genaues Abzählen der Arbeitsschritte sinnvoll.

Nach soviel Vorarbeit und Bla Bla komme ich jetzt endlich zum interessanten Punkt des Artikels: die

Beschreibung der Sortieralgorithmen.

Sechs Algorithmen, die sich in drei Gruppen unterteilen lassen, habe ich untersucht: Insertion Sort und zwei Varianten, Merge Sort und das berühmte Quick Sort nebst einem Derivat. Im folgenden benenne ich die Algorithmen mit den Namen, die ich den entsprechenden Programmen gegeben habe.

Ich zähle bei allen Algorithmen nur die benötigten Vergleiche und nicht die Zeit, die die Programme beschäftigt waren, da sich die Programme nahezu beliebig optimieren lassen, die Anzahl der Vergleiche jedoch bleibt. Außerdem ist der Aufwand für einen Vergleich von den zu sortierenden Daten abhängig. So lassen sich Integerzahlen in der Regel auf jedem Computer mit einer einzigen CPU-Instruktion vergleichen. Um z.B, Zeichenketten mit deutschen Umlauten zu vergleichen, ist jedoch schon ein recht aufwendiges Programm nötig, das auch entsprechend mehr Zeit in Anspruch nimmt, sodaß der Overhead, der durch das eigentliche Sortierprogramm entsteht, eventuell vernachlässigt werden kann.

ISORT

Als erstes wäre da ISORT, das Sortieren durch einfügen. ISORT startet bei dem zweiten Element und vergleicht es mit dem ersten. Ist es größer als das erste, so wird zum nächsten Element übergegangen, ansonsten das zweite Element vor das erste gesetzt. Das nächste Element wird nun mit dem vorherigen verglichen und, falls es nicht größer ist, an die passende Stelle in die bereits sortierten Elemente eingefügt. Dies geschieht dadurch, daß schrittweise das einzufügende Element rückwärts mit allen bereits sortierten Elementen verglichen wird, bis eins kleiner ist, oder wir am Anfang der Liste angekommen sind. Dort wird das Element dann eingefügt.

Ist die Liste bereits sortiert, so wird jedes Element vom zweiten bis zum letzten nur mit seinem Vorgänger verglichen. Es werden also nur $(n-1)$ Vergleiche gemacht.

Sind nur wenige Elemente falsch platziert, so werden für jedes Element so viele zusätzliche Vergleiche benötigt, wie es Positionen vom richtigen Platz entfernt ist. Es ist also egal, ob ein falsches Element weit vorne oder weit hinten steht. Es kommt nur darauf an, wie weit es bewegt werden muß.

Im durchschnittliche Fall, also bei zufällig verteilten Daten, braucht im Mittel nur bis zum $(k-1)/2$-ten Element geschaufelt zu werden. Daraus ergibt sich ein Aufwand von $(n-1)/2 \cdot (n-1)/2$ also $(n-1)^2/4$ Vergleichen.

Im ungünstigsten Fall, wenn die Liste in entgegengesetzter Reihenfolge sortiert ist, muß jedes Element an den Anfang der

Liste geschaufelt werden, was für das k-te Element bedeutet, daß es mit allen (k-1) Vorgängern verglichen werden muß. Da k von 2 bis n geht, bedeutet dies, daß insgesamt $(n-1)^2$ Vergleiche durchgeführt werden müssen.

Also gilt:

$$T_{ISORT}(n) = O(n^2)$$

BISORT

SORT kann deutlich verbessert werden, wenn man die Methode des Einfügens überarbeitet, da hierin die meiste Zeit verbraten wird. Da die vorherigen Elemente bereits sortiert sind, kann mit binärer Suche die richtige Position für das einzufügende Element schnell gefunden werden, da die binäre Suche in einer n-elementigen Liste nur $\log_2 n$ Schritte benötigt. Die lineare Suche aus dem vorherigen Algorithmus benötigt jedoch bis zu n Schritte.

Bei der binären Suche wird jeweils das mittlere Element der Liste genommen und mit dem einzufügenden Datum verglichen. Abhängig von diesem Testergebnis wird dann in der linken oder der rechten Hälfte der Liste wie zuvor fortgefahren, bis die Position, in die das Datum eingefügt werden soll, gefunden ist.

Es müssen insgesamt etwa $n \cdot \log_2 (n/2)$ Vergleiche ausgeführt werden. Also gilt:

$$T_{BITORT}(n) = O(n \log_2 n)$$

BITSORT

Bei BISORT ist leider die schöne Eigenschaft von ISORT, nahezu sortierte Listen sehr viel schneller zu sortieren, verloren gegangen. War ein Element größer als sein Vorgänger, so wurde nur ein Vergleich durchgeführt. Bei BISORT war das allerdings egal. Es muß immer die Liste ganz durchsucht werden.

Wird BISORT so modifiziert, daß bei jedem Element vor dem Einfügen geprüft wird, ob es nicht schon an seiner richtigen Stelle steht, so entsteht ein Binary Insertion Sort mit Test. Bei sortierten Listen ist es genau so schnell wie ISORT. Ist jedoch ein Element nicht an seinen Platz, so wird dieses mit der Geschwindigkeit des BISORT eingefügt. Allerdings wurde dann ein Test zuviel gemacht, wodurch BITSORT bei nicht gut sortierten Listen gegenüber BISORT einen Overhead von bis zu n Vergleichen hat. Auch reagiert es empfindlich darauf, ob ein Element nach vorne oder nach hinten bewegt werden muß. Muß es nach vorne einsortiert werden, so ist das mit einem Einfügungsschritt erledigt. Muß es jedoch nach hinten verschoben werden, so steht es für so viele nachfolgende Elemente vor einem kleineren, wie es nach hinten verschoben werden muß. Jedes dieser Elemente muß dann mit $\log_2 (k-1)$ Schritten eingefügt werden, obwohl es reichen würde, einfach das k-te mit dem (k-1)-ten Element zu vertauschen. Will man in eine sortierte Liste neue Elemente einfügen, so empfiehlt es sich, diese an das Ende der Liste anzuhängen.

Im ungünstigsten Fall müssen etwa $(n-1)\log_2 (n-1) + (n-1)$ Vergleiche durchgeführt werden. "Etwa" deshalb, weil sich eine Liste mit einer ungeradzahligen Anzahl Elemente nicht in zwei gleich große Hälften teilen läßt. Dadurch kann die Zielposition eines Elementes mal etwas früher, mal etwas später gefunden werden.

Das asymptotische Verhalten ist also:

$$T_{BITSOTR}(n) = O(n \log_2 n)$$

MSORT

Zu einer anderen Klasse von Sortieralgorithmen gehört Merge Sort. Es teilt die zu sortierende Liste in zwei gleich große Teillisten auf, die durch einen rekursiven Aufruf von MSORT sortiert werden, und mischt diese dann so: Zuerst werden die beiden ersten Elemente verglichen. Das kleinere Element der beiden wird übernommen. Im nächsten Schritt wird das verbliebene Element der einen Liste mit dem nun ersten der anderen verglichen und das kleinere von beiden ausgewählt. Dies wiederholt sich, bis eine Liste keine Elemente mehr enthält. Dann werden die restlichen der anderen Liste an die bereits gemischte Liste angehängt.

Bei diesem Vorgang weden im Extremfall (m+n-1) Schritte gebraucht, wenn m und n die Längen der beiden zu mischenden Listen sind.

Im günstigsten Fall brauch nur jedes Element der kürzeren Liste mit dem ersten Element der längeren verglichen zu werden, also MIN(m,n) , oder, bei gleich langen Listen: Gesamtlänge/2 Vergleiche. Das bedeutet im Ganzen, daß höchstens $n \cdot \log n$ Vergleiche und mindestens $(n \cdot \log n)/2$ Vergleiche benötigt werden. MSORT ist also bei nahezu sortierten Listen bis zu zwei mal schneller, als bei unsortierten.

Auf alle Fälle gilt folgendes:

$$T_{MSORT}(n) = O(n \log_2 n)$$

QuickSort

Qick Sort ist eigentlich ein ziemlich einfacher Algorithmus: es wird einfach das erste Element der Liste genommen, die restliche Liste in die Elemente aufgeteilt, die größer sind als das entfernte Element und die, die kleiner oder gleich sind. Diese Teillisten werden dann durch einen rekursiven Aufruf von Quick Sort sortiert und das zuvor entfernte Element zwischen die beiden sortierten Teillisten eingefügt.

Sind diese Teillisten jeweis etwa gleich groß, so werden ca. $(n \log_2 n)$ Vergleiche durchgeführt. Bei zufällig verteilten Elementen trifft das wahrscheinlich auch weitestgehend zu. Bei gut sortierten Listen ist die eine Teilliste jedoch sehr klein, oder sogar leer, während die andere die restlichen Elemente enthält. In diesem Fall werden $(n-1)^2$ Vergleiche durchgeführt.

Wegen des Verhaltes bei sortierten Listen (wobei es keine Rolle spielt, ob in auf- oder absteigender Reihenfolge sortiert ist) gilt:

$$T_{QuickSort}(n) = O(n^2) .$$

Jedoch kann man sagen, daß das wahrscheinliche Verhalten in einem durchschnittlichen Fall

$$T^A_{QuickSort}(n) = O(n \log_2 n)$$

ist, wobei $T^A_{QuickSort}(n)$ die Durchschnittsperformance ist. Was ein Durchschnittsfall ist, und eine gründliche Diskussion darüber kann in [1] nachgelesen werden.

QuickerSort

Daß das schlechteste Verhalten von Quick Sort ausgerechnet bei sortierten Datensätzen zutage tritt, läßt sich recht einfach beheben: es muß ja nicht unbedingt das erste Element genommen werden, nach dem der Rest der Liste aufgeteilt wird. Nimmt man das mittlere Element, so ändert sich das Verhalten bei Zufallsverteilungen nicht, bei sortierten Listen tritt dann aber der günstigste Fall ein, daß alle Teillisten genau gleich groß sind (1 Element). Dieser Algorithmus ist als Quicker Sort bekannt.

Allerdings bleibt die Möglichkeit des quadratischen Zeitverbrauchs bestehen.

Deshalb gilt beim Zeitverhalten das zu Quick Sort gesagte.

Auswertung:

Die Tabelle zeigt die Anzahl der Vergleichsoperationen, die die einzelnen Programme zum sortieren der einzelnen Listen benötigten.

Bei Auswerung der Tabelle ist zu beachten, daß die Implementierungen der drei ISort-Algorithmen entgegen den Beschreibungen der Algorithmen, NICHT von vorne nach hinten sortieren, sondern (wegen einer Eigenart der verwendeten Funktionen des HP48) von hinten nach vorne! Einen Unterschied macht das bei der achten und neunten Spalte der Tabelle.

Ich habe alle sechs Programme mit jeweils zehn verschiedenen Listen der Längen 5, 10, 20, 50, 100 und 200 Elemente getestet. Die Listen einer Größe waren je Spalte für alle Programme identisch.

Die ersten sechs Listen (Spalte 1-6) enthalten Zufallszahlen. Liste 7 ist sortiert. In der 8. Liste ist aus einer sortierten Liste das mittlere Element an den Anfang ver-

schoben worden und in der neunten Liste an das Ende. Die 10. Liste ist in umgekehrter Reihenfolge sortiert.

Bei kurzen Listen (5 und 10 Elemente) schneiden noch alle Algorithmen mehr oder weniger gleich gut ab. Gewisse Unterschiede lassen sich bei den sortierten Listen ausmachen.

Bei 20 Elementen kann man schon gut das quadratische Verhalten von ISORT und QuickSort beobachten. Auch wird der Unterschied bei sortierten Listen zwischen Quick Sort und Quicker Sort deutlich. Man darf aber nicht vergessen, daß Quicker Sort sich bei entsprechenden Datensätzen auch so schlecht wie Quick Sort verhalten kann.

Generell kann man sagen, daß MSORT immer schneller als Quick Sort und Quicker Sort ist. Es ist sogar im schlechtesten Fall nahezu genau so schnell, wie die anderen beiden im besten Fall.

Über ISORT muß man sich ja wohl keine Gedanken machen?

BISORT ist für Zufallslisten etwa genau so schnell, wie MSORT, jedoch wird es bei sortierten Listen von MSORT um etwa den Faktor zwei überrundet.

Als einzige Alternative zu MSORT erscheint nur noch BITSORT. Es ist zwar meistens etwas langsamer, eignet sich jedoch wegen des linearen Verhaltens bei sortierten Listen sehr gut zum Einfügen neuer Daten in eine bereits sortierte Liste.

Bei den Listings der Programme ist zu beachten, daß einige Funktionen aus meinem Artikel über Listenmanipulation aus PRISMA 04.90 benutzt werden.

Die Programme (insbesondere alle drei Insertion-Sort Varianten) sind natürlich in der Praxis völlig unbrauchbar, da sie viel zu langsam sind und/oder zu viel Speicherplatz benötigen. Will man die Algorithmen wirklich anwenden, so müssen erst optimierte Versionen geschrieben werden. Eine optimierte Version von BITSORT ist SORTWITH aus dem oben genannten Artikel.

Literatur:

[1] Donald E. Knuth (1973)
The Art of Computer Programming
Vol.3: Sorting and Searching
Addison-Wesley, Reading, Mass.

Größe	Algorithmus	LISTE 1	2	3	4	5	6	7	8	9	10
5	ISORT	7	8	6	8	10	8	4	6	5	10
.	BISORT	7	7	7	8	8	7	6	6	7	8
.	BITSORT	8	9	9	10	12	11	4	6	7	12
.	MSORT	7	7	6	8	8	8	5	6	6	7
.	QuickSort	6	8	7	7	8	6	10	6	9	10
.	QuickerSort	10	7	8	8	6	7	7	8	6	7
10	ISORT	33	34	36	32	25	29	9	14	12	45
.	BISORT	22	22	21	24	23	23	19	19	20	25
.	BITSORT	30	27	29	33	28	29	9	12	16	34
.	MSORT	22	24	22	25	23	23	15	18	17	19
.	QuickSort	25	20	23	23	21	22	45	25	42	45
.	QuickerSort	32	21	22	30	25	22	20	20	19	20
20	ISORT	115	118	102	101	119	106	19	29	27	190
.	BISORT	62	65	61	60	62	62	54	54	57	69
.	BITSORT	77	77	75	74	73	79	19	23	41	88
.	MSORT	61	63	65	65	67	65	40	47	43	48
.	QuickSort	85	76	56	57	71	65	190	100	182	190
.	QuickerSort	59	68	88	88	77	65	57	58	57	57
50	ISORT	638	676	579	750	653	641	49	74	72	1225
.	BISORT	219	222	217	223	218	216	193	193	202	237
.	BITSORT	255	257	261	261	252	253	49	54	132	286
.	MSORT	222	224	225	227	223	226	133	154	137	153
.	QuickSort	339	267	237	257	246	265	1225	625	1202	1225
.	QuickerSort	253	284	239	277	252	236	199	201	200	199
100	ISORT	2539	2329	2658	2555	2686	2425	99	149	147	4950
.	BISORT	529	533	533	530	535	526	480	480	498	573
.	BITSORT	619	615	617	615	619	588	99	105	310	672
.	MSORT	550	533	547	540	538	544	316	361	321	356
.	QuickSort	600	556	643	689	622	718	4950	2500	4902	4950
.	QuickerSort	585	651	632	653	616	621	492	493	492	492
200	ISORT	9981	9810	10311	10242	9368	10197	199	299	297	19900
.	BISORT	1256	1252	1261	1254	1256	1255	1153	1153	1189	1345
.	BITSORT	1441	1434	1443	1439	1434	1432	199	206	715	1544
.	MSORT	1288	1286	1282	1284	1277	1284	732	826	738	812
.	QuickSort	1585	1350	1656	1406	1530	1532	19900	10000	19802	19900
.	QuickerSort	1482	1503	1468	1441	1339	1456	1177	1178	1177	1177

ISORT

```
Checksum: # 8661h                Size: 47.5 Bytes
« { } { INSERT } FOLDR
»
```

INSERT

```
Checksum: # 514Fh                Size: 135 Bytes
«
 IF DUP SIZE
 THEN
   IF DUP 1 GET 3 PICK test
   THEN DUP HD ROT ROT TL INSERT PREFIX
   ELSE PREFIX
 END
 ELSE DROP 1 →LIST
 END
»
```

BISORT

```
Checksum: # 1762h                Size: 49.5 Bytes
« { } { BINSERT } FOLDR
»
```

BITSORT

```
Checksum: # 119Ch    Size: 119.5 Bytes
« { } {
  IF DUP SIZE
  THEN
    IF DUP2 1 GET test
    THEN PREFIX
    ELSE BINSERT
    END
  ELSE PREFIX
  END } FOLDR
»
```

BINSERT

```
Checksum: # C536h                Size: 227 Bytes
«
  IF DUP SIZE
  THEN
    IF DUP SIZE 2 / CEIL DUP2 GET 4 PICK test
    THEN DUP2 1 SWAP SUB ROT ROT 1 + 1000000
         SUB ROT SWAP BINSERT CONCAT2
    ELSE DUP2 5 ROLL ROT ROT 1 - 1 SWAP
         SUB BINSERT ROT ROT 10000 SUB CONCAT2
    END
  ELSE DROP 1 →LIST
  END
»
```

MSORT

```
Checksum: # 756Eh                Size: 109 Bytes
«
  IF DUP SIZE 1 >
  THEN DUP SIZE 2 / IP 1 SWAP SUB LASTARG +
       1000000 SUB MSORT SWAP MSORT MERGEL
  END
»
```

MERGEL

```
Checksum: # A28Eh                Size: 178 Bytes
«
  CASE OVER SIZE NOT
    THEN SWAP DROP
    END DUP SIZE NOT
    THEN DROP
    END OVER 1 GET OVER 1 GET test
    THEN SWAP DUP HD SWAP TL ROT MERGEL PREFIX
    END DUP HD ROT ROT TL MERGEL PREFIX
  END
»
```

MERGELISTS

ist eine etwas übersichtlichere Variante von MERGEL

```
Checksum: # 43D2h                Size: 212.5 Bytes
« → a b
  «
    CASE a SIZE NOT
      THEN b
      END b SIZE NOT
      THEN a
      END a 1 GET b 1 GET test
      THEN a HD a TL b MERGEL PREFIX
      END b HD a b TL MERGEL PREFIX
    END
  »
»
```

QuickSort

```
Checksum: # EE64h                Size: 143 Bytes
«
  IF DUP SIZE 1 >
  THEN DUP HD SWAP TL OVER 'test' 2 →LIST SPLIT
    QuickSort SWAP QuickSort ROT SUFFIX
       SWAP CONCAT2
  END
»
```

QuickerSort

```
Checksum: # 40C5h    Size: 200.5 Bytes
«
  IF DUP SIZE 1 >
  THEN DUP SIZE DUP 2 / IP 3 PICK OVER DUP SUB
       1 GET 4 ROLLD 1 + SWAP SUB LASTARG
       DROP 2 - 1 SWAP SUB + OVER 'test' 2
       →LIST SPLIT QuickerSort SWAP QuickerSort
       ROT SUFFIX SWAP CONCAT2
  END
»
```

---

test
ist die in allen Programmen benutzte Testinstruktion. Zum zählen der Vergleiche wird die globale Variable n erhöht. Dann werden nur noch die beiden ersten Argumente auf dem Stack verglichen.

```
Checksum: # 604h    Size: 35.5 Bytes
« 'n' 1 STO+ <
»
```

Matthias Rabe
Teichsheide 13
4800 Bielefeld

Welt-hunger. Ernte-dank.

Die Deutsche Welthungerhilfe unterstützt Selbsthilfe-Projekte von Bauern der Dritten Welt, damit für sie Ernährung aus eigener Kraft möglich wird. Und sie hilft den Bauern, Natur und Umwelt als Lebensgrundlage zu erhalten, damit Entwicklung auch Zukunft hat.

DEUTSCHE WELTHUNGERHILFE
Spendenkonto Sparkasse Bonn: 111
Adenauerallee 134 · 5300 Bonn 1 · Tel.: 02 28/22 88 0

Pas de deux - Teil III

Programme und Tips für 48SX und 28S

von Ralf Pfeifer

Um die Kompatibilität zum 28S zu erhalten, hat HP einige Funktionen in den 48SX übernommen, so z.B. stehen der Funktion OBJ→ (48SX) auch noch die Funktionen STR→, ARRY→ und LIST→ zur Seite. Obwohl nur OBJ→ in einem Menue steht, empfehle ich, möglichst die übrigen drei zu gebrauchen, denn einerseits sind sie schneller (nur in Schleifen wichtig) und andererseits testen sie das eingegebene Objekt. Ein Programm welches nur Arrays als Eingabe zuläßt, muß auf eine Liste so schnell wie möglich mit einem Error reagieren, bevor gespeicherte Daten verändert werden.

Zur Kompatibilität trägt auch die Funktion FACT bei, die sich von ! nicht unterscheidet. In Programmdokumentationen ist FACT wesentlich besser lesbar als !, was mein Programm GLN in PRISMA 90.3.28 abschreckend beweist, dort habe ich die Funktion | (where, wobei) und dann ! (statt FACT) benutzt.

Außerdem darf man bei der Eingabe eines Programms auch LAST benutzen, der 48SX konvertiert es aber dann zu LASTARG.

Um die Kompatibilität zwischen den Modellen noch zu steigern, hier noch ein paar Programme: **UPDIR** bewirkt auf dem 28S den Aufstieg ins nächsthöhere Directory. Beim 48SX steht diese Funktion als UP auf der Tastatur, in Programmen heißt sie auch UPDIR.

OBJ→ löst auf dem 28S nur Arrays und Listen auf, in Ebene 1 verbleibt die Anzahl der Elemente. Auch bei Vektoren und Matrizen steht in Ebene 1 nur eine Zahl, die Zahl eben dieser Elemente!

Manchem ist die LASTARG-Funktion ein Rätsel. Klar ist, daß bei mathematischen Funktionen alle Argumente gesichert werden, z.B. zwei bei + oder sogar vier bei ∫. Die Vergleichsfunktionen z.B. == oder SAME kann man in diesem Sinne auch als mathematische Funktionen sehen, die zwei Argumente vom Stack nehmen um sie durch ein neues zu ersetzen und somit LASTARG aktivieren. Ganz allgemein kann man sagen: Alles was irgendeine Funktion vom Stack nimmt sichert LASTARG.

Beispielsweise nimmt IFTE den Inhalt von Ebene 1-3 ins LASTARG-Grab, und die Befehle FOR, START, STEP usw. sichern ihre Parameter. Aber es gibt auch Ausnahmen wie CLEAR (CLR) die LASTARG nicht beeinflussen, bei STO gibt es (nur 48SX) sogar eine sehr intelligente Ausnahme, denn diese Funktion sichert denn letzten Inhalt der Variablen, nicht den des Stacks!

Und noch etwas: Ob LASTARG ein- oder ausgeschaltet ist beeinflußt den IFERR-Test. Sollte ein Fehler auftreten, verschwinden die Argumente ohne LASTARG vom Stack, bei eingeschaltetem LASTARG bleiben sie unverändert erhalten.

Die LAST-Funktion STACK wirkt oft ähnlich wie LASTARG. Zusammen mit CMD gibt es folgende praktische Anwendung: Programme verändern diese beiden Funktionen nicht. Sollte also ein Programmaufruf in die Hose gehen (falsches Programm, Error, falsch Eingaben) bringt STACK und ggf. CMD den Stackinhalt vor Programmstart zurück. Flags und die Inhalte von Variablen korrigieren diese Funktionen allerdings nicht.

Um Programme zu optimieren braucht man zwei wichtige Informationen: Die Programmlänge und die Laufzeit (28S: PRISMA 89.4.37). Für die Programmlänge ist die eingebaute Funktion BYTES (28S: Programm LNG in PRISMA 90.3.28) zuständig, die Laufzeit ermittelt **TIMER**. Zunächst richtet man alles so ein, wie es das Programm braucht, dann muß dieses oder dessen 'Name' in Ebene 1 um schließlich TIMER zu starten. Die Vorbereitungen für TIMER entsprechen also ganz genau denen der Standardfunktion DEBUG. Die Ausgabe umfasst dann die Ergebnisse des Programms und in Ebene 1 die Laufzeit in Einheiten der eingebauten Uhr (durch 8192 teilen ergibt die Zeit in Sekunden).

Für die Optimierung ist der Zahlenwert unbedeutend, wichtig ist nur, daß die Laufzeiten immer kürzer werden, mit der feinen Unterteilung von TICKS ist das oft sogar zu genau möglich, denn trotz gleicher Startbedingungen hat das gleiche Programm selten die gleiche Laufzeit. Um diese Gleichheit wenigstens anzunähern, ruft TIMER zunächst das Programm zurück (falls gespeichert) und bewirkt ein Speicher-Packing mit MEM DROP (s. auch Programm MORSE in Pas de deux II).

Vor der ersten Anwendung muß TIMER noch justiert werden, die 125 in Zeile 7 sollte man nämlich so verändern, daß ein leeres Programm (das hier: « ») die Laufzeit 0 ergibt. Nur selten (z.B. für die Benchmark-Rallys) braucht man die richtige Laufzeit. Bei gleicher Bedienung wie TIMER hinterläßt **TMHR** die Laufzeit im HH,MMSSss-Format in Ebene 1.

Etwas Interessantes gibt es zur Rundung, hier verschaffen nämlich Kaufleute einer alten Regel neue Ehren. Wer ab und zu die Dollarkurse verfolgt, bemerkt, daß man die Umrechnungen hier nach Zehnteln und Hundersteln eines Pfennigs berechnet. Ziel der Rundung ist ja, Fehler beim Runden so klein wie möglich zu halten. 4,4 Pf werden auf 4 Pf abgerundet, 4,6 auf 5 Pfennig aufgerundet. Auch bei 4,51 Pf ist der Fall klar: Aufrunden auf 5 Pfennig bringt weniger Fehler als abrunden auf 4 Pfennig. Im Spezialfall 4,500.. Pf bringt aufrunden genausoviel Fehler wie abrunden, aber die üblichen Regeln für Zahlendarstellungen im Computer sehen hier immer das Aufrunden vor. So treiben es auch die hierzu geprüften HP-Modelle 28S, 42S oder 48SX, wenn sie die Funktion RND anwenden oder eine Zahl im FIX/SCI/ENG-Format anzeigen.

Doch die neue alte "Geradzahlregel" sagt hierzu, daß im Fall ..,5 jeweils zur nächsten geraden Zahl gerundet werden soll, also z.B. 4,5 Pf rundet man zu 4 Pf ab, 5,5 Pf zu 6 Pf auf. Der Vorteil: Wenn die Zahlen alle mit gleicher Wahrscheinlichkeit berechnet werden, sind auch die Abweichungen durch Rundungsfehler im Mittel Null, während bei der einfachen Rundung die Zahlen im Laufe einer längeren Rechnung wachsen. HP wäre nicht HP, wenn sie diesem Umstand nicht Rechnung trügen, intern runden die genannten Modelle nämlich nach der Geradzahlregel!

Dazu folgendes Beispiel: Zur Zahl 4,5 addiert man 100 Billionen (=1E11). Das Ergebnis ist eine dreizehnstellige Zahl, die genannten Rechner benutzen zwar intern 15-stellige Zahlen, aber das Ergebnis bekommt der Anwender immer auf 12 Stellen gerundet zurück. Nach den Regeln der RND-Funktion muß das Ergebnis jetzt 100 000 000 005 lauten, die Rechner aber wenden die eben genannte Geradzahlregel an, so daß das korrekte Ergebnis durch Abrundung 100 000 000 004 beträgt. Führt man die Addition von 1E11 mit 5,5 durch erhält man hier durch Aufrundung 100 000 000 006.

Auf dem Prinzip der Addition und Subtraktion beruht dann auch das Programm **RND2** (auch 28S), welches Zahlen (speziell Geldbeträge in DM) auf zwei Nachkommastellen rundet. Über das Runden kann man auch bei Rüdiger Schulz in PRISMA 83.4.18 nachlesen, von Franz A. Riedlinger wissen wir aus PRISMA 86.6.24, daß die Schweizer Geldbeträge

auf volle 5 oder 10 Rappen runden. Das Programm **RND2CH** (nicht 28S, CH steht für Schweiz) arbeitet nach den einfachen Regeln der RND-Funktion. **RND2CH2** (auch 28S) ist eher mit dem RND2 Programm verwandt, denn es rundet 2,5 Rappen immer ab, aber 7,5 Rappen immer auf.

Die folgenden Programme helfen bei der Auswertung von Polynomen (Gleichungen n-ten Grades), einer besonders wichtigen Klasse von Funktionen. So wäre die Funktion

$$f(x) = -2x^3 + 6x^2 - 4;$$

ein Beispiel für eine Gleichung dritten Grades (kubische Gleichung). Zur Darstellung dieser Funktionen verwenden die nachfolgenden Programme Vektoren. Dabei schreibt man - wie in der Mathematik üblich- so, daß der Koeffizient der höchsten Potenz ganz links im Vektor (für die Funktion GET und PUT wäre das Position 1) und das absolute Glied (=ohne x) ganz rechts steht. Außerdem müssen Koeffizienten fehlender Glieder mit Null eingetragen werden. Für das Beispiel ergäbe sich also folgender Vektor:

[-2 6 0 -4] ;

Zu einer Gleichung n-ten Grades gehört also immer ein Vektor mit n+1 Elementen. Vektoren können auch komplexe Zahlen speichern, was bei den folgenden Programmen auch zulässig ist. Von der Anwendung der Funktionen im MTH VECTR Menue möchte ich allerdings dringend warnen, da einige Funktionen von Winkel- oder Koordinatenmodus abhängen.

Eine wichtige Anwendung ist das Hornerschema um den Funktionswert und die ersten n Ableitungen (noch höhere Ableitungen sind ohnehin Null) für eine bestimmte Stelle zu berechnen. Das Hornerschema vermeidet das Potenzieren und rechnet ein Polynom nur mit Hilfe von Additionen und Subtraktionen aus. Die Zeitersparnis und geringer Ärger mit Rundungsfehlern machen sich auch auf dem neuen 48SX bezahlt. Zudem kann man sich leicht überlegen: Wer eine Tabelle mit Funktionswerten und Ableitungen eines Polynoms anlegen will, kann zwar die Ableitungen aufstellen, aber das Einsetzen braucht genausoviele Rechenschritte wie das Hornerschema, welches dieses Ableiten einspart.

HORNER verlangt in Ebene 1 die Stelle, für die Funktionswert und die ersten n Ableitungen berechnet werden sollen. In Ebene 2 steht der Vektor mit den Koeffizienten (s.o.). Das Ergebnis von HORNER hat wiederum die Form eines Vektors: Der Funktionswert an erster Stelle, die erste Ableitung an zweiter Stelle usw. HORNER arbeitet nur im Stack, wer also Ergebnisse nur ansehen oder abschreiben will, kann hinterher mit LAST STACK und LAST CMD den Vektor des Polynoms zurückholen, das spart Arbeit.

Zwei Polynome kann man auch durcheinander dividieren, z.B. um Nullstellen zu entfernen (=Deflation) oder um aus einer unecht gebrochenen rationalen Funktion (=Bruch, dessen Zählerpolynom einen höheren Grad hat als das Nennerpolynom) eine geeignetere Darstellung zu finden. **POLY÷** (28S: Das Programm könnte auch POLYDIV heißen) erwartet dazu das Zählerpolynom in Ebene 2 und das Nennerpolynom in Ebene 1, also wie bei der Division zweier Zahlen. Wenn der Grad des Zählers (=z) nicht kleiner als der des Nenners (=n) ist, berechnet POLY÷ den Quotienten in Ebene 2 und den Rest in Ebene 1. Dabei hat der Quotient den Grad z-n+1 und der Rest höchstens den Grad n-1. Das mit dem Rest muß man sich so vorstellen: 13 durch 4 gibt 3 Rest 1. Die 3 ist der Quotient, der Rest ist auf den Nenner bezogen, also als "Rest durch Nenner" auszudrücken, hier ist es 1 durch 4 (landläufig ein Viertel genannt). Schreibt man den Rest der Polynomdivision auf, muß man also auch den "Rest durch Nenner" schreiben!

Um zwei Polynome zu multiplizieren (in Ebene 1 und 2 einzugeben) habe ich gleich zwei Programme anzubieten: **PMUL** und **PMUL2** um z.B. Polynome aus Linearfaktoren zu konstruieren. Der mathematische Weg ist gleich, aber bei PMUL läuft alles durch Matrizenmultiplikation ab. Das ist zwar speicherfressend, aber recht genau, weil der 48SX die Berechnung mit seinen internen 15 Stellen durchführt und erst das Ergebnis auf 12 Stellen rundet. Außerdem ist PMUL sehr viel schneller als PMUL2.

Für Rechner mit begrenztem Speicherplatz wie dem 28S empfiehlt sich dennoch PMUL2, da es nur 2a+3(b+1) Speicherplätze für die Polynome mit dem Grad a bzw. b braucht. PMUL speichert zusätzlich weitere b(a+b+1) Zahlen, wobei beide Programme aus Zeit- und Platzgründen die Polynome so vertauschen, daß b der kleinere Grad ist. Leider ergeben sich bei der Matrizenmethode von PMUL quadratische Matrizen nur in Spezialfällen, sonst wäre die Umkehr des Verfahrens zur Polynomdivision möglich.

PADD dient zur Addition zweier Polynome. Die Funktion + kann das nur ausführen, wenn beide Polynome in Vektoren gleicher Größe gespeichert werden. Die Vektoren mit dem Koeffizienten erwartet PADD in Ebene 1 und 2, zur Subtraktion verwendet man die Schritte +/-, PADD oder NEG, PADD.

Manchmal kann man die Ableitung eines Polynoms ganz gut gebrauchen, möchte aber nicht auf die Darstellung mit Vektoren verzichten. Deshalb berechnet **ABL1** die erste Ableitung der Funktion in Ebene 1. Für höhere Ableitungen kann man dieses Programm entsprechend wiederholen. Die 48SX-Versionen der Programme POLY÷, PMUL, PADD, ABL1 und HORNER laufen auch auf dem 28S, wenn man die EVAL-Befehle in den Listings durch 1 GET ersetzt.

Bei manchen Operationen verschwindet der Koeffizient der höchsten Potenz, das Polynom wird also um (mindestens) einen Grad niedriger. Um unnötige Errors und Wartezeiten zu vermeiden, verwenden einige Programme **COL** (COLCT), um den Grad des Polynoms zu überprüfen. COL verkleinert den Vektor solange, bis auf Position 1 keine Null mehr steht bzw. der Vektor nur noch ein Element enthält (welches dann Null sein kann). Außerdem verwandelt COL einen komplexen Vektor in einen reellen, falls alle Imaginärteile Null sind. Beim 48SX führt der Test 0 == sowohl bei einer reellen Null als auch bei der komplexen Null (0;0) zum richtigen Ergebnis 1, bei der 28S-Version von COL müssen diese Fälle jedoch getrennt behandelt werden.

Wer von der Vektordarstellung zur algebraischen Syntax wechseln möchte, kann dazu →**EQ** einsetzen. Das Polynom erscheint in der Variablen, die das Programm INDP? als unabhängig ausweist.

INDP? (28S: Zeile 4 im Listing streichen) fragt nach der unabhängigen Variablen in der Liste PPAR. Normalerweise verändert man diese mit der Funktion INDEP. Das Programm INDP? sucht zuerst in Menue des aufrufenden Programms (z.B. →EQ oder POLYFIT) nach PPAR, falls es dort keines findet im nächsthöheren usw. bis es im HOME-Directory ankommt (hier sollte man es auch speichern). Ist auch hier kein PPAR zu finden wählt das Programm X als Unabhängige.

NULLWEG ist eine zusätzliche Hilfe für den Löser (SOLVE) des 48SX. Hat dieser die Nullstelle einer beliebigen Funktion berechnet, so könnte Interesse bestehen, weitere Nullstellen zu finden. Besonders wenn Nullstellen dicht beieinander liegen, kann es sein, daß SOLVR immer auf die gleiche dominante Nullstelle zuläuft. Polynome können außerdem mehrfache Nullstellen haben, deren Anzahl sich nicht auf den ersten Blick offenbart. Dem hilft man ab, indem man die Funktion durch $(x-x_0)$, also "Variable minus gefundene Nullstelle", dividiert.

Folgendes Beispiel macht das deutlich: Der Plotter zeigt, daß die Funktion

$$f(x) = x^2 - 2 - \ln x$$

zwei Nullstellen hat. Nun gibt man SOLVR den Startwert 3 und erhält die erste Nullstelle X = 1,564...; der 48SX gibt diese Nullstelle als markiertes Objekt aus, welche man sofort an das Programm NULLWEG verfüttert. Erneutes Plotten der Funktion zeigt, daß diese plötzlich einen weiten Bogen um die alte Nullstelle

```
48SX HORNER
  « OVER SIZE EVAL →
002 x s
    « ARRY→ EVAL 1
004     FOR m 0 m 1 + 2
          FOR n x * n
006 ROLL + DUP n ROLLD
    -1
008       STEP DROP s m
    - FACT * m ROLLD -1
010     STEP s →ARRY
    »
012 »  Bytes : 139
    Checksum : # 183Eh
```

$f(X)=X^4-X^3-21X^2+X+20$

Gesucht: Funktionswert und alle Ableitungen für X = 3

```
2:      [ 1 -1 -21 1 20 ]
1:                       3
  HORNER
1: [ -112 -44 48 66 24 ]
```

Ergibt:

f(3) = -112
f'(3) = -44
f''(3) = 48
f'''(3) = 66
f''''(3) = 24

```
48SX ABL1
  « ARRY→ SWAP DROP
002   IF LIST→ - DUP
      THEN 1 SWAP
004     FOR n n ROLL
    LASTARG * n ROLLD
006     NEXT LASTARG
      ELSE 1
008   END →ARRY
  »  Bytes : 81
010 Checksum : # A0D1h
48SX →EQ
  « ARRY→ LIST→ OVER
002 INDP? → g x
    « - 0 SWAP 0
004     FOR n n 2 +
    ROLL x n ^ * + -1
006     STEP
    »
008 »  Bytes : 93
    Checksum : # 25F3h
```

f(X)=2*X^3-5*X-3
f'(X) = ?

```
1:          [ 2 0 -5 -3 ]
  ABL1
1:            [ 6 0 -5 ]
  →EQ
1:              '6*X^2-5'
```

```
48SX POLY÷
  « COL SWAP COL SWAP
002 DUP 1 GET DUP 4
    ROLLD / 1 0 PUT
004 OVER SIZE OVER SIZE
    LIST→ - OVER EVAL
006 DUP2 < / OVER - → a
    b
008   « RDM 1 b
        FOR n SWAP DUP2
010 n GET * - SWAP
    ARRY→ EVAL ROLLD
012 LASTARG →ARRY
        NEXT DROP ARRY→
014 DROP a
      IF DUP NOT
016   THEN LASTARG
      END →ARRY b 1 +
018 ROLLD b →ARRY ROT /
      » SWAP COL
020 »  Bytes : 215,5
    Checksum : # 37D6h
```

$$\frac{-2X^4-5X^3+2X+6}{-2X^2+3X-2}=?$$

```
2:     [ -2 -5 0 2 6 ]
1:          [ -2 3 -2 ]
  POLY÷
2:          [ 1 4 5 ]
1:            [ -5 16 ]
```

Lösung:

$$X^2+4X+5+\frac{-5X+16}{-2X^2+3X-2}$$

```
48SX PADD
  « OVER SIZE EVAL
002 OVER SIZE EVAL
      IF DUP2 <
004   THEN 4 ROLL SWAP
    ROT
006   END - → a
      « OVER SIZE RDM
008 ARRY→ EVAL 1 a
      »
010   START ROLLD
    LASTARG
012   NEXT →ARRY + COL
    »  Bytes : 108
014 Checksum : # 974Ah

2:      [ 1 -1 -21 1 20 ]
1:                [ 1 0 -1 ]
  PADD
1:      [ 1 -1 -20 1 19 ]
```

```
48SX PMUL
  « OVER SIZE EVAL
002 OVER SIZE EVAL
      IF DUP2 <
004   THEN 4 ROLL SWAP
    ROT
006   END
      IF OVER 1 SAME
008   THEN 1 + SWAP 0.5
    ROLL 1 GET 2 →ARRY
010 4 ROLLD
      END OVER → n
012   « + 1 - 1 →LIST
    RDM ARRY→ EVAL 2 n
014   START DUPN
    LASTARG ROLLD
016 LASTARG
        NEXT n SWAP 2
018 →LIST →ARRY TRN
      » SWAP * COL
020 »  Bytes : 187,5
    Checksum : # 38EDh
```

```
28S PMUL2
  « OVER SIZE 1 GET
002 OVER SIZE 1 GET DUP2
    IF >
004   THEN 4 ROLL SWAP
    ROT
006   END OVER + 1 - ROT
    SWAP 1 →LIST RDM DUP
008 0 CON 1 4 ROLL
      FOR a OVER 4 PICK
010 a GET * + SWAP ARRY→
    1 GET ROLLD LAST
012 →ARRY SWAP
      NEXT ROT ROT DROP2
014 COL
  »  Bytes : 158
016 Checksum : # F4A2h
```

(X^3+6*X+1)*(X^2-1) = ?

```
2:              [ 1 0 6 1 ]
1:                [ 1 0 -1 ]
  PMUL
1:      [ 1 0 5 1 -6 -1 ]
  →EQ
1: 'X^5+5*X^3+X^2-6*X-1'
```

```
48SX COL
  «
002   WHILE DUP 1 GET 0
    == OVER SIZE ( 1 )
004 ≠ AND
      REPEAT ARRY→
006 LIST→ - →ARRY SWAP
    DROP
008   END
      IF DUP IM ABS NOT
010   THEN RE
      END
012 »  Bytes : 87,5
    Checksum : # D46Ch
```

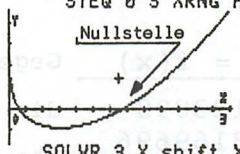
```
28S COL
  «
002   WHILE DUP 1 GET 0
    SAME LAST 0 R→C SAME
004 OR OVER SIZE ( 1 ) ≠
    AND
006   REPEAT ARRY→ LIST→
    - →ARRY SWAP DROP
008   END DUP IM ABS
      IF NOT
010   THEN RE
      END
012 »  Bytes : 100
    Checksum : # 788Eh
```

```
48SX INDP?
  « PPAR 3
002   IFERR GET
      THEN DROP2 'X'
004   ELSE ( ) + 1 GET
      END
006 »  Bytes : 67
    Checksum : # 23A2h

  INDP?      (kein PPAR)
1:                   'X'
```

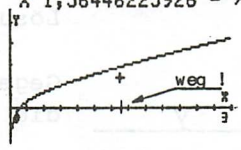
```
48SX NULLWEG
  « RCLF STD SWAP
002 OBJ→ RCEQ ( ) + 1
    GET
004   IF DUP TYPE 8
    SAME
006   THEN →STR DUP
    SIZE ROT REPL " " +
008 SWAP + " =/" + STR→
      ELSE "-" ROT +
010 STR→ ROT - /
      END RCEQ
012   IF TYPE 5 SAME
      THEN RCEQ 1 ROT
014 PUT
      END STEQ STOF
016 »  Bytes : 162,5
    Checksum : # 121Ch

1:  « X SQ 2 - X LN - »
    STEQ 0 3 XRNG AUTO
```




```
  SOLVR 3 X shift X
    (Sign Reversal)
1:      X: 1,56446225926

  NULLWEG ERASE DRAW
EQ : « X SQ 2 - X LN -
  X 1,56446225926 - / »
```




```
48SX KR
  «
002   IFERR RCL
      THEN 0
004   ELSE LASTARG DUP
    PURGE ROT OVER 1
006   END ROT ROT SWAP
    OVER ∂ DUP ROT ∂
008 SWAP SQ 1 + 1,5 ^ /
      IF SWAP
010   THEN ROT ROT STO
      END "1/R " →TAG
012 »  Bytes : 124,5
    Checksum : # BBDAh

2:                'SIN(X)^2'
1:                          'X'
  KR
1: 1/R : '(-(SIN(X)^2*
SIN(X))+COS(X)^2*COS(X))
/(SQ(COS(X)^2*SIN(X))+1)
^1,5'  (RAD wählen !)
```

```
48SX BOGEN
  «
002   IFERR RCL
      THEN 0
004   ELSE LASTARG DUP
    PURGE 5 ROLLD 4
006 ROLLD 5 PICK 1
      END 5 ROLLD SWAP
008 OVER ∂ SQ 1 + ∫
    SWAP ∫
010   IF SWAP
      THEN SWAP ROT STO
012   END →NUM
    »  Bytes : 107,5
014 Checksum : # F157h

4:                        ,5
3:                      1,5
2:              'SINH(X)^'
1:                        'X'
  BOGEN
1:          1,90262963968
```

```
48SX GGT
  «
002   WHILE SWAP OVER
    MOD
004   REPEAT LASTARG
      END
006 »  Bytes : 27,5
    Checksum : # EE7Bh
```

```
28S UPDIR
« PATH DUP SIZE 1
002 DUP2
      IF SAME
004   THEN DROP
      ELSE -
006   END GET EVAL
    »  Bytes : 45
008 Checksum : # 59CCh
```

```
48SX POLYFIT
  « 1 + NΣ MIN DUP 1
002 2 →LIST DUP 0 CON
    DUP DUP TRN * ROT
004 ΣPAR 1 1 SUB
    LASTARG + 2 SUB → b
006 x y
      « "X-COL : " x
008 EVAL + 1 DISP
    "Y-COL : " y EVAL +
010 2 DISP 1 NΣ
        FOR a RCLΣ a x
012 + GET 4 PICK 1 OVER
        START 1 - ^
014 LASTARG
        NEXT DROP2 b
016 →ARRY ROT OVER RCLΣ
    a y + GET * + ROT
018 ROT DUP TRN * +
      NEXT
020   » SWAP OVER /
    LASTARG 3 PICK RSD
022 ROT / + OVER 1
    →LIST RDM 'L' STO
024 'n' 1 ROT 'L(n)'
    INDP? 3 PICK 6 PICK
026 - ^ * Σ
    »  Bytes : 372
028 Checksum : # 6622h
```

```
48SX NLIN
  « RCLΣ ARRY→ EVAL 1
002 - DUP 1 2 →LIST 0
    CON DUP DUP TRN * →
004 n b a
      « 1 SWAP
006     START n 1 +
    ROLLD n 1 2 →LIST
008 →ARRY DUP DUP TRN *
    'a' STO+ * 'b' STO+
010     NEXT b a /
    LASTARG 3 PICK RSD
012 a / + n 1 →LIST RDM
    'L' STO 'j' 1 n 'L(
014 j)*X(j)' Σ
      »
016 »  Bytes : 261
    Checksum : # CDE6h
```

```
48SX TIMER
  «
002   IFERR RCL
      THEN TEXT
004   END MEM DROP
    TICKS → t
006   « EVAL TICKS t -
      » B→R 125 - 1000
008 ,05 BEEP
    »  Bytes : 93
010 Checksum : # 29D5h
```

```
48SX TMHR
  « TIMER 29491200 /
002 →HMS 6 RND 0 HMS+
    »  Bytes : 44
004 Checksum : # 8837h
```

```
48SX RND2
  « SIGN -1000000000
002 LASTARG ABS OVER -
    + *
004 »  Bytes : 38
    Checksum : # DB99h

1:                  13,545
  RND2
1:                   13,54

1:                  13,555
  RND2
1:                   13,56
```

```
48SX RND2CH
  « 2 * 1 RND 2 /
002 »  Bytes : 25
    Checksum : # EE21h
48SX RND2CH2
  « SIGN -10000000000
002 LASTARG 2 * ABS
    OVER - + * 2 /
004 »  Bytes : 48
    Checksum : # 58E9h

1:                  13,225
  RND2CH2
1:                    13,2
```

```
28S OBJ→
  «
002   IFERR LIST→
      THEN ARRY→ LIST→
004 DUP2 DROPN *
    END
006 »  Bytes : 37,5
    Checksum : # D7B7h
```

Ralf Pfeifer (116)
Rubensstr. 5
D-5000 Köln 50

POLYFIT - Beispiel 1:

Gegeben: Wertetabelle. Vermutet wird eine Funktion der Form: $f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$;

Lösung: $f(x) = x^4 - 2x^2 + x - 9$;

x	y
-6	1209
-3	51
-1	-11
1	-9
2	1
4	219
5	571

```
ΣDAT (Beispiel 1)
[[ -6 1209 ]
 [ -3 51 ]
 [ -1 -11 ]
 [ 1 -9 ]
 [ 2 1 ]
 [ 4 219 ]
 [ 5 571 ]]

1 XCOL 2 YCOL 4 POLYFIT
1:          '-9+X-2*X^2-
8,8460325E-14*X^3+X^4'
```

```
ΣDAT (Beispiel 2)
[[ ,1353353 ,6123098 ]
 [ ,3678794 ,03169696 ]
 [ 3,004166 1,012515 ]
 [ 6,049647 19,44929 ]
 [ 14,87973 177,7672 ]]

1 XCOL 2 YCOL 2 POLYFIT
1:          'T,000000-3,000001*
X+1,000000*X^2'
```

```
ΣDAT (Beispiel 3)
[[ -,29552 -1,45239 ]
 [ ,00999983 -2,0098 ]
 [ ,109778 -2,08964 ]
 [ ,389418 -2,26329 ]
 [ ,932039 -3,62362 ]
 [ ,971148 -3,83265 ]]

1 XCOL 2 YCOL 3 POLYFIT
1:          '-2,0000-1,0000*X+
2,0000*X^2-3,0000*X^3'
```

POLYFIT - Beispiel 2:

x	exp(x) = z	y = f(x)
-2	,1353353	,6123098
-1	,3678794	,03169696
1,1	3,004166	1,012515
1,8	6,049647	19,44929
2,7	14,87973	177,7672

Gegeben: Wertetabelle mit x, y. Gesucht wird die Funktion: $f(x) = a_2e^{2x} + a_1e^x + a_0$;

Lösung: Substitution $e^x = z$, dann Regression mit einem quadratischen Polynom der Form: $f(z) = a_2z^2 + a_1z + a_0$;

Lösung: $f(x) = e^{2x} - 3e^x + 1$;

POLYFIT - Beispiel 3:

x	sin(x) = z	y
-0,3	-0,29552	-1,45239
0,01	0,00999983	-2,0098
0,11	0,109778	-2,08964
0,4	0,389418	-2,26329
1,2	0,932039	-3,62362
1,33	0,971148	-3,83265

Gegeben: Wertetabelle mit x, y. Vermutet wird die Funktion:

$y = a_3\sin^3 x + a_2\sin^2 x + a_1\sin x + a_0$

Lösung: Substitution sin x = z, dann Regression mit einem kubischen Polynom der Form:

$f(z) = a_3z^3 + a_2z^2 + a_1z + a_0$;

Lösung: $y = -3\sin^3 x + 2\sin^2 x - \sin x - 1$;

NLIN - Beispiel 1:

Gegeben: Wertetabelle mit x_1, x_2, x_3, y. Gesucht wird eine Funktion der Form: $f(x) = y = a_1x_1 + a_2x_2 + a_3x_3$;

Lösung: $f(x) = y = 3x_1 - 4x_2 - x_3$;

x_1	x_2	x_3	y
3	0	1	8
4	6	2	-14
-1	-1	2	-1
0	1	-1	-3
6	-1	3	19

```
ΣDAT (Beispiel 1)
[[ 3 0 1 8 ]
 [ 4 6 2 -14 ]
 [ -1 -1 2 -1 ]
 [ 0 1 -1 -3 ]
 [ 6 -1 3 19 ]]

NLIN
'3*X(1)-4*X(2)-X(3)'
```

```
ΣDAT (Beispiel 2)
[[ 2 4 -3 1 -21 ]
 [ -1 -4 2 1 22 ]
 [ 3 2 0 1 2 ]
 [ 7 28 1 1 -63 ]
 [ 1 18 4 1 -30 ]]

NLIN
'2*X(1)-3*X(2)+5*X(
3)+2*X(4)'
```

```
48SX GM
« 1 RCLΣ SIZE 2 GET
  FOR a 0 1 NΣ
    FOR b RCLΣ b a
  NEXT NΣ / Γ
  NEXT RCLΣ SIZE 2
GET →ARRY
» Bytes : 93
Checksum : # AF57h
```

NLIN - Beispiel 2:

X_1	X_2	X_3	X_4	y
2	1	-2	-3	-21
-1	-1	2	2	22
3	2	1	0	2
7	7	2	1	-63
1	2	3	4	-30

Um die folgende Funktion zu ermitteln ist eine neue Wertetabelle einzuführen, die man mit der Substitution $z_2 = x_2x_3^2$ berechnet. Die Variablen $z_1 = x_1$ und $z_3 = x_4$ brauchen keine Substitution. NLIN berechnet keine Konstanten, um solche dennoch zu finden (hier: a_4), führt man die Variable $z_4 = 1$ = const. ein.
Vermutet wird eine Funktion der Form:

z_1	z_2	z_3	z_4	y
2	4	-3	1	-21
-1	-4	2	1	22
3	2	0	1	2
7	28	1	1	-63
1	18	4	1	-30

$f(x) = a_1X_1 + a_2X_2X_3^2 + a_3X_4 + a_4$;

Substitution: $f(z) = a_1z_1 + a_2z_2 + a_3z_3 + a_4z_4$;

Lösung: $f(x) = 2x_1 - 3x_2x_3^2 + 5x_4 + 2$;

```
48SX GM
« 1 RCLΣ SIZE 2 GET
  FOR a 1 1 NΣ
    FOR b RCLΣ b a
  2 →LIST GET ARG 0
  LASTARG IFTE *
    NEXT NΣ XROOT
  NEXT RCLΣ SIZE 2
GET →ARRY
» Bytes : 98
Checksum : # 886h
```

```
48SX HM
« 1 RCLΣ SIZE 2 GET
  FOR a NΣ 0 1 NΣ
    FOR b RCLΣ b a
  2 →LIST GET
    IF DUP
    THEN INV
    ELSE SWAP ROT
    END +
  NEXT /
NEXT RCLΣ SIZE 2
GET →ARRY
» Bytes : 113
Checksum : # 3498h
```

Ralf Pfeifer (116)
Rubensstr. 5
5000 Köln 50

macht, die zweite und letzte kann SOLVR jetzt ohne Probleme angehen.

NULLWEG nimmt zuerst die von SOLVR ermittelte Lösung und trennt Name der Variablen und Zahl voneinander. Dann holt NULLWEG die aktuelle Gleichung, entfernt das evtl. vorhandene Tastenmenue und prüft, ob die Funktion als algebraisches Objekt oder als Programm vorliegt. Entsprechend dieser Syntax dividiert NULLWEG durch "Variable minus Nullstelle" und speichert die Gleichung zurück, wobei Tastendefinitionen erhalten bleiben.

Daß bei diesem System der Nullstellenbeseitigung eine Division durch Null herauskommen kann , führt bei SOLVR nicht zum Error (sehr intelligent!), bei Polynomen findet dies Funktion trotzdem mehrfache Nullstellen! Bei Polynomen bietet sich zwar die Polynomdivision (Deflation) an, aber eine ungenaue Nullstelle (mehrfache Nullstellen sind immer ungenau!) verändert die Koeffizienten des reduzierten Polynoms, so daß am Ende vielleicht komplexe Nullstellen berechnet werden, obwohl das Originalpolynom nur reelle hatte. Dagegen verändert die von NULLWEG durchgeführte Methode die Lage der noch unentdeckten Nullstellen nicht.

Bei den nächsten beiden Programmen handelt es sich um 48SX-Versionen von bereits veröffentlichten 28S-Programmen. **POLYFIT** (siehe "Regression mit Polynomen", PRISMA 88.3.42 und "Komfort bei Polynomberechnungen", PRISMA 89.3.38) und **NLIN** ("Der Zinsling", PRISMA 89.4.30) haben folgende Dinge gemeinsam: Sie beziehen die benötigten statistischen Daten aus der Statistikmatrix ∑DAT, sie stellen mit Hilfe der Matrixalgebra nach der Methode der kleinsten Quadrate ein Gleichungssystem auf und verbessern die gewonnen Ergebnisse mit dem Gradientenverfahren (1-Schritt-Residuumskorrektur).

Mit POLYFIT kann man einer Reihe von (n+1) Meßwerten nicht nur eine Gerade anpassen(=lineare Regression), sondern auch quadratische, kubische Parabeln usw., höchstens bis zum Grad n. Um die Koeffizienten dieses Polynoms möglichst genau zu berechnen, ermittelt man bei Messungen oft viel mehr als n+1 Punkte, obwohl man nur ein Polynom n-ten Grades sucht. POLYFIT erwartet in Ebene 1 den Grad des zu errechnenden Polynoms. Dabei kann das Ergebnis durchaus einen niedrigeren Grad haben, keinesfalls aber einen höheren. Aus der Liste ∑PAR holt sich POLYFIT die Nummer der x- und y-Spalte in ∑DAT, das Ergebnis ist ein Polynom in Ebene 1, dessen Variable das Programm INDP? festlegt. In L speichert POLYFIT einen Vektor, der die Koeffizienten des Regressionspolynoms enthält, so daß man sofort HORNER, ABL1 usw. anwenden kann. Mehr dazu zeigen die Beispiele.

Ebenfalls zur Lösung von Regressionsproblemen taugt **NLIN**, welches eine lineare Funktion mit mehr als einer Variablen an die Punkte in ∑DAT anpaßt. Der Unterschied zu POLYFIT besteht darin, daß eine Funktion mit m Variablen auch m x-Werte braucht und nicht nur einen, wie POLYFIT.

∑DAT braucht für m Variable m+1 Spalten: Die Spalten 1 bis m enthalten die x-Werte, die (m+1)ste Spalte enthält die y-Werte der gesuchten Funktion.

Eingaben braucht NLIN nicht, bei der Ausgabe wählt NLIN als Variable X(i), wobei i von 1 bis m läuft; INDP? verwendet das Programm nicht, da der Plotter nur eindimensionale Funktionen darstellen kann. Das Einsetzen geht auch ganz einfach, indem man die X-Werte von X(1) bis X(m) in einen Vektor speichert (hat man wie im zweiten Beispiel eine Substitution mit z gewählt, sind die entsprechenden z-Werte zu wählen) und diesen wiederum in die Variable 'X' speichert. Jetzt wendet man ein einfaches EVAL auf das algebraische Objekt in Ebene 1 an und bekommt y. In der Variablen L speichert NLIN einen Vektor der die Koeffizienten der Gleichung enthält. Dabei steht a_1 an der Position 1, a_2 an Position 2 ...

Für die Berechnung von y gibt es so noch eine zweite Möglichkeit: Den Vektor X wie eben beschrieben erzeugen, dann X und L in den Stack und mit der Funktion DOT verknüpfen. Wer weitere Möglichkeiten des Programms erforschen will, sollte unbedingt die Beispiele in PRISMA 89.4.31 probieren.

Zum weiteren Dunstkreis statistischer Programme gehören **QM**, **GM** (28S: XROOT in Zeile 6 durch INV ^ ersetzen) und **HM**. Die Programme benötigen keine Eingaben im Stack und ähneln der eingebauten Funktion MEAN, die das arithmetische Mittel jeder Spalte in ∑DAT ermittelt und in einen Vektor schreibt.

Einen solchen Vektor ermitteln auch die drei Programme, nur daß QM das quadratische, GM das geometrische und HM das harmonische Mittel berechnet. Sollte in einer Spalte eine Null vorkommen, ist das harmonische Mittel dieser Spalte Null, enthält eine Spalte Null oder einen negativen Wert, ist das geometrische Mittel dieser Spalte Null.

Die 28S-Programme KR und S aus PRISMA 88.4.38 habe ich ebenfalls an den 48SX angepasst. **KR** berechnet die Krümmung einer Funktion. In Ebene 2 erwartet KR die Funktion und in Ebene 1 die Variable, also wie auch die Funktion ∂ ihre Argumente verlangt. Das Ergebnis ist eine Funktion, welche die Krümmung an jeder Stelle berechnen kann, und falls die Variable existiert, ein EVAL zum Einsetzen ausreicht.

Die Krümmung Null bedeutet entweder einen Wende-/Sattelpunkt der Funktion oder daß diese eine Gerade ist. Vorstellen kann man sich das so: Fährt man mit dem Fahrrad auf dem Graph einer Funktion entlang, dann gibt die Krümmung an, wie weit man den Lenker einschlagen muß (Krümmung 0 = geradeaus fahren). In der Praxis fragt man nach der Krümmung, wenn es darum geht, wie weit sich Bauteile (z.B. Flügel am Flugzeug) durchbiegen dürfen, bevor sie brechen. Die meisten technischen Werkstoffe vertragen allerdings so wenig Durchbiegung, daß man näherungsweise sagt, die Krümmung ist gleich der zweiten Ableitung der Durchbiegung.

Die Funktion **BOGEN** berechnet die Bogenlänge einer Funktion, also den Weg, den man zurücklegt, wenn man auf der Funktion wandert. Praktisches Beispiel: Wenn die Wäscheleine durchhängt, folgt sie der Funktion Cosinus hyperbolicus (Kettenlinie, Befehl: COSH). Kennt man den Abstand zwischen den Aufhängungspunkten und eine weitere Größe (tiefste Stelle, Vorspannung), läßt sich berechnen, wieviel Wäscheleine man kaufen muß. BOGEN verlangt die gleichen Eingaben wie die Funktion ∫, also die Stelle bei der begonnen wird, in Ebene 4, das Integrationsende in Ebene 3, die Funktion in Ebene 2 und die Variable in Ebene 1.

Sowohl KR als auch BOGEN prüfen, ob die Variable, die bei der Eingabe in Ebene 1 spezifiert wurde, bereits existiert. Falls ja, retten die Programme deren Inhalt im Stack und speichern ihn nach Programmende zurück, weil sonst das Ableiten nicht klappt. Allerdings verlegen die Programme die Variable dabei an den Anfang des VARS-Menues.

In Pas de deux II habe ich zwar viele Programme zur Berechnung des größten gemeinsamen Teilers vorgestellt, hier kommt aber die absolut kürzeste Version: **GGT**.

Zum Schluß noch ein paar 48SX-Bugs:

1. In Ebene 1 gibt man das algebraische Objekt '-5/2' ein und führt EXPAN oder →Q aus. Nach einer dieser Operationen hinterläßt die Funktion OBJ→ das algebraische Objekt '-5' in Ebene 4. Abhilfe mit EVAL möglich.
2. Speichert man in einer Variablen (z.B. MAT1) eine Matrix und möchte ein Element (z.B. in der zweiten Zeile und dritten Spalte) bearbeiten, so kann man das mit dem Namen (hier: 'MAT(2/3)') und den Funktionen STO, STO+ usw. machen, nur RCL funktioniert nicht.

Ralf Pfeifer (116)
Rubenstraße 5
5000 Köln 50

Extended IL ROM für den HP41

Mit dem Ext IL Rom ist es möglich Massenspeicher bis zu 1 Megabyte zu adressieren (beim HP-IL Modul werden lediglich 128 Kilobyte adressiert/benutzt). 3 neue Datentypen werden unterstützt:

a XMEM - der erweiterte Speicher,
b CALC - sämtliche Konstanten des Rechners,
c BUFF - individuelle Bufferfiles.

Mit dem Modul wird es möglich, den Rechner teilweise oder komplett als Datendatei auf dem Massenspeicher abzulegen (Tastaturzuweisungen, Rechnerkonstanten, Hauptspeicher und Hauptspeicher zusammen mit dem erweiterten Speicher = XMemory).

Erläuterungen zu den Begriffen

IL - Abkürzung für Interface Loop.

Loop - Term für die HP-IL Schleife.

Programmzeiger - Ein Register welches die gerade aktuelle Adresse enthält.

Massenspeicher - Ein Gerät das Daten speichert (Kassettenlaufwerk).

Medium - Physikalisches Objekt, welches Daten speichern kann (Kassette).

Format - Vordefinierte Einteilung des Mediums.

Volume Label - Mit dem Volume Label wird dem Medium ein Name zugeteilt, der bei einem Listing mit SDIR in der ersten Zeile gedruckt wird.

Adresse - jedes Gerät in der Loop bekommt eine bestimmte Nummer (Adresse) zugeteilt. Der HP 41 hat Adresse 0, das erste Gerät in der Loop 1 usw..

Aid - Abkürzung für Accessory Id. Dies ist eine Nummer für ein HP IL Gerät. Der Bereich reicht von 0 bis 255 und ist in Unterbereiche (16 er Schritte) eingeteilt. So bedeutet eine AID von 0 bis 15 (16-31), daß das Gerät zur Klasse der Loop Kontroller (Massenspeicher) gehört.

Buffer - sind Regionen im Speicher unterhalb der .END. Marke. Buffer können nur von ROM's eingerichtet werden.

Selected Device - ist das aktive Gerät in der Loop von dem aus begonnen wird nach Druckern und Massenspeicher zu suchen.

Struktur eines Massenspeichers

Das Medium eines HP IL Massenspeichers ist in Sektionen mit 256 Bytes Länge partitioniert. Jede Sektion ist ein Record bzw. die kleinste benutzbare Speichereinheit. Dies bedeutet, daß ein Programm, das 20 Bytes lang ist, 256 Bytes oder einen Record beim Abspeichern auf das Medium benötigt.

Beim Formatieren werden alle Bytes auf den Wert FF (Hex) gesetzt. Danach werden die ersten beiden Records (512 Bytes) durch den Wert 00 (Hex) überschrieben. Danach enthalten die ersten 20 Bytes des ersten Records (Record 0) Informationen über die Größe des Verzeichnisses und das Volume Label.

Das Verzeichnis beginnt mit dem 2. Record (0, 1, 2). Jeder Verzeichniseintrag kann 32 Bytes lang sein, sodaß 8 Verzeichnisse in einen Record passen. Der letzte Verzeichniseintrag des letzten Verzeichnisrecords wird von dem System als Markierung für das Ende des Verzeichnisses benutzt.

Im folgenden werden die Befehle des Moduls erläutert, der analoge Befehl des HP-IL Modules wird, sofern vorhanden, in eckigen Klammern angegeben:

SNEWM [NEW]
modifizierter NEW Medium Befehl des HP-IL Modules, formatiert ein Medium.

NAMEMED
(ohne zu formatieren) im Gegensatz zum Befehl zuvor wird dem Medium ein Name gegeben.

SCREATE [CREATE]
gleiche Funktion wie CREATE aus dem HP-IL Modul, die Dateigröße ist auf 65535 Register erweitert worden (das entspricht einer Dateigröße von einem halben Megabyte) sofern das Medium soviel Platz hat.

SWRTA [WRTA]
schreibt den Hauptspeicher des Rechners auf das Medium (für den HP41 CV/CX beträgt die Dateigröße 336 Register).

SWRTK
schreibt die Tastaturzuweisungen der Katalogfunktionen 2 und 3 auf's Medium.

SWRTP, SWRTPV [WRTP, WRTPV]
schreibt ein Programm auf das Medium.

SWRTS
schreibt den Status des Rechners auf das Medium. Diese Datei ist immer 10 Register groß und enthält Informationen über die Position der Speichergrenze, Programmregister/Datenregister, sowie des Sigmaregisters, den Zustand der Flags 00 bis 43, den Inhalt der Register X, Y, Z, T, LASTX und den des Alpharegisters.

WRTBUFX, READBUF
schreibt/liest einen Buffer auf das/vom Medium

WRTXM, READXM
schreibt/liest den erweiterten Speicher des Rechners. Lediglich der benutzte Teil des erweiterten Speichers wird gesichert.

WRTCAL, READCAL
schreibt/liest den gesamten Rechnerinhalt (Haupt- und erweiterter Speicher). Die Dateigröße beträgt 337 Register plus die Anzahl der Datenregister des benutzten erweiterten Speichers.

SDIR [DIR]
Verzeichnisliste. Diese Funktion erkennt die neuen Dateitypen.

DIRSIZE
ergibt die Anzahl der freien Verzeichniseinträge.

DIRLEFT
ergibt die Anzahl der freien Verzeichniseinträge welche noch auf dem Medium Platz haben.

RECLEFT
ergibt die Anzahl der freien Records welche noch auf dem Medium Platz haben.

SCOPYFL
kopiert Dateien zwischen zwei verschiedenen Massenspeichern sofern der Dateityp vom HP41 erkannt wird.

XAR
diese Funktion konvertiert eine Zahl zwischen 0 und 255 in ein Alphazeichen welches als letztes Zeichen des Alpha Registers angehängt wird.

CLRBUF
löscht einen Buffer um beispielsweise mehr Platz im Rechner zu haben.

PRTAID
diese Funktion (printer accessory id) erlaubt es das Gerät zu wählen, zu dem die zu druckenden Daten gesendet werden sollen (ausgehend vom Ext-IL Rom).

ATOBUFX, SACA
eine Möglichkeit ist das komfortable Umschalten verschiedener Druckmodi.

ATOBUFX
speichert einen Alpha String im Ext-IL Rom Buffer. Diese Strings werden durch eine Zahl von 0 bis 31 identifiziert. SACA greift auf diese Strings zurück.

SACA [ACA]
akkumuliert Zeichen vom ALpha Register in den Drucker Buffer. Das Zeichen 13 (das Winkelzeichen) wird nicht in das Zeichen 124 umgewandelt, weil der HP Drucker der einzige Drucker ist, bei dem das Winkelzeichen durch das Zeichen 124 definiert ist.

Das Zeichen 13 ist bei den meisten Druckern ein CR (Carriage Return). Außerdem wird das Sigmazeichen 126 nicht in Zeichen 28 umgewandelt weil dieses Zeichen in den meisten Druckerzeichensätzen nicht vorhanden ist.

Es können Zeichen im Wertebereich von 0 bis 255 an den Drucker geschickt werden. Bei Zeichen mit Werten zwischen 1 und 31 wird zuerst im Ext-IL Rom Buffer nachgesehen ob dort ein String mit diesem Wert abgespeichert wurde. Wenn dies der Fall ist wird der abgespeicherte String an den Drucker geschickt. Ist kein String mit dem Wert abgespeichert wirt der Wert selbst an den Drucker geleitet.

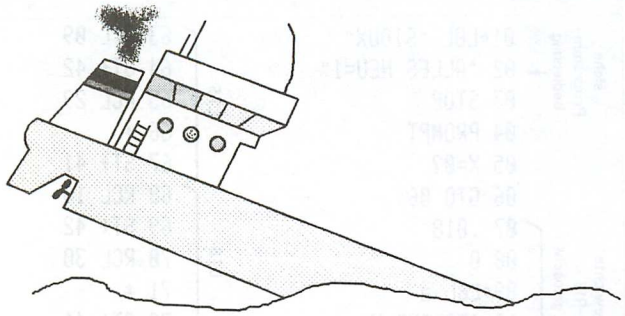
MCPRP, MCLIST
multiple Spalten Druck/Listroutine. Mit diesen Funktionen können mehrspaltige Programmlistings auf einfache Art und Weise erstellt werden.

Hinweis: Da das Ext IL Rom dieselbe Id-Nummer wie der Barcodelesestift bnutzt, muß das Ext IL Rom in den Port mit der kleineren Nummer als der Barcodelesestift. Mit dieser Konfiguration ist es dann möglich Barcodes zu lesen. Die Rom Funktionen des Barcodelesestiftes (WANDSCN, WANDSUB etc.) können aber nicht benutzt werden.

Wer weitere Details zu diesem Modul wissen möchte,, kann sich an Dr. Martin Hochenegger, Heidelberger Landstraße 97, 6100 Darmstadt 13 wenden, der auch das Handbuch für diesen Artikel zur Verfügung gestellt hat. mik

Schiffstrimmung

mit dem HP41



HP41CV
624 Zeilen, 1118 Bytes, 160 REGs., SIZE 100

Damit einem nicht das gerade gezeigte Mißgeschick passiert hat Wolfgang Teggler das nun folgende Programm geschrieben, das als Grundlage für weitere Lösungen auf diesem Gebiet angesehen werden kann. Die Dokumentation der einzelnen Schritte und Verfahrensweisen soll dies erleichtern, da es ebensoviele Schiffstypen wie Binnenseen gibt. Alle haben beim Befahren der Weltmeere eins gemeinsam: falsche Beladung bzw. Verteilung derselben kann im Extremfall zum Untergang führen...
Anhand der Skizze kann man sich eine bessere Vorstellung darüber machen, welche Probleme die Beladung eines solchen Schiffes bereitet.

Bei diesem Programm handelt es sich um ein Trimmprogramm für einen bestimmten Schiffstyp *(M/T INDIO)*.
Abhilfe leistet das Programm, indem es die verschiedenen Tiefgänge und Vertrimmung des Schiffes berechnet, nachdem man das Schiff über die Speicher 00 bis 18 "beladen" oder "entladen" hat, d.h. die Gewichte in den Speicher eingegeben und ggf. variiert hat.
Dabei beachtet das Programm einige Grenzwerte, zuviel oder zuwenig "Displacement" und quittiert diese mit "NEIN". Eine Schlagseite wird bei dieser Programmversion vernachlässigt. In der Praxis wird eine Schlagseite durch Verteilung mit Ladung oder durch Füllen oder Lenzen von Ballasttanks ausgeglichen.

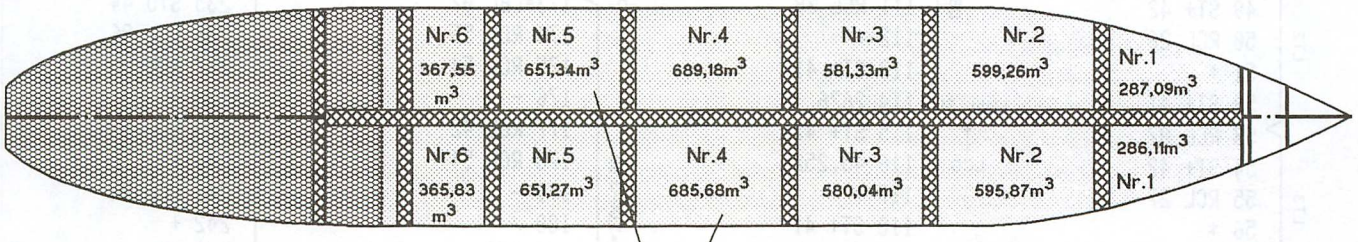
Die Stärke des Programmes liegt darin, daß man *vor* der Beladung eines Schiffes "durchspielen" kann, wie das Schiff liegt, wenn man diesen oder jenen Tank befüllt bzw. lenzt und ob man besser Ballast nimmt. Außerdem eignet sich dieses Programm gut zur Tiefgangbeschränkung:
KOPF bedeutet das Schiff liegt vorne tiefer als achtern oder hinten. Umgekehrt bedeutet GATT, daß das Schiff hinten tiefer liegt. Zum Fahren möchte man das Schiff ca. 0,35 cm in Glattlage haben, weil es dann auf See und beim Befahren enger Gewässer besser zu steuern ist. Zum Vermessen der Ladung soll es dann wieder auf EBENER KIEL liegen. Im Programm sind sicherlich noch Verbesserungen/Straffungen möglich.

Programmbedienung:
Nach dem Programmstart mit XEQ "SIOUX" erscheint "ALLES NEU = 1".

Wird das Program mit 1 "R/S" fortgesetzt, so werden alle Speicher für die Gewichte auf NULL (0) gesetzt und müssen neu eingegeben werden.
Wird das Program hingegen mit 0 "R/S" nach "ALLES NEU = 1" fortgesetzt, so bedeutet dies, daß alle bereits gespeicherten Gewichte erhalten bleiben und einzeln variiert werden können.

Benutzte Abkürzungen:

HFO	=	HEAVY FUEL OIL
LO	=	LUBEOIL
GO	=	GASOIL
CON	=	CONSTANTS (feste Ausrüstung SIOUX/UNKAS 80MT SIOUX/INDIO 140MT)
LT	=	LADETANK
VP	=	VORPIEK (Ballastwasser)
DB	=	DOPPELBODEN (Ballastwasser)
LCG	=	LONGITUDINAL CENTER OF GRAVITY (before APP)
MCT	=	MOMENT TO CHANGE TRIMM (MTO/M)
LCB	=	LONGITUDINAL CENTER OF BUOYANCE (before APP)
LPP	=	105,92M
FPP-AHMING	=	-0,76m
APP-AHMING	=	+6,00m
L AHM	=	99,16m
LIGHT SHIP	=	2626

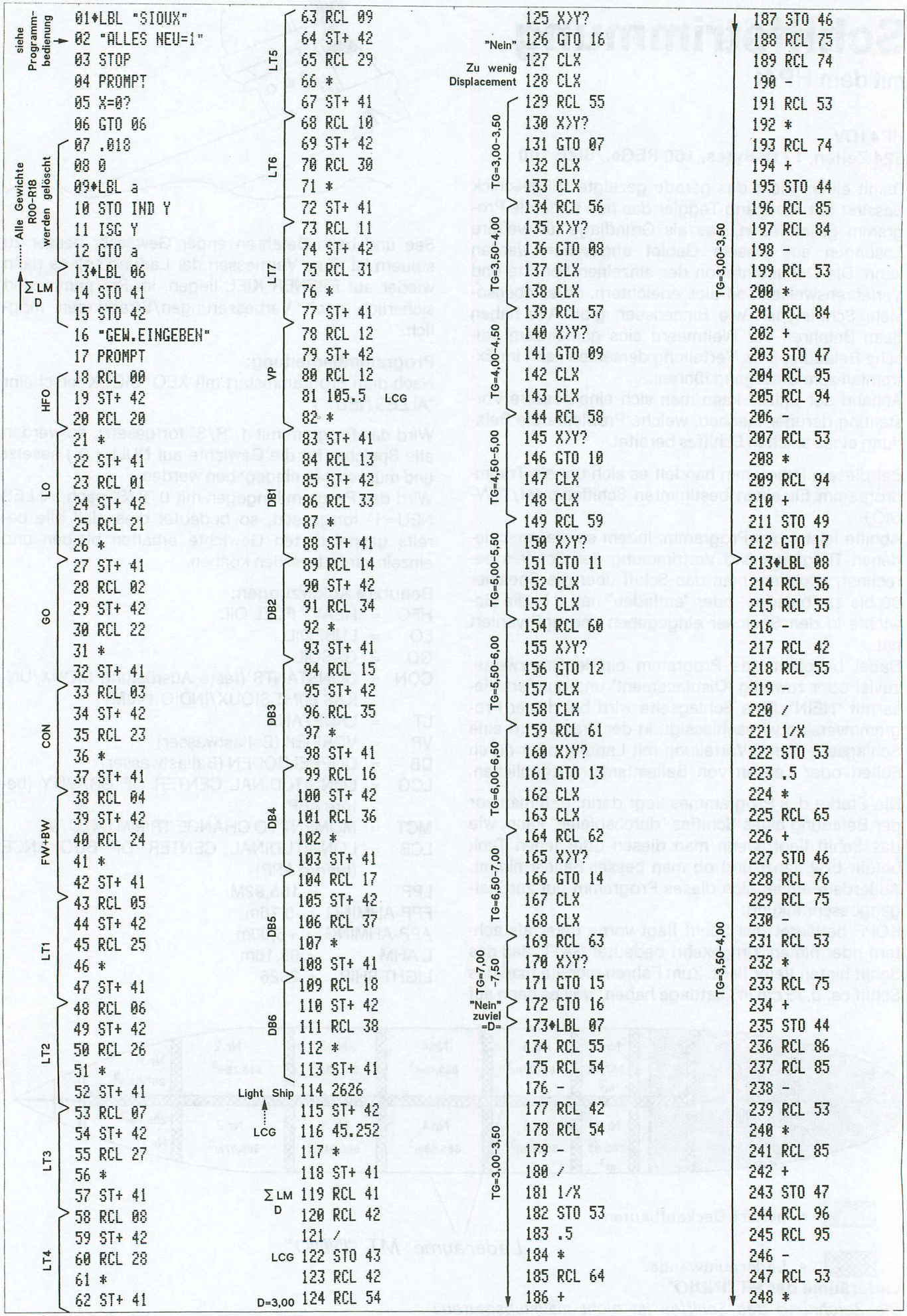


= hintere Deckaufbauten

= Laderaumwände

Laderäume MT "INDIO"

Laderäume der MT "INDIO"
Die Zeichnung des Schiffes ist nicht maßstabsgetreu!


siehe Programm-bedienung
01♦LBL "SIOUX"
02 "ALLES NEU=1"
03 STOP
04 PROMPT
05 X=0?
06 GTO 06
07 .018
08 0
09♦LBL a
10 STO IND Y
11 ISG Y
12 GTO a
13♦LBL 06
14 STO 41
15 STO 42
16 "GEW.EINGEBEN"
17 PROMPT
18 RCL 00
19 ST+ 42
20 RCL 20
21 *
22 ST+ 41
23 RCL 01
24 ST+ 42
25 RCL 21
26 *
27 ST+ 41
28 RCL 02
29 ST+ 42
30 RCL 22
31 *
32 ST+ 41
33 RCL 03
34 ST+ 42
35 RCL 23
36 *
37 ST+ 41
38 RCL 04
39 ST+ 42
40 RCL 24
41 *
42 ST+ 41
43 RCL 05
44 ST+ 42
45 RCL 25
46 *
47 ST+ 41
48 RCL 06
49 ST+ 42
50 RCL 26
51 *
52 ST+ 41
53 RCL 07
54 ST+ 42
55 RCL 27
56 *
57 ST+ 41
58 RCL 08
59 ST+ 42
60 RCL 28
61 *
62 ST+ 41
Alle Gewichte R00-R18 werden gelöscht
Σ LM D
HFO
LO
GO
CON
FW+BW
LT1
LT2
LT3
LT4
63 RCL 09
64 ST+ 42
65 RCL 29
66 *
67 ST+ 41
68 RCL 10
69 ST+ 42
70 RCL 30
71 *
72 ST+ 41
73 RCL 11
74 ST+ 42
75 RCL 31
76 *
77 ST+ 41
78 RCL 12
79 ST+ 42
80 RCL 12
81 105.5
82 *
83 ST+ 41
84 RCL 13
85 ST+ 42
86 RCL 33
87 *
88 ST+ 41
89 RCL 14
90 ST+ 42
91 RCL 34
92 *
93 ST+ 41
94 RCL 15
95 ST+ 42
96 RCL 35
97 *
98 ST+ 41
99 RCL 16
100 ST+ 42
101 RCL 36
102 *
103 ST+ 41
104 RCL 17
105 ST+ 42
106 RCL 37
107 *
108 ST+ 41
109 RCL 18
110 ST+ 42
111 RCL 38
112 *
113 ST+ 41
114 2626
115 ST+ 42
116 45.252
117 *
118 ST+ 41
119 RCL 41
120 RCL 42
121 /
122 STO 43
123 RCL 42
124 RCL 54
LT5
LT6
LT7
VP
LCG
DB1
DB2
DB3
DB4
DB5
DB6
Light Ship
LCG
Σ LM D
LCG
D=3,00
125 X>Y?
126 GTO 16
"Nein"
Zu wenig Displacement
127 CLX
128 CLX
129 RCL 55
130 X>Y?
131 GTO 07
132 CLX
133 CLX
134 RCL 56
135 X>Y?
136 GTO 08
137 CLX
138 CLX
139 RCL 57
140 X>Y?
141 GTO 09
142 CLX
143 CLX
144 RCL 58
145 X>Y?
146 GTO 10
147 CLX
148 CLX
149 RCL 59
150 X>Y?
151 GTO 11
152 CLX
153 CLX
154 RCL 60
155 X>Y?
156 GTO 12
157 CLX
158 CLX
159 RCL 61
160 X>Y?
161 GTO 13
162 CLX
163 CLX
164 RCL 62
165 X>Y?
166 GTO 14
167 CLX
168 CLX
169 RCL 63
170 X>Y?
171 GTO 15
172 GTO 16
"Nein" zuviel =D=
173♦LBL 07
174 RCL 55
175 RCL 54
176 -
177 RCL 42
178 RCL 54
179 -
180 /
181 1/X
182 STO 53
183 .5
184 *
185 RCL 64
186 +
TG=3,00-3,50
TG=3,50-4,00
TG=4,00-4,50
TG=4,50-5,00
TG=5,00-5,50
TG=5,50-6,00
TG=6,00-6,50
TG=6,50-7,00
TG=7,00 -7,50
TG=3,00-3,50
187 STO 46
188 RCL 75
189 RCL 74
190 -
191 RCL 53
192 *
193 RCL 74
194 +
195 STO 44
196 RCL 85
197 RCL 84
198 -
199 RCL 53
200 *
201 RCL 84
202 +
203 STO 47
204 RCL 95
205 RCL 94
206 -
207 RCL 53
208 *
209 RCL 94
210 +
211 STO 49
212 GTO 17
213♦LBL 08
214 RCL 56
215 RCL 55
216 -
217 RCL 42
218 RCL 55
219 -
220 /
221 1/X
222 STO 53
223 .5
224 *
225 RCL 65
226 +
227 STO 46
228 RCL 76
229 RCL 75
230 -
231 RCL 53
232 *
233 RCL 75
234 +
235 STO 44
236 RCL 86
237 RCL 85
238 -
239 RCL 53
240 *
241 RCL 85
242 +
243 STO 47
244 RCL 96
245 RCL 95
246 -
247 RCL 53
248 *
TG=3,00-3,50
TG=3,50-4,00

```
TG=4,00-4,50
249 RCL 95
250 +
251 STO 49
252 GTO 17
253♦LBL 09
254 RCL 57
255 RCL 56
256 -
257 RCL 42
258 RCL 56
259 -
260 /
261 1/X
262 STO 53
263 .5
264 *
265 RCL 66
266 +
267 STO 66
268 RCL 77
269 RCL 76
270 -
271 RCL 53
272 *
273 RCL 76
274 +
275 STO 44
276 RCL 87
277 RCL 86
278 -
279 RCL 53
280 *
281 RCL 86
282 +
283 STO 47
284 RCL 97
285 RCL 96
286 -
287 RCL 53
288 *
289 RCL 96
290 +
291 STO 49
292 GTO 17
293♦LBL 10
294 RCL 58
295 RCL 57
296 -
297 RCL 42
298 RCL 57
299 -
300 /
301 1/X
302 STO 53
303 .5
304 *
305 RCL 67
306 +
307 STO 46
308 RCL 78
309 RCL 77
310 -
TG=4,50-5,00
311 RCL 53
312 *
313 RCL 77
314 +
315 STO 44
316 RCL 88
317 RCL 87
318 -
319 RCL 53
320 *
321 RCL 87
322 +
323 STO 47
324 RCL 98
325 RCL 97
326 -
327 RCL 53
328 *
329 RCL 97
330 +
331 STO 49
332 GTO 17
333♦LBL 11
334 RCL 59
335 RCL 58
336 -
337 RCL 42
338 RCL 58
339 -
340 /
341 1/X
342 STO 53
343 .5
344 *
345 RCL 68
346 +
347 STO 46
348 RCL 79
349 RCL 78
350 -
351 RCL 53
352 *
353 RCL 78
354 +
355 STO 44
356 RCL 89
357 RCL 88
358 -
359 RCL 53
360 *
361 RCL 88
362 +
363 STO 47
364 RCL 99
365 RCL 98
366 -
367 RCL 53
368 *
369 RCL 98
370 +
371 STO 49
372 GTO 17
TG=5,00-5,50
TG=5,50-6,00
373♦LBL 12
374 RCL 60
375 RCL 59
376 -
377 RCL 42
378 RCL 59
379 -
380 /
381 1/X
382 STO 53
383 .5
384 *
385 RCL 69
386 +
387 STO 46
388 RCL 80
389 RCL 79
390 -
391 RCL 53
392 *
393 RCL 79
394 +
395 STO 44
396 RCL 90
397 RCL 89
398 -
399 RCL 53
400 *
401 RCL 89
402 +
403 STO 47
404 RCL 19
405 RCL 99
406 -
407 RCL 53
408 *
409 RCL 99
410 +
411 STO 49
412 GTO 17
413♦LBL 13
414 RCL 61
415 RCL 60
416 -
417 RCL 42
418 RCL 60
419 -
420 /
421 1/X
422 STO 53
423 .5
424 *
425 RCL 70
426 +
427 STO 46
428 RCL 81
429 RCL 80
430 -
431 RCL 53
432 *
433 RCL 80
434 +
TG=6,00-6,50
435 STO 44
436 RCL 91
437 RCL 90
438 -
439 RCL 53
440 *
441 RCL 90
442 +
443 STO 47
444 RCL 39
445 RCL 19
446 -
447 RCL 53
448 *
449 RCL 19
450 +
451 STO 49
452 GTO 17
453♦LBL 14
454 RCL 62
455 RCL 61
456 -
457 RCL 42
458 RCL 61
459 -
460 /
461 1/X
462 STO 53
463 .5
464 *
465 RCL 71
466 +
467 STO 46
468 RCL 82
469 RCL 81
470 -
471 RCL 53
472 *
473 RCL 81
474 +
475 STO 44
476 RCL 92
477 RCL 91
478 -
479 RCL 53
480 *
481 RCL 91
482 +
483 STO 47
484 RCL 40
485 RCL 39
486 -
487 RCL 53
488 *
489 RCL 39
490 +
491 STO 49
492 GTO 17
493♦LBL 15
494 RCL 63
495 RCL 62
496 -
TG=6,50-7,00
TG=7,00-7,50
```

```
497 RCL 42
498 RCL 62
499 -
500 /
501 1/X
502 STO 53
503 .5
504 *
505 RCL 72
506 +
507 STO 46
508 RCL 83
509 STO 82
510 -
511 RCL 53
512 *
513 RCL 82
514 +
515 STO 44
516 RCL 93
517 RCL 92
518 -
519 RCL 53
520 *
521 RCL 92
522 +
523 STO 47
524 RCL 32
525 RCL 40
526 -
527 RCL 53
528 *
529 RCL 40
530 +
531 STO 49
532 GTO 17
533♦LBL 16
534 "NEIN"
535 PROMPT
536♦LBL 17
537 RCL 43
538 RCL 44
539 -
540 STO 45
541 RCL 42
542 RCL 47
543 /
544 105.92
545 /
546 RCL 45
547 *
548 STO 48
549 RCL 49
550 CHS
551 RCL 48
552 *
553 RCL 46
554 +
555 STO 50
556 105.92
557 RCL 49
558 -
559 RCL 48
560 *
561 RCL 46
562 +
563 STO 51
564 RCL 50
565 -
566 100
567 *
568 .5
569 +
570 INT
571 100
572 /
573 STO 52
574 FIX 2
575 X>0?
576 GTO 01
577 GTO 02
578♦LBL 01
579 RCL 52
580 "KOPF"
581 ARCL X
582 PROMPT
583 GTO 04
584♦LBL 02
585 X<0?
586 RCL 03
587 "EBENER KIEL"
588 PROMPT
589 GTO 04
590♦LBL 03
591 RCL 52
592 CHS
593 "GATT"
594 ARCL X
595 PROMPT
596 GTO 04
597♦LBL 04
598 RCL 51
599 "VORD.LOT"
600 ARCL X
601 PROMPT
602 RCL 50
603 "ACHT.LOT"
604 ARCL X
605 PROMPT
606 -.76
607 RCL 52
608 *
609 99.16
610 /
611 RCL 51
612 +
613 "AHMING V"
614 ARCL X
615 PROMPT
616 6
617 RCL 52
618 *
619 99.16
620 /
621 RCL 50
622 +
623 "AHMING H"
624 ARCL X
625 AVIEW
626 END
```

TG=7,00-7,50 · zuviel oder zuwenig Displace-ment · Trimmhebel · TGH · TGV · Lpp · TGV · Trimm · Kopf- Gatt- · Kopflage · Ebener Kiel · Gattlage · TGV-Lot · TGM-Lot · TGV-Ahming · Fpp-Ahm. · Lpp · TGH-Ahming · TGH-Ahming

Benutzte Register der Gewichte:

HFO	= R 00	LT6	= R 10
LO	= R 01	LT7	= R 11
GO	= R 02	VP	= R 12
CON	= R 03	DB1	= R 13
FW + BW	= R 04	DB2	= R 14
LT1	= R 05	DB3	= R 15
LT2	= R 06	DB4	= R 16
LT3	= R 07	DB5	= R 17
LT4	= R 08	DB6	= R 18
LT5	= R 09		

Allgemeine Speicherbelegung (Register):

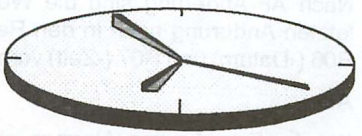
19	51,23	LCF 6,00m	59	6628	D-5,50/SW
20	21,50	ϕLCG HFO	60	7328	D-6,00/SW
21	12,16	ϕLCG LO	61	8050	D-6,50/SW
22	19,91	ϕLCG GO	62	8793	D-7,00/SW
23	26,00	ϕLCG CON	63	9556	D-7,50/SW
24	2,03	ϕLCG FW+BW			
25	92,70	LCG LT1	64	3,00	
26	80,70	LCG LT2	65	3,50	
27	68,42	LCG LT3	66	4,00	
28	55,89	LCG LT4	67	4,50	
29	42,62	LCG LT5	68	5,00	
30	32,60	LCG LT6	69	5,50	
31	27,54	LCG LT7	70	6,00	
			71	6,50	
32	49,88	LCF 7,50m	72	7,00	
			73	7,50	
33	93,61	LCG DB1			
34	80,12	LCG DB2	74	54,25	LCB/3,00m
35	68,07	LCG DB3	75	54,25	LCB/3,50m
36	55,58	LCG DB4	76	54,23	LCB/4,00m
37	42,69	LCG DB5	77	54,19	LCB/4,50m
38	31,40	LCG DB6	78	54,12	LCB/5,00m
			79	53,99	LCB/5,50m
39	50,49	LCF 6,50m	80	53,80	LCB/6,00m
40	50,07	LCF 7,00m	81	53,52	LCB/6,50m
			82	53,23	LCB/7,00m
41	.. Summe LM		83	52,94	LCB/7,50m
42	.. Summe Einz.,Gew.=D				
43	.. LM/D = LCG		84	6557	MCT/3,00m
44	.. LCG für aktuellen TFG		85	6818	MCT/3,50m
45	.. Trimmhebel		86	7085	MCT/4,00m
46	.. TGM aktuell		87	7379	MCT/4,50m
47	.. MCT für aktuellen TFG		88	7783	MCT/5,00m
48	.. D/MCT akt.TFG/LPP*TH		89	8400	MCT/5,50m
49	.. LCF für aktuellen TFG		90	9196	MCT/6,00m
50	.. TGH		91	9965	MCT/6,50m
51	.. TGV		92	10641	MCT/7,00m
52	.. TRIMM (- = Gattlage)		93	11252	MCT/7,50m
53	.. Zwischenergebnisse				
			94	54,15	LCF/3,00m
54	3357	D -3,00/SW	95	54,05	LCF/3,50m
55	3988	D -3,50/SW	96	53,84	LCF/4,00m
56	4630	D -4,00/SW	97	53,56	LCF/4,50m
57	5282	D -4,50/SW	98	53,13	LCF/5,00m
58	5947	D -5,00/SW	99	52,19	LCF/5,50m

R19-R40 und R54-R99 müssen für einen anderen Schiffstyp erst angepaßt werden!

Fortsetzung Seite 69

ALARMVERWALTUNG

Programmpaket für den HP-41



von Günter Schapka

HP-41 CX, (oder CV + TIME + Extended Memory), CCD-Modul, (Drucker), (Rambox oder RSU-Ram Storage Unit), (Assembler); Hinweis zu RSU siehe am Ende der allgemeinen Übersicht

Einzelne Programme dieses Pakets laufen bei mir bereits seit langer Zeit. Seit Einsatz der Rambox brauche ich weder einzelne Programmteile noch Alarme manuell nachzuladen. Das alljährliche Vergessen von z.B. Hochzeitstag oder Geburtstag der reichen Erbtante gehören damit der Vergangenheit an.

Hier eine Übersicht über die einzelnen Programme (die nicht alle auf meinem Mist gewachsen sind) in der Reihenfolge Name, Bytes und Erläuterung:

1.) CAL 368 (mit WIZ und SOZ)

Kalibriert die Uhr
(Nach Jeremy Smith, PPC V10, N9, P27)
und paßt den AF-Faktor [AF = Accuracy Factor = Kompen-sationsfaktor] anhand der Änderungsgröße an. (Mit der Funktion CORRECT ist eine derartige Genauigkeit nie erreichbar !)

2.) AL 248 (mit BEN und <)

Setzt Alarme menügesteuert
(ebenfalls nach Jeremy Smith, PPC V10, N10, P35)

3.) KAL 931

In der Rambox gespeicherte Alarme werden zur Korrektur in den Rechner geholt und nach Korrektur wieder zurückgeladen.

ANL
Automatisches Nachladen von Alarmen aus der Rambox.

RSAL
Menügeführtes Korrigieren von Alarmen.

PRAL
Kompletter Ausdruck der aktuellen und Rambox-Alarme.

MAL
Setzt zwei Alarme auf bestimmte Wochentage.

Hier für jeweils den ersten Dienstag (Text SFC) und letzten Samstag im Monat (Text CCD für regionales CCD-Treffen)

4.) W1 307

Wecker Steuerprogramm
(Anregung durch P.Jemian, PPC V11,N7,P28)

W2/WECK
Weckprogramm

DW
Wecker-Änderungsprogramm

5.) UMDAK 344

Rambox Umspeicherprogramm für einzelne Daten-, Alarm- und Keyfiles; UMDAK löscht die Ausgangsfiles.

UMLAM 154
Kopiert alle 12 ALAM-Alarmfiles auf eine andere Rambox-Page ohne die Ausgangs-Files zu löschen.

6.) SUSAL 29 (Synth. Version)

Suspend Alarms - Alarme vorübergehend außer Kraft setzen, ohne sie zu löschen.
Synthetische Version von Tapani Tarvainen (PPC V10, N7, P10)
MCODE-Version von Kari Pasanen (PPC V10, N8, P36)

7.) WIZ

Umschalten von Sommer auf Winterzeit und umgekehrt

SOZ
inclusive Korrektur der letzten Kalibrierdaten der Uhr.

8.) BEN

Viertelstündliche akustische Uhranzeige "Big Ben" (Nach Idee von Chuck Wheat PPC V12, N4, P4)

Die Programme 1, 2 sowie 4 bis 8 können jedes einzeln für sich benutzt werden. Soll Programm 3 arbeiten können, so müssen auch 5 + 7 bereitstehen (für KAL ist Nr. 5 und für ANL Nr. 7 notwendig). Programm 6 wird vor Programmen eingesetzt, die auf keinen Fall von Alarmen gestört werden dürfen (z.B. Programme die während ihres Ablaufs Zustandsregister c verändern).

Alle Programme sind in der Rambox abgelegt, im Hauptspeicher bleibt lediglich ein Steuerteil PG= (von 198 Bytes Länge), der von ANL, BEN und Weckalarmen angesprungen wird und diese in die Speichererweiterung auf die richtige Ramboxhälfte (nur bei 64K-Rambox erforderlich) weiterleitet.

In der RSU können keine Buffer-(Alarm)-files direkt abgespeichert werden, dieses wäre nur möglich nach Umwandlung der Alarme im Extended Memory in Daten, Speicherung als Datenfile und der entsprechenden Zurückwandlung beim fälligen Nachladen. Es ist natürlich auch möglich Alarmfiles auf Magnetkarten zu speichern und auf Anforderung die nächsten zu laden. Siehe z.B. Handbuch zum CCD-Modul Seite 8.17, Programm WF. Da dieses Programm für alle Filetypen benutzt werden kann sollte man über die Namen bereits Buffertyp und Anzahl der Register erkennen können um Problemen aus dem Weg zu gehen. In diesem Fall könnte ein Name so aussehen: 11AL22A - 11 für Alarme für Monat 11, AL als textliche Kennzeichnung für Alarme, 22 als Anzahl der belegten Buffer-Register, A als Kennzeichen für den Buffertyp: A (hex) = 10 (dec).

Weitere MCODE-Funktionen bzw. z.T. entsprechende synthetische Unterprogramme, die häufig benutzt werden, sind in einem Abschnitt zusammengefaßt: SK2 (Skip 2 Programmschritte), SK3, SK4, SKN (Skip N Programmschritte), X+1 und X-1 (Erhöhen bzw. Erniedrigen des X-Registers um 1 ohne den Stack zu benutzen), VA (View Alpha). POP (Löschen der ersten Rücksprungadresse) war bereits im Prisma 6/89 von mir veröffentlicht worden.

Details der einzelnen Programme:

CAL
CAL holt Datum und Zeit (R00 und R01) der letzten Kalibrierung sowie den dabei errechneten Faktor AF (R02) aus einem Datenfile CaL3, 3 Register groß. Ist kein Datenfile vorhanden, werden die letzten Kalibrierdaten abgefragt (Zeilen 33-35 und 49-51).

Die + und - Tasten werden den Programmteilen "+" und "-" zugewiesen (ab Z. 61).

Nach Anzeige der Uhrzeit (Z. 83) kann damit die Zeit schrittweise solange korrigiert werden, bis sie mit einer Referenzuhr übereinstimmt.

Die Eingabe der Änderungen erfolgt jeweils in Sekunden, ohne Angabe eines Werts wird um 0.02 Sekunden geändert (Z. 82). Die gesamte Änderung wird in R03 gespeichert (Z. 80). Ist die Zeit eingestellt, wird bei Zeitanzeige mit R/S die Berechnung des aus der Änderung resultierenden AF ausgelöst sowie die Werte der Änderung abgespeichert (LBL 15).

Es wird keine Änderung durchgeführt:

- wenn aktueller AF und der Wert von der letzten Kalibrierung (in Reg. 02) nicht übereinstimmen (z.B. weil bei Memory Lost auch der AF auf Null gesetzt wurde).
- wenn AF-Wert und R02 übereinstimmen, aber keine Zeitkorrektur durchgeführt wurde (Sonst würden Datum und Zeit der letzten Änderung durch die aktuellen Zeit und Datum-Werte überschrieben und bei der nächsten Kalibrierung dieser Zeitbereich fehlen, folglich auch der berechnete AF falsch werden).

Nach AF-Änderung sind die Werte der letzten Änderung noch in den Registern R06 (-Datum) und R07 (-Zeit) vorhanden.

AL

Bei der Eingabe von Alarmen stehen 2 Möglichkeiten für die Eingabe von Zeit und Datum zur Verfügung:

Absolutwerte oder relativ zur aktuellen Zeit bzw. Datum.

Wird bei Aufforderung zur Alarmzeiteingabe nichts eingegeben, folgt der Alarmkatalog und die Anzeige der aktuellen Uhrzeit (Z. 44-45). Ein relativer Alarm, in z.B. 3:55 Std., wird nach der Zifferneingabe durch SST ausgelöst (setzt Flag 51). Ziffereingabe ohne SST setzt die Alarmzeit absolut. Datumüberschreitung wird berücksichtigt (Z. 17 bis 31)

DT/dd mit angehängtem aktuellem Datum fordert zur Datumseingabe auf.

Ohne Eingabe bleibt der aktuelle Wert gültig ("heute"). Bei Eingabe eines gültigen Datums kann die Funktion DOW einen Wochentag zuordnen, Errorflag 25 bleibt gesetzt und dieses Datum wird Auslösetag für den Alarm. Bei Eingabe einer Ziffer wird F25 gelöscht und das aktuelle Datum um die Anzahl der eingegebenen Tage erhöht.

Für Wiederholzeit und Message sind keine Besonderheiten notwendig.

KAL

ist das **Lade- und Speicherprogramm** zum Ändern **von Rambox Alarmfiles**.

Es erinnert als erstes daran, *alle* nach dem zu ändernden File stehenden Files mittels Programm UMDAK auf eine andere Page zu verschieben. Die Alarmfiles mit Namen ALAMxx stehen bei mir am Ende von Page 13a.

Wird beim Menü KORR JN mit Ja geantwortet, dann steuert dies zu LBL 04; dort wird bei vorhandenen aktuellen Alarmen F01 gesetzt und diese im Extended Memory unter dem Namen al aufgehoben. Dann wird der Filename erfragt (Z. 65) - nicht mehr als 6 Zeichen benutzen, da er in Register 03 zum späteren Zurückladen gespeichert wird. Dann wird das zu korrigierende File geladen und der Vorhang um 4 Register nach unten verschoben, damit RSAL beim Ändern keine Daten von KAL überschreibt.

LBL 20 ist das CU-Programm von Wolfgang Padberg, Prisma 86.1.20 . Mittels Programm RSAL können die jetzt im Rechner befindlichen Alarme korrigiert oder gelöscht, mittels AL neue hinzugefügt werden.

Danach wird wieder KAL aufgerufen, bei der Antwort Nein auf die Frage KORR? JN wird zu LBL 08 gesteuert, die Alarme unter ihrem alten Namen (R03) in die Rambox (bei mir auf Page 13) zurückgespeichert sowie in Abhängigkeit vom Zustand der Flag 00 die ursprünglichen Alarme (al) aus dem Extended Memory geholt und zum Abschluß angezeigt "ALAMxx in 13".

ANL

ist der Alarm-Nachladeteil, ausgeführt über einen Steueralarm ↑↑ANL.

Der Ablauf geschieht nach folgender Reihenfolge:

1. Vorhandene Alarme werden im Extended Memory gespeichert
2. Neue und vorhandene Alarme werden in einen Buffer kombiniert
3. Update der Alarme, wenn notwendig, d.h.
   a) solche, die mehr als 35 Tage im Voraus liegen, werden gelöscht,
   b) solche, die in der Jahreszahl zurückliegen werden aufs aktuelle Jahr gesetzt.
4. Nach dem Zusammenführen läuft der Alarmkatalog zur Übersicht ab.
5. Der nächste Steueralarm ANL wird auf den Ersten des nächsten Monats weitergesetzt.

ANL stellt den Monat xx [xx=FRC(Datum) bei MDY-Format] fest, erhöht um 1 (bei Überschreiten von 12 werden 12 abgezogen, d.h. Jahresende überschritten), setzt den nächstfälligen ANL-Steueralarm und speichert die aktuellen Alarme mit Namen "al" im Extended Memory ab.

Dann wird Datenfile ALAMxx+1 geholt, in der Schleife von LBL 10 der erste Alarm in Stack und Alpha geholt, die Meldung in Register R04 bis R06 zwischengespeichert, dieser Alarm gelöscht, Flag 01 gesetzt falls weitere vorhanden und diese im Extended Memory mit Namen "bl" gespeichert. Sodann wird "al" zurückgeholt, der erste Alarm dazugeschrieben, alles als "al" zurückgespeichert und die Schleife so lange durchlaufen, bis File "bl" leer ist.

Die Namen für die Zwischenfiles wurden absichtlich mit kleinen Buchstaben gewählt um nicht bei zufällig existierenden Files solcher Namen Fehlermeldungen bzw. Programmabbruch zu erhalten.

Danach werden die Alarme in der Schleife LBL 00 auf ihre Fälligkeit geprüft (Anzeige UPDT für Update): Wird ein Alarm erst nach mehr als 35 Tagen (1 Monat) fällig oder liegt er bis zu 270 Tage zurück, dann wird er gelöscht. Liegt er mehr als 270 Tage zurück, dann wird die Jahreszahl auf das aktuelle Jahr gesetzt. D.h. die Alarme müssen nicht jedes Jahr korrigiert werden.

Bei ANL ist die Funktion POP (Löschen einer Rücksprungadresse) in Benutzung, siehe auch mein Artikel in Prisma 6/89 Seite 34.

RSAL

Die Alarme werden nacheinander in Stack und Alpha geholt und für jeden einzeln abgefragt, ob **geändert werden** soll:

Bei Eingabe **N**=Nein weiter zum nächsten Alarm

Bei **e**=Ende zum Programmende

Bei **J**=Ja erscheint das Menü KORR? TDRM -. T für Zeit, D für Datum, R für Wiederholungszeit, M für Message, "↑"für Ende der Änderung und - für Löschen des Alarms. Bei den einzelnen Punkten wird der vorhandene Wert angezeigt, ohne Ziffern- oder Texteingabe bleibt dieser Wert erhalten. Wird bei Datumänderung kein Datumformat eingegeben, so wird das aktuelle Datum um die Anzahl der eingegebenen Tage verschoben.

PRAL

veranlaßt zuerst den Ausdruck der aktuellen Alarme, anschließend den Ausdruck sämtlicher Alarmfiles ALAMxx aus der Rambox, dabei wird die Steuerziffer xx aus R00 an den Namen angehängt. Am Ende werden die aktuellen Alarme wieder aus dem Extended Memory restauriert.

Registerbelegung bei KAL-ANL-RSAL-PRAL:

		Programm			
Reg.		KAL	ANL	RSAL	PRAL
00	21	21 / .016	.016	.01202	Size=12 bei
01	E1	E1 / T	Time	E1	allen Vieren.
02	al	bl / D	Date	ALAKT	Nach Aufruf von
03	Name	al / R	Repeat		UP 20 (CU)
04		A-Text1	A-Text1		bleiben für RSAL
05		A-Text2	A-Text2		dann die
06		A-Text3	A-Text3		erforderlichen
07		IND	IND		8 Register
08					übrig
09					
10					
11					

MAL

wird z.B. am letzten Samstag (DOW=6) eines Monats aufgerufen und setzt einen Alarm für den letzten Samstag des nächsten Monats. Für Dienstags (DOW=2) wird er auf den ersten Dienstag des nächsten Monats gesetzt. In beiden Fällen fängt der Rechner zwei Tage vor dem Termin bereits an auf diesen hinzuweisen. (Zeile 459 bis 461)

W1

Programm zum Setzen des Steueralarms ↑↑W1 und des Weckalarms ↑↑W2.Der Steueralarm wird um 23.59 eines Tages aktiv und setzt sich selbst für den Folgetag. Aus dem Datenfile W7 (7 Register, eines für jeden Tag) holt er die Weckzeit für den Folgetag: Positiver Zahlenwert = Wecken, negativer Wert = nicht Wecken.

W2/WECK

Weckprogramm, kann ganz nach Geschmack angepaßt werden. Bei mir wird mit zwei verschiedenen Tonfolgen abwechselnd geweckt: T1 (alternativer Beep aus dem PPC-ROM) und normaler BEEP mit einigen Folgetönen.

Die gesamte Weckdauer - Anzahl der Tonsequenzen inclusive Pausen - wird über die Steuerziffer (Z. 31) festgelegt, bei mir maximal 5 Tonfolgen. Bei Drücken irgendeiner Taste (GETKEYX ungleich Null, Z. 65 und 66) wird der Alarm gestoppt und mit 3 Tönen quittiert. Wehe man dreht sich um und schläft weiter ! Solange keine Taste gedrückt wird erfolgt der Rücksprung an die Zeile 32 über die im Stack abgelegte Programmzeiger-Adresse (STO b / RCL b).

Tonfolge T1: Vorsilbe 159, Nachsilbe jeweils: 57, 57, 57, 89, 89, 89, 57, 89, 89, 57, 89, 89, 89. Zeile 69 ist ein NOP, hier Byte 240.

DW

Änderungsteil für Wecker-Datenfile W7. Fragt im Menü TG SMDIOFAX für welchen Tag geändert werden soll: Bei X werden nacheinander alle Weckzeiten angezeigt, S steht für Sonntag, I für Mittwoch, A für Samstag.

Wird ein einzelner Tag angewählt, dann kann die Alarmzeit geändert werden: Nach Anzeige der gespeicherten Weckzeit kann beim Menü WECK NJ oder Menü N-WECK JN umgeschaltet werden. Das Resultat wird im Prompt zur Kontrolle nocheinmal gezeigt und kann weiter, z.B. Ziffernmäßig verändert werden, R/S übernimmt den letzten Wert in den Speicher. Negative Zeit heißt kein Weckalarm für diesen Tag.

UMDAK

ist notwendig wenn Rambox Files geändert werden sollen.

Da immer nur das letzte File einer Page gelöscht werden kann, müssen die nach dem zu ändernden File stehenden auf eine andere Page verlagert werden.

Das Programm kann Daten-, Buffer- und Keyfiles verschieben, Abfrage im Menü "Typ? DAK".

Im Rechner vorhandene Alarme werden unter dem Namen ALAKT (Aktuelle Alarme) im Extended Memory zwischengelagert. Nach Umspeichern wird daran erinnert, gegebenenfalls ALAKT wieder zurückzuholen. UMDAK gibt eine entsprechende Fehlermeldung aus, wenn ein File mit dem eingegebenen Namen auf der Page nicht vorhanden oder nicht das Letzte ist. Zeilen 92 bis 94 erfragen die Page-Nummern in der üblichen ISG / DSE Form von.nach, z.B. 12.013.

Im Register a wird der mehrfach benutzte Wert E3 = 1000 zwischengespeichert. Bei Datenfiles weiß man die Größe normalerweise nicht (es sei denn, sie ist gleich im Filename mit angegeben, wie z.B. bei W7), in Zeile 14 wird man aufgefordert den Size so groß vorzugeben, daß das File auf alle Fälle Platz hat. In der Schleife bei LBL 01 wird er dann auf die Datenfilegröße reduziert, da das Laden mit der Funktion LDREG erfolgt (Zeile 134). In den Zeilen 49 bis 54 wird der reduzierte Wert angezeigt und kann, wenn erwünscht oder erforderlich, geändert werden.

UMLAM

ist ein Hilfsprogramm zum Kopieren aller ALAM Files auf eine andere Page.

Da es Page-Umschaltfunktionen der Rambox benutzt darf es nur im Hauptspeicher oder Extended Memory ablaufen, sonst Absturz ! Deshalb Schleife 10. Bereits im Extended Memory abgelegte aktuelle ALAKT Alarme werden durch die im Rechner vorhandenen überschrieben. Die Nr. der Zielpage wird in Register 01 abgelegt.

SUSAL

Synthetische Version: Die .END.-Adresse wird vorübergehend unterhalb von Register 0C0 geschoben, der Befehl ALMNOW ausgeführt und Reg. c wiederhergestellt. Bei Abschalten des Rechners werden die Alarme wieder aktiviert. Zeile 2 ist in Hex: F0 01 69 01 00 10. **Dieses Programm darf nicht im SST-Modus durchgetastet werden, Folge sonst Memory Lost!!**

MCODE-Version: In das interne Register des Time Moduls, das die Zeit für den nächstfälligen Alarm enthält wird Null geschrieben, das bedeutet Null Uhr am 1. Januar 1900. Eine frühere Zeit kann im HP-41 nicht gesetzt werden, damit kann auch kein Alarm aktiviert werden. Ebenfalls bei Abschalten des Rechners werden die Alarme wieder aktiv. Die Funktionen CLOCK und OFF aktivieren überfällige Alarme, ALMNOW aktiviert nur Kontrollalarme, CLX mit T+X aktivieren auch zukünftige Alarme.

WIZ/SOZ

(Winterzeit-Sommerzeit-Umschaltung) stellen die Zeit über einen Steueralarm ↑↑WIZ bzw. ↑↑SOZ um eine Stunde zurück bzw. vor.

Zusätzlich wird die Änderung zum R01 des Kalibrierdatenfiles CaL3 addiert, damit beim nächsten CAL-Aufruf nicht eine Stunde als Änderung auftaucht und damit zu völlig falscher AF-Berechnung führt.

Abschließend wird mit der Nachricht SO/WI-Zeit im Alpha Register und gesetztem Autostartmodus des CCD-Moduls abgeschaltet - beim nächsten Einschalten morgens erscheint die Meldung wenn kein Weckalarm fällig wurde.

Nach einem Weckalarm erinnert der gesetzte Autostartmodus mit seinem Piepston, daß etwas fällig war: Die Meldung steht noch im Alpha Register, da das Weckprogramm dieses nicht überschreibt oder löscht. Eine gesetzte Flag 11 würde bereits vom Alarm gelöscht und nicht mehr auf die Meldung hinweisen.

Der Autostart-Modus des CCD-Moduls kann nur gelöscht werden, wenn der Rücksprung in Zeile 172 im Programm-Modus manuell mit SST übersprungen wird.

BEN

Es besteht aus 2 Teilen: BEN setzt das eigentliche Alarmprogramm (Bei mir LBL genannt), wenn es nicht aktiv war und löscht es wieder, falls es aktiv war. Der bedingte Alarm wird auf 15 Minuten Wiederholzeit gesetzt, eventuelle Datumüberschreitung wird berücksichtigt, zum Schluß wird cL'd oder SeT angezeigt. Register a wird zum Zwischenspeichern des Intervalls 15 Minuten benutzt.

BEN zeigt jede Viertelstunde die Zeit an und gibt bei 15 Minuten nach der vollen Stunde einen Ton, bei 30 Minuten zwei und bei 45 Minuten drei kurze Töne. Zur vollen Stunde ertönt Big Ben und danach die Stundenschläge.

Die Töne sind bis auf einen synthetisch, Vorsilbe dez. 159, Nachsilbe für Big Ben 72, 70, 71, 21, 53, 87, 8 und 54, für Zwischentöne und Stundenschlag 72.

PG=

Hauptspeicherprogramm für die Page-Umschaltfunktionen PG, PG10, PG01, die nicht in der Rambox ablaufen dürfen (Absturz!), sowie Ansprungstelle für die Alarme ANL, BEN, W1 und WECK (W2). Bei manuellem Aufruf der Umschaltprogramme wird zur besseren Orientierung jeweils für Page 8 und 9 angezeigt, welche Ramboxhälfte aktiv ist.

Bei gesetzter Flag 11 sind die Umschaltprogramme als Unterprogramme benutzbar, dann wird die Orientierungsanzeige unterdrückt.

Nach Eintippen von LBL ANL und LBL W1 müssen die Ramboxaufrufe wegen der Namensgleichheit als XROM xx,yy eingegeben werden (oder zuerst die Aufrufe per XEQ und danach erst die Label).

MCODE-Funktionen

Hier sind einige sehr angenehme Funktionen aufgeführt. Ideen dazu stammen aus PPC-Heften, waren z.T. aber erst nach einiger Umbauerei mit Erfolg zum Laufen zu bringen. Die Adressen sind nur als Beispiel zu betrachten.

Ge GTO END

```
E0B0   C GO 0131   05C  E
E0B2   READ 13(c)  007  G
E0B3   C=0 M
E0B4   R= 3
E0B5   LD@R 3
E0B6   CLRF 10          setzt Rechner auf RAM
E0B7   SETF 13          macht dem Rechner vor daß ein Programm läuft, d.h. die Funktion läuft auch nach manuellem Aufruf ab.
E0B8   WRIT 12(b)       schreibt End-Adresse in Reg. b
E0B9   RTN
```

VA VIEW ALPHA

```
E24A   NC GO 0520   081  V
E24B                016  A
E24C   CLRF 8       Scrollt, kein Prompt
                    bei ARGOUT
E24D   NC XQ ARGOUT  Entry 2C10
                    Anzeige des Alpha
                    Registers
E24F   NC GO 0380   setzt Message Flag
```

SKN, SK2, SK3, SK4, SKIP N, 2, 3, 4 LINES

```
E251   B=A ALL          08E  N
E252   JNC +01 E253     00B  K
E253   JNC +02 E255     013  S
                    Zeilensprung-Vorga-
                    bewert wird aus dem
E254   READ 3(X)      X-Register geholt.
E255   NC XQ BCDBIN   Entry 02E3
                    X-Wert wird in BCD-
                    Binär umgewandelt
E257   JNC +12 E269
E258   AC P-Q         0B2  2
E259   JNC +01 E25A   00B  K
E25A   JNC +02 E25C   013  S
E25B   LDI
E25C   HEX 002   B  Zeilensprung-Vorga-
                    be für 2 Zeilen
E25D   JNC +0C E269
E25E   JNC +16 E274   0B3  3
E25F   JNC +01 E260   00B  K
E260   JNC +02 E262   013  S
E261   LDI
E262   HEX 003   C  Zeilensprung-Vorga-
                    be für 3 Zeilen
E263   JNC +06 E269
E264   ???           0B4  4
E265   JNC +01 E266   00B  K
E266   JNC +02 E268   013  S
E267   LDI
E268   HEX 004   D
                    Zeilensprung-Vorga-
                    be für 4 Zeilen
E269   C=C-1   S&X
E26A   JC +0B E275  Carry-Flag gesetzt >>
                    zum Ende
E26B   M=C
E26C   NC XQ GETPC    Entry 2950
                    Holt Programmzeiger
E26E   NC XQ SKPLIN  Entry 2AF9
                    Überspringt aktuelle
                    PRGM-Zeile
E270   NC XQ PUTPCX  Entry 232F
                    Bringt PRGM-Zeiger
                    auf neuen Stand
E272   C=M
E273   C=C-1   S&X
E274   JNC -09 E26B
E275   NC GO NFRPU   Entry 00F0
                    Rücksprung in Rech-
                    ner-Hauptbereich
```

Die Zeilen E26C bis E270 entsprechen der Funktion DOSKP (Entry 1631) des Systems, die nach Überspringen einer PRGM-Zeile über Funktion NFRPU zurückkehrt.

X+1, X-1 X-Register um 1 erhöhen bzw. erniedrigen

```
E49A   C GO 0A2C    0B1  1
                    (Nach Idee von Ross
                    Cooling)
E49B                02B  +
E49C   ???          018  X
E49D   CLRF 8       Flag 8 gelöscht bei
                    X+1
E49E   JNC +05 E4A3
                    Sprung zum Hauptteil
E49F   C XQ 0B2C    0B1  1
E4A0                02D  -
E4A1   ???          018  X
E4A2   SETF 8       Flag 8 gesetzt bei X-1
E4A3   READ 3(X)    Liest Wert im X-Regi-
                    ster
E4A4   A=0 ALL      Setzt CPU-Register A
                    zu Null
E4A5   A=A+1 MS     Setzt Nybble 13 auf 1
E4A6   ?A#C MS      Test auf Text
E4A7   NC GO ERRAD   Entry 14E2
                    bei Text: Fehleranzei-
                    ge "Alpha Data"
E4A9   RSHFA ALL    Nybble 1 um 1 Stelle
                    nach rechts rotiert,
E4AA   SETDEC       ergibt User-1
E4AB   ?FSET 8      überspringt nächste
                    Zeile bei X+1
E4AC   JNC +03 E4AF
E4AD   A=A-1 MS     macht User Code 1
                    zu -1
E4AF   NOP          beliebige Funktion um
                    Carry zurückzusetzen
E4B1   NC XQ AD2-10  Entry 1807
                    führt User Code Addi-
                    tion aus
E4B1   WRIT 3(X)    Ergebnis zurück in X-
                    Register
E4B2   RTN
```

SUSAL Suspend Alarms

```
E120   ?FSET 5      08C  L
E121   C GO 0400     001  A
E122                013  S
E123   C GO 0405     015  U
E124                013  S
E125   C=0  ALL
E126   NC XQ 50E2   Time Module Entry
                    für Schreiben/Lesen
                    der Zeit
E128   WRIT 2(Y)    Schreibt Null in Time-
                    Module Register für
                    nächsten
E129   RTN          Alarmtermin
```

```
 01♦LBL "KAL"
XEQ 21  "UMDAK >LST.FL"
PROMPT  4  "KORR JN"
PMTK  *  GTO IND X


 10♦LBL 21
.  X<>F  SIZE?  12  X>Y?
PSIZE   E1  STO 01  +
STO 00  TONE 9  CLST
RTN

 24♦LBL 20
16  *  STO L  13.1
PEEKB  LASTX  +  STO Z
256  MOD  POKEB  RCL Z
LASTX  /  RCL X  E  MOD
-  STO L  13.2  PEEKB
LASTX  +  POKEB  CLST
RTN

 51♦LBL 04
"al"  ASTO 02  RCL 01
B?  SF 00  FC? 00
GTO 09  SAVEB  CLA  CLB

 62♦LBL 09
SF 27  "FILE="  PMTA
CF 27  ASTO 03  UNPTCT
GTBUF  13  CLLSTFL  4
XEQ 20  ">RSAL >KAL"
GTO 15

 76♦LBL 08
4  CHS  XEQ 20
"AL.SICH: "  ARCL 03  VA
BEEP  CLA  ARCL 03
RCL 01  BUFLNG?  13
CRFLBUF  RCL 01  LDBUF
PTCT  CLB  CLA  FC? 00
GTO 09  ARCL 02  RCL 01
GETB  PURFL  CLST  +
STO 02

104♦LBL 09
X<> 03  ARCL X  CLX
"⊢ IN 13"

109♦LBL 15
TONE 9  VA  CLST  +  PSE
CLA  Ge

117♦LBL "ANL"
XEQ 21  "↑↑ANL"  VA  MDY
12  DATE  INT  -  X≠0?
LASTX  X<>Y  X≠0?  CLX
X<>Y  X+1  DATE  FRC  +
2  XYZALM  "al"  ASTO 03
RCL 01  SF 25  SAVEB
CLB  DATE  "ALAM"  ARCLI
DMY  X<>Y  SF 25  GTBUF
"bl"  ASTO 02  CLA
FC? 25  GTO 09

156♦LBL 10
SIGN  RCLALM  CLALMA
RCL 01  B?  SF 00
XEQ 18  ARCL 02  FC? 00
GTO 09  SAVEB  CLB

169♦LBL 09
CLA  ARCL 03  GETB  RDN
SF 01  FS? 25  XEQ 17
CLA  R↑  ARCL 03
FC?C 00  GTO 09  SAVEB
CLA  ARCL 02  GETB
PURFL  GTO 10

188♦LBL 09
SF 25  PURFL  "UPDT"  VA
12  STO 07  XEQ 00
ALMNOW  ALMCAT  CLRG
CLST  +  "ANL OK"
XROM "VW"

203♦LBL 00
.016  STO 00

206♦LBL 16
RCL 00  SF 25  RCLALM
FS?C 25  XEQ IND 07
ISG 00  GTO 16  RTN

215♦LBL 12
STO 01  X<> Z  STO 03
RCL d  DATE  RCL T
STO 02  DDAYS  X<0?  SK4
35  X<Y?  CLALMA  RTN
CLALMA  -270  X<Y?  RTN
RDN  365.25  /  LASTX
X<>Y  FIX 0  RND  ABS  *
RND  X<>Y  STO d  RCL 02
RCL Z  DATE+  RCL 03
X<>Y  RCL 01  XYZALM
POP  GTO 16
```

```
255♦LBL "RSAL"
XEQ 21

257♦LBL 19
13 STO 07 XEQ 00
GTO 15

262♦LBL 13
POP XEQ 18 ARCL 04
ARCL 05 ARCL 06
"├ JNe" PMTK RDN
GTO IND T

272♦LBL 02
ISG 00 GTO 16

275♦LBL 03
"END" GTO 15

278♦LBL 18
ASTO 04 ASHF ASTO 05
ASHF ASTO 06 CLA RTN


286♦LBL 01
RCL 00 RCLALM CLALMA
STO 01 RDN STO 02 RDN
STO 03

295♦LBL 11
3 "KOR? TORM↑-" PMTK
X>Y? GTO IND X CLA
SF IND X FS? 02 FIX 6
RCL IND X ARCL X
PROMPT FS? 02 FC? 22
GTO 09 SF 25 DOW
FS? 25 LASTX FS?C 25
GTO 09 DATE+ ENTER↑
ENTER↑

320♦LBL 09
FC?C 02 ENTER↑ FS?C 22
STO IND T FIX 3 .
X<>F CLX GTO 11


330♦LBL 04
ARCL 04 ARCL 05
ARCL 06 "├:" PMTA
ALENG X≠0? XEQ 18
GTO 11

340♦LBL 14
ISG 00 GTO 09 RCL 01
CLA ARCL 02 SF 25
GETB GTO 03

349♦LBL 09
RCL 00 "ALAM" ARCLI
PRA RCL 01 SF 25
GTBUF FS?C 25 GTO 07
"N. VORH." PRA CLA
GTO 14

363♦LBL 07
ADV E RCLALM CLALMA
ACA ADV CLA FIX 2
ATIME "├ " X<>Y
FIX 4 ADATE ACA ADV
CLA X<> Z ATIME ACA
ADV RCL 01 B? GTO 07
GTO 14

388♦LBL "PRAL"
CLA DATE ADATE PRA
.012 STO 00 E1
STO 01 SF 25 B?
FC?C 25 GTO 14 "ALAKT"
ASTO 02 SF 25 PURFL
SAVEB PRA GTO 07

408♦LBL 05
RCL 03 RCL 02 RCL 01


412♦LBL 17
CLA ARCL 04 ARCL 05
ARCL 06 XYZALM FS?C 01
RTN

420♦LBL 06
FIX 3 GTO 16

423♦LBL "MAL"
CF 01 2 DATE DOW
X=Y? SF 01 X=Y?
"SFC " 6 X=Y? "CCD "
DATE 28 DATE+ STO 00
FRC E2 * INT STO 01
RCL 00 7 DATE+ STO 02
FRC E2 * INT RCL 01
FS?C 01 SK3 X=Y?
XEQ 09 SK2 X≠Y?
XEQ 09 12 RCL 00
ADATE 2 CHS DATE+
19.1 XYZALM GTO 15

469♦LBL 09
RCL 02 STO 00 RTN END
```

```
01♦LBL "SUSAL"
02 "×i×♦0"
03 X<> [
04 X<> c
05 ALMNOW
06 X<> c
07 X<> [
08 CLA
09 END
                    PLNG ""
   29 BYTES
SUSAL:
02 F5 01 69 01 00 10
```

```
01♦LBL "BEN"
SF 25 "↑∠" CLALMA
FS? 25 "cL'd" FS?C 25
GTO 11 TIME 4 1/X
HMS STO a HMS+ RCL X
24 STO T MOD ST- Y
X<>Y R↑ / DATE X<>Y
DATE+ X<>Y RCL X FRC
RCL a MOD - RCL a R↑
RCL Z XYZALM "SeT"

37♦LBL 11
VA TONE 2 TONE 2 CLA
RTN

43♦LBL 02
ALMCAT CLOCK

46♦LBL "AL"
DMY CLK24 CF 22 TIME
"H.MS" PROMPT FC? 51
. FC?C 22 GTO 02 X=0?
GTO 11 HMS+ HR RCL X
24 STO T MOD ST- Y
HMS X<>Y R↑ /

70♦LBL 11
DATE X<>Y DATE+
"DT:dd " ADATE PROMPT
FC?C 22 GTO 11 SF 25
DOW FC? 25 DATE+
FC?C 25 GTO 11 RCL Z
LASTX

87♦LBL 11
"RPT" PROMPT FC?C 22
. X<> Z CLA PMTA
XYZALM CLST CLA RTN


99♦LBL "∠"
CLA TIME ATIME AVIEW
HR RND ENTER↑ FRC 4
* INT X≠0? GTO 11
SF 19 RDN 12 MOD
X=0? XEQ 03 TONE 2
TONE 0 TONE 1 TONE 1
TONE 3 TONE 7 TONE 8
TONE 4

127♦LBL 11
X<>Y

129♦LBL 04
FS? 19 PSE TONE 2
DSE Y GTO 04 RTN

136♦LBL 03
LASTX + END
```

```
01♦LBL "PG="
Ge PG<> GTO 00

05♦LBL "PGab"
PG01 GTO 00

08♦LBL "PGba"
PG10

10♦LBL 00
FS?C 11 RTN 8 PG? E
CHS AROT ATOX CLA 9
PG? R↑ AROT R↑
ENTER↑ ATOX CLA
"PG8=" X<>Y XTOA X<>Y
"├ 9=" XTOA TONE 7 VA
PSE CLX CLA Ge

40♦LBL "ANL"
SF 25 XROM "ANL" PG<>
XROM "ANL" Ge

46♦LBL "BEN"
FIX 2 65 XEQ "SS"

50♦LBL 01
CLST CLA OFF

54♦LBL "∠"
FIX 2 66 XEQ "SS"
GTO 01

59♦LBL "SS"
SF 10

61♦LBL "UP"
SF 25 XROM "OP" PG<>
SF 25 XROM "OP" PG<>
RTN

69♦LBL "W1"
SF 25 XROM "W1" PG<>
XROM "W1"

74♦LBL "WECK"
SF 25 XROM "W2" PG<>
XROM "W2" Ge END
                PLNG "PG="
   198 BYTES
```

```
01♦LBL "UMDAK"

02♦LBL 00
. X<>F CLST
"TYP? DAK" PMTK
SF IND X CLA PMTA
FC? 01 GTO 02 ASTO X
")=SIZE:" PROMPT STO Z
E3 STO a / CLA
ARCL Y RDN CLX R↑
```

```
 25↑LBL 01
SF 25  GTREGXY  FS? 25
GTO 02  X<> Z  DSE X
RCL L  /  SF 25  GTREGXY
FS? 25  GTO 02  X<> Z
RCL L  *  DSE X  PSIZE
X<> Z  GTO 01

 45↑LBL 02
SIZE?  ASTO Z  FC? 01
GTO 04  "SIZE "  ARCLI
"⊢ K.?"  RDN  CF 22
PROMPT  FC? 22  R↑
FS?C 22  PSIZE  FS? 01
GTO 06

 62↑LBL 04
FC? 02  GTO 05  SF 25
 E1  B?  FC?C 25  GTO 00
ASTO L  "ALAKT"  SAVEB
CLB  CLA  ARCL L

 76↑LBL 00
SF 25  GTBUF  RDN
FS?C 25  GTO 06
"NO BUF."  PROMPT
GTO 00

 85↑LBL 05
CLA  ARCL Z  GTKEY

 89↑LBL 06
"VON.NACH PG:"  TONE 9
PROMPT  CLA  SF 25
CLLSTFL  FS?C 25  GTO 03
"FL UNPT >R/S"  PROMPT
CLA  ARCL T  UNPTCT
SF 25  CLLSTFL  FS?C 25
GTO 03  "NOT LST FL"
PROMPT

109↑LBL 03
FRC  ST* a  X<> a  CLA
ARCL T  FS? 01  CRFLDTA
FC? 02  GTO 10  E1
BUFLNG?  X<>Y  X<> Z
CRFLBUF

124↑LBL 10
FC? 03  GTO 08  KEYAS?
X<>Y  CRFLKEY

130↑LBL 08
FS?C 01  LDREG  FC? 02
GTO 07  X<> Z  LDBUF
CLB  X<> Z

139↑LBL 07
FS?C 03  LDKEY  PTCT
TONE 9  "⊢:"  ARCLI  VA
PSE  "⊢:ALAKT?"  CLST
STO a  VA  Ge  END

 01↑LBL "UMLAM"

 02↑LBL 10
"!NUR HS/EFM!"  PROMPT
GTO 10  CF 00  E1
STO 02  BUFLNG?  X=0?
GTO 00  X<>Y  "ALAKT"
SF 25  SEEKPTA  FS?C 25
PURFL  SAVEB

 19↑LBL 00
"N. PG:"  PMTD  STO 01
2  MOD  X=0?  SF 00
1.012  STO 00

 29↑LBL 05
CLA  RCL 00  "ALAM"
ARCLI  RCL 02  SF 25
GETB  FS? 25  SK2  SF 25
GTBUF  FC?C 25  GTO 06
BUFLNG?  RCL 01  FS?C 00
PG<>  PG<>  CRFLBUF
X<> Z  LDBUF  CLB  PG<>

 53↑LBL 06
ISG 00  GTO 05  SF 25
"ALAKT"  GETB  CF 25
CLST  "⊢ OK"  TONE 8  VA
CLA  Ge  END

 01↑LBL "CAL"
.  X<>F  SIZE?  8  X>Y?
PSIZE  .007  CLRGX
CF 22  CLX  CF 23
"CaL3.FL?"  PMTA  SF 25
GTREG  ASTO 05  FC?C 25
SF 03  FS? 03  GTO 01
RCLAF  RCL 02  X=Y?
GTO 02  99.9  X<Y?
GTO 02  RDN  SETAF
"R:02≠AF"  GTO "VW"

 33↑LBL 01
"LAST DATE?"  PROMPT
FC? 22  GTO 03

 38↑LBL 02
FC? 03  RCL 00  DATE
STO 06  DDAYS  24  *
FC? 03  RCL 01  FS? 03
"LAST TIME?"  FS?C 03
PROMPT  HR  -  TIME
STO 07  HR  +  STO 04

 59↑LBL 03
CF 22  "+"  61  SF 25
PASN  "-"  51  FS?C 25
PASN  GTO 05

 70↑LBL "-"
CHS

 72↑LBL "+"
FC? 22  GTO 04  *  50  *

 78↑LBL 04
T+X  ST+ 03

 81↑LBL 05
2 E-6  CLOCK  FC? 22
GTO 06  RCL 03  HR
RCL 04  /  10240  *
RCLAF  STO 02  X≠0?  1/X
+  1/X  RCL d  X<>Y
FIX 1  RND  SETAF

103↑LBL 06
CLA  51  SF 25  PASN  61
FS?C 25  PASN  FC?C 22
GTO 09  "AF:"  ARCL 02
"⊢ TO "  ARCL Z  R↑
STO d  AVIEW  RCL T
STO 02  PSE  RCL 00
RCL 06  STO 00  RDN
STO 06  RCL 01  RCL 07
STO 01  RDN  STO 07

133↑LBL 15
CLA  ARCL 05  UNPTCT
.002  LDREGX  PTCT

140↑LBL 09
CLX  Ge

143↑LBL "WIZ"
CLX  SIGN  CHS  SK3

148↑LBL "SOZ"
CLX  SIGN  T+X  "CaL3"
GTREG  ST+ 01  UNPTCT
.002  LDREGX  CLRGX
PTCT  RDN  X<0?  "WI"
X>0?  "SO"  "⊢-ZEIT"

166↑LBL "VW"
SAS  OFF

169↑LBL 07
VA  STOP  GTO 07  CAS
Ge  END

 01↑LBL "PG"
Ge  PG<>  GTO 00

 05↑LBL "PGab"
PG01  GTO 00

 08↑LBL "PGba"
PG10

 10↑LBL 00
FS?C 11  RTN  8  PG?  E
CHS  AROT  ATOX  CLA  9
PG?  R↑  AROT  R↑
ENTER↑  ATOX  CLA
"PG8="  X<>Y  XTOA  X<>Y
"⊢ 9="  XTOA  TONE 7  VA
PSE  CLX  CLA  Ge

 40↑LBL "ANL"
SF 25  XROM "ANL"  PG<>
XROM "ANL"  Ge

 46↑LBL "BEN"
FIX 2  65  XEQ "SS"

 50↑LBL 01
CLST  CLA  OFF

 54↑LBL "∠"
FIX 2  66  XEQ "SS"
GTO 01

 59↑LBL "SS"
SF 10

 61↑LBL "UP"
SF 25  XROM "OP"  PG<>
SF 25  XROM "OP"  PG<>
RTN

 69↑LBL "W1"
SF 25  XROM "W1"  PG<>
XROM "W1"

 74↑LBL "WECK"
SF 25  XROM "W2"  PG<>
XROM "W2"  Ge  END

 01↑LBL "W1"
SIZE?  8  X>Y?  PSIZE
"W7"  DATE  E  DATE+
STO L  DOW  .  GTREGXY
RCL L  RCL 00  X<0?
GTO 01  "↑↑WECK"  XYZALM

 20↑LBL 01
"↑↑W1"  CLST  LASTX
23.59  XYZALM  CLST
STO 00  OFF  Ge

 30↑LBL "W2"
1.005  RCL b  X<>Y
STO Z  2  MOD  INT  X=0?
GTO 04  TONE 7  TONE 7
TONE 7  TONE 9  TONE 9
TONE 9  TONE 7  TONE 9
TONE 9  TONE 7  TONE 9
TONE 9  TONE 9  FS? 47
GTO 03  GTO 05

 56↑LBL 04
BEEP  TONE 7  TONE 9
TONE 8  TONE 7
```

Fortsetzung Seite 70

Innereien des HP48SX

Wer möchte nicht gerne wissen, wie denn das schmucke Kästchen, daß er sich gerade für mehrere hundert DM gekauft hat, intern aufgebaut ist? Im Handbuch finden sich natürlich nur Angaben über die "normale" Bedienung unseres Rechners, HP läßt sich nicht so gerne in die Karten sehen, auch wenn dies für den Anwender sehr nützlich sein kann.

Hewlett Packard vertritt aber die Devise, daß seine Rechner idiotensicher zu handhaben sein sollen. Es sollen also keine Fehler möglich sein, die zu Datenverlusten führen können. Dazu darf man natürlich nicht mit den Internas eines Rechners herumspielen dürfen, da hier ein Fehler nicht mehr abgefangen werden kann. Die Nutzung der in diesem Artikel vorkommenden Informationen geschieht also absolut auf eigene Gefahr, ein Backup der im Rechner befindlichen Daten empfiehlt sich also dringend, bevor man irgendwelche Experimente macht!!

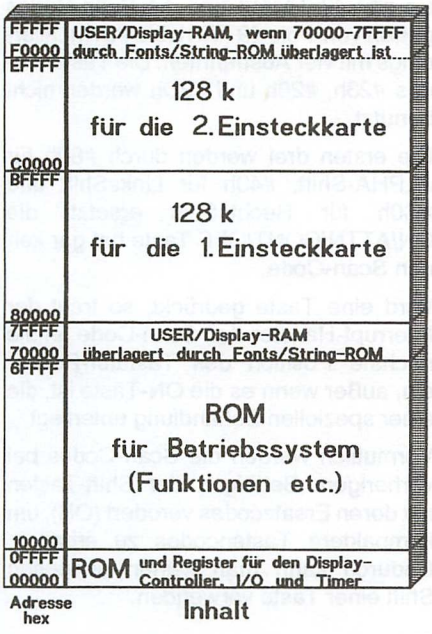


Bild 1: Speicherbelegung HP48

Alle Angaben in diesem Artikel beziehen sich übrigens auf die **Betriebssystemversion A** von 1989. Dies gilt vor allem für die Einsprungadressen innerhalb des Betriebssystems! In Bild 1 ist die Verteilung des 1MByte großen internen Adressraums zu sehen, zu der Besonderheit der Überlagerung kommen wir noch.

Speicher Scan-Modus

Im Betriebssystem des HP48SX ist ein kleiner Trick eingebaut, mit dessen Hilfe es vermutlich möglich sein sollte Programme in Steckmodulen direkt zu starten. Man kann damit auch bequem Routinen im Speicher ausführen oder den Speicher selbst verändern. Ebenso ist es damit einfach möglich Speicherbereiche über die serielle Schnittstelle zu einem anderen Rechner zu übertragen.

Immer die Taste direkt und ohne SHIFT drücken!!

Der Aufruf des Speicher Scan-Modus erfolgt folgendermaßen:
[ON] [D] gleichzeitig drücken, danach [⇐] (Taste Backspace). Jetzt befindet man sich im gewünschten Modus, in dem die Tasten der Tabelle ihre Funktion erhalten.

Taste	Aktion
[⇐]	Refresh
[↑]	inkrement um 1000h
[↓]	dekrement um 1000h
[/]	dekrement um 100h
[*]	inkrement um 100h
[-]	dekrement um 1h
[+]	inkrement um 1h
[ENTER]	goto Adresse 00100h (wahrscheinlich der Display-Kontroller)
[±]	goto Adresse F000Ah
[1/x]	goto Adresse F0A8Ch oder F1210 (Display)
[EEX]	goto Adresse 80000h (Steckmodul 1)
[DEL]	goto Adresse C0000h (Steckmodul 2)
[0]-[F]	gibt entsprechende Hex-Ziffer (Nibble), inkrementiert um 1, ein
[.]	überträgt 16 Nibbles zur Infrarotschnittstelle oder zur seriellen Schnittstelle, wobei auch um 10 inkrementiert wird
[SPC]	kopiert 1000h Speicherstellen auf die serielle Schnittstelle (9600 Baud)
[EVAL]	startet ein Programm an der angegebenen Adresse.

Drückt man das erste Mal die Taste [⇐], um in den Speicher Scan-Modus zu gelangen, so befindet man sich auf Adresse 705D9h. Drückt man an dieser Stelle [EVAL], so wird die Betriebssystemversion angezeigt:
Version HP-48A Copyright HP 1989
Das Bemerkenswerteste an diesem Speicher Scan-Modus ist vor allem, daß die Speicherverteilung anders als im Normalbetrieb ist.

Die 32k USER/Display-RAM sind von 70000h nach F0000h verschoben, so daß man das ROM sehen kann, das normalerweise verborgen ist. Während des normalen Rechnerbetriebes wechselt der Adressraum ab 70000h zwischen ROM und Display-RAM. Das hier untergebrachte ROM enthält Programmcode und Daten für I/O-Operationen, d.h. Strings und Fontinformationen (Rastergrafiken), ebenso befinden sich hier Diagnosefunktionen wie der Selbsttest und unser gerade betrachteter Speicher Scan-Modus.

Will man auf dieses versteckte ROM während der normalen Arbeit zugreifen, so muß man etwas zaubern, um die Adressierung umzuschalten. Dies muß in der Zeit geschehen, in der nicht auf das Display zugegriffen wird, d.h. zwischen jedem Refresh desselben.

F0A8Ch und F1210h sind die Adressen der zweiten Zeile des Displays, abhängig davon, ob man in der Stackanzeige oder in der Plotanzeige war, als man den Scan-Modus aufgerufen hat. Drückt man nun die Tasten [0]-[F] nach der Taste [1/x], dann kann man sofort den Effekt sehen, den das Schreiben eines Nibbles in das Display hat. Man kann auf diese Art und Weise natürlich auch auf dem Display zeichnen.

Die erste Variable im HOME-Verzeichnis ist so abgespeichert, daß ihre Adresse auf 7FFFAh endet, ganz im Gegensatz zum HP28, wo sie bei CFFFFh endete. Die Adresse 7FFFAh erscheint als FFFFAh im Scan-Modus.

Der sogenannte Tiefschlaf Modus ist durch Drücken der Tasten [ON] [SPC] zu erreichen. Drückt man weitere Tasten, so wird ein Großteil des Rechners neu initialisiert.

Ein bislang undokumentiertes Kommando ist WSLOG, das einen Block von Warmstarts auszulösen scheint. Dies scheint in dem verborgenen ROM abgespeichert zu sein.

Nun ein kleiner Ausflug in die Welt der Maschinensprache:

PEEK und POKE

Der bequemste Weg Maschinensprache einzugeben ist wie schon erwähnt der Speicher Scan-Modus. Man braucht keine Zeit mit Hex-Strings Umwandlern und anderen Werkzeugen zu vergeuden. Wir speichern nun das folgende PEEK Programm als erste Variable in das HOME-Verzeichnis:

```
« RCWS SWAP 64 STWS #0h OR
SWAP STWS
"ABCDEFGHIJKLMNOPQRSUVWXYZ"
»
HOME 'PEEK' DUP PURGE STO
```

Nun schalten wir in den Scan-Modus und suchen den Anfang des String-Objektes: [ON] [D], [⇐], [ENTER], [/], wir halten die Taste [-], bis wir FFFB3h sehen.

Nun ändern wir den Objekttyp (C2A20) in CCD20h. Zu beachten ist, daß die Länge des Objekts folgendermaßen errechnet wird:
93000h -> 39h = 57d = 26*2+5.

Display-Register für die verschiedenen Objekttypen des HP48SX:

Adr.	Typ	Name	Anzeige (Bsp.)	(event. Syntax)
02933	0	Real	3.14159265359	
02977	1	Complex	(1,2)	
02A2C	2	String	"hello" oder $C 5 hello	
029E8	3	Real Array	[1 2 3]	
029E8	4	Complex Array	[(1,2) (1,2)]	
02A74	5	List	{ DUP 3 '3/4' }	
02E48	6	Global Name	'PEEK'	
02E6D	7	Local Name	't'	
02D9D	8	Programm	« 440 .5 BEEP »	
02AB8	9	Algebraisches Objekt	SIN(X)	
02A4E	10	Binary Integer	# 454432h	
02B1E	11	Grafisches Objekt	Grafik 4*4	GROB 4 4 70607050
02AFC	12	angehängtes (tagged) Objekt	root1: 2.6	:root1: 2.6
02ADA	13	Einheiten Objekt	1_lyr	1_lyr
02E92	14	XLIB Name	XLIB 2 261	
02A96	15	Directory	DIR Avog 6.02E23 END	
02B40	16	Bibliothek (library)	Library	keiner
02B62	17	Backup Objekt	Backup Objekt	keiner
02E92	18	Funktion	SIN	
02E92	19	Kommando	SWAP	
02911	20	Adresse	FCh	keiner
02955	21	Long Real	Long Real	keiner
0299D	22	Long Complex	Long Complex	keiner
02A0A	23	verbundener (linked) Array	Linked Array	keiner
029BF	24	Zeichen	"X"	keiner
02DCC	25	Code	Code	keiner
02B88	26	Daten der Bibliothek	Bibl. Daten	keiner
other	27	External	External	keiner

Wir bewegen uns weiter zur Adresse FFFBDh, indem wir die Taste [+] benutzen, und tippen den Maschinencode ein. Achtung, nichts oberhalb der Adresse FFFF0h eingeben!!
13210314313019614613615671 30
16915471131321421648 08C0
Mit [ON] [C] gelangen wir zurück in den normalen Rechenbetrieb.

Am besten bilden wir jetzt die Prüfsumme über unser PEEK-Programm mit Hilfe des BYTES Kommando: Das Ergebnis sollte #8568h sein. Dies ist die binäre Prüfsumme, d.h. sie enthält das gesamte Objekt und ist gleichzeitig unabhängig vom Displaymodus.
Nun speichern wir das POKE-Programm im HOME Verzeichnis:

```
« SWAP OVER #0h AND OR SWAP
"123456789 123456789
123456789 123" »
HOME 'POKE' DUP PURGE STO
```

Wir schalten wieder in den Scan-Modus wechseln wie oben beschrieben den Typ unseres Objektes von String in den Code-Typ und geben den nun folgenden Maschinencode ein.

Der String ist wesentlich länger, so daß wir wesentlich weiter zurückgehen müssen als zuvor, um zum Anfang zu gelangen. Der Maschinencode endet trotzdem an derselben Stelle wie eben.
13210314317414717413416414613618513680D01641561E7130169
14213015411131321 42E72016480 8C0
Mit [ON] [C] gelangen wir zurück in den normalen Rechenbetrieb.

Die Prüfsumme sollte für POKE #725Dh betragen, ansonsten ist irgendwas bei der Eingabe schief gegangen.

Weitere interessante Adressen

Die Adresse des Stackendes, das normalerweise im CPU-Register D1 zu finden ist, wird in #70579h gesichert.

Um im Stack etwas auszuführen werden wir im Scan-Modus zu 70579h gehen:
[1/x], [/] 5 mal, [-] 19 mal
Die Adresse des Stackendes wird in umgekehrter Reihenfolge angezeigt. Wenn im Display
F0579 : 9F8E7309E7...
steht, dann beginnt der Stack bei 7E903 und endet bei 7E8F9. Es sind anscheinend zwei Objekte auf dem Stack. Wenn wir zu FE8F9 gehen, so sehen wir den Stack selbst sowie die beiden Adressen, die er enthält.

Jetzt kommt eine Liste mit interessanten Speicherstellen (die führende 7 wird im Scan-Modus durch F überlappt):

Adresse	Beschreibung
704EAh	Tasten-Puffer
70579h	Spitze (Ende) Stack
7057Eh	Boden (Start) Stack
70583h	lokale Variablen ??
70588h	interne Schleife ??
7058Dh	Menü-Tasten ??
70592h	HOME Verzeichnis
70597h	Ende des HOME Verzeichnisses (zeigt auf fünf 0 Nibbles, wahrscheinlich für ATTACH)
7059Ch	aktives Verzeichnis
70713h	grafisches Objekt zum Zusammensetzen der Stackzeilen: Es ist immer genug Speicher reserviert, um eine 19 Zeichen umfassende Grafik unterbringen zu können.
70844h	grafisches Objekt für die Menüanzeige: 8 bis 131
70968h	grafisches Objekt für den Rest des Bildschirms: 56 bis 131
710ECh	grafisches Objekt für die Plotanzeige: variable Größe

Die fraglichen Werte sind von der Speicherverteilung HP28 übernommen, wie sie Dave Kaffine veröffentlicht hatte. Da sind einige Gemeinsamkeiten zwischen den beiden Rechnern. Bei der Benutzung von PEEK muß man natürlich die echten Adressen #7xxxxh verwenden.

Tastaturpuffer und Scan-Codes

Der Tastaturpuffer arbeitet genauso wie beim HP28. Der Scan-Code läuft seriell: #01h für A, #02h für B, . . ., #30h für [SPC] und #31h für [+].
Es gibt 49 Tasten und die Scan-Codes gehen von 1 bis 49 (#01h bis #31h), allerdings mit vier Ausnahmen: Die Tastencodes #23h, #28h und #2Dh werden nicht benutzt.

Die ersten drei werden durch #80h für ALPHA-Shift, #40h für LinksShift und #C0h für RechtsShift ersetzt, die ON/ATTN/CONT/OFF Taste hat gar keinen Scan-Code.

Wird eine Taste gedrückt, so trägt der Interrupt-Handler den Scan-Code an die nächste Position des Tastatur-Puffers ein, außer wenn es die ON-Taste ist, die einer speziellen Behandlung unterliegt.

Vermutlich werden die Scan-Codes bei vorherigem Betätigen der Shift-Tasten mit deren Ersatzcodes verodert (OR), um kompaktere Tastencodes zu erhalten. Dadurch kann man keinen doppelten Shift einer Taste verwenden.

Kommandos und ihre Adressen für SYSEVAL

In der folgenden Tabelle finden sich alle im HP48 vorkommenden Kommandos mit ihren XLIB-Nummern und ihrer jeweiligen Adresse im ROM. Jede Adresse ist der Anfang der RPL-Routine. Bemerkenswert ist die geringe Größe der Routinen.

Die Adressen werden absolut verwendet, dürfen sich also nicht ändern. Wenn man die Kommandos disassemblieren würde, dann könnte man wahrscheinlich den restlichen Teil der Routinen finden, die an anderer Stelle irgendwo im ROM liegen müssen.

Dem ersten Anschein nach müßte also das Programmieren mit diesen SYSE-

VAL-Adressen recht einfach sein, indem man diese auf einem externen Rechner verwendet und dann das Programm auf den HP48 herunterlädt. Das setzt natürlich die festen Einsprungadressen im ROM voraus.
Im Prinzip könnte man auf diese Art und Weise auch einfach Compiler aufbauen, die den Interpretationsaufwand bei der Programmausführung erübrigen, indem ein Programm gleich nur noch die Routinen im ROM selber aufruft. Ebenso ließen sich auf diese Weise Fehlerabfangroutinen umgehen, die dann eine weitere Leistungssteigerung ermöglichen würden, und schon sind wir nicht mehr weit von den großen Brüdern, den sogenannten Personalcomputern entfernt.

In der Tabelle wurden einige der Sonderzeichen als Text geschrieben, um diese über Mailboxen übertragen zu können.

Quelle: Alonzo Gariepy USA
Überarbeitung: MM

Addr	XLIB #	Name
1957B	000002	ASR
1959B	001002	RL
195BB	002002	RLB
195DB	003002	RR
195FB	004002	RRB
1961B	005002	SL
1963B	006002	SLB
1965B	007002	SR
1967B	008002	SRB
1969B	009002	R->B
196BB	00A002	B->R
196DB	00B002	CONVERT
1971B	00C002	UVAL
1974F	00D002	UNIT
19771	00E002	UBASE
197A5	00F002	UFACT
197F7	010002	TIME
19812	011002	DATE
1982D	012002	TICKS
19848	013002	WSLOG
19863	014002	ACKALL
1987E	015002	ACK
1989E	016002	->DATE
198BE	017002	->TIME
198DE	018002	CLKADJ
198FE	019002	STOALRM
19928	01A002	RCLALRM
19948	01B002	FNDALRM
19972	01C002	DELALRM
19992	01D002	TSTR
199B2	01E002	DDAYS
199D2	01F002	DATE+
1A105	020002	CRDIR
1A125	021002	PATH
1A140	022002	HOME
1A15B	023002	UPDIR
1A194	024002	VARS
1A1AF	025002	TVARS
1A1D9	026002	BYTES
1A2BC	027002	NEWOB
1A303	028002	KILL
1A31E	029002	OFF
1A339	02A002	DOERR
1A36D	02B002	ERR0
1A388	02C002	ERRN
1A3A3	02D002	ERRM
1A3BE	02E002	EVAL
1A3FE	02F002	IFTE
1A4C0	030002	IFT
1A52E	031002	SYSEVAL
1A584	032002	DISP
1A5A4	033002	FREEZE
1A5C4	034002	BEEP
1A5E4	035002	->NUM
1A604	036002	LASTARG
1A71F	037002	WAIT
1A858	038002	CLLCD
1A873	039002	KEY
1A8BB	03A002	CONT
1A8D8	03B002	=
1A995	03C002	NEG
1AA1F	03D002	ABS
1AA6E	03E002	CONJ

Addr	XLIB #	Name
1C3CF	089002	GRAD
1C3EA	08A002	FIX
1C41E	08B002	SCI
1C452	08C002	ENG
1C486	08D002	STD
1C4A1	08E002	FS?C
1C520	08F002	FC?C
1C559	090002	BIN
1C574	091002	DEC
1C58F	092002	HEX
1C5AA	093002	OCT
1C5C5	094002	STWS
1C5FE	095002	RCWS
1C619	096002	RCLF
1C67F	097002	STOF
1C78C	098002	->LIST
1C79E	099002	R->C
1C7CA	09A002	RE
1C819	09B002	IM
1C85C	09C002	SUB
1C8EA	09D002	REPL
1C95A	09E002	LIST->
1C98E	09F002	C->R
1C9B8	0A0002	SIZE
1CAB4	0A1002	POS
1CB0B	0A2002	->STR
1CB26	0A3002	STR->
1CB46	0A4002	NUM
1CB66	0A5002	CHR
1CB86	0A6002	TYPE
1CE28	0A7002	VTYPE
1CEE3	0A8002	EQ->
1CF7B	0A9002	OBJ->
1D009	0AA002	->ARRAY
1D092	0AB002	ARRY->
1D0DE	0AC002	RDM
1D186	0AD002	CON
1D2DC	0AE002	IDN
1D392	0AF002	TRN
1D407	0B0002	PUT
1D5DF	0B1002	PUTI
1D7C6	0B2002	GET
1D8C7	0B3002	GETI
1DD06	0B4002	V->
1DE66	0B5002	->V2
1DEC2	0B6002	->V3
1E04A	0B7002	INDEP
1E07E	0B8002	PMIN
1E09E	0B9002	PMAX
1E0BE	0BA002	AXES
1E0E8	0BB002	CENTR
1E126	0BC002	RES
1E150	0BD002	*H
1E170	0BE002	*W
1E190	0BF002	DRAW
1E1AB	0C0002	AUTO
1E1C6	0C1002	DRAX
1E1E1	0C2002	SCALE
1E201	0C3002	PDIM
1E22B	0C4002	DEPND
1E25F	0C5002	ERASE
1E27A	0C6002	PX->C
1E29A	0C7002	C->PX

Addr	XLIB #	Name
1FC7F	116002	DUPN
1FC9A	117002	PICK
1FCB5	118002	ROLL
1FCD0	119002	ROLLD
1FCEB	11A002	CLEAR
1FD0B	11B002	STOsigma
1FD2B	11C002	CLsigma
1FD46	11D002	RCLsigma
1FD61	11E002	sigma+
1FD8B	11F002	sigma-
1FDA6	120002	Nsigma
1FDC1	121002	CORR
1FDDC	122002	COV
1FDF7	123002	sigmaX
1FE12	124002	sigmaY
1FE2D	125002	sigmaX^2
1FE48	126002	sigmaY^2
1FE63	127002	sigmaX*Y
1FE7E	128002	MAXsigma
1FE99	129002	MEAN
1FEB4	12A002	MINsigma
1FECF	12B002	SDEV
1FEEA	12C002	TOT
1FF05	12D002	VAR
1FF20	12E002	LR
1FF7A	12F002	PREDV
1FF9A	130002	PREDY
1FFBA	131002	PREDX
1FFDA	132002	XCOL
1FFFA	133002	YCOL
2001A	134002	UTPC
2003A	135002	UTPN
2005A	136002	UTPF
2007A	137002	UTPT
2009A	138002	COLsigma
200C4	139002	SCLsigma
200F3	13A002	sigmaLINE
2010E	13B003	BINS
20133	13C002	BARPLOT
20167	13D002	HISTPLT
2018C	13E002	SCTRPLT
201B1	13F002	LINFIT
201D6	140002	LOGFIT
201FB	141002	EXPFIT
20220	142002	PWRFIT
2025E	143002	BESTFIT
202CE	144002	SINV
2034D	145002	SNEG
203CC	146002	SCONJ
2044B	147002	STO+
20538	148002	STO-
2060C	149002	STO/
20753	14A002	STO*
208F4	14B002	INCR
209AA	14C002	DECR
20A15	14D002	COLCT
20A49	14E002	EXPAN
20A7D	14F002	RULES
20A93	150002	ISOL
20AB3	151002	QUAD
20AD3	152002	SHOW
20B20	153002	TAYLR
20B40	154002	RCL

```
1AABD  03F002 pi
1AADF  040002 MAXR
1AB01  041002 MINR
1AB23  042002 e
1AB45  043002 i
1AB67  044002 +
1ACD7  045002 +
1AD09  046002 -
1ADEE  047002 *
1AF05  048002 /
1B02D  049002 ^
1B185  04A002 XROOT
1B1C4  04B002 XROOT
1B278  04C002 INV
1B2DB  04D002 ARG
1B32A  04E002 sigmaN
1B374  04F002 sqrt
1B426  050002 SQ
1B4AC  051002 SIN
1B505  052002 COS
1B55E  053002 TAN
1B5B7  054002 SINH
1B606  055002 COSH
1B655  056002 TANH
1B6A4  057002 ASIN
1B72F  058002 ACOS
1B79C  059002 ATAN
1B7EB  05A002 ASINH
1B830  05B002 ACOSH
1B8A2  05C002 ATANH
1B905  05D002 EXP
1B94F  05E002 LN
1B9C6  05F002 LOG
1BA3D  060002 ALOG
1BA8C  061002 LNP1
1BAC2  062002 EXPM
1BB02  063002 !
1BB41  064002 FACT
1BB6D  065002 IP
1BBA3  066002 FP
1BBD9  067002 FLOOR
1BC0F  068002 CEIL
1BC45  069002 XPON
1BC71  06A002 MAX
1BCE3  06B002 MIN
1BD55  06C002 RND
1BDD1  06D002 TRNC
1BE4D  06E002 MOD
1BE9C  06F002 MANT
1BEC8  070002 D->R
1BEF4  071002 R->D
1BF1E  072002 ->HMS
1BF3E  073002 HMS->
1BF5E  074002 HMS+
1BF7E  075002 HMS-
1BF9E  076002 RNRM
1BFBE  077002 CNRM
1BFDE  078002 DET
1BFFE  079002 DOT
1C007  07D002 %T
1C01E  07A002 CROSS
1C03E  07B002 RSD
1C060  07C002 %
1C149  07E002 %CH
1C1B9  07F002 RAND
1C1D4  080002 RDZ
1C1F6  081002 COMB
1C236  082002 PERM
1C274  083002 SF
1C2D5  084002 CF
1C313  085002 FS?
1C360  086002 FC?
1C399  087002 DEG
1C3B4  088002 RAD
1E2BA  0C8002 GRAPH
1E2D5  0C9002 LABEL
1E2F0  0CA002 PVIEW
1E31A  0CB002 PIXON
1E344  0CC002 PIXOFF
1E36E  0CD002 PIX?
1E398  0CE002 LINE
1E3C2  0CF002 TLINE
1E3EC  0D0002 BOX
1E416  0D1002 BLANK
1E436  0D2002 PICT
1E456  0D3002 GOR
1E4E4  0D4002 GXOR
1E572  0D5002 LCD->
1E58D  0D6002 ->LCD
1E5AD  0D7002 ->GROB
1E5D2  0D8002 ARC
1E606  0D9002 TEXT
1E621  0DA002 XRNG
1E641  0DB002 YRNG
1E661  0DC002 FUNCTN
1E681  0DD002 CONIC
1E6A1  0DE002 POLAR
1E6C1  0DF002 PARMTRC
1E6E1  0E0002 TRUTH
1E701  0E1002 SCATTER
1E721  0E2002 HISTGRM
1E741  0E3002 BAR
1E761  0E4002 SAME
1E783  0E5002 AND
1E809  0E6002 OR
1E88F  0E7002 NOT
1E8F6  0E8002 XOR
1E972  0E9002 ==
1EA9D  0EA002 !=
1EBBE  0EB002 <
1EC5D  0EC002 >
1ECFC  0ED002 <=
1ED9B  0EE002 >=
1EE38  0EF002 OLDPRT
1EE53  0F0002 PR1
1EE6E  0F1002 PRSTC
1EE89  0F2002 PRST
1EEA4  0F3002 CR
1EEBF  0F4002 PRVAR
1EF43  0F5002 DELAY
1EF63  0F6002 PRLCD
1EF7E  0F7002 delta
1EFCC  0F8002 delta
1F133  0F9002 RCEQ
1F14E  0FA002 STEQ
1F16E  0FB002 ROOT
1F1D4  0FC002 integral
1F21D  0FD002 integral
1F2C9  0FE002 sigma
1F354  0FF002 ∫
1F3ED  100002 ∫
1F500  101002 QUOTE
1F55D  102002 APPLY
1F9C4  107002 ->Q
1F9E9  108002 ->Qpi
1FA59  109002 ^MATCH
1FA8D  10A002 vMATCH
1FAEB  10B002 _
1FB5D  10C002 RATIO
1FB87  10D002 DUP
1FBA2  10E002 DUP2
1FBBD  10F002 SWAP
1FBD8  110002 DROP
1FBF3  111002 DROP2
1FC0E  112002 ROT
1FC29  113002 OVER
1FC44  114002 DEPTH
1FC64  115002 DROPN
20CCD  155002 STO
20D65  156002 DEFINE
20EFE  157002 PURGE
20FAA  158002 MEM
20FD9  159002 ORDER
210FC  15A002 CLVAR
2115D  15B002 TMENU
21196  15C002 MENU
211E1  15D002 RCLMENU
211FC  15E002 PVARS
2123A  15F002 PGDIR
2125A  160002 ARCHIVE
2133C  161002 RESTORE
2137F  162003 MERGE
213D1  163002 FREE
2142D  164002 LIBS
21448  165002 ATTACH
2147C  166002 DETACH
21E75  167002 XMIT
21E95  168002 SRECV
21EB5  169002 OPENIO
21ED5  16A002 CLOSEIO
21EF0  16B002 SEND
21F24  16C002 KGET
21F62  16D002 RECN
21F96  16E002 RECV
21FB6  16F002 FINISH
21FD1  170002 SERVER
21FEC  171002 CKSM
2200C  172002 BAUD
2202C  173002 PARITY
2204C  174002 TRANSIO
2206C  175002 KERRM
22087  176002 BUFLEN
220A2  177002 STIME
220C2  178002 SBRK
220DD  179002 PKT
224CA  17A002 INPUT
224F4  17B002 ASN
22514  17C002 STOKEYS
22548  17D002 DELKEYS
22586  17E002 RCLKEYS
225BE  17F002 ->TAG
22633  180002 DTAG
22EC3  000700 IF
22EFA  001700 THEN
22FB5  002700 ELSE
22FD5  003700 END
22FEB  004700 ->
23033  005700 WHILE
2305D  006700 REPEAT
230C3  007700 DO
230ED  008700 UNTIL
23103  009700 START
231A0  00A700 FOR
2324C  00B700 NEXT
23380  00C700 STEP
233DF  00D700 IFERR
23472  00E700 HALT
2349C  00F700
234C1  010700 ->
235FE  011700 >>
2361E  012700 <<
23639  013700 >>
23654  014700 '
23679  015700 '
23694  016700 END
236B9  017700 END
2371F  018700 THEN
2378D  019700 CASE
237A8  01A700 THEN
23813  01B700 DIR
23824  01C700 PROMPT
```

Noch einmal Ephemeriden

von Dr. Heilmann

Diese Version für den Neuen entspricht in den wesentlichen Teilen dem Programm des Autors für den HP41 (PRISMA 89.06.27). Die Ausführungszeit beträgt nur noch ein Fünftel der Zeit, die der 41er braucht. Die Darstellung ist viel übersichtlicher. Die Bedienung des Programmes ergibt sich aus der Auflistung.

PROGRAMM: Ephemeriden Sonne, Mond und kleine Planeten (Merkur, Venus, Mars)

Directory EPH: Routinen

Page 1:	IPL	IPS	SZ	LS	LP	E-A
Page 2:	PRZ	A-H	P-R	C-S	PH	KP
Page 3:	SKP	FKP	RM	HS	SH	MM
Page 4:	TOG	DA	DQ	KC	B	L
Page 5:	Q					

B, L, Q: Variable.

Subdirectory IPL Hauptprogramme.

Page 1:	STZ	SON	MON	MER	VEN	MAR
Page 2:	IN	DT	PZ	AZ	S	T
Page 3:	E	M				

S, T, E, M: Variable.

```
# 16620d
  5056.0
```

I P S

Hauptprogramm Sonne und Planeten

```
« SZ LS →V3
  CASE 7 FS?
    THEN .387099
.205614 .00002
102.2794
149472.5153 28.7537
.3705 7.0029 .0019
47.146 1.185
     END 8 FS?
    THEN .723332
.006821 -.000048
212.6032 58517.8039
54.3838 .508 3.3936
.001 75.78 .9
    END 9 FS?
    THEN 1.52392
.093313 .000092
319.5294 19139.8585
285.4322 1.0698
1.8503 -.0007
48.786 .771
    END
  END
  IF 6 FC?
  THEN LP DUP ROT +
DUP
  END V→
  IF 3 4 FC? SWAP
FS? AND
  THEN PRZ
  END E→A
  IF 4 FS?
  THEN DA
   ELSE DQ SWAP 'r'
→TAG →STR +
  END
  CASE 6 FS?
    THEN "Sonne"
    END 7 FS?C
     THEN "Merkur"
    END 8 FS?C
     THEN "Venus"
    END 9 FS?C
     THEN "Mars"
    END
  END "
" + SWAP +
  IF  6 FC?C
   THEN "
" + SWAP
ROT PH
  END KC
»
# 22427d
   702.0
```

Bahnelemente Merkur

Bahnelemente Venus

Bahnelemente Mars

Berechnung des geozentrischen Planetenortes, ekliptisch

Ort von Sonne und Planeten vektoriell in Polarkoordinaten

Präzession für äquatoriale Ortsangabe Transformation ekliptisch-äquatorial

S Z

Errechnet für die gewählte Epoche T (Anzahl bis zur Epoche vergangenen julianischen Jahrhunderte vom 31.12.1899 12 Uhr UT angezählt) und S, die lokale Sternzeit

```
«
  IF 1 FS?
  THEN "Datum, Zeit:" {
":D:
:Z:
" { 1 0 }
V } INPUT OBJ→
   ELSE DATE TIME
  END HMS→
- IF 2 FS?
   THEN
"Breite, Länge:" Äquin."
{
":B:50.1839
:L:-7.5119
:Q:1950"
{ 1 0 } V } INPUT
OBJ→ 1900 -
365.24219878 *
.31352 + 36525 /
```

Vorgaben, geographische Koordinaten des Hauses des Autors, das Äquinoktium 1950

```
'Q' STO HMS→ 15 /
'L' STO HMS→ 'B'
STO 2 CF
  END DUP 1.0027379
* SWAP 1.0119 4
ROLL DDAYS .5 + DUP
ROT 24 / + 36525
SWAP OVER / 'T' STO
/ DUP .0929 *
8640184.542 + *
23925.836 + SH +
'L' RCL - 15 * MM
'S' STO
»
# 34954d
   515.5
```

L S

Errechnet die geozentrisch-ekliptischen Koordinaten der Sonne

```
« 'T' RCL → t
  « 1 .016751 t
.000042 * -
358.4758 35999.0498
t * + KP 281.2208
1.7192 t * + + 90
  »
»
# 54683d
   153.0
```

L P

Errechnet die geozentrisch-ekliptischen Koordinaten der Planeten aus den Bahnelementen in I P S

```
« 'T' RCL → t
  « t * + SWAP
ROT + ROT t * 4
ROLL + 8 ROLL 7
ROLL t * 8 ROLL + 6
ROLL t * 7 ROLL +
  » KP ROT + P→R
ROT P→R C→S SWAP 4
ROLL + SWAP →V3
»
# 26683d
   173.0
```

E → A

Transformation ekliptisch nach äquatorial

```
« 23.452295 'T' RCL
.013004 * - 4 ROLL
ROT P→R ROLL P→R
4 PICK P→R 4 ROLL 5
ROLL P→R NEG 4 ROLL
+ SWAP ROT + C→S
»
# 48409d
   133.0
```

P R Z

Präzession, ekliptisch, nur für äquatoriale Koordinaten

```
« 'T' RCL DUP 'Q'
RCL SWAP - SWAP 1 -
→ t u
  « t DUP .04 * u
.5 * 869.81 + - * u
DUP .61 * 3289.48 +
* 629554.98 + + SH
t DUP -.33 * u
-.067 * 47.003 + +
* SH t DUP 1.111 *
u 2.222 * 5029.097
+ + * SH
  » 6 ROLL 5 ROLL
P→R 6 ROLL 6 PICK -
P→R 5 PICK P→R 4
ROLL 6 ROLL P→R
SWAP ROT - SWAP ROT
+ SWAP C→S SWAP 5
ROLL + 4 ROLL +
SWAP
»
# 8129d
   414.5
```

A → H

Transformation äquatorial nach azimuthal, Azimuth von N über O 360 Grad

```
« 4 ROLL ROT P→R 4
ROLL P→R NEG SWAP 4
PICK P→R NEG 4 ROLL
5 ROLL P→R SWAP ROT
+ SWAP ROT + ROT
SWAP C→S
»
# 28978d
   105.0
```

P → R

Verwandelt ebene Polarkoordenaten in ebene kartesische Koordinaten

```
« -15 CF -16 SF →V2
-16 CF V→
»
# 19170d
    61.5
```

C → S

Verwandelt räumliche kartesische Koordinate in räumliche Polarkoordinaten

```
« -15 CF -16 CF →V3
-15 SF -16 SF V→
»
# 54599d
    74.5
```

P H

Hilfsprogramm Phasenberechnung

```
« DOT LASTARG ABS
SWAP ABS * / 1 + 2
/ 3 FIX 'k' →TAG
→STR +
»
# 24594d
    66.0
```

K P

Berechnung der wahren Anomalie aus der mittleren Anomalie für Kepler-Ellipse

```
« -3 SF RAD D→R
DUP + MOD DUP
  IF π >
  THEN 10 SF π  DUP
+ SWAP -
  END DUP 'M' STO
DUP ROT DUP 'E' STO
+ DUP
  IF π  >
   THEN DROP π
```

```
  END SWAP
  DO SKP DUP FKP 7
RND 0
  UNTIL ==
  END
  IF 10 FS?C
   THEN π DUP + SWAP

  END SWAP DROP R→D
DEG 1 'E' RCL DUP2
5 PICK COS * - 5
ROLL * 4 ROLLD +
LASTARG - /   SWAP
2 / TAN * ATAN DUP
+ MM
»
# 27735d
   307.0
```

S K P — Sekantenschnittpunkt

```
« DUP2 DUP2 SWAP -
ROT FKP ROT FKP / 1
- / +
»
# 890d
  58.0
```

F K P — Kepler-Gleichung

```
« DUP SIN 'E' RCL *
- 'M' RCL -
»
 # 19823d
    54.0
```

R M — Errechnet aus gegebenem T die topozentrisch-ekliptischen Koordinaten des Mondes

```
«  DUP DUP -.01133
* 481267.883142 + *
270.434164 + MM
OVER DUP .009192 *
477198.849108 + *
296.104608 + MM 3
PICK DUP .002078 *
1934.142008 - *
259.183275 + MM 4
PICK DUP .000303 *
36000.768925 + *
279.696678 + MM 5
ROLL DUP -.00015 *
35999.04975 + *
358.475833 + MM → a
b c d e
  « 6378.14 3423 b
DUP DUP + COS 10 *
SWAP COS 187 * + +
b a d - DUP + DUP
ROT DUP2 - SWAP ROT
+ COS 3 * ROT COS
28 * + SWAP COS 34
* + + SH SIN / b
DUP DUP + SWAP +
LASTARG e - LASTARG
+ LASTARG a d - DUP
DUP + 4 ROLL +
LASTARG - LASTARG
ROT SWAP - LASTARG
SWAP 7 ROLL -
LASTARG 6 ROLL SWAP
- LASTARG SWAP SIN
2370 * SWAP SIN
-668 HS 165 HS 4586
HS 206 HS 22640 HS
212 HS 192 HS -125
HS -110 HS 148 HS
769 HS 36 * + a c -
DUP + SIN 412 *
SWAP OVER - SH a +
DUP c - ROT e SIN
541 * + SH + SIN
18520 * d DUP + a c
+ - b LASTARG -
LASTARG DROP e +
LASTARG - LASTARG
DROP a c - b - DUP
b - 8 ROLL SWAP SIN
-25 HS 21 HS -526
HS 11 HS -23 HS -31
HS 44 * + SH
  »
»
# 14338d
  1125.5
```

H S — Häufige Befehlsfolge in Programmteil RM

```
« * + SWAP SIN
»
# 8347d
   26.5
```

S H — Verwandlung von Sekunden in dezimale Stunden.

```
« 3600 /
»
# 53312d
    29.5
```

M M — Reduktiom auf Winkelbereich 0...360 Grad.

```
« 360 MOD
»
# 25790d
    29.5
```

T O G — Toggles Flags.

```
« DUP
  IF FS?
  THEN  CF
  ELSE SF
  END
»
# 64191d
    37.5
```

D A — Erzeugt Ausgabe für azimuthale Darstellung (ohne Angabe des Abstandes von der Erde). Azimuth und Höhe in dezimalen Grad, k: beleuchtete Scheibe in Prozent, außer Sonne

```
« 1 FIX 'S' RCL ROT
- SWAP 'B' RCL A→H
90 SWAP - ROT DROP
```

```
SWAP MM 'a' →TAG
→STR "
" + SWAP 'h'
→TAG →STR +
»
# 15592d
   130.5
```

D Q

Erzeugt Ausgabe für äquatoriale Koordinaten. Ausgabe in ddd.mmss (Grad, Minuten Sekunden), k: beleuchtete Scheibe in Prozent, außer Sonne, r in astronomischen Einheiten

```
« 4 FIX 90 SWAP -
SWAP MM 15 / →HMS
'α' →TAG →STR "
" +
SWAP →HMS 'δ ' →TAG
→STR + "
" +
»
# 43947d
  114.0
```

K C

Display-Routine

```
« CLLCD 3 DISP 3
FREEZE
»
# 3046d
   29.0
```

S T Z

Errechnet die Sternzeit, Flag 1 gelöscht: momentane Epoche Flag 1 gesetzt : Eingegebene Epoche.

```
« SZ 'S' RCL 15 / →HMS
4 FIX →STR
"Sternzeit:" SWAP +
CLLCD 1 DISP 1
FREEZE
»
# 60579d
    86.0
```

S O N

Sonnenort

```
« 6 SF IPS
»
# 53119d
    29.0
```

M O N

Mondort, Flag 4 gelöscht: topozentrisch-äquatorial, Flag 4 gesetzt : topozentrisch azimuthal Mondentfernung r in km Präzessionskorrektur nicht vorgesehen

```
« SZ 'T' RCL RM 90
SWAP - →V3 DUP ABS
SWAP DUP V→ E-A →V3
6378.14 'S' RCL 90
'B' RCL - →V3 - V→
  IF 4 FS?
   THEN DA ROT DROP
   ELSE ROT DROP DQ
ROT 0 FIX 'r' →TAG
→STR +
  END "
" + SWAP
NEG DUP LS →V3
149597870 * SWAP
PH "Mond" "
" +
SWAP + KC
»
# 49118d
   281.0
```

M E R

Merkurort.

```
« 7 SF IPS
»
# 22105d
    29.0
```

V E N

Venusort.

```
« 8 IPS
»
# 45339d
     26.5
```

M A R

Marsort.

```
« 9 IPS
»
# 30356d
    26.5
```

I N

Setzt Flag zur Initiierung Wird nach der Initiierung gelöscht, Koordinaten es Beobachtungsortes bleiben erhalten

```
« 2 TOG d
»
# 53304d
    25.5
```

D T

Toggles mit Flag 1 die Eingabe von der momentanen Epoche verschiedener Epochen

```
« 1 TOG
»
# 30312d
    25.5
```

P Z

Toggles mit Flag 3 die Berechnung der Präzession. Voreinstellung Besseljahr 1950. Azimuthale Koordinaten werden stets nur für das momentane Äquinoktium errechnet.

```
« 3 TOG
»
# 6071d
   25.5
```

A Z

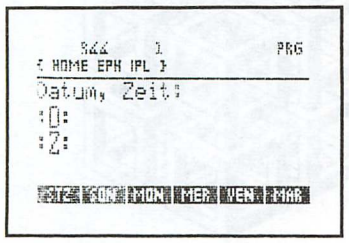
Toggles Flag 4 für die Darstellung des Ortes in azimuthalen Koordinaten

```
« 4 TOG
»
# 51358d
    25.5
```

Es empfiehlt sich, die Variablen vor der Eingabe der Programme zu erzeugen, um die Menus in der angegebenen Form zu sehen.

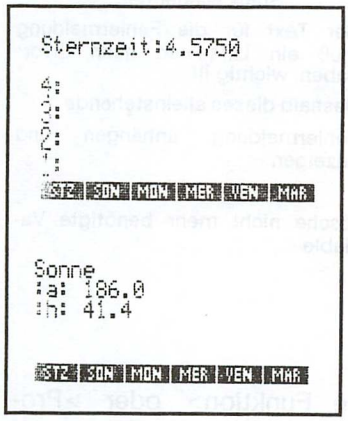
Inzwischen sind zwei spezielle Ephemeridenprogramme für HP41CX fertig, die die Positionen von Kometen und Kleinplaneten berechnen. Beim Kometenprogramm sind sehr ausführliche

Beispielausdrucke:

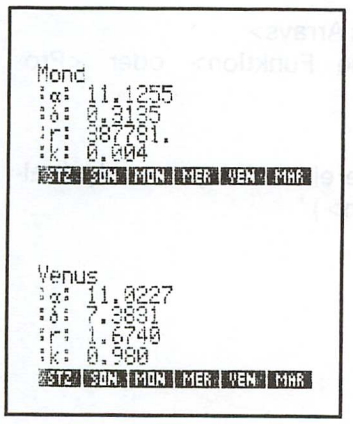


Aufforderung
Zeiteingabe
Flag 1 gesetzt
(Toggle: DT)

Aufforderung
Ortseingabe und Äquinoktium mit Voreinstellung (PRISMA 90.03.34),
Flag 2 gesetzt
(Toggle: IN)



Ausgabe: Azimutal (Flag 4)
(Toggle: AZ)



Ausgabe: Äquatorial
(Toggle: DT)

Untersuchungen zur Genauigkeit und Konvergenz des benutzten Verfahrens angestellt worden, die jede "Katastrophe" ausschließen.

Es ist vorgesehen diese Programme für den HP48SX umzuschreiben.

Interessenten werden gebeten, sich wegen der Programme direkt an den Autor zu wenden.

Dr. G. Heilmann
Obernhofer Straße 15
5408 Seelbach

PUT und GET

auf dem HP41

HP41C, 60 Zeilen, 115 Bytes, 17 Regs., SIZE n+20, CCD, (XF/M)

Mit PUT und GET möchte ich ein weiteres Programm zur kompakten Datensicherung vorstellen. Es legt jeweils sieben Zwei-Byte-Integerzahlen in zwei Registern ab. Der Aufruf der Befehle (Programme) ist ähnlich den Befehlen PEEKR und POKER des CCD-Moduls.

PUT legt den in Y liegenden Wert in dem in X angegebenen Zwei-Byte-Register ab.

GET liest das in X angegebene Register. Der Inhalt des ALPHA-Registers geht verloren. Bei PUT geht auch die Nummer des Registers verloren, bei GET wird sie durch den Inhalt des Registers ersetzt. Die Stackregister Y, Z und T bleiben unverändert. Der Wertebereich für Zwei-Byte-Integerzahlen ist 0-65535, geht aber wegen des Vorzeichenbits in diesen Programmen von -32767 bis +32768 (Zeilen 4, 5 und 56, 57).

Die absolute Registeradresse wird in den Zeilen 33-44 berechnet:

A=INT(reg/3.5)+492
B=2(reg MOD 3.5)
=> A+B

In den Zeilen 6-12 wird der zu speichernde Wert zerlegt:

1. Byte: INT(2/256)
2. Byte: n MOD 256

Die Routinen entstammen, leicht verändert, dem Dearing: 15-11, I/.

LBL 00: Kopiert die Inhalte von Y, Z und T nach M, N und O und berechnet die absoluten Registeradresse.

Zeile 40 (492) kann vom Anwender geändert werden. In der jetzigen Form werden die obersten 20 Register (492-511 absolut - siehe CCD-ROM Handbuch, Seite 1.07 ff) zur Speicherung der Zwei-Byte-Integerzahlen verwendet. Ein Register für 3.5 Pseudoregister => 0-69.

Zahlen können auch im X-Memory abgelegt werden. Ein Datenfile ist nicht nötig, aber durchaus von Vorteil. Hierbei sind die Hinweise auf den Seiten 7.18/19 des Handbuchs zu beachten. Vorsicht beim Löschen von Files !

Warnung: Bei einem Zugriff auf die Zwei-Byte-Register mit RCL, X<>, VIEW und anderen normalisierenden Funktionen können Daten u.U. verändert werden !

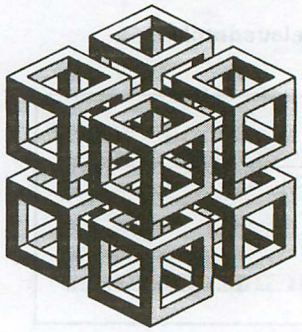
Quellen: Dearing, Tricks, Tips und Routinen . . . ; CCD-ROM Handbuch; Jerry Smith, Quick Reference Guide

```
01*LBL "PUT"
   XEQ 00
   X<>Y
   32767
05 +
   RCL X
   256
   MOD
   ST- Y
10 X<>Y
   LASTX
   /
   R^
   X<>Y
15 POKEB
   R^
   A+
   R^
   POKEB
20*LBL 01
   RCL M
   RCL N
   RCL O
   RTN
25*LBL 00
   R^
   STO M
   R^
   STO N
30 R^
   STO O
   R^
   RCL X
   3,5
35 MOD
   ST- Y
   X<>Y
   LASTX
   /
40 492
   +
   X<>Y
   ST+ X
   A+B
45 RTN
  *LBL "GET"
   XEQ 00
   PEEKB
   256
50 *
   X<>Y
   A+
   PEEKB
   RCL Z
55 +
   32767
   -
   XEQ 01
   R^
60 END
```

Werner Büsing
Herrenbreite 7
3340 Wolfenbüttel

Matritzenelemente verrechnen

mit dem HP28



MOP ist eine Funktion, die beliebige algebraische Operationen bzw. ein Programm auf jedes Element einer gegebenen Matrix oder eines Vektors auszuführen erlaubt, solange dies mathematisch möglich ist.
Konkret heißt dies z.B. ich will alle Zahlen einer Matrix mit 14,123 multiplizieren, um nur einen trivialen Fall zu benennen. Ich muß also nicht mehr erst jedes Element suchen, es normal eintippen, verrechnen (-multiplizieren), das Ergebnis aufschreiben und an der richtige Stelle im Array manuell (mit EDIT) wieder einsetzen.
Natürlich sind auch komplizierte Aktionen wie die Anwendung eines Programms auf ein Matrixelement möglich, ebenso wie eine mathematische Funktionskette wie der SIN(COS(TAN(Matrix bzw. Vektor))).
Nun aber zum Programmlisting mit Dokumentation:

```
MOP                          Länge = 365 Byte
« → a o «                    speichere Array und auszuführende Operation in lokalen Variablen
 DEPTH DUPN                  verdopple den Stack
 DEPTH 2 / →LIST             kopiere den Stack in eine Liste
 a IFERR RCL                 Wenn der Array ein Name ist (d.h. in einem Objekt gespeichert),
 SWAP OVER a 3 →LIST         dann hänge den Namen und den Inhalt des Objekts an die eben erzeugte List an.
 THEN + a a ROT END          Ansonsten hänge den Array selbst an die Liste.
 → b «                       Die Liste speichern: Dies ist ein Backup für den Fall eines Fehlers
 1 SWAP SIZE LIST→           Bestimmung der Größe des Arrays
 IF 1 – THEN * END           Wenn es ein 2-dimensionaler Array ist, dann multipliziere die Anzahl der Reihen und Spalten.
 IFERR                       Start der Fehlerroutine für verbotene Operationen.
  FOR i                      Schleife für die Ausführung der Operation auf jedem Element.
   IF a TYPE 6 = =           Wenn der Array ein Name ist (d.h. in einem Objekt gespeichert),
   THEN a END                dann bringe Array-Namen auf den Stack.
   i OVER i GET              hole nächstes Array-Element
   'X' STO                   speichere es
   o EVAL                    Operation auf nächstes Element anwenden
   IFERR                     \   Wenn das Resultat eine koplexe Zahl ist, und der
    PUT THEN                 |   Array nur Real-Zahlen (= = > error) enthält,
    ROT (1,0) *              |   dann wandle den Array in einen komplexen Array um
    ROT ROT PUT              /   und speichere ihn.
   END
  NEXT
 THEN                        Wenn eine verbotene Operation (wie INV(0)) auftritt,
  CLEAR                      dann lösche Data-Garbage, die der Fehler erzeugt hat.
  b LIST→ DROP               hole Backup wieder zurück
  IF a TYPE 6 = = THEN STO   Wenn der Array ein Name war, dann speichere ihn wieder zurück
  LIST→ DROP END             und stelle den alten Stack wieder her.
  "MOP Error:                Der Text für die Fehlermeldung muß ein Line-Feed nach Error: haben, wichtig !!!
"                            Deshalb dieses alleinstehende "
  ERRM + 1 DISP              Fehlermeldung anhängen und anzeigen
 END
 'X' PURGE                   lösche nicht mehr benötigte Variable
 » »
»
```

Anwendung:

```
1)  2:  <Array>
    1:  <algebraische Funktion> oder >Programm>
    MOP

2)  2:  <Name eines Arrays>
    1:  <algebraische Funktion> oder <Programm>
    MOP

3)  1:  'MOP(<Name eines Arrays>, <algebraische Funktion>) '
    EVAL
```

Beispiel:

```
2:  [ [  1  2.3  ]
      [ - 3  4.4
      [  1 - 1.1  ] ]
1:  'LOG(SQR(X))^-3 '
MOP

2:  [ [  1  2.3  ]
      [ - 3  4.4
      [  1 - 1.1  ] ]
1:  « IF X 1 < THEN X DUP R→C ELSE X END »
MOP

2:  'ΣDAT'          |   2:  'ΣDAT'
1:  'INV(X)'        |   1:  « X INV »
MOP                 |   MOP

1:  'MOP(MA,X^INV(3)) '
EVAL
```

42S: Trigonometrie

In PRISMA 90.3.25 habe ich für 28S und 48SX Programme zur Berechnung von Dreiecken aus drei beliebigen Stücken vorgestellt. Der 42S, die Arche Noah des 41er Betriebssystems, verfügt über eine Menuesteuerung mit benannten Softkeys, weshalb sich hier eine Überarbeitung der 28/48 Programme anbot. Wie man sieht, benötigt der 42S gerade 'mal ein Drittel des Speicherplatzes der "großen" Rechner, und zeitlich arbeitet er auch nicht langsamer. Die Bedienung ist ganz einfach: Mit XEQ "TRI" das Programm starten, die gegebenen Stücke (mindestens 3) eingeben, und das Programm mit R/S fortsetzen. Nach Programmende kann man sich die fehlenden Stücke mit RCL und der entspechenden Menuetaste holen. Falls sich aus den Eingaben kein Dreieck konstruieren läßt, bricht das Programm mit einer Fehlermeldung ab, falls sich sogar zwei Lösungen ergeben, weist der Rechner darauf hin, und berechnet die Lösung mit der größeren Fläche (= dem größeren Umfang).

Ralf Pfeifer (116)

```
00 { 267-Byte Prgm }
01▶LBL "TRI"
02 MVAR "A"
03 MVAR "B"
04 MVAR "C"
05 MVAR "∠A"
06 MVAR "∠B"
07 MVAR "∠C"
08 CLRG
09 XEQ 09
10 VARMENU "TRI"
11 STOP
12 CF 00
13 XEQ 09
14 8
15 STO 00
16▶LBL 00
17 RCL 04
18 X≠0?
19 GTO 05
20 RCL 05
21 X=0?
22 GTO 01
23 RCL 06
24 X=0?
25 GTO 01
26 +
27 -1
28 ACOS
29 X<>Y
30 -
31 GTO 05
32▶LBL 01
33 RCL 01
34 X=0?
35 GTO 05
36 X^2
37 RCL 02
38 X=0?
39 GTO 03
40 X^2
41 RCL 03
42 X=0?
43 GTO 02
44 X^2
45 +
46 X<>Y
47 -
48 2
49 RCL 02
50 x
51 RCL 03
52 x
53 ÷
54 ACOS
55 GTO 05
56▶LBL 02
57 LASTX
58 RCL 05
59 X≠0?
60 GTO 04
61▶LBL 03
62 RCL 03
63 X=0?
64 GTO 05
65 RCL 06
66 X=0?
67 GTO 05
68▶LBL 04
69 SIN
70 RCL 01
71 x
72 X<Y?
73 SF 00
74 X<>Y
75 RCL 01
76 X<Y?
77 CF 00
78 Rv
79 ÷
80 ASIN
81▶LBL 05
82 STO 04
83 X=0?
84 GTO 08
85 SIN
86 RCL 01
87 X≠0?
88 GTO 08
89 CLX
90 RCL 02
91 X=0?
92 GTO 06
93 RCL 05
94 X=0?
95 GTO 06
96 SIN
97 ÷
98 x
99 STO 01
100 GTO 08
101▶LBL 06
102 RCL 02
103 X=0?
104 GTO 08
105 X^2
106 RCL 03
107 X^2
108 +
109 2
110 RCL 02
111 x
112 RCL 03
113 x
114 RCL 04
115 COS
116 x
117 -
118 SQRT
119 STO 01
120▶LBL 08
121 RCL 01
122 X<> 02
123 X<> 03
124 STO 01
125 RCL 04
126 X<> 05
127 X<> 06
128 STO 04
129 DSE 00
130 GTO 00
131 SCI 08
132 RCL 01
133 RCL 02
134 +
135 RCL 06
136 2
137 ÷
138 SIN
139 x
140 RND
141 RCL 04
142 RCL 05
143 -
144 2
145 ÷
146 COS
147 RCL 03
148 x
149 RND
150 ALL
151 X=Y?
152 X=0?
153 SF 40
154 "2 Dreiecke"
155▶LBL 09
156 RCL 01
157 X<> "A"
158 X<> 02
159 X<> "B"
160 X<> 03
161 X<> "C"
162 STO 01
163 RCL 04
164 X<> "∠A"
165 X<> 05
166 X<> "∠B"
167 X<> 06
168 X<> "∠C"
169 STO 04
170 CLST
171 FS?C 00
172 AVIEW
173 END
```

Bei den algebraischen Operationen muß als Argument X angegeben werden und nicht das Feld selbst. Das ist sehr bedauerlich, aber "Calling by Reference" wie z.B. 'MOP(INV(<array>*3 - 2) ' ist leider nicht möglich, oder weiß jemand einen noch unbekannten neuen Weg?

Jetzt folgt das gleiche Programm MOP, nur benötigt diese Version weniger Speicherplatz, da sich hier das Feld direkt im Stack befinden muß:

MOP Länge = 360,5 Byte

```
« → a o «                    speichere Array und auszuführende Operation in lokalen Variablen
  DEPTH DUPN                 verdopple den Stack
  DEPTH 2 / →LIST            kopiere den Stack in eine Liste
  a +                        hänge den Array an die Liste an
  → b «                      Die Liste speichern: Dies ist ein Backup für den Fall eines Fehlers
    1 SWAP SIZE LIST→        Bestimmung der Größe des Arrays
    IF 1 – THEN * END        Wenn es ein 2-dimensionaler Array ist, dann multipliziere die Anzahl der Reihen und Spalten.
    IFERR                    Start der Fehlerroutine für verbotene Operationen.
      FOR i                  Schleife für die Ausführung der Operation auf jedem Element.
        IF a TYPE 6 = =      Wenn der Array ein Name ist (d.h. in einem Objekt gespeichert),
          THEN a END         dann bringe Array-Namen auf den Stack.
        i OVER i GET         hole nächstes Array-Element
        'X' STO              speichere es
        o EVAL               Operation auf nächstes Element anwenden
        IFERR                \   Wenn das Resultat eine
          PUT THEN           |   koplexe Zahl ist, und der
          ROT (1,0) *        |   Array nur Real-Zahlen (= = > error) enthält.
                             |   dann wandle den Array in einen komplexen Array um
          ROT ROT PUT        /   und speichere ihn.
        END
      NEXT
    THEN
      CLEAR                  Wenn eine verbotene Operation (wie INV(0)) auftritt,
      b LIST→ DROP           dann lösche Data-Garbage, die der Fehler erzeugt hat.
      "MOP Error:            hole Backup wieder zurück
                             Der Text für die Fehlermeldung muß ein Line-Feed nach Error: haben, wichtig !!!
      "                      Deshalb dieses alleinstehende "
      ERRM + 1 DISP          Fehlermeldung anhängen und anzeigen
    END
    'X' PURGE                lösche nicht mehr benötigte Variable
  » »
»
```

HP28 Usergroup ULM
Überarbeitung: MM (1000)

Zeiterfassung und Druckutilities

von Georg Hoppen

Da ich ohne Computer - und somit ohne externes Speichermedium auskommen muß (noch nicht einmal eine RAM-Karte nennt der Kerl sein eigen) - habe ich meinen HP48SX so "eingerichtet", daß im HOME-Directory Programme, Daten und ähnliches sind, die zentral für weitere Unterdirectories zur Verfügung stehen sollen. Es handelt sich in erster Linie um Druckersteuerungen, Tools etc. Aus diesem Grunde ist es unvermeidlich, selbst für einfache Prgrämmchen zusätzliche Erklärungen mitzugeben.

Natürlich könnte das Programm "DS", welches bei mir den PR1-Befehl so gut wie ersetzt, durch den o.a. Befehl ersetzt werden - aber ich würde mich der vielfältigen Gestaltung von Ausdrucken im Einzelfall entledigen ...

Weiterhin fühle ich mich nicht als der "Guru", der ständig etwas Neues und Besseres erfindet, ich sehe mich als (leidenschaftlich) interessierten Anwender, der immer wieder spielerisch versucht, aus gegebenen Anwendungen eigene (neue) Verbindungen zu suchen. Damit möchte ich einmal eine Lanze für den Teil der Anwender brechen, die meinen, daß Ihre eigenen Entwicklungen nicht "gut genug" für die Gurus und Freaks im Prisma sind.

Aus diesem Grund folgen einige Anwendungsbeispiele von mehr oder weniger vorgegebenen Programmen für den HP28 aus Prisma 4/89.

Das erste Programm "Check" soll verdeutlichen, wie man mit Hilfe der vorgestellten Zeitmessung von Ralf Pfeiffer ein Entwicklungswerkzeug für die eigene Programmierung erstellen kann. Durch die Gegenüberstellung von Byteverbrauch und verbrauchter Rechenzeit ist der Entwickler in der Lage, verschiedene Lösungsansätze auszuprobieren und die geeignetste zu optimieren. Dabei wurde das Ursprungsprogramm nur geringfügig verändert. Eigentlich wurde nur eine sinnvolle Anwendung dazu gepackt.

```
485X CHECK
     « ,8 DELAY CLEAR
001  MEM → a
     « ZE "Pruefterm"
004  ANZ 'a' STO a CR
     BYTES M10 →TAG SWAP
006  M11 →TAG DS DS CR
     TICKS → t
008      « a EVAL TICKS
     TEST 1 MENU t -
010      » 8→R 136 -
     29491200 / →hMS
012  RCLF "t=" ROT 7 FIX
     7 RND HMS+ →HMS
014  →STR DUP ":" POS 1
     - 1 SWAP SUB ":"
016  LASTARG + 1 + DUP 1
     + SUB ":" LASTARG 1
018  + SWAP 3 + SUB ":"
     LASTARG DROP 2 +
020  OVER SIZE SUB 1 7
          START +
022       NEXT 'a' STO
     STOF CLEAR CR
024  "Zeit" a
     "h:min:SSsss" PRSTC
026  CR CR CR CR
     »
028  »  Bytes : 39€
     Checksum : # ED5Eh
```

Die Listings sind alle mit Hilfe des Programms PRP aus 3/90 erstellt, um eine einheitliche Dokumentation zu gewährleisten.

Anmerkungen zu CHECK:
In Zeile 001 wird die Ausgabegeschwindigkeit des Thermodruckers manipuliert, indem die Wartezeiten des HP48 verkürzt werden. 0,8 ist ein ermittelter Wert, der auch für mittlere Programmausdrucke ausreichend Wartezeit übrig läßt, um keinen Datenverlust zu verursachen - selber ausprobieren ....

Zeile 002 verursacht einen "internen Packing"-Befehl, so daß eine einigermaßen "echte" Zeitmessung gewährleistet werden kann.

Die verwendeten Unterprogramme "ZE, ANZ, DS, TEST werden im Anschluß vorgestellt. Sie können bei Bedarf wie folgt ersetzt werden: ZE kann ersatzlos entfallen (s.u.), ANZ kann durch den PROMPT-Befehl ersetzt werden und DS durch PR1 - dafür ist dann zusätzlich nach PR1 ein DROP einzufügen.

M10 (Zeile 005) und M11 (Zeile 006) sind Strings aus dem HOME-Directory und lauten "Bytes" (M10) und "Check " (M11).

Zeile 003 - 006 liefert einmal das Datum mit der Uhrzeit für das zu prüfende Objekt, speichert dieses nach der Eingabeaufforderung in die lokale Variable a und ermittelt den Bytebedarf für das zu testende Objekt.

Danach läuft im Großen und Ganzen das Programm HR aus Prisma 4/89 ab. Die Veränderungen beziehen sich darauf, daß das Programm TMHR ohne die SYSEVAL-Befehle des HP28 mit übernommen wurde und eine andere Kontrollzahl (136 anstelle der 146) in Zeile 010 verwandt wird. Dies hat seine Begründung darin, daß ich häufiger Programme teste und 136 Ticks auf meinem HP48 gerade ein "Nullprogramm" («») benötigt. Für reine STACK-Operationen (algebraische Objekte etc.) sind andere, zum Teil erheblich kleinere Korrekturwerte anzusetzen. auch wird durch FIX 7 die Meßlatte auf 1/1000 s gelegt. Zeile 024 - 026 ist eine Mischung für Anwender mit und ohne Drucker. Zum einen wird die Anzeige "übersetzt" und zum anderen wird der Faulheit Vorschub geleistet, in dem mittels Papiervorschub genau die Abrißkante in Aktion treten kann.

Die etwas verwirrende Einfügung in Zeile 009 ... Test 1 Menu ... erscheint deshalb, weil dieses Programm CHECK in meinem Unterdirectory TEST steht und ich eventuell mehrere Objekte testen möchte (kann also entfallen).

Zum Gebrauch zwei Beispiele:
Es wurde das nachfolgende Programm ZE getestet, beim ersten Mal wurde nur der Name mit dem Paranthesezeichen '..' in die erste Zeile geholt, während beim zweiten Mal 'ZE' RCL erfolgte. Auf diese Weise kommt die unterschiedliche Bytezahl zustande. Der Regelfall dürfte die zweite Art der Prüfung sein. Zu beachten ist jedoch dann, daß innerhalb von Programmaufrufen etwas mehr Bytes benötigt werden - einen ersten Überblick kann die Tabelle für den Byteverbrauch in Prisma 3/90 geben.

```
485X ZE
     « DATE TIME TSTR
002  PR1 CR DROP
     »  Bytes : 25
004  Checksum : # 2529h



SAT 17.11.90  10:58:57

Pruefterm: ZE

CheckΣ : # 9513d
Bytes : 31,5

SAT 17.11.90  10:59:10

3:                  "Zeit"
2:        "t=:0,:00:00528"
1:           "h:min:SSsss"


SAT 17.11.90  10:59:41

Pruefterm:
« DATE TIME TSTR PR1 CR
DROP
»

CheckΣ : # 9513d
Bytes : 25

SAT 17.11.90  10:59:57

3:                  "Zeit"
2:        "t=:0,:00:00521"
1:           "h:min:SSsss"
```

Da das Programm ZE selbsterklärend ist, verzichte ich hier auf eine Kommentie-

```
485X ANZ
     « DUP " eingeben" +
002  PROMPT SWAP →TAG
     DUP DS DTAG
004  »  Bytes : 47
     Checksum : # 7FF9h
```

```
485X DRG
     « "Druckgroeße" DUP
002  ": 1=n 2=nD
     3=k 4=kD 5=G"
004  + PROMPT SWAP →TAG
     PR1 DTAG → X
006    « 1 5
         FOR n
008        IF n X ==
           THEN n SF
010        ELSE n CF
           END
012      NEXT
       »
014  »  Bytes : 143,5
     Checksum : # DFDEh


485X DS
     «
002    CASE 1 FS?
         THEN PR1 DROP
004      END 2 FS?
         THEN DICHT PR1
006  DROP
         END 3 FS?
008      THEN PRR
         END 4 FS?
010      THEN DICHT PRR
         END 5 FS?
012      THEN AN1
         END
014    END
     »  Bytes : 131,5
016  Checksum : # F233h
```

rung. Für mich ist es halt wichtig, daß ich eine zusammenfassende Routine der Zeitdarstellung jederzeit zur Verfügung habe, um ggf. diese in Programme einbinden zu können.

ANZ: Ein beliebiger String wird als Eingabeaufforderung genutzt und durch die Verdopplung mittels DUP gleichzeitig für ein 'Tagged Objekt' genutzt. Der Unterprogrammteil DS regelt die gewünschte Druckerausgabe in Abhängigkeit von der gewählten Schriftgröße in DRG.

DRG: Für die Darstellung der Schriftgröße werden die sichtbaren USER-Flags gewählt, damit jederzeit sichtbar ist, welche Druckausgabe erfolgt. Gleichzeitig werden mit der Festlegung der Schriftgröße die nicht benötigten Flags gelöscht. Die Zeile 002 schließt mit dem Zeichen für New Line (↵) ab, um den String innerhalb von zwei Zeilen im Display zu positionieren. Für Anwendungen ohne Drukker könnte mit dem Flag 6 der Druckbefehl durch eine sinnvolle Anzeige ersetzt werden (Prompt, Disp, Freeze etc.) <in DS dann einarbeiten>

DS Es wurde das Programm DICHT (Prisma 4/89 S. 35) verwandt, um die verschiedenste Druckdarstellung zu gewährleisten.

Die eigentlichen Druckroutinen sind demnach PR1, DICHT PR1, PRR, DICHT PRR und AN1.

Diese Routinen werden in Abhängigkeit der Flags aufgerufen und bestimmen das Schriftbild. Wichtig ist, daß nach jedem Druckbefehl der jeweilige Ausdruck gelöscht wird. Falls er für die weitere Anwendung benötigt wird, ist er ggf. mit DUP zu sichern.

Die Aufforderungen im String des Programmes DRG bedeuten sinngemäß:
1 = Normaldruck
2 = normal + Dicht
3 = kleine Schriftgröße
4 = klein + Dicht
5 = Großdruck

```
485X DICHT
     « 0 → t
002    « →STR 1000000 →
     s
004    «
         WHILE DUP 10
006  CHR POS DUP
         REPEAT DUP2 1
008  - 1 SWAP SUB 32 CHR
     + 3 ROLLD 1 + s SUB
010  +
         END DROP ""
012  SWAP
         WHILE DUP 32
014  CHR POS DUP
         REPEAT DUP2 1
016  SWAP SUB 4 ROLL
     SWAP DUP SIZE t +
018  DUP 't' STO
         IF 24 >
020      THEN DUP
     SIZE 't' STO SWAP
022  DUP SIZE 1 - 1 SWAP
     SUB 10 CHR + SWAP
024      END + 3
     ROLLD 1 + s SUB
026      DO DUP 1 1
     SUB
028      UNTIL 32
     CHR ≠
030        IF DUP
     NOT
032        THEN SWAP
     2 s SUB SWAP
034      END
         END
036    END DROP +
       »
038  »
     »  Bytes : 389,5
040  Checksum : # AC86h
```

Anm.: Es ist sicher möglich, DRG zu ersetzen (z.B. 1 DS; 2 DS etc.), wobei die Abfrage in DS entsprechend gestaltet werden kann ≙ XEQ IND... auf dem HP41; mir hat die Darstellung mit den Flags bisher am besten gefallen - dafür habe ich den größeren Speicherbedarf in Kauf genommen.

```
485X PRR
     « →STR → p
002    « 0 → n
         « p SIZE 38 / 1
004  + IP → m
           « 1 m
006          FOR X p n 1
     + 38 X * SUB 38 'n'
008  STO+
           NEXT 1 m
010        FOR X m
     ROLL AN2 m 1 - 'm'
012  STO
           NEXT
014      »
       »
016  »
     »  Bytes : 203
018  Checksum : # E0D6h

485X AN2
     « 1 →GROB PR1 DROP
002  »  Bytes : 20
     Checksum : # 8A10h

485X AN1
     « 27 CHR 253 CHR +
002  SWAP + 27 CHR + 252
     CHR + PR1 DROP CR
004  »  Bytes : 82
     Checksum : # E217h
```

DICHT (Prisma 4/89 S. 35)

Die Änderungen beziehen sich lediglich auf die lokale Variable '!' und ':', diese wurden in Namen abgeändert, die von meinem HP akzeptiert wurden, die Struktur als Unterprogramm erspart mir das Löschen der Variablen am Schluß. Ebenso wurde das letzte PR1 DROP aus dem Programm entfernt, um vielseitigere Anwendungen zu gewährleisten.

PRR

Dieses Programm formt den STACK 1 zu einem String um. Dieser Gesamtstring wird dann portionsgerecht für das Unterprogramm AN2 zerlegt (Zeile 003 - 009). Die lokalen Variablen n und m dienen als Platzhalter für die Zerlegungs- und Ausgaberoutine, es kann nicht gerundet werden (Zeile 004), da auch noch Reste in nachfolgenden Zeilen gedruckt werden müssen. Nach dem Ausdruck ist der ursprüngliche Inhalt der STACK-Zeile 1 verschwunden (s. Bemerkungen zu den Druckroutinen). 38 in Zeile 007 ist eine experimental ermittelte Zahl, um für das Grafikobjekt keine störenden Trennlinien zu produzieren (eigentlich müßte es ja 39 sein - aber was ist schon Mathemetik gegenüber dem praktischen Leben...).

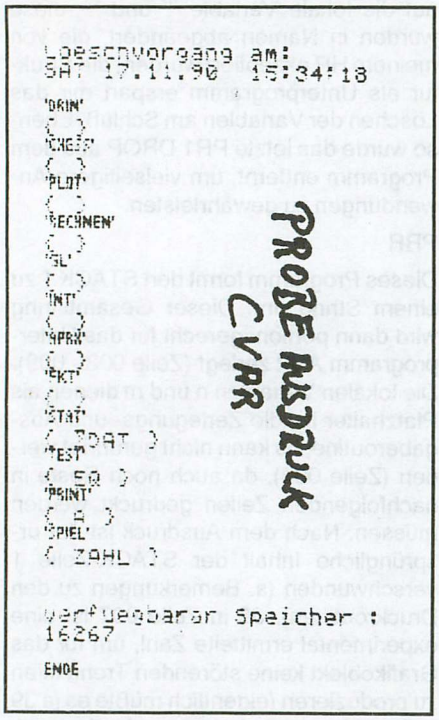
Da ich der Meinung bin, daß einmal vorhandene Bausteine immer wieder neu genutzt werden sollten, füge ich gleich noch ein Programm an, das überflüssigen Datenmüll entfernt. Das Programm CVAR löscht mit Ausnahme des HOME-Directories sämtliche Datenvariablen, es werden nur Programme, Listen und Directories gesichert.

Gleichzeitig wird der Löschvorgang protokolliert (da bricht der Beamte wieder in mir durch - wen diese Funktion stört, der kann sie ja weglassen), so daß ggf. wichtige Daten wieder eingegeben werden können.

```
485X CVAR
     «
002  "Loeschvorgang am:"
     PR1 DROP ZE CLEAR
004  HOME 15 TVARS OBJ→
     DROP
006    DO DUP PRR EVAL 8
     TVARS 5 TVARS 15
008  TVARS + + DUP ORDER
     OBJ→ → a
010      « a DROPN VARS
     OBJ→ → b
012        « b a - →LIST
     PR1 PURGE a DROPN
014  15 TVARS OBJ→ DROP
       »
016    »
       UNTIL DEPTH 0 ==
018    END SPK 1 MENU
     »  Bytes : 219
020  Checksum : # 85257h


485X SPK
     « CLEAR CR CR MEM
002  117,5 + M7 →TAG DE
     CR M9 AN2 CR CR CR
004  CR HOME 1 MENU
     »  Bytes : 73,5
006  Checksum : # 730Ah
```

Unterprogramme sind diesmal: ZE, PRR und SPK, sowie die Strings aus den HO-



ME-Dirctory M7 ("verfügbarer Speicher") M9 ("Ende"); in SPK werden die oben vorgestellten Programme DS und AN2 wieder verwandt.

Anm.:

Clear Variable gibt in Zeile 001 - 003 das Datum aus, danach wird ausgehend von den in Home vorhandenen Directories der HP nach Datenmüll durchforstet (in Home stehen wie gehabt immer noch die "Dauervariablen"). Da ja nicht immer bekannt ist, wie die Datenvariablen bezeichnet wurden, bleibt nur die Suche über die TYPE-Funktion. In Zeile 004 werden die einzelnen Directories auf den Stack geholt und die Liste aufgelöst. Die einzelnen Directories werden nun mit EVAL initialisiert und aus ihnen lediglich die Programme, Listen und ggf. vorhandene Directories in Listenform gesichert. Es erfolgt dann (wieder der Ordnung halber) eine Umgruppierung nach Typen bevor die Liste in a zwischengespeichert wird (nur die Anzahl der Variablen in der Liste). Da sich jetzt neben dem noch nicht geprüften Directories noch die Listenvariablen im Stack befinden, werden diese jetzt gelöscht - mit Hilfe von Vars wird eine Kontrolliste gebildet, deren Kontrollzahl ebenfalls zwischengespeichert wird. Da durch das vorangegangene Umgruppieren die zu sichernden Variablen am Anfang stehen, kann nun mittels "Listenarithmetik" eine Liste gebildet werden, die Grundlage für den PURGE-Befehl ist. Danach werden die gesicherten Variablen vom Stack gelöscht und weiter gehts ...

Das Programm SPK gibt dann den noch verfügbaren Speicherplatz aus, so daß man für kommende Anwendungen wieder bestens gerüstet ist.

Die Kontrollzahl 117,5 in Zeile 002 ist empirisch ermittelt, also mit Fehlern behaftet, da ungefähr genauso viel Bytes irgendwo im Rechner "versteckt" werden, sobald das Programm in Aktion getreten ist (zu dem Papiervorschub siehe Stichwort Faulheit oben).

Für das Programm SORT kann dasselbe aus Prisma 6/89 S. 38 verwendet werden.

Georg Hoppen
Hubertusring 5
4512 Wallenhorst

Adressverwaltung auf dem HP48SX

von M. Breidenbach und A. Lehmann

Dies ist ein einfaches Adressverwaltungsprogramm für den HP48SX. Man legt es am besten in einem Unterverzeichnis ab.

Die Adressen werden als Strings in der Liste FILE gespeichert.

Dieses Programm ist genaugenommen eine Sammlung kleiner Routinen, die über das Custom-Menu aufgerufen werden. Es gibt kein umfassendes "Hauptprogramm". Deshalb wird das Programm gestartet, indem man in das entsprechende Verzeichnis wechselt und die CST-Taste drückt.

Die Routine DSP zeigt denjenigen Datensatz an, dessen Nummer in N gespeichert ist. Am Ende aller Routinen wird DSP aufgerufen. Wenn man das "Programm" durch Drücken der Taste CST startet, ist allerdings kein Datensatz in der Anzeige, da DSP noch nicht aufgerufen wurde. Zur Abhilfe kann man sich eine kleine Routine schreiben, mit der man dieses Programm startet, die in das entsprechende Verzeichnis wechselt, das CUSTOM-Menu initialisiert und DSP aufruft.

Das Programm enthält keine Sicherheitsabfragen für den Fall, daß FILE nicht existiert bzw. leer ist (d.h. keine Datensätze eingegeben sind). Zu Anfang sollte FILE daher eine Liste beinhalten, die aus einem Leerstring besteht. Der Wert der Variablen N muß immer zwischen 1 und der Anzahl der Datensätze liegen (d.h. am Anfang N=1). Darauf hat der Benutzer zu achten.

Das Custom-Menu wurde in zwei Teile aufgeteilt. Dies hat folgenden Grund: DSP zeigt einen Datensatz an und friert die Anzeige mit FREEZE bis zum nächsten Tastendruck ein. Würde man mit einem Custom-Menu arbeiten, so würde das Betätigen der Next-Taste den aktuellen Datensatz aus der Anzeige löschen. Wenn man mit zwei Custom-Menus arbeitet, kann man nach dem Umschalten in die zweite Menuhälfte gleich DSP aufrufen, damit der aktuelle Datensatz wieder in die Anzeige kommt.

Die Bedeutung der einzelnen Variablen bzw. Routinen ist:

CST	Variable für CUSTOM-Menu
CST1	erste Custom-Menu-Hälfte
CST2	zweite Custom-Menu-Hälfte
FILE	die "Datei"
N	enthält den aktuellen Datensatz
SS	enthält den aktuellen Suchstring
QSORT	Quicksort, sortiert eine Liste auf dem Stack
ICR	nach oben blättern
DCR	nach unten blättern
ED	Datensatz editieren
ADD	Datensatz hinzufügen (am Anfang der Liste FILE)
GON	Gehe zu dem Datensatz, dessen Nummer auf dem Stack liegt
FIND	Suchstring eingeben und suchen
DEL	Datensatz löschen
DELJN	Sicherheitsabfrage für DEL

M. Breitenbach
A. Lehmann

```
DIR
  QSORT
    « OBJ→ DUP 1
      IF >
      THEN DUP 2 / 1 + ROLL { } { } → t kl
gr
        « 1 - 1 SWAP
          FOR i DUP t <
            IF
            THEN 'kl'
            ELSE 'gr'
            END STO+
          NEXT
           kl QSORT t + gr QSORT +
        »
      ELSE →LIST
      END
    »
  CST CST1
  CST1 { { "<"     « DCR DSP » }
         { ">"     « ICR DSP » }
         { "FIND" « "Suchstring" SS -1 2
           →LIST INPUT 'SS' STO FIND DSP
                 » }
         { "NEXT" « N FILE SIZE MOD 1
                + 'N' STO FIND DSP » }
         { "GOTO" « GON DSP » }
         { "=>"   « 'CST2' MENU DSP » }
       }
  CST2 { { "<="   « 'CST1' MENU DSP » }
         { "ADD" « ADD DSP » }
         { "DEL"  « DELJN DSP » }
         { "EDIT" « ED DSP » }
         { "SORT" « FILE QSORT 'FILE'
                  STO 1 'N' STO DSP » }
         { "Ende" HOME }
       }
  GON
    « 1 - FILE SIZE MOD 1 + 'N' STO »
  FIND
    «
      IF SS "" <>
      THEN 0 → found
        « FILE N
          DO GETI SS POS
            IF 0 <> DUP 'found' STO
            THEN DUP 2 - FILE SIZE
                 MOD 1 + 'N' STO
            END
          UNTIL DUP N == found OR
          END DROP2
        »
      END
    »
  DELJN
    « "Löschen ?" 7 DISP 0 FREEZE
      { { "Ja" } { } { } { } { }
        { "Nein" } } TMENU -1 WAIT
      IF 11.1 SAME
      THEN DEL
      END
      0 MENU
    »
  DEL
    « FILE 1 N 1 - SUB
      FILE DUP SIZE N 1 + SWAP SUB
      + 'FILE' STO
      FILE SIZE N MIN 'N' STO
    »
  ADD
    « "" { "" α } INPUT →STR DUP
      IF "" <>
      THEN 'FILE' STO+ 1 'N' STO
      ELSE DROP
      END
    »
  ED
    « "" FILE N GET 1 2 →LIST INPUT
      →STR FILE SWAP N
      SWAP PUT 'FILE' STO
    »
  ICR
    « N FILE SIZE MOD 1 + 'N' STO
    »
  DCR
    « N 2 - FILE SIZE MOD 1 + 'N' STO
    »
  DSP
    « FILE N GET CLLCD 1 DISP N 7
                  DISP 3 FREEZE
    »
  N 1
  SS ""
  FILE { "" }
END
```

Zwei kleine Programme für den HP 48SX

Nach dem Erscheinen des HP 48SX lege ich zwei Programme aus der Ephemeridenrechnung vor (STZ und KP), von denen eines (KP) das Analogon vom HP41 Programm gleichen Namens in PRISMA 89-4-22 ist.

Die Sternzeit ist die Rektaszension des Frühlingspunktes.

Das Programm STZ errechnet die Sternzeit:

Flag 1 gesetzt:
Datum(dd.mmjjjj) und Zeit(UT) eingeben, ENTER.

Flag 2 gesetzt:
Geographische Länge des Beobachtungsortes eingeben, östliche Länge negetiv, Grad, Minuten Sekunden, ENTER.

Beide Flags gelöscht:
Uhr nach UT stellen. Es wird die momentane Sternzeit des vorgegebenen Ortes ausgegeben. Im ausgedruckten Programm -7.5219 für den Heimatort des Autors (Zeile 6). Wird am Ende des Programmes hinter FREEZE STZ, der Name des Programms eingefügt, so ruft sich das Programm selbst auf. Die Sternzeit wird laufend angezeigt. Anhalten mit ATTN.

Nach meinen Untersuchungen bedarf ein Durchlauf von KP im neuen Rechner 1/5 der Zeit (entsprechend dem analogen Basic-Programm im HP71).

Literatur: siehe PRISMA 89-6-30

```
K P
« CLEAR AEM
RAD D→R π  NUM DUP
+ MOD FMM DUP 'M'
STO DUP ROT
DUP 'E'STO + DUP
  IF π  >
  THEN DROP π
  END SWAP
  DO SKP DUP FKP 7
RND Ø
  UNTIL ==
  END
  IF 10 FS?C
  THEN SPM
  END SWAP DROP R→D
DEG DUP 2 / TAN 'E'
RCL DUP 1 + 1 ROT -
/ √ * ATAN DUP +
360 MOD SWAP COS
'E' RCL * 1 SWAP
ROT * EAG
```

```
#1274d
   385
```

```
S T Z

« CLEAR STD
  IF 2 FS?
  THEN "Länge:" {
":L:" { 1 Ø } V }
INPUT OBJ→
  ELSE -7.5119
  END HMS  15 / NEG
  IF 1 FS?
  THEN
"Datum, Zeit: {
":D:
:Z:" { 1 Ø } V
} INPUT OBJ→
  ELSE DATE TIME
  END HMS→
1.0027379 * 1.0119
ROT DDAYS .5 +
36525 / DUP .0929 *
8640184.542 + *
23925.836 + 3600 /
+ + 24 MOD →HMS 4
FIX  STR STD
"Sternzeit:" SWAP +
CLLCD 1 DISP 1
FREEZE
»
        # 32365d
          372.5
```

```
A E M
« "Eingabe a,ε,M" {
":a:■
:ε:■
:M: { 1 Ø
} V } INPUT OBJ→
»
  # 44659d
       76

F F M
« DUP
  IF π >
   THEN 10 SF SPM
  END
«
  # 60945d
       57

S K P
« DUP2 DUP2 SWAP -
ROT FKP ROT FKP / 1
- / +
»
  # 890d
    58
```

```
F K P
« DUP SIN 'E' RCL *
- 'M' RCL -
»
  # 19823d
       54

S P M
« π →NUM DUP + SWAP
-
»
  # 49544d
      32.5

E A G
« →STR "r = " SWAP
+ SWAP →STR "v = "
SWAP +
»
  # 8246d
      53
```

Programmschutz

im HP48

Die Variable CST bietet einen Weg, Programme eines Verzeichnisses vor unbeabsichtigter Veränderung besser zu schützen.

Beispiel:
In einem bestimmten Verzeichnis liegen die gefährdeten Programme STZ und SON, die weder durch rechts noch durch links Umschaltung beeinflußbar gemacht werden sollen.
Dazu wird die Liste {{ "STZ" { STZ "" "" }}{ "SON" { SON "" "" }}} aus der Befehlsebene in Ebene 1 gebracht und der Befehl MENU ausgeführt (ganz rechts im umgeschalteten Modes Menü).
CST dieses Verzeichnisses enthält dann STZ und SON mit den gewünschten Eigenschaften im Menü.

VORSICHT! Jede Änderung eines Programms in VAR ändert diese auch in CST.

Dr.G.Heilmann
Obernhofer Straße 15
5408 Seelbach

Alarmsystem für den HP71B

von Wolf-Thorsten Gerdts

ALMCCD1

LEX-File, 3539 Bytes, ID #E1, Token #F0

Ver$ ALM:1D

Neue BASIC-Worte:

ALMSET
ALMREL
ALMCAT
ALMGET
ALMCLR
ALMON
ALMOFF
ALMACK
ALMEXC

Erweiterte Befehle:

LIST
PLIST

Gliederung:

1 Kurze Zusammenfassung

Ein Alarmsystem existiert im HP41 und im HP75. Auch der HP71B schien darauf vorbereitet (Anzeige ((0))).

ALMCCD1 ist ein LEX-File, der den HP71 mit allen neuen Funktionen versieht, die notwendig sind, um den HP71 als Terminkalender, Wecker o.ä. zu verwenden.

Das Alarmsystem funktioniert weitestgehend so wie beim HP75; kein Wunder, sein Anleitungsbuch stand Pate.

Ganz anders sieht die Eingabe und das Editieren von Alarmen aus: Der HP75 hat einen speziellen ALARM-Modus, ALMCCD1 benutzt zur Eingabe von Alarmen BASIC-Worte und ist damit viel flexibler: Alarme können Alarme setzen und löschen, und falls besonderer Wert auf einen Alarm-Modus gelegt wird, so kann ein entsprechendes Programm in BASIC geschrieben werden.

2 Funktionsweise

2.1 Alarme

Ein Alarm ist vergleichbar mit dem STARTUP-String des Rechners: Der STARTUP-String wird bei jedem Einschalten des Rechners ausgeführt; der Alarm ist flexibler, der Zeitpunkt zu dem er ausgeführt werden soll, kann mittels der neuen BASIC-Worte bestimmt werden. Zusätzlich läßt sich ein Wiederholungsintervall eingeben. Ein Alarm muß nicht geräuschlos ablaufen, sondern kann mit 10 verschiedenen akustischen Signalen versehen werden.

Gespeichert werden Alarme in einem neuen Filetyp:

APPT ID #E220

Es können mehrere APPT-Files im Speicher existieren, der Rechner greift immer nur auf den APPT-File zurück, der im Hauptspeicher steht und den Namen APPTS trägt.

Nur dieser File kann mit ALMCAT (s. u.) bearbeitet werden. Alle APPT-Files hingegen können mit LIST bzw. PLIS angesehen werden.

Beim ersten Setzen eines Alarms wird der APPT-File APPTS im Hauptspeicher automatisch kreiert. Er existiert weiter, auch wenn alle Alarme gelöscht sind.

Mit RENAME und COPY lassen sich andere APPT-Files zum aktuellen Alarmfile machen, der aktuelle Alarmfile muß, wie erwähnt, den Namen APPTS tragen und im Hauptspeicher stehen. Der aktuelle APPT-File APPTS wird im folgenden immer als Alarmfile bezeichnet.

2.2 Wenn ein Alarm fällig wird

2.2.1 Rechner ausgeschaltet

Bei ausgeschaltetem Rechner schaltet sich der Rechner zum Alarmzeitpunkt ein, Flag -60 wird gesetzt, die Anzeige ((0)) erscheint und es erklingt ein akustisches Signal gemäß dem Alarmtyp. Dannn wird der Alarm ausgeführt und der Rechner schaltet sich wieder aus.

Flag -60 bleibt solange gesetzt (und bei eingeschaltetem Rechner ist solange die Anzeige ((0)) zu sehen), wie im Alarmfile ein abgelaufener Alarm steht, der noch nicht quittiert ist(s. u.).

2.2.2 Rechner eingeschaltet

Bei eingeschaltetem Rechner wird Flag -60 gesetzt, die Anzeige ((0)) erscheint und das entsprechende Signal ertönt.

Erst beim Ausschalten des Rechners - oder durch Ausführung der Anweisung ALMEXC - wird der Alarm ausgeführt.

2.3 Zustände eines Alarms

2.3.1 Alarm in der Zukunft

2.3.2 Alarm wird fällig

Der Rechner reagiert gemäß 2.2; im Alarmkatalog wird der Alarm mit P (groß P) für past due markiert, außerdem mit x (klein x) als Markierung, daß dieser Alarm noch ausgeführt werden muß, x verschwindet nach Ausführung des Alarmes.

2.3.3 Alarm wird quittiert

Ein Alarm wird solange im Katalog mit P markiert, ((0)) und Flag -60 bleiben gesetzt, bis der Alarm durch den Benutzer quittiert wird. Dies geschieht durch die Anweisung ALMACK n (für acknowledge, n steht für den n-ten Alarm im Katalog).

Ist der Alarm quittiert, so wird P (großes P für past due) durch a (kleines a für acknowledge) in der Katalogseite ersetzt.

3 Alarmeingabe

Die Eingabe eines Alarms erfolgt mit - auch programmierbaren - BASIC-Anweisungen.

SETALM Alarmtyp, Alarmstring [[Rep.Intervall,]Datum,]Zeitpunkt

ALMREL Alarmtyp, Alarmstring, [Rep.Intervall,] Abstand zum jetzigen Zeitpunkt.

3.1 Alarmstring

Der Alarmstring ist ein Stringausdruck mit maximal 95 Zeichen. Er sollte eine syntaktisch einwandfreie BASIC-Eingabezeile sein, Ausnahme siehe 3.1.2.

Also: "BEEP" oder "FORR=1 TO 10@BEEP@NEXTR" oder " 'jetzt gehts los'@BEEP", bzw. besser lesbar mit Leerzeichen: "FOR R=1 TO 10 @ BEEP @ NEXT R".

3.1.1 >$

Aus Gründen der Kompatibilität zum HP75 kann der Alarmstring auch mit dem Zeichen > beginnen.

Also: ">BEEP" oder ">DISP 'so also auch ';TIME$@WAIT3".

3.1.2 !$

Aus Gründen der Kompatibilität zum HP75, aber weil es sonst auch nicht schlecht ist: Beginnt der Alarmstring mit einem ! (Ausrufezeichen), so wird der gesamte String - ohne Ausrufezeichen am Anfang - für ca. 5 sek angezeigt. Wird vor Ablauf der 5 sek eine Taste gedrückt, verschwindet diese Anzeige wieder.

Also: "!Es geht auch so . . . " statt "DISP'es geht auch so'@ WAIT 5".

3.2 Der Alarmtyp

Die Angabe des Alarmtyps besagt, welches akustische Signal beim Fälligwerden des Alarms ertönen soll.

Wenn die Rechnerbatterien sehr leer (Zustand very low battery) sind, ertönen keine Geräusche mehr, um nicht durch völlig entleerte Batterien ein memory lost zu verursachen.

Alarmtyp
- 0 es ertönt kein Signal
- 1 es ertönt ein Geräusch wie der ERROR-Beep
- 2 ein langer, tiefer Ton
- 3 dreimal ein hoher und ein tiefer Ton
- 4 40mal Kingeln
- 5 ein langer tiefer und ein langer hoher Ton
- 6 viermal die Sirene
- 7 wie Alarmtyp 2, aber der lange tiefe Ton wird alle 15 sek wiederholt, bis eine Taste gedrückt wird oder die Batterien leer sind. Der Rechner selbst ist solange gesperrt.
- 8 wie Alarmtyp 7, es wird jedoch Alarmtyp 4 wiederholt.
- 9 wie Alarmtyp 7, es wird jedoch Alarmtyp 6 wiederholt.

Jedes akustische Signal kann durch beliebigen Tastendruck sofort gestoppt werden.

Alle Alarmgeräusche können durch Flag -25 laut oder leise gestellt werden.

Alle Alarmgeräusche außer Typ 6 und Typ 9 können durch BEEP OFF bzw. SFLAG -2 unterdrückt werden.

Typ 6 und Typ 9 erklingen immer, außer im very-low-battery-Zustand.

3.3 Der Alarmzeitpunkt

Der Alarmzeitpunkt wird angegeben in HH.MMSSss
HH Stunden, kann auch über 24 sein; 30 ist 6.00 morgen, 54 ist 6.00 übermorgen.

MM Minuten

SSss Sekunden und Sekundenbruchteile

3.4 Alarmdatum

Das Alarmdatum wird eingegeben in der Form WS.TTMMJJJJ

- TT Tag des Monats
- MM Monat
- JJJJ Jahr

Das mit TTMMJJJJ eingegebene Datum kann modifiziert werden durch den Vorkommaanteil WS (Wochentag-Spezifikator).

W gibt den Wochentag an:
- 1 Montag
- 2 Dienstag
- 3 Mittwoch
- 4 Donnerstag
- 5 Freitag
- 6 Samstag
- 7 Sonntag
- 8 Werktag

S spezifiert den Wochentag:
- 0 nächste Woche
- 8 nächster entsp. Wochentag
- 9 entsprechender Wochentag vor dem eingegebenen Datum
- 1,2,3,4,5 entsprechender Wochentag in der nächstmögl. 1,2,3,4,5 Woche im nächtmöglichen Monat.

Achtung! Wochenanfang ist der Montag (wichtig für den Begriff "nächste Woche"!).

Werktag (8) arbeitet nur mit den Spezifikatoren 8 und 9.

Also:
16.11091990 ist der nächstmögliche Montag nach dem 11. September, also der 17.9.
25.25071990 ist der nächstmögliche 5. Dienstag im Monat nach dem 25. Juli 1990, das wäre der 30. Oktober 1990.

Das heutige Datum kann durch 0 ersetzt werden:
88 wäre der nächste Werktag nach dem heutigen Datum.

3.5 Wiederholungsintervall

Bei Wiederholungsalarmen wird das Wiederholungsintervall in der Form WS.MMTTHHmmss eingegeben:

- W Wochentag
- S Spezifikator
- MM Monate
- TT Tage
- HHmmss Stunden Minuten Sekunden.

Für den Zeitpunkt der Wiederholung wird erst das Intervall MMTTHHmmss addiert, dann gemäß Wochentag und Spezifikator modifiziert.

Also:
- 46 als Wiederholungsintervall: Der Alarm erscheint jeden Donnerstag.
- 51 der Alarm erscheint an jedem ersten Freitag im Monat.
- 36.0007 dieser Alarm erscheint vierzehntäglich am Mittwoch. Der Alarm erscheint an einem Mittwoch. Dann werden 7 Tage addiert (.0007), das ist dann der nächste Mittwoch, danach wird gemäß 36 modifiziert, das heißt, der dann nächstmögliche Mittwoch genommen.

Bei einem Alarm ohne Wiederholungsintervall kann das Wiederholungsintervall ganz weggelassen werden, oder es wird 0 eingegeben.

3.5.1 N-Alarme

Ein Alarm ohne Wiederholungsintervall wird im Alarmkatalog mit N bezeichnet (non-repeating alarm).

Bei einem Alarm ohne Wiederholungsintervall kann das aktuelle Datum auch, statt 0 einzusetzen, ganz weggelassen werden.

3.5.2 R-Alarme

Wird das Wiederholungsintervall als positive Zahl eingegeben, so wird der Alarm im Alarmkatalog als R bezeichnet (automatic resheduling alarm).

Ein R-Alarm quittiert sich selbst, addiert dann das Wiederholungsintervall und stellt sich automatisch auf den nächsten Zeitpunkt ein.

Wenn ein R-Alarm fällig wird, erscheint das ((0))-Zeichen nur ganz kurz, da der Alarm sich selbst unverzüglich quittiert.

R-Alarme sind sinnvolle Vorgänge, die keine Beachtung durch den Benutzer zu finden brauchen:
z.B. ein Alarm, der zur vollen Stunde die Stunden schlägt. Ein Alarm, der täglich oder auch werktäglich ein Programm aufruft, das die täglichen Alarme setzt, wie z.B. Termin Dienstbeginn, Dienstschluß etc.

3.5.3 A-Alarme

Ein A-Alarm ist ein Wiederholungsalarm, dessen Wiederholungsintervall als **negative Zahl** eingegeben wurde.

Er wird im Alarmkatalog mit A bezeichnet (appointment alarm).

Wird ein A-Alarm fällig, so erscheint ((0)), der Alarm wird aber erst **dann** auf sein neues Datum gesetzt, wenn er vom Benutzer quittiert wurde durch ALMACK n oder durch fA im Alarmkatalog (s.u.).

Erst dann wird das Wiederholungsintervall zum Alarmzeitpunkt (nicht zum Zeitpunkt der Quittierung!!) addiert. ((0)) verschwindet wieder, wenn sonst kein noch nicht quittierter Alarm im Katalog ist.

A-Alarme sind sinnvoll, wenn der Benutzer den Inhalt auf jeden Fall bestätigen soll, wie Erinnerung an regelmäßige Termine usw.

4 Neue BASIC-Worte

`ALMSET #,$,[[#,]#,]#`

Typ, Alarmstring,[[Wiederholungsintervall,]Datum,]Alarmzeit

Form der Eingabe s.o.

ALMSET dient dazu, einen Alarm auf einen bestimmten Zeitpunkt zu setzen. In R-Alarmen wird das Wiederholungsintervall positiv gesetzt, in A-Alarmen negativ.

Das heutige Datum kann durch 0 ersetzt werden, in Alarmen ohne Wiederholungsintervall auch ganz weggelassen werden.

`ALMREL #,$,[#,]#`

Typ, Alarmstring,[Wiederholungs-Intervall,]zu addierende Zeit

Zu addierende Zeit: HHHH.MMSSss

ALMREL dient dazu, einen Alarm relativ zur augenblicklichen Zeit einzugeben.

ALMOFF

deaktiviert das Alarmsystem, es können zwar weiterhin Alarme eingegeben werden, es werden jedoch keine Alarme mehr ausgeführt.

Die Deaktivierung wird im aktuellen Alarmfile gespeichert. Abfrage siehe ALMGET.

ALMON

aktiviert das Alarmsystem. Abfrage siehe ALMGET.

Das Flag für ALARMON/OFF wird im aktuellen Alarmfile (APPT-File mit dem Namen APPTS im Main Memory) in dessen FLAG-Field gespeichert. Wird ein Alarmfile auf Massenspeicher abgespeichert und anschliessend zurückgeladen, so ist dieses Flag immer gelöscht und damit ALMON.

Bei COPY und RENAME im RAM-Bereich bleibt das FLAG-Field erhalten, und die Information, on ALMON oder ALMOFF bleibt erhalten.

ALMACK #

n. Alarm

Mit ALMACK n wird der n-te Alarm quittiert.

Solange im Alarmkatalog noch ein abgelaufener Alarm steht, der noch nicht quittiert worden ist, bleiben Flag -60 und die Anzeige ((0)) gesetzt.

ALMCLR # oder ALMCLR $

löscht den n-ten Alarm oder den ersten Alarm, der mit dem eingegebenen String *beginnt*.

Existiert der Alarm nicht, erscheint die Warnung End of file.

ALMEXC

führt den ersten Alarm aus, der abgelaufen, aber noch nicht ausgeführt ist. Solche Alarme sind im Alarmkatalog mit x bezeichnet. Nach Ausführung verschwindet das x.

ALMEXC kann auch in einem Programm stehen. Falls der ausgeführte Alarmstring allerdings nicht mit einem ! beginnt stoppt das Programm.

Flag -60 und die Anzeige ((0)) wird durch ALMEXC nicht beeinflußt.

ALMGET #;var[,var$[,var[,var[,var[,var[]]]]]]

n;Typ[,Almstring[,Wiederh.-Int.[,Datum[,Zeitpunkt[,Zustand]]]]]

ALMGET dient zum Ausgeben eines bestimmten Alarmes.

n ist die Nummer in der Reihenfolge der Alarme. Typ, Alarmstring, Wiederholungsintervall, Datum und Zeitpunkt werden in die jeweiligen Variablen ausgegeben wie eingegeben.

Falls kein Wiederholungsintervall existiert (N-Alarme!), wird für das Wiederholungsintervall 0 ausgegeben.

Zustand gibt ein Byte aus mit folgender Bit-Bedeutung:

Bit-Nr.	Wert	Bedeutung, wenn gesetzt
0	1	Alarm ist quittiert
1	2	Alarm ist abgelaufen
2	4	Alarm muß noch ausgeführt werden.

Der Zustand des Alarms ergibt sich aus der Summe der gesetzten Bits. Ein Alarm mit dem Zustand 6 ist abgelaufen, muß noch ausgeführt werden und ist noch nicht quittiert worden.

Wird ein Alarm aufgerufen, der nicht existiert (n zu groß), so erscheint zwar keine Warnung, aber es wird ein Alarmtyp von -1 ausgegeben, sonst jeweils 0 oder der Nullstring.

2 wichtige Sonderfälle:

ALMGET0;var
var ist eine beliebige numerische Variable.
Besteht der Zustand ALMON, wird in var 1 gespeichert, ist ALMOFF, wird 0 gespeichert.

ALMGET-#;var
Wird mit ALMGET ein negativer Alarm abgefragt, wird in der Variablen die Nummer des ersten Alarms gespeichert, der nicht past due ist.

ALMCAT

Mit dieser Funktion wird der Alarmkatalog aufgerufen. Er bietet eingeschränkte Editiermöglichkeiten für den aktuellen Alarmfile.

Während der Ausführung von ALMCAT ist die Ausführung aller Alarme gesperrt. Wird ALMCAT aufgerufen, so erscheint in der Anzeige der zuletzt fällig gewordene Alarm. Ist noch kein Alarm fällig geworden, erscheint der als nächster fällig werdende Alarm.

Im Alarmkatalog sind eine Reihe von Tasten neu definiert:

fOFF Verlassen des Alarmkataloges
ATTN Verlassen des Alarmkataloges

Pfeil links, Pfeil rechts, gPfeil links, gPfeil rechts: Scrollen des angezeigten Alarms

Pfeil nach oben: Anzeigen des vorherigen Alarms. Wird schon der erste Alarm angezeigt, hat diese Taste keine Wirkung.

Pfeil nach unten: Anzeigen des nächsten Alarms. Wird der letzte Alarm bereits angezeigt, hat diese Taste keine Funktion.

gPfeil nach unten: Anzeige des ersten Alarms.

gPfeil nach oben: Anzeige des letzten Alarms.

T Anzeige von Datum und Uhrzeit, solange T gedrückt ist.

fDELETE Löschen des angezeigten Alarms. Angezeigt wird dann der nächste Alarm. wird der letzte Alarm gelöscht, wird der dann letzte angezeigt. Sind alle Alarme gelöscht, wird automatisch der Alarmkatalog verlassen.

fA Quittieren des angezeigten Alarmes. Sind alle Alarme quittiert, verschwindet die Anzeige ((0)).

Beim Quittieren eines Repeat-Alarms wird dieser auf den neuen Zeitpunkt gesetzt.

fD setzt einen Repeat-Alarm um ein Repeat-Intervall weiter. Liegt der Alarmzeitpunkt dann immer noch in der Vergangenheit, wird ein weiteres Intervall addiert usw.

fX toggelt die x-Anzeige bei einem Alarm. Damit kann entschieden werden, ob ein Alarm beim nächsten Ausschalten des Rechners ausgeführt wird oder nicht.

RUN der angezeigte Alarm wird ausgeführt, wenn er mit x markiert ist, der Alarmkatalog wird verlassen.

Darstellung der Alarme im Alarmkatalog:
DD.MM.JJ hh:mm:ss** wT $

**	abgelaufen	quittiert	noch auszuführen
P	ja	nein	nein
x	nein		ja
xP	ja	nein	nein
a	ja	ja	ja
xa	ja	ja	nein
	nein		

w N für Normaler nicht-Wiederholungsalarm
A für Wiederholungsalarm, der quittiert werden muß
R für Wiederholungsalarm, der sich automatisch quittiert.

T Alarmtyp

$ Alarmstring

LIST file[,#[,#]]
PLIST file [,#[,#]]

Mit LIST und PLIST kann jeder APPT-File gelistet werden. Ohne Parameter wird der gesamte File gelistet. Wird nur ein Parameter angegeben, so wird nur dieser eine n-te Alarm gelistet, bei zwei Parametern geben diese den ersten und den letzten zu listenden Alarm an.

Die Parameter müssen Zahlen sein, Ausdrücke (wie LIST ALARME,A,2*A) können nicht verwendet werden.

5 Beispiel

ALMSET 0,CALLSTSCH,.000001,0,19

ruft ab 19.00 Uhr heute stündlich das Programm STSCH auf.

ALMSET 4,!Gem.Mittagessen,-28, 28.01011991,12.15

Dieser Alarm zeigt ab dem auf den 1. Januar 1991 folgenden Dienstag jeden Dienstag jeweils um 12.35 Uhr "Gem.Mittagessen" für 5 sek oder bis Tastendruck an. Bis zur Quittung durch ALMACK n oder im Alarmkatalog mit fA bleibt ((0)) in

der Anzeige und Flag -60 gesetzt. Nach dem Quittieren wird der Alarm auf den folgenden Dienstag gesetzt.

Flag -60 wird nur dann gelöscht und ((0)) verschwindet nur dann, wenn sich im Alarmfile kein Alarm mehr befindet, der past due und noch nicht quittiert ist.

ALMREL 5,Spaghetti fertig!!,.12

In 12 Minuten zeigt dieser Alarm an, daß die Spaghetti fertig sind.

ALMSET 0,CALLWECKER,88,10,6.00

ruft werktäglich ab Montag nächster Woche das Programm Wecker auf.

Abschließend ein herzliches Dankeschön an Matthias Rabe, der mir viele Tips und Ideen zur Weiterentwicklung des Programms gegeben hat.

Über Zuschriften mit Kritik, weiteren Ideen, Bugs o.ä. würde ich mich sehr freuen.

```
  LEX 'ALMCCD1'
  ID #E1
  MSG 0
  POLL PH
  ENTRY WCC
  CHAR #D
  KEY 'ALM'
  TOKEN #F0
  ENDTXT
ATYP EQU #E220
TBL GOSUB TBL1
  NIBASC 'ONACKCAT'
  NIBASC 'CLREXCRE'
  NIBASC 'LSETOFF'
  NIBASC 'GET'
TBL1 C=RSTK
  CD0EX
  R3=C
  P= 3
CC RTNCC
DP CD0EX
  C+P+1
  CD0EX
  P= 5
  RTNCC
PS GOSBVL #3FB4
  GOSUB TBL
  B=0 A
PS1 B=B+1 A
  C=DAT0 W
  CSL W
  ?C=0 B
  GOYES PSERR
  CSR W
  ?A=C WP
  GOYES PS2
  GOSUB DP
  GONC PS1
PS2 C=R3
  CD0EX
  CD1EX
  C+P+1
  CD1EX
  C=B A
  GOSBVL #2D28
  C=C-1 A
  GOSBVL #2426
  REL(3) CC
  REL(3) FIXP
  REL(3) CC
  REL(3) AA
  REL(3) CC
  REL(3) PSS
  REL(3) PSS
  REL(3) CC
  REL(3) GPS
FIXP GOVLNG #2A6E
AA GOSBVL #3FD9
  ?ST=0 0
  GOYES PX
PSERR GOVLNG #2E35
PX GOVLNG #3172
PSS GOSUB FIXP
  GOSBVL #36CD
  GOSBVL #379D
  GOSBVL #36CD
  GOVLNG #3A03
DC GOSUB TBL
  A=0 A
  A=DAT1 1
  A=A-1 A
  B=A A
DC1 C=DAT0 W
  A=A-1 A
  GOC DCE
  GOSUB DP
  GONC DC1
DCE A=R3
  AD0EX
  D0=D0- 2
  P=P+1
  LCASC ' '
  P=P+1
  GOSBVL #5423
  D1=D1+ 1
  A=B A
  GOVLNG #5470
GPS GOSBVL #369D
  LCHEX F2
  ?A#C B
  GOYES V5
  GOSBVL #2CEB
  CR3EX
  C=0 W
  LCHEX 111121
  CR3EX
V0 GOSUB V1
  P= 0
  C=R3
  C=C-1 A
  ?C=0 P
  GOYES V2
  ?ST=1 3
  GOYES V3
  GONC V4
V2 ?ST=0 3
  GOYES V3
V4 CSR W
  R3=C
  ?C=0 W
  GOYES V8
  LCHEX F1
  ?A#C B
  GOYES V6
  GOSBVL #2CEB
  GONC V0
V8 GOSBVL #2A7A
  GOC V6
V5 GOVLNG #2E2B
V1 CD1EX
  D1=C
  R2=C
  GOSBVL #3FD9
  RTNC
  ?ST=1 0
  GOYES V3
  ?XM=0
  GOYES V9
V3 C=R2
  D1=C
  ST=1 4
  GOVLNG #2E66
V6 GOTO PX
V9 RTNCC
GSUXE GOTO AFN1
GSUX GOSBVL #F178
  ?A#0 W
  GOYES GSUX1
  GOSUB GG0
  GOSUBL BU
  GONC GSUXE
  D1=D1- 5
  D1=D1- 12
  C=DAT1 B
  ST=C
  A=0 W
  ?ST=1 2
  GOYES GSUX2
  A=A+1 S
GSUX2 GOTO GNIX
GSUX1 ?A#0 S
  GOYES GSUX3
  GOTO GSUXZ
GSUX3 GOSUB GG0
  GOSUBL B2
  GONC GSUXE
  A=0 W
GSUX4 CD1EX
  ?C=D A
  GOYES GSUX5
  CD1EX
  A=A+1 W
  GOSUBL CN
  ?ST=0 1
  GOYES GSUX6
  GOSUBL NXA
  GONC GSUX4
GSUX6 GOSBVL #1B322
GSUX5 GOTO GNIX1
V9Y RTNCC
AFNF C=RSTK
AFN1 P= 0
  LCHEX 39
  GOTO BC
SBSU GOSBVL #F178
SBSU1 GOSBVL #1B223
  B=A W
  GOSUBL B2
  GONC AFNF
SBS1 CD1EX
  ?C=D A
  GOYES V9Y
  CD1EX
  B=B-1 A
  ?B=0 A
  RTNYES
  GOSUBL NXA
  GONC SBS1
GSU GOTO GSUX
GSUXZ GOSUB SBSU1
  GOC GGET
  GOSUB GG0
  C=0 W
  P= 0
  LCHEX 91
  A=C W
  ASRC
  GOTO GNIX
GGET CD1EX
  D1=(5) #2F6CF
  DAT1=C A
  GOSUB GG0
  A=DAT1 W
  GOSBVL #ED21
  A=0 M
  ASR W
  GOSUB GG2
  GOSUB GG1
  C=DAT1 B
  C=C+C B
  CD0EX
  GOSUB GS4
  GONC GNOR
  D0=D0- 10
  D1=D1+ 10
GNOR D0=D0- 14
  CD0EX
  CSRB
  R2=C
  C=R3
  GOSBVL #ED2C
  CD1EX
  CD0EX
  AD1EX
  R1=A
  GOSBVL #1A460
  D1=A
  A=R2
GB10 C=DAT0 B
  ?A=0 B
  GOYES GB20
  A=A-1 A
  GOSBVL #18504
  D0=D0+ 2
  GONC GB10
GB20 ST=1 0
  P= 0
  GOSBVL #181B7
  C=R3
  CD0EX
  GOSBVL #F23C
  A=DAT1 W
  GOSUB GSTO
  GOSUB GG1
  GOSUB GS4
  A=0 W
  GONC GNA
  C=0 W
  C=DAT1 B
  GOSUB CL
  D1=D1+ 2
  C=DAT1 B
  R1=C
  C=0 W
  D1=D1+ 2
  C=DAT1 6
  GOSUB GS11
  R2=C
  C=A W
  GOSBVL #ECB4
  GOSUB CL1
  R1=C
  C=R2
  SETHEX
  GOSUB GS31
  GOSBVL #1B322
  P= 0
  LCHEX 010
  A=A-C X
  LCHEX 01
  ?A#C X
  GOYES NZM
  P= 14
  LCHEX 8
  ?A=C P
  GOYES NZM
  LCHEX 2
  ?AC P
  GOYES NZM1
  LCHEX 5
NZM1 A=A-C P
NZM P= 0
  ?ST=0 2
  GOYES GNA
  A=A-1 S
GNA GOSUB GG2
  GOSUB GS1
  C=A W
  GOSBVL #13335
  C=0 W
  C=D B
  GOSUB CL
  C=B B
  GOSBVL #ED40
  A=A+C W
  P= 0
  LCHEX 08
  GOSUB GS2
  GOSUB GS1
  A=0 W
  R1=A
  GOSUB GS31
  P= 0
  LCHEX 04
  GOSUB GS2
  GOSUB GG1
  B=0 W
  B=B+1 S
  A=0 W
  C=DAT1 B
  C=C+C B
  GONC GNIC
  A=A+B S
GNIC B=B+B S
  GOSUBL CN
  ?ST=0 1
  GOYES GNIP
  A=A+B S
GNIP B=B+B S
  ?ST=0 0
  GOYES GNIX
  A=A+B S
GNIX ASRC
GNIX1 GOSUB GG2
  GOTO WES
GG1 A=DAT0 B
  P= 0
  LCHEX F1
  ?A=C B
  GOYES GG0
  C=RSTK
  GOTO WES
GG0 D0=D0+ 2
  GOSBVL #FA35
  GOSBVL #F186
  CD1EX
  D1=C
  GOSBVL #ED3D
  CD0EX
  R3=C
  GOSBVL #F7B0
  D1=(5) #2F6CF
  C=DAT1 A
  CD1EX
  C=0 W
  RTN
GS2 C=0 XS
  GOSBVL #1B322
  A=A-C X
GG2 SETHEX
  C=R3
  CD0EX
  GOSBVL #ED2C
  CD1EX
  C=A W
  GOSBVL #F216
GSTO GOVLNG #F5F8
GS1 GOSUB GG1
  C=DAT1 12
  GOSUB CS1
GS11 GOVLNG #13229
GS31 GOSBVL #13252
  GOSUB CL1
GS3 GOSUB CL
  C=B B
  GOSUB CL
  C=D B
  A=C W
  RTN
GS4 D1=D1+ 13
  C=DAT1 A
  D1=D1+ 1
  ST=C
  ?ST=1 3
  RTNYES
  RTN
CL1 C=R1
CL CSL W
  CSL W
  C=A B
  RTN
  REL(5) DC
  REL(5) PS
WCC C=0 A
  C=DAT0 1
  D0=D0+ 1
  B=C A
  GOSBVL #2426
  REL(3) EXT
  REL(3) ALA
  REL(3) AG
  REL(3) W5V
  REL(3) ACV
  REL(3) W4V
  REL(3) W2
  REL(3) W1
  REL(3) ALA
  REL(3) GSU
W5V GOLONG W5
ACV GOLONG AC
W4V GOLONG W4
ALA GOSUB BU
  GONC ALAE
  D1=D1- 12
  D1=D1- 5
  C=DAT1 A
  ST=C
  ST=0 2
  B=B-1 A
  ?B=0 P
  GOYES ALA1
  ST=1 2
ALA1 C=ST
  DAT1=C A
ALAE GOTO COE
IV P= 0
  LCHEX 36
  GOTO BC
AG GOSUB SBSU
  GONC IV
  GOSUB AGO
WES GOTO WEX
BU SETHEX
  P= 0
  LCASC '   STPPA'
  D1=(5) #2F558
  A=DAT1 A
  AD1EX
BU1 A=DAT1 W
  ?A=C W
  GOYES BUE
  ?A#0 B
  GOYES BU2
BUX RTNCC
BU2 D1=D1+ 16
  D1=D1+ 16
  A=DAT1 A
  CD1EX
  C=C+A A
  CD1EX
  GONC BU1
BUE D1=D1+ 16
  A=0 A
  A=DAT1 4
  LC(5) ATYP
  ?A=C A
  GOYES BUE1
  C=0 A
  RTN
BUE1 D1=D1+ 16
  A=DAT1 A
  D1=D1+ 5
  C=0 A
  RTNSC
ST D0=(5) #2F8C5
  C=DAT0 W
  CR2EX
  DAT0=C W
  D0=D0+ 16
ST1 C=DAT0 W
```

```
  CR1EX
  DAT0=C W
  D0=D0+ 16
  C=DAT0 W
  CR0EX
  DAT0=C W
  RTN
INP GOSBVL #E90C
  D1=D1+ 16
  RTN
HMSB GOSBVL #F186
  GOSUB INP
  C=0 A
  LCHEX 4
  GOSUB RJA
  GOSUB CS
  ACEX W
  GOSBVL #13274
  R3=A
  RTN
ARJ SETDEC
  A=A+C X
RJU GOVLNG #12AE2
RJA GOSUB ARJ
RJA2 ACEX W
RJA1 B=0 W
  D=0 W
  D=C B
  RTN
CS GOSUB CS1
CS2 B=C B
CS1 CSR W
  CSR W
  RTN
W2 GOSUB HMSB
  CD1EX
  R2=C
  GOSUB SNA3
  GOSUB CS1
  CSRB
  A=R2
  AD1EX
  A=DAT1 W
  R2=A
  GOSUB JPHR
  GOTO PL2
W1 GOSUB HMSB
  C=DAT1 A
  A=0 W
  C=C+1 P
  GOC DATBL
  GOSUB INP
DATBL R1=A
  C=DAT1 W
  R2=C
  GOSUB JPH
PL2 SETHEX
  C=DAT1 W
  C=C+1 P
  GOC TEXT
  D1=D1+ 16
TEXT GOSBVL #BD31
  SETHEX
  C=R3
  B=A A
  BSRB
  B=B+C W
  R0=A
  CD1EX
  R1=C
  C=C+A A
  CD1EX
  C=0 A
  LCHEX BE
  ?CA A
  GOYES TOK
  LCHEX 25
BC GOVLNG #9393
TOK D1=D1+ 14
  C=0 W
  P= 12
  C=DAT1 P
  C=C+B W
  R3=C
EXT GOSUB ST
  GOSUB BU
  GOC TOK1
  R2=C
  ?C#0 A
  GOYES NNV
  LCHEX 3F
  GONC BC
NNV C=0 A
  LCHEX 25
  D=0 S
  GOSBVL #1AF01
  CON(5) #84C4
  GONC CRF1
TFE GOVLNG #944D
CRF1 C=R1
  D1=C
  C=R2
  DAT1=C W
  D1=D1+ 16
  LC(5) ATYP
  DAT1=C A
TOK1 GOSUB BU
  AD1EX
  D1=A
  D1=D1- 16
  D1=D1- 16
  D1=D1- 5
  B=0 W
  C=R3
  C=C+C B
  B=C B
  CD1EX
  GOSBVL #1AF01
  CON(5) #133C
  A=R0
  GOSUB ST
  C=A A
  CR0EX
  B=0 W
  B=C B
  AD1EX
  C=R3
  DAT1=C 14
  D1=D1+ 14
  C=R2
  ?C=0 W
  GOYES CO5
  DAT1=C 10
  D1=D1+ 10
CO5 A=R1
  CD1EX
  GOSBVL #1308
  C=R0
  CD1EX
  GOSUB POS
COE GOSUB P1
WEX GOLONG W4E
POS CD1EX
  D0=C
  GOSUB B2
  C=DAT0 M
  CSR M
  CSR M
  B=C M
POS1 CD1EX
  D1=C
  ?C=D A
  GOYES POS2
  C=DAT1 M
  CSR M
  CSR M
  ?CB M
  GOYES POS2
  GOSUB NXA
  GONC POS1
NXA C=0 A
  C=DAT1 B
  C=C+C B
  AD1EX
  A=A+C A
  AD1EX
  RTNCC
POS2 A=0 W
  A=DAT0 B
  A=A+A B
  CD1EX
  D1=C
  C=C-A A
  GOSBVL #ED3D
  C=A B
  D=C W
  AD0EX
  CD1EX
  ?CA A
  GOYES POS3
  B=C A
  B=B-A A
  B=B-1 A
  GOSUB SHI1
  C=A A
  GOSUB SHI
  C=A A
  A=A+1 A
  GOTO POS4
POS3 ACEX A
  C=C+D A
  ACEX A
  B=A A
  B=B-C A
  B=B-1 A
  D1=C
  GOSUB SHI
  C=A A
  GOSUB SHI1
  CD1EX
  A=C A
  C=C+1 A
POS4 GOSBVL #1B41B
  D0=C
  GOSUB SHI
  C=DAT0 S
  GOSBVL #1308
  GOSBVL #1B432
  D0=C
  GOSBVL #1B418
  DAT0=C S
  D=D-1 B
  ?D#0 B
  GOYES POS4
  D1=A
  ?CA A
  RTNYES
  C=D W
  GOSBVL #ED2C
  D1=C
  RTN
SHI1 C=C-1 A
SHI GOVLNG #1B435
JPH A=R1
  GOSUB RJU
  ?A=0 B
  GOYES JU
  ACEX W
  D=C W
  GOSUB G
  C=D W
  A=C W
JU D0=(5) #2F8C5
  DAT0=A A
  A=R1
  C=0 W
  P= 0
  LCHEX 8
  GOSUB ARJ
  SETHEX
  ?A#0 A
  GOYES X1
  CD1EX
  RSTK=C
  GOSUB SNA3
  D0=(5) #2F8C5
  GOSUB CS1
  CSRB
  GOSUB GS11
  C=RSTK
  CD1EX
  GOTO X2
X1 GOSUB RJA2
  P= 3
  A=0 W
  A=C WP
  P= 0
  GOSBVL #ED2F
  GOSUB CS2
  D=C B
  GOSBVL #13304
X2 C=A W
  D=C W
  GOSUB KEIN
  GOSUB MAL
JPHR A=R3
  C=A+C W
  CSL W
  CSL W
  P= 0
  LCHEX 7
  R3=C
  C=0 W
  CR2EX
  A=C W
  C=C+1 P
  GONC X8
X81 RTNCC
X8 ?A=0 W
  GOYES X81
  CR3EX
  P= 8
  ?A=0 S
  GOYES X4
  P= 12
X4 CPEX 13
  LCHEX C
  CR3EX
  LCHEX 010
  GOSUB ARJ
  SETHEX
  C=A W
  GOSBVL #1B42C
  B=0 W
  GOSUB CS2
  BSL W
  BSL W
  B=B+C B
  ?C=0 B
  GOYES JD8
  D=C A
  GOSUB G
  C=D A
  B=C B
JD8 C=B W
  R2=C
  C=A W
  GOSUB RJA1
  GOSUB CS
  A=0 W
  ACEX B
  CSR W
  C=0 M
  SETDEC
  C=C+C A
  A=A+C A
  CSR A
  C=C+C A
  A=A+C A
  GOSBVL #13274
  GOSBVL #ED40
  A=R2
  C=C+A W
  R2=C
  RTNCC
MAL C=0 W
  P= 1
  SETHEX
  LCHEX 1518
  GOVLNG #ECBB
KEIN A=DAT0 B
  ?A=0 B
  GOYES KX
KEIX P= 0
  LCHEX 9
  ?A#C P
  GOYES K1
K2 D=D-1 W
  GOSUB D7B
  P= 0
  ?C#0 P
  GOYES K2U
  LCHEX 7
K2U A=DAT0 B
  ASR A
  GOSUB WG
  ?A#C P
  GOYES K2
KX GOTO KF
K1 ?A#0 P
  GOYES K3
K4 GOSUB D7A
  P= 0
  LCHEX 3
  ?B#C P
  GOYES K4
  D=D-1 W
K3 GOSUB D7A
  P= 0
  ?C#0 P
  GOYES K5
  LCHEX 7
K5 A=DAT0 B
  ASR A
  GOSUB WG
  ?A#C P
  GOYES K3
  A=DAT0 B
  LCHEX 8
  ?A=C P
  GOYES KF
  ?A=0 P
  GOYES KF
K7 C=D W
K9 R0=C
  GOSBVL #13335
  GOSUB KF
  SETDEC
  A=A-1 W
  GOSUB DIV7
  SETHEX
  A=A+1 W
  C=DAT0 B
  P= 0
  ?C=A P
  GOYES K8
  A=R0
  GOSUB LC7
  C=C+A W
  GONC K9
K8 C=R0
  D=C W
KF C=D W
  A=C W
  RTN
WG ST=C
  LCHEX 8
  ?A=C P
  GOYES WG1
  C=ST
  RTN
WG1 C=ST
  ACEX P
  ST=C
  LCHEX 1
  ?A=C P
  GOYES WGEN
  C=C+1 A
  ?A=C P
  GOYES WGEN
  C=A P
  RTN
WGEN C=ST
  ACEX P
  RTN
LC7 C=0 W
  P= 0
  LCHEX 7
  RTN
D7A D=D+1 W
D7B GOSUB KF
  A=A+1 W
DIV7 GOSUB LC7
  GOVLNG #EC7B
G P= 0
  SETDEC
  DSRC
  ?D=0 P
  GOYES GR
  LCHEX 8
  ?D#C P
  GOYES GH
  DSLC
* C=C+1 A
  ?D=C P
  RTNYES
* LCHEX 6
* ?D=C P
* RTNYES
  ?D=0 P
  RTNYES
GR C=RSTK
  GOVLNG #E920
GH ?DC P
  GOYES GR
  LCHEX 5
  ?DC P
  GOYES GH1
  LCHEX 2
GH1 D=D+C P
  DSLC
  LCHEX 6
  ?D=C P
  GOYES GR
  C=C+1 A
  ?D=C P
  GOYES GR
  RTN
AGO C=DAT1 W
  CD1EX
  R0=C
  CD1EX
  GOSBVL #1B438
  ST=C
  ?ST=0 5
  RTNYES
  ?ST=1 7
  GOYES ARE
  C=DAT1 A
  ST=C
  ST=1 7
  C=ST
  DAT1=C A
  GOTO AFI
ARE ST=0 5
  C=ST
  GOSBVL #1B41E
  DAT1=C W
  CD1EX
  RSTK=C
  GOSUB SNA3
  GOSUB CS1
  B=C W
  BSRB
  C=RSTK
  D1=C
  D0=C
  D0=D0+ 14
  A=B W
  R1=A
NEURE C=0 W
  C=DAT1 12
  GOSUB CS1
  D=C W
  C=0 W
  C=DAT0 10
  GOSBVL #ED2F
  C=C+D W
  GOSBVL #13229
  D1=D1+ 2
  DAT1=C 10
  D1=D1- 2
  C=A W
  GOSBVL #13335
  SETDEC
  C=0 W
  D0=D0+ 2
  C=DAT0 B
  D0=D0- 2
  B=B+C W
  P= 0
  LCHEX 12
MTE ?B B
  GOYES MOK
  B=B-C B
  A=A+1 A
  GOTO MTE
MOK GOSBVL #13304
  D=C W
  A=DAT0 B
  ?A=0 B
  GOYES ND
  GOSUB KEIX
ND GOSUB KF
  GOSUB MAL
```

```
MIKO D1=D1+ 2
  C=0 W
  C=DAT1 10
  C=C+A W
  DAT1=C 10
  D1=D1- 2
  A=R1
  ?A W
  GOYES NDOK
  A=DAT0 B
  P= 0
  LCHEX 9
  ?A=C P
  GOYES NDER
  GOTO NEURE
NDER LCHEX 93A80
  A=0 W
  A=C A
  GOTO MIKO
NDOK CD1EX
  RSTK=C
  GOSUB SNA3
  B=C W
  C=RSTK
  RSTK=C
  CD1EX
  C=0 W
  D1=D1+ 2
  C=DAT1 10
  GOSUB SNA
  C=RSTK
  CD1EX
  GOSUB POS
AFI CD1EX
  R0=C
  GOSUB B2
  R1=C
AF1 CD1EX
  D1=C
  A=R1
  ?C=A A
  GOYES AF3
  C=0 W
  C=DAT1 14
  GOSBVL #1B438
  ST=C
  ?ST=0 5
  GOYES AF3
  C=DAT1 A
  C=C+C B
  GONC AF2
  GOSUB NXA
  GONC AF1
AF3 P= 0
  LCHEX C4
  GOSBVL #13601
AF2 CD1EX
  A=R1
  ?CA A
  GOYES AF21
  C=R0
AF21 A=R0
  CD1EX
  AD0EX
  RTN
SNA D0=(5) #2F755
  A=0 W
  A=DAT0 12
  CSL W
  CSL W
  C=C+C W
  ?A W
  GOYES SNA1
  ACEX W
SNA1 ?AB W
  GOYES SNA2
  ACEX W
SNA2 DAT0=A 12
SNA3 GOVLNG #125B2
B2 C=RSTK
  D=C A
  GOSUB BUL
  C=D A
  RSTK=C
  RTNNC
  D1=D1- 5
B3 CD1EX
  D1=C
  C=C+A A
  D1=D1+ 5
  D=C A
  RTNSC
PH ABEX A
  ?A=0 B
  GOYES C3
  GOSBVL #23E3
  CON(2) #F9
  REL(3) P1
  CON(2) #FC
  REL(3) Y7
  CON(2) #FE
  REL(3) Y7
  CON(2) #FD
  REL(3) DS
  CON(2) #2D
  REL(3) FTY
  CON(2) #2E
  REL(3) LI
  CON(2) 0
  ABEX A
  RTNSXM
C3 C=R3
  D1=C
  A=R2
  D1=D1- 14
  CD1EX
  ?AC A
  GOYES C6
  D1=C
  R3=C
  LCASC ' ALM:1D'
  DAT1=C 14
C6 RTNSXM
LI ABEX A
  LC(5) ATYP
  ?A=C A
  GOYES LI2
  RTNSXM
LI2 D1=D1+ 16
  D1=D1+ 16
  CD1EX
  B=C A
  GOSUBL W5RA1
  C=B A
  D1=C
  A=DAT1 A
  GOSUB B3
  GOSBVL #ED3D
  CD1EX
  R3=C
  A=0 W
  R1=A
  A=A-1 A
  R2=A
  A=DAT0 B
  LCHEX F1
  B=C A
  ?A#C B
  GOYES LI7
  D0=D0+ 1
  A=DAT0 A
  ASR A
  R1=A
  DQ=D0+ 5
  C=DAT0 B
  ?B#C B
  GOYES LI8
  D0=D0+ 2
  A=DAT0 4
LI8 GOSBVL #1B2D2
  R2=A
LI7 A=R1
  GOSBVL #1B2D2
  C=R2
  GOSBVL #ED3D
  C=A A
  GOSBVL #ED3D
  R2=C
  C=R3
  CD1EX
LI3 CD1EX
  A=R3
  GOSBVL #ED0A
  ?C=A A
  GOYES ZZE
  CD1EX
  C=R3
  CD1EX
  D1=C
  R3=C
  C=R2
  C=C+1 A
  R2=C
  A=C A
  GOSBVL #ED2C
  B=C A
  GOSBVL #ED2C
  ?A A
  GOYES LI4
  ?AC A
  GOYES ZZE
  GOSUB ZEIG
  GOSUB ZZEIG
  GOSUB CN
  ?ST=0 3
  GOYES LI4
  D1=D1+ 14
  GOSUBL JHI
  GOSUB ZZEIG
LI4 GOSUB NXA
  GONC LI3
ZZEIG GOSUBL CRLF
  GOSBVL #76AD
  C=R3
  CD1EX
  RTNC
  C=RSTK
ZZE GOSUBL W5RA
  XM=0
  RTNCC
FTY ABEX A
  LC(5) ATYP
  ?A=C A
  GOYES FTY1
  RTNSXM
FTY1 GOSUB FTY2
  BSS 3
  CON(2) 5
  NIBASC 'APPT '
  CON(1) 1
  CON(4) ATYP
  NIBHEX FF
FTY2 C=RSTK
  CD1EX
  A=0 S
  A=A+1 S
  RTNCC
ST2 D0=(5) #2F901
  GOLONG ST1
P1 GOSUB BUL
  GOC P1J
P1E RTNSXM
P1J D1=D1- 5
  D1=D1- 12
  C=DAT1 A
  ST=C
  ?ST=1 2
  GOYES P1E
  GOSUB ST2
  D0=(2) #41
  P= 1
C5 C=RSTK
  DAT0=C A
  D0=D0+ 16
  P=P-1
  GONC C5
  C=D A
  DAT0=C A
LBA GOSUB SNA3
  B=C W
  BSR W
  BSR W
  BSRB
  GOSUB B2
LB0 CD1EX
  D1=C
  ?C=D A
  GOYES PE
  GOSUB CN
  ?ST=0 1
  GOYES LB2X
  GOSUB NXA
  GONC LB0
LB2X C=0 W
  C=DAT1 12
  CSR W
  CSR W
  ?B W
  GOYES LB3X
  GOSUB SAN
  GOLONG W8
SAN LCHEX C4
SAN1 GOVLNG #135FA
W8Z D1=D1+ 13
  C=DAT1 A
  ST=C
  ST=1 0
  ST=1 1
  C=ST
  DAT1=C A
  D1=D1- 13
  ?ST=0 3
  GOYES LB5X
  ?ST=1 2
  GOYES LB5X
  GOSUB AGO
LB5X GOTO LBA
LB3X BSL W
  BSL W
  B=B+B W
  GOSUB SNA
PE GOSUB DSX
  GOSUB ST2
  D0=(2) #61
  C=DAT0 A
  D=C A
  P= 1
C4 D0=D0- 16
  C=DAT0 A
  RSTK=C
  P=P-1
  GONC C4
  RTNSXM
W4E GOTO WEND
W4 GOSUB WIPE
  GOSUB B2
  GONC W4E
PRA1 CD1EX
  ?C=D A
  GOYES W4E
PRN CD1EX
  A=0 A
  A=DAT1 B
  GOSUB CN
  ?ST=1 0
  GOYES PRAG
  GOSUB NXL
  GONC PRA1
PRAG D1=D1+ 13
  ST=0 0
  C=ST
  DAT1=C P
  D1=D1+ 1
  A=A+A B
  AD0EX
  D0=D0- 14
  ?ST=0 3
  GOYES PRANR
  D1=D1+ 10
  D0=D0- 10
PRANR A=DAT1 B
  LCASC '|'
  ?A#C B
  GOYES PRA3
  D1=D1+ 2
  D0=D0- 2
  A=0 W
  AD0EX
  ASRB
  P= 2
  GOSBVL #9723
  GOSBVL #2296
  GOSUB VNT
  LCHEX 20000
PRA4 C=C-1 A
  GOC PRA5
  ?ST=0 12
  GOYES PRA4
PRA5 GOSUB IRT
PRAA GOTO W4E
PRA3 AD1EX
  CD0EX
  B=C A
  D0=(5) #2F480
  CD0EX
  GOSBVL #1308
  CD0EX
  A=DAT0 B
  LCASC ''
  ?A#C B
  GOYES PRA6
  LCASC ' '
  DAT0=C B
PRA6 GOSBVL #14C8A
  GOVLNG #2620
VNT GOSUB VNT1
  INTOFF
  ST=1 12
  C=0 A
I1 C=C+1 X
  GONC I1
  INTON
  GOVLNG #DB
VNT1 ST=0 12
  D0=(5) #2F43C
  C=DAT0 A
  D0=(4) #F971
  DAT0=C A
  C=RSTK
  GOTO IRT1
IRT D0=(5) #2F971
  C=DAT0 A
  ST=1 12
IRT1 D0=(4) #F43C
  DAT0=C A
  P= 0
  RTN
CN2 C=R3
CN3 CD1EX
CN D1=D1+ 13
  C=DAT1 A
  D1=D1- 13
  ST=C
  RTN
DS C=D A
  R2=C
DSX GOSUB B2
  GONC DSE
DS1 CD1EX
  ?C=D A
  GOYES DSE1
  CD1EX
  C=DAT1 A
  ST=C
  ?ST=1 7
  GOYES DS2
  GOSUB CN
  ?ST=1 1
  GOYES DS3
DS2 GOSUB NXL
  GONC DS1
DSE1 LCHEX C4
  GOSBVL #13601
  GONC DSE
DS3 GOSUB SAN
DSE C=R2
  D=C A
Y7E RTNSXM
Y7 C=D A
  R2=C
  R3=A
  GOSUB P1
  GOSUB BUL
  GONC DSE
  D1=D1- 5
  D1=D1- 12
  C=DAT1 B
  ST=C
  ?ST=1 2
  GOYES DSE
  GOSUB B2
Y71 CD1EX
  ?C=D A
  GOYES P
  GOSUB CN3
  ?ST=1 0
  GOYES Y72
  GOSUB NXL
  GONC Y71
Y72 AR3EX
  A=A+1 A
  A=A+1 B
  GOC UA
  A=0 W
  LCHEX 35
  A=C B
  GOSBVL #12917
P GOTO DSE
UA C=RSTK
  R3=C
  LCHEX CD
  GOSBVL #13601
  LCHEX CE
  GOSUB SAN1
  C=0 A
  LCHEX E
  B=C A
  LCHEX 809
  GOSBVL #1197D
  GONC UA1
  LCHEX
0D4358454D4C41
  DAT1=C 14
UA1 C=R3
  RSTK=C
  GOTO P
CLEAR GOSUB BUL
  D1=D1- 16
  D1=D1- 16
  D1=D1- 5
  A=R3
  D0=A
  C=0 A
  C=DAT0 B
  C=C+C B
  A=A+C A
  C=-C A
  B=C A
  CD1EX
  RTN
ACFN GOLONG AFN1
AC GOSBVL #F186
  CD1EX
  R0=C
  R1=A
  GOSUB B2
  GONC ACFN
  R3=C
  CD1EX
  R2=C
  C=R0
  CD1EX
  C=DAT1 S
  A=R1
  C=C+1 S
  GOC AY
  GOTO AM
NX1 C=R3
NX2 CD1EX
NXL GOLONG NXA
BUL GOLONG BU
AY GOSBVL #BD31
  R1=A
  CD1EX
  R0=C
  C=R3
  D=C A
  C=R2
  CD1EX
AD C=R0
  CD0EX
  CD1EX
  R3=C
  D1=C
  ?C=D A
  GOYES Q
  GOSUB CN
  D1=D1+ 14
  ?ST=0 3
  GOYES AD1
  D1=D1+ 10
AD1 C=R1
  AD0EX
  A=A+C A
  AD0EX
```

```
   AD1EX
   A=A+C A
   AD1EX
   GOSBVL #1B1C7
   GOC AK
NES GOSUB CLEAR
   GOSBVL #1AF01
   CON(5) #133C
   GOSUB AFI
   GOTO PRAA
AK GOSUB NX1
   GONC AD
Q P= 0
   LCHEX  0036
   P= 9
   GOSBVL #93BC
WEND GOVLNG #8A48
AM GOSBVL #1B223
   B=A A
   C=R3
   D=C A
   C=R2
   CD1EX
AM1 CD1EX
   R3=C
   ?C=D A
   GOYES Q
   GOSUB NX2
   B=B-1 A
   ?B#0 A
   GOYES AM1
   GONC NES
TIMH D0=(5) #2F8C5
   P= 0
   LCASC '00.00.00'
   GOSUB XSPC
   LCASC ':00:00'
XSPC DAT0=C W
   D0=D0+ 16
SPC LCASC ' '
   DAT0=C W
   D0=D0+ 2
   RTN
U4 D0=(5) #2F8E5
U3 CDEX W
   GOSUB U2
   C=B A
   GOSUB U2
   C=A A
   GOSUB U2
   CDEX W
   RTN
U2 DAT0=C 1
   D0=D0- 2
   CSR W
   DAT0=C 1
   D0=D0- 4
   RTN
WIPE P= 0
   D1=(5) #2F480
   C=0 A
   LCHEX D8
   GOVLNG #1B0AF
ZEIG GOSUB TIMH
   GOSUB CN2
   ?ST=0 0
   GOYES ZEI1
   LCASC 'x'
   D0=D0- 2
   DAT0=C B
   D0=D0+ 2
ZEI1 LCASC '  '
   ?ST=0 1
   GOYES ZEI2
   LCASC 'P'
   A=DAT1 B
   A=A+A B
   GONC ZEI2
   LCASC 'a'
ZEI2 DAT0=C A
   D0=D0+ 4
   C=0 A
   P= 3
   CPEX 3
   D1=D1+ 12
   C=DAT1 XS
   LCASC 'N'
   ?ST=0 3
   GOYES ZEI4
   LCASC 'R'
   ?ST=0 2
   GOYES ZEI4
   LCASC 'A'
ZEI4 DAT0=C A
   D0=D0+ 4
   C=0 A
   GOSUB SPC
   C=R3
   CD1EX
   C=0 W
   D1=D1+ 2
   C=DAT1 10
   GOSBVL #130E5
   GOSUB U4
   GOSBVL #1B418
   A=C B
   GOSUBL CS
   D=C B
   GOSUB RNSU
   C=R3
   CD1EX
   C=0 A
   C=DAT1 B
   C=C+C B
   CD0EX
   D1=D1+ 13
   D0=D0- 14
   C=DAT1 A
   ST=C
   D1=D1+ 1
   ?ST=0 3
   GOYES ZEI5
   D1=D1+ 10
   D0=D0- 10
ZEI5 A=0 W
   AD0EX
   ASRB
   GOTO ZSP
SE1? CD1EX
SE? C=R2
   GOSBVL #ED2C
   D=C W
   CD1EX
   D1=C
   ?C A
   RTNYES
   RTN
W5RA1 P= 0
   LCHEX 4
   D1=(5) #2F6C1
   DAT1=C 1
W5RA GOSUBL BU
   RTNNC
   D1=D1- 12
   D1=D1- 5
   C=DAT1 S
   CD1EX
   D1=(5) #2F6C1
   A=DAT1 S
   DAT1=C S
   CD1EX
   DAT1=A S
   RTNSC
QUIT GOSUB W5RA
   GOSBVL #14C8A
WOUT GOLONG COE
W5 P= 0
   GOSBVL #18534
   GOSUB W5RA1
   GONC QUIT
   GOSUBL B2
   CD1EX
   R3=C
   ?C=D A
   GOYES QUIT
   CDEX A
   GOSBVL #ED3D
   CDEX A
   R2=C
   CD1EX
W5B GOSUB SE?
   GONC W51
   AD1EX
   D1=A
   GOSUB CN
   ?ST=0 1
   GOYES W51
   R3=A
   GOSUB NXL
   GONC W5B
W51 GOSUB ZEIG
W52 GOSUB CLN
W53 GOSBVL #CF7
   GOSBVL #152BA
   GONC TA1
   LCHEX 32
   ?B=C B
   GOYES TA2
   C=C+1 A
   ?B=C B
   GOYES TA2
TA1 GOSBVL #212E
   GOSBVL #10EE
   GOC W53
TA2 A=B A
   ST=0 0
   GOSBVL #23E3
   CON(2) 51
   REL(3) DWN
   CON(2) 163
   REL(3) FDWN
   CON(2) 50
   REL(3) UP
   CON(2) 162
   REL(3) FUP
   CON(2) 99
   REL(3) QUIT
   CON(2) 43
   REL(3) QUIT
   CON(2) 5
   REL(3) TIM
   CON(2) 4
   REL(3) REPI
   CON(2) 86
   REL(3) XRU
   CON(2) 91
   REL(3) DEL
   CON(2) 71
   REL(3) AGOW
   CON(2) 73
   REL(3) RST
   CON(2) 46
   REL(3) RUN
   CON(2) 0
DWN1 GOTO W51
FDWN ST=1 0
DWN GOSUB NX1
   GOSUB SE?
   GONC DWN1
   CD1EX
   R3=C
   ?ST=0 0
   GOYES DWN1
   GONC FDWN
UP C=R2
   A=R3
   B=A A
UP1 ?C=B A
   GOYES DWN1
   R3=C
   GOSUB NX2
   CD1EX
   GONC UP1
FUP C=R2
   R3=C
FUP1 GOTO W51
TIM GOSUB TIMH
   C=0 W
   DAT0=C W
   GOSBVL #130DB
   GOSUB U4
   GOSBVL #1B418
   A=C B
   GOSUBL CS
   D=C B
   GOSUB U3
   P= 0
   D1=(5) #2F8C5
   GOSBVL #15147
TIM1 GOTO W51
REPI GOSUB CN2
   D1=D1+ 14
   ?ST=0 3
   GOYES TIM1
   GOSUB JHI
   GOTO W52
JHI P= 0
   D0=(5) #2F8C5
   LCASC '0+00    '
   DAT0=C W
   D0=D0+ 16
   LCASC ':00:00+0'
   DAT0=C W
   D0=D0+ 16
   LCHEX 003030
   DAT0=C 6
   A=0 W
   A=DAT1 B
   ?A#0 B
   GOYES RN3
   GOTO RN2
RN3 ASR W
   GOSUB RN1
   NIBASC 'SaSoMoDi'
   NIBASC 'MiDoFrWe'
RN1 C=RSTK
   A=A-1 A
   A=A+A A
   A=A+A A
   A=A+C A
   AD0EX
   C=DAT0 A
   D0=(5) #2F8C5
   DAT0=C A
   A=0 A
   A=DAT1 1
   GOSUB RN4
   NIBASC ' 12345+-'
RN4 LCHEX 6
   ?A P
   GOYES RN5
   A=A-1 A
   A=A-1 A
RN5 A=A+A A
   C=RSTK
   C=C+A A
   CD0EX
   C=DAT0 B
   D0=(5) #2F8C9
   DAT0=C B
RN2 D0=(5) #2F8CF
   D1=D1+ 2
   C=DAT1 A
   GOSUB U2
   D1=D1+ 2
   C=0 W
   C=DAT1 6
   SETHEX
   GOSBVL #13229
   D=C W
   GOSBVL #ECAF
   SETHEX
   D0=(5) #2F8D5
   GOSUB U2
   CDEX W
   GOSBVL #13252
   D0=(5) #2F8E7
RNSU GOSUB U3
   P= 0
   D1=(5) #2F8C5
   A=0 A
ZSP2 D1=D1+ 2
   A=A+1 A
   C=DAT1 B
   ?C#0 B
   GOYES ZSP2
   D1=(2) #C5
ZSP D0=A
   P= 0
   GOSBVL #18542
   AD0EX
   ST=1 4
   C=R2
   R0=C
   C=RSTK
   CD0EX
   GOSBVL #17E15
   C=R0
   R2=C
   CD0EX
   RSTK=C
   RTN
CLN D0=(5) #2F478
   C=DAT0 A
   CSTEX
   ST=1 9
   CSTEX
   DAT0=C A
CRLF C=R2
   CD0EX
   GOSBVL #17DC1
   C=R2
   CD0EX
   R2=C
   RTN
XRU GOSUB CN2
   D1=D1+ 13
   ?ST=0 0
   GOYES XRU1
   ST=0 0
   GONC XRU2
XRU1 ST=1 0
XRU2 C=ST
   DAT1=C A
   GOTO W51
DEL C=R2
   R4=C
   GOSUB CLEAR
   GOSBVL #1AF01
   CON(5) #133C
   GOSUBL AFI
   C=R2
   CD1EX
   D1=D1+ 16
   D1=D1+ 16
   C=DAT1 A
   AD1EX
   D1=A
   C=C+A A
   GOSBVL #ED3D
   D1=D1+ 5
   CD1EX
   A=R4
   C=A S
   R2=C
   GOSUB SE1?
   GOC DEL1
   GOTO QUIT
DEL1 C=R3
   GOSUB SE1?
   GOC RST2
   C=R2
   R3=C
   GOTO FDWN
AGOW GOSUB CN2
   ?ST=0 1
   GOYES RST2
   GOSUBL AGO
   GOSUB SE?
   GONC RST2
   CD1EX
   GOC RST3
RST GOSUB CN2
   D1=D1+ 13
   ?ST=0 3
   GOYES RST2
   ST=1 1
   C=ST
   DAT1=C A
   D1=D1- 13
   GOSUBL AGO
   CD0EX
RST3 R3=C
RST2 GOTO W51
RUN ?ST=1 13
   GOYES RST2
   GOSUB CN2
   ?ST#1 0
   GOYES RST2
   GOSUB W5RA
   GOSUB WIPE
   C=R3
   GOLONG PRN
W8 D1=D1+ 12
   P= 0
   A=0 A
   A=DAT1 P
   D1=D1- 12
   LCHEX 9
   ?CA P
   GOYES OUT
   R1=A
   GOSUBL VNT
W81 C=R1
   GOSBVL #2426
   REL(3) OUT
   REL(3) T1
   REL(3) T2
   REL(3) T3
   REL(3) T4
   REL(3) T5
   REL(3) T6
   REL(3) T2
   REL(3) T4
   REL(3) T6
T1 GOSBVL #EC5A
OUT GOSUBL IRT
   GOLONG W8Z
T2 LCHEX 100
   D=C A
   LCHEX 800
   GOSUB TUT
   GOTO WT
T31 P= 0
   LCHEX 400
   GOSUB T33
   P= 0
   LCHEX 300
T33 D=C A
   LCHEX 200
TUT GOSUB TUTE
   RTNC
   C=0 M
   D=0 M
   GOVLNG #EB40
T3 GOSUB T31
   GOSUB T31
   GOSUB T31
T32 GOTO OUT
T4 LCHEX 40
T41 ST=C
   GOSBVL #EC5A
   GOSUB TUTE
   GOC T32
   C=ST
   C=C-1 B
   GONC T41
   GOTO WT
T5 LCHEX 0B0
   D=C A
   LCHEX E00
   GOSUB TUT
   P= 0
   LCHEX A00
   D=C A
   LCHEX E00
   GOSUB TUT
T59 GOTO OUT
T6 LCHEX 4
T61 ST=C
   C=0 W
   LCHEX 60
   R0=C
   ST=0 11
T62 B=0 W
   B=B-1 P
   C=R0
   GOSBVL #EBEB
   P= 0
   GOSUB TUTE
   GOC T59
   A=R0
   ?ST=1 11
   GOYES T63
   A=A-1 A
   LCHEX 10
   GONC T64
T63 A=A+1 A
   LCHEX 60
T64 R0=A
   ?A#C B
   GOYES T62
   ?ST=1 11
   GOYES T65
   ST=1 11
```

```
  GONC T62
T65 C=ST
  C=C-1 P
  GONC T61
WT P= 0
  A=R1
  LCHEX 10007
  ?A P
  GOYES T59
WT4 GOSUB TUTE
  GOC T59
  C=C-1 A
  GONC WT4
  GOTO W81
TUTE ACEX W
  D0=(5) #2E3FF
  C=DAT0 A
  CSTEX
  ?ST=1 2
  GOYES TUTE1
TUTE1 CSTEX
  ACEX W
  RTNC
  ?ST=1 12
  RTNYES
  RTN
  LEX 'ALMCCD1'
  ID #E1
  MSG 0
  POLL PH
  ENTRY WCC
  CHAR #D
  KEY 'ALM'
  TOKEN #F0
  ENDTXT
ATYP EQU #E220
TBL GOSUB TBL1
  NIBASC 'ONACKCAT'
  NIBASC 'CLREXCRE'
  NIBASC 'LSETOFF'
  NIBASC 'GET'
TBL1 C=RSTK
  CD0EX
  R3=C
  P= 3
CC RTNCC
DP CD0EX
  C+P+1
  CD0EX
  P= 5
  RTNCC
PS GOSBVL #3FB4
  GOSUB TBL
  B=0 A
PS1 B=B+1 A
  C=DAT0 W
  CSL W
  ?C=0 B
  GOYES PSERR
  CSR W
  ?A=C WP
  GOYES PS2
  GOSUB DP
  GONC PS1
PS2 C=R3
  CD0EX
  CD1EX
  C+P+1
  CD1EX
  C=B A
  GOSBVL #2D28
  C=C-1 A
  GOSBVL #2426
  REL(3) CC
  REL(3) FIXP
  REL(3) CC
  REL(3) AA
  REL(3) CC
  REL(3) PSS
  REL(3) PSS
  REL(3) CC
  REL(3) GPS
FIXP GOVLNG #2A6E
AA GOSBVL #3FD9
  ?ST=0 0
  GOYES PX
PSERR GOVLNG #2E35
PX GOVLNG #3172
PSS GOSUB FIXP
  GOSBVL #36CD
  GOSBVL #379D
  GOSBVL #36CD
  GOVLNG #3A03
DC GOSUB TBL
  A=0 A
  A=DAT1 1
  A=A-1 A
  B=A A
DC1 C=DAT0 W
  A=A-1 A
  GOC DCE
  GOSUB DP
  GONC DC1
DCE A=R3
  AD0EX
  D0=D0- 2
  P=P+1
  LCASC ' '
  P=P+1
  GOSBVL #5423
  D1=D1+ 1
  A=B A
  GOVLNG #5470
GPS GOSBVL #369D
  LCHEX F2
  ?A#C B
  GOYES V5
  GOSBVL #2CEB
  CR3EX
  C=0 W
  LCHEX 111121
  CR3EX
V0 GOSUB V1
  P= 0
  C=R3
  C=C-1 A
  ?C=0 P
  GOYES V2
  ?ST=1 3
  GOYES V3
  GONC V4
V2 ?ST=0 3
  GOYES V3
V4 CSR W
  R3=C
  ?C=0 W
  GOYES V8
  LCHEX F1
  ?A#C B
  GOYES V6
  GOSBVL #2CEB
  GONC V0
V8 GOSBVL #2A7A
  GOC V6
V5 GOVLNG #2E2B
V1 CD1EX
  D1=C
  R2=C
  GOSBVL #3FD9
  RTNC
  ?ST=1 0
  GOYES V3
  ?XM=0
  GOYES V9
V3 C=R2
  D1=C
  ST=1 4
  GOVLNG #2E66
V6 GOTO PX
V9 RTNCC
GSUXE GOTO AFN1
GSUX GOSBVL #F178
  ?A#0 W
  GOYES GSUX1
  GOSUB GG0
  GOSUBL BU
  GONC GSUXE
  D1=D1- 5
  D1=D1- 12
  C=DAT1 B
  ST=C
  A=0 W
  ?ST=1 2
  GOYES GSUX2
  A=A+1 S
GSUX2 GOTO GNIX
GSUX1 ?A#0 S
  GOYES GSUX3
  GOTO GSUXZ
GSUX3 GOSUB GG0
  GOSUBL B2
  GONC GSUXE
  A=0 W
GSUX4 CD1EX
  ?C=D A
  GOYES GSUX5
  CD1EX
  A=A+1 W
  GOSUBL CN
  ?ST=0 1
  GOYES GSUX6
  GOSUBL NXA
  GONC GSUX4
GSUX6 GOSBVL #1B322
GSUX5 GOTO GNIX1
V9Y RTNCC
AFNF C=RSTK
AFN1 P= 0
  LCHEX 39
  GOTO BC
SBSU GOSBVL #F178
SBSU1 GOSBVL #1B223
  B=A W
  GOSUBL B2
  GONC AFNF
SBS1 CD1EX
  ?C=D A
  GOYES V9Y
  CD1EX
  B=B-1 A
  ?B=0 A
  RTNYES
  GOSUBL NXA
  GONC SBS1
GSU GOTO GSUX
GSUXZ GOSUB SBSU1
  GOC GGET
  GOSUB GG0
  C=0 W
  P= 0
  LCHEX 91
  A=C W
  ASRC
  GOTO GNIX
GGET CD1EX
  D1=(5) #2F6CF
  DAT1=C A
  GOSUB GG0
  A=DAT1 W
  GOSBVL #ED21
  A=0 M
  ASR W
  GOSUB GG2
  GOSUB GG1
  C=DAT1 B
  C=C+C B
  CD0EX
  GOSUB GS4
  GONC GNOR
  D0=D0- 10
  D1=D1+ 10
GNOR D0=D0- 14
  CD0EX
  CSRB
  R2=C
  C=R3
  GOSBVL #ED2C
  CD1EX
  CD0EX
  AD1EX
  R1=A
  GOSBVL #1A460
  D1=A
  A=R2
GB10 C=DAT0 B
  ?A=0 B
  GOYES GB20
  A=A-1 A
  GOSBVL #18504
  D0=D0+ 2
  GONC GB10
GB20 ST=1 0
  P= 0
  GOSBVL #181B7
  C=R3
  CD0EX
  GOSBVL #F23C
  A=DAT1 W
  GOSUB GSTO
  GOSUB GG1
  GOSUB GS4
  A=0 W
  GONC GNA
  C=0 W
  C=DAT1 B
  GOSUB CL
  D1=D1+ 2
  C=DAT1 B
  R1=C
  C=0 W
  D1=D1+ 2
  C=DAT1 6
  GOSUB GS11
  R2=C
  C=A W
  GOSBVL #ECB4
  GOSUB CL1
  R1=C
  C=R2
  SETHEX
  GOSUB GS31
  GOSBVL #1B322
  P= 0
  LCHEX 010
  A=A-C X
  LCHEX 01
  ?A#C X
  GOYES NZM
  P= 14
  LCHEX 8
  ?A=C P
  GOYES NZM
  LCHEX 2
  ?AC P
  GOYES NZM1
  LCHEX 5
NZM1 A=A-C P
NZM P= 0
  ?ST=0 2
  GOYES GNA
  A=A-1 S
GNA GOSUB GG2
  GOSUB GS1
  C=A W
  GOSBVL #13335
  C=0 W
  C=D B
  GOSUB CL
  C=B B
  GOSBVL #ED40
  A=A+C W
  P= 0
  LCHEX 08
  GOSUB GS2
  GOSUB GS1
  A=0 W
  R1=A
  GOSUB GS31
  P= 0
  LCHEX 04
  GOSUB GS2
  GOSUB GG1
  B=0 W
  B=B+1 S
  A=0 W
  C=DAT1 B
  C=C+C B
  GONC GNIC
  A=A+B S
GNIC B=B+B S
  GOSUBL CN
  ?ST=0 1
  GOYES GNIP
  A=A+B S
GNIP B=B+B S
  ?ST=0 0
  GOYES GNIX
  A=A+B S
GNIX ASRC
GNIX1 GOSUB GG2
  GOTO WES
GG1 A=DAT0 B
  P= 0
  LCHEX F1
  ?A=C B
  GOYES GG0
  C=RSTK
  GOTO WES
GG0 D0=D0+ 2
  GOSBVL #FA35
  GOSBVL #F186
  CD1EX
  D1=C
  GOSBVL #ED3D
  CD0EX
  R3=C
  GOSBVL #F7B0
  D1=(5) #2F6CF
  C=DAT1 A
  CD1EX
  C=0 W
  RTN
GS2 C=0 XS
  GOSBVL #1B322
  A=A-C X
GG2 SETHEX
  C=R3
  CD0EX
  GOSBVL #ED2C
  CD1EX
  C=A W
  GOSBVL #F216
GSTO GOVLNG #F5F8
GS1 GOSUB GG1
  C=DAT1 12
  GOSUB CS1
GS11 GOVLNG #13229
GS31 GOSBVL #13252
  GOSUB CL1
GS3 GOSUB CL
  C=B B
  GOSUB CL
  C=D B
  A=C W
  RTN
GS4 D1=D1+ 13
  C=DAT1 A
  D1=D1+ 1
  ST=C
  ?ST=1 3
  RTNYES
  RTN
CL1 C=R1
CL CSL W
  CSL W
  C=A B
  RTN
  REL(5) DC
  REL(5) PS
WCC C=0 A
  C=DAT0 1
  D0=D0+ 1
  B=C A
  GOSBVL #2426
  REL(3) EXT
  REL(3) ALA
  REL(3) AG
  REL(3) W5V
  REL(3) ACV
  REL(3) W4V
  REL(3) W2
  REL(3) W1
  REL(3) ALA
  REL(3) GSU
W5V GOLONG W5
ACV GOLONG AC
W4V GOLONG W4
ALA GOSUB BU
  GONC ALAE
  D1=D1- 12
  D1=D1- 5
  C=DAT1 A
  ST=C
  ST=0 2
  B=B-1 A
  ?B=0 P
  GOYES ALA1
  ST=1 2
ALA1 C=ST
  DAT1=C A
ALAE GOTO COE
IV P= 0
  LCHEX 36
  GOTO BC
AG GOSUB SBSU
  GONC IV
  GOSUB AGO
WES GOTO WEX
BU SETHEX
  P= 0
  LCASC '   STPPA'
  D1=(5) #2F558
  A=DAT1 A
  AD1EX
BU1 A=DAT1 W
  ?A=C W
  GOYES BUE
  ?A#0 B
  GOYES BU2
BUX RTNCC
BU2 D1=D1+ 16
  D1=D1+ 16
  A=DAT1 A
  CD1EX
  C=C+A A
  CD1EX
  GONC BU1
BUE D1=D1+ 16
  A=0 A
  A=DAT1 4
  LC(5) ATYP
  ?A=C A
  GOYES BUE1
  C=0 A
  RTN
BUE1 D1=D1+ 16
  A=DAT1 A
  D1=D1+ 5
  C=0 A
  RTNSC
ST D0=(5) #2F8C5
  C=DAT0 W
  CR2EX
  DAT0=C W
  D0=D0+ 16
ST1 C=DAT0 W
  CR1EX
  DAT0=C W
  D0=D0+ 16
  C=DAT0 W
  CR0EX
  DAT0=C W
  RTN
INP GOSBVL #E90C
  D1=D1+ 16
  RTN
HMSB GOSBVL #F186
  GOSUB INP
  C=0 A
  LCHEX 4
  GOSUB RJA
  GOSUB CS
  ACEX W
  GOSBVL #13274
  R3=A
  RTN
ARJ SETDEC
  A=A+C X
RJU GOVLNG #12AE2
RJA GOSUB ARJ
RJA2 ACEX W
RJA1 B=0 W
  D=0 W
  D=C B
  RTN
CS GOSUB CS1
CS2 B=C B
CS1 CSR W
  CSR W
  RTN
W2 GOSUB HMSB
  CD1EX
  R2=C
  GOSUB SNA3
  GOSUB CS1
```

```
   CSRB
   A=R2
   AD1EX
   A=DAT1 W
   R2=A
   GOSUB JPHR
   GOTO PL2
W1 GOSUB HMSB
   C=DAT1 A
   A=0 W
   C=C+1 P
   GOC DATBL
   GOSUB INP
DATBL R1=A
   C=DAT1 W
   R2=C
   GOSUB JPH
PL2 SETHEX
   C=DAT1 W
   C=C+1 P
   GOC TEXT
   D1=D1+ 16
TEXT GOSBVL #BD31
   SETHEX
   C=R3
   B=A A
   BSRB
   B=B+C W
   R0=A
   CD1EX
   R1=C
   C=C+A A
   CD1EX
   C=0 A
   LCHEX BE
   ?CA A
   GOYES TOK
   LCHEX 25
BC GOVLNG #9393
TOK D1=D1+ 14
   C=0 W
   P= 12
   C=DAT1 P
   C=C+B W
   R3=C
EXT GOSUB ST
   GOSUB BU
   GOC TOK1
   R2=C
   ?C#0 A
   GOYES NNV
   LCHEX 3F
   GONC BC
NNV C=0 A
   LCHEX 25
   D=0 S
   GOSBVL #1AF01
   CON(5) #84C4
   GONC CRF1
TFE GOVLNG #944D
CRF1 C=R1
   D1=C
   C=R2
   DAT1=C W
   D1=D1+ 16
   LC(5) ATYP
   DAT1=C A
TOK1 GOSUB BU
   AD1EX
   D1=A
   D1=D1- 16
   D1=D1- 16
   D1=D1- 5
   B=0 W
   C=R3
   C=C+C B
   B=C B
   CD1EX
   GOSBVL #1AF01
   CON(5) #133C
   A=R0
   GOSUB ST
   C=A A
   CR0EX
   B=0 W
   B=C B
   AD1EX
   C=R3
   DAT1=C 14
   D1=D1+ 14
   C=R2
   ?C=0 W
   GOYES CO5
   DAT1=C 10
   D1=D1+ 10
CO5 A=R1
   CD1EX
   GOSBVL #1308
   C=R0
   CD1EX
   GOSUB POS
COE GOSUB P1
WEX GOLONG W4E
POS CD1EX
   D0=C
   GOSUB B2
   C=DAT0 M
   CSR M
   CSR M
   B=C M
POS1 CD1EX
   D1=C
   ?C=D A
   GOYES POS2
   C=DAT1 M
   CSR M
   CSR M
   ?CB M
   GOYES POS2
   GOSUB NXA
   GONC POS1
NXA C=0 A
   C=DAT1 B
   C=C+C B
   AD1EX
   A=A+C A
   AD1EX
   RTNCC
POS2 A=0 W
   A=DAT0 B
   A=A+A B
   CD1EX
   D1=C
   C=C-A A
   GOSBVL #ED3D
   C=A B
   D=C W
   AD0EX
   CD1EX
   ?CA A
   GOYES POS3
   B=C A
   B=B-A A
   B=B-1 A
   GOSUB SHI1
   C=A A
   GOSUB SHI
   C=A A
   A=A+1 A
   GOTO POS4
POS3 ACEX A
   C=C+D A
   ACEX A
   B=A A
   B=B-C A
   B=B-1 A
   D1=C
   GOSUB SHI
   C=A A
   GOSUB SHI1
   CD1EX
   A=C A
   C=C+1 A
POS4 GOSBVL #1B41B
   D0=C
   GOSUB SHI
   C=DAT0 S
   GOSBVL #1308
   GOSBVL #1B432
   D0=C
   GOSBVL #1B418
   DAT0=C S
   D=D-1 B
   ?D#0 B
   GOYES POS4
   D1=A
   ?CA A
   RTNYES
   C=D W
   GOSBVL #ED2C
   D1=C
   RTN
SHI1 C=C-1 A
SHI GOVLNG #1B435
JPH A=R1
   GOSUB RJU
   ?A=0 B
   GOYES JU
   ACEX W
   D=C W
   GOSUB G
   C=D W
   A=C W
JU D0=(5) #2F8C5
   DAT0=A A
   A=R1
   C=0 W
   P= 0
   LCHEX 8
   GOSUB ARJ
   SETHEX
   ?A#0 A
   GOYES X1
   CD1EX
   RSTK=C
   GOSUB SNA3
   D0=(5) #2F8C5
   GOSUB CS1
   CSRB
   GOSUB GS11
   C=RSTK
   CD1EX
   GOTO X2
X1 GOSUB RJA2
   P= 3
   A=0 W
   A=C WP
   P= 0
   GOSBVL #ED2F
   GOSUB CS2
   D=C B
   GOSBVL #13304
X2 C=A W
   D=C W
   GOSUB KEIN
   GOSUB MAL
JPHR A=R3
   C=A+C W
   CSL W
   CSL W
   P= 0
   LCHEX 7
   R3=C
   C=0 W
   CR2EX
   A=C W
   C=C+1 P
   GONC X8
X81 RTNCC
X8 ?A=0 W
   GOYES X81
   CR3EX
   P= 8
   ?A=0 S
   GOYES X4
   P= 12
X4 CPEX 13
   LCHEX C
   CR3EX
   LCHEX 010
   GOSUB ARJ
   SETHEX
   C=A W
   GOSBVL #1B42C
   B=0 W
   GOSUB CS2
   BSL W
   BSL W
   B=B+C B
   ?C=0 B
   GOYES JD8
   D=C A
   GOSUB G
   C=D A
   B=C B
JD8 C=B W
   R2=C
   C=A W
   GOSUB RJA1
   GOSUB CS
   A=0 W
   ACEX B
   CSR W
   C=0 M
   SETDEC
   C=C+C A
   A=A+C A
   CSR A
   C=C+C A
   A=A+C A
   GOSBVL #13274
   GOSBVL #ED40
   A=R2
   C=C+A W
   R2=C
   RTNCC
MAL C=0 W
   P= 1
   SETHEX
   LCHEX 1518
   GOVLNG #ECBB
KEIN A=DAT0 B
   ?A=0 B
   GOYES KX
KEIX P= 0
   LCHEX 9
   ?A#C P
   GOYES K1
K2 D=D-1 W
   GOSUB D7B
   P= 0
   ?C#0 P
   GOYES K2U
   LCHEX 7
K2U A=DAT0 B
   ASR A
   GOSUB WG
   ?A#C P
   GOYES K2
KX GOTO KF
K1 ?A#0 P
   GOYES K3
K4 GOSUB D7A
   P= 0
   LCHEX 3
   ?B#C P
   GOYES K4
   D=D-1 W
K3 GOSUB D7A
   P= 0
   ?C#0 P
   GOYES K5
   LCHEX 7
K5 A=DAT0 B
   ASR A
   GOSUB WG
   ?A#C P
   GOYES K3
   A=DAT0 B
   LCHEX 8
   ?A=C P
   GOYES KF
   ?A=0 P
   GOYES KF
K7 C=D W
K9 R0=C
   GOSBVL #13335
   GOSUB KF
   SETDEC
   A=A-1 W
   GOSUB DIV7
   SETHEX
   A=A+1 W
   C=DAT0 B
   P= 0
   ?C=A P
   GOYES K8
   A=R0
   GOSUB LC7
   C=C+A W
   GONC K9
K8 C=R0
   D=C W
KF C=D W
   A=C W
   RTN
WG ST=C
   LCHEX 8
   ?A=C P
   GOYES WG1
   C=ST
   RTN
WG1 C=ST
   ACEX P
   ST=C
   LCHEX 1
   ?A=C P
   GOYES WGEN
   C=C+1 A
   ?A=C P
   GOYES WGEN
   C=A P
   RTN
WGEN C=ST
   ACEX P
   RTN
LC7 C=0 W
   P= 0
   LCHEX 7
   RTN
D7A D=D+1 W
D7B GOSUB KF
   A=A+1 W
DIV7 GOSUB LC7
   GOVLNG #EC7B
G P= 0
   SETDEC
   DSRC
   ?D=0 P
   GOYES GR
   LCHEX 8
   ?D#C P
   GOYES GH
   DSLC
* C=C+1 A
   ?D=C P
   RTNYES
* LCHEX 6
* ?D=C P
* RTNYES
   ?D=0 P
   RTNYES
GR C=RSTK
   GOVLNG #E920
GH ?DC P
   GOYES GR
   LCHEX 5
   ?DC P
   GOYES GH1
   LCHEX 2
GH1 D=D+C P
   DSLC
   LCHEX 6
   ?D=C P
   GOYES GR
   C=C+1 A
   ?D=C P
   GOYES GR
   RTN
AGO C=DAT1 W
   CD1EX
   R0=C
   CD1EX
   GOSBVL #1B438
   ST=C
   ?ST=0 5
   RTNYES
   ?ST=1 7
   GOYES ARE
   C=DAT1 A
   ST=C
   ST=1 7
   C=ST
   DAT1=C A
   GOTO AFI
ARE ST=0 5
   C=ST
   GOSBVL #1B41E
   DAT1=C W
   CD1EX
   RSTK=C
   GOSUB SNA3
   GOSUB CS1
   B=C W
   BSRB
   C=RSTK
   D1=C
   D0=C
   D0=D0+ 14
   A=B W
   R1=A
NEURE C=0 W
   C=DAT1 12
   GOSUB CS1
   D=C W
   C=0 W
   C=DAT0 10
   GOSBVL #ED2F
   C=C+D W
   GOSBVL #13229
   D1=D1+ 2
   DAT1=C 10
   D1=D1- 2
   C=A W
   GOSBVL #13335
   SETDEC
   C=0 W
   D0=D0+ 2
   C=DAT0 B
   D0=D0- 2
   B=B+C W
   P= 0
   LCHEX 12
MTE ?B B
   GOYES MOK
   B=B-C B
   A=A+1 A
   GOTO MTE
MOK GOSBVL #13304
   D=C W
   A=DAT0 B
   ?A=0 B
   GOYES ND
   GOSUB KEIX
ND GOSUB KF
   GOSUB MAL
MIKO D1=D1+ 2
   C=0 W
   C=DAT1 10
   C=C+A W
   DAT1=C 10
   D1=D1- 2
   A=R1
   ?A W
   GOYES NDOK
   A=DAT0 B
   P= 0
   LCHEX 9
   ?A=C P
   GOYES NDER
   GOTO NEURE
NDER LCHEX 93A80
   A=0 W
   A=C A
   GOTO MIKO
NDOK CD1EX
   RSTK=C
   GOSUB SNA3
   B=C W
   C=RSTK
   RSTK=C
   CD1EX
   C=0 W
   D1=D1+ 2
   C=DAT1 10
   GOSUB SNA
   C=RSTK
   CD1EX
   GOSUB POS
AFI CD1EX
   R0=C
   GOSUB B2
   R1=C
AF1 CD1EX
   D1=C
```

```
  A=R1
  ?C=A A
  GOYES AF3
  C=0 W
  C=DAT1 14
  GOSBVL #1B438
  ST=C
  ?ST=0 5
  GOYES AF3
  C=DAT1 A
  C=C+C B
  GONC AF2
  GOSUB NXA
  GONC AF1
AF3 P= 0
  LCHEX C4
  GOSBVL #13601
AF2 CD1EX
  A=R1
  ?CA A
  GOYES AF21
  C=R0
AF21 A=R0
  CD1EX
  AD0EX
  RTN
SNA D0=(5) #2F755
  A=0 W
  A=DAT0 12
  CSL W
  CSL W
  C=C+C W
  ?A W
  GOYES SNA1
  ACEX W
SNA1 ?AB W
  GOYES SNA2
  ACEX W
SNA2 DAT0=A 12
SNA3 GOVLNG #125B2
B2 C=RSTK
  D=C A
  GOSUB BUL
  C=D A
  RSTK=C
  RTNNC
  D1=D1- 5
B3 CD1EX
  D1=C
  C=C+A A
  D1=D1+ 5
  D=C A
  RTNSC
PH ABEX A
  ?A=0 B
  GOYES C3
  GOSBVL #23E3
  CON(2) #F9
  REL(3) P1
  CON(2) #FC
  REL(3) Y7
  CON(2) #FE
  REL(3) Y7
  CON(2) #FD
  REL(3) DS
  CON(2) #2D
  REL(3) FTY
  CON(2) #2E
  REL(3) LI
  CON(2) 0
  ABEX A
  RTNSXM
C3 C=R3
  D1=C
  A=R2
  D1=D1- 14
  CD1EX
  ?AC A
  GOYES C6
  D1=C
  R3=C
  LCASC ' ALM:1D'
  DAT1=C 14
C6 RTNSXM
LI ABEX A
  LC(5) ATYP
  ?A=C A
  GOYES LI2
  RTNSXM
LI2 D1=D1+ 16
  D1=D1+ 16
  CD1EX
  B=C A
  GOSUBL W5RA1
  C=B A
  D1=C
  A=DAT1 A
  GOSUB B3
  GOSBVL #ED3D
  CD1EX
  R3=C
  A=0 W
  R1=A
  A=A-1 A
  R2=A
  A=DAT0 B
  LCHEX F1
  B=C A
  ?A#C B
  GOYES LI7
  D0=D0+ 1
  A=DAT0 A
  ASR A
  R1=A
  D0=D0+ 5
  C=DAT0 B
  ?B#C B
  GOYES LI8
  D0=D0+ 2
  A=DAT0 4
LI8 GOSBVL #1B2D2
  R2=A
LI7 A=R1
  GOSBVL #1B2D2
  C=R2
  GOSBVL #ED3D
  C=A A
  GOSBVL #ED3D
  R2=C
  C=R3
  CD1EX
LI3 CD1EX
  A=R3
  GOSBVL #ED0A
  ?C=A A
  GOYES ZZE
  CD1EX
  C=R3
  CD1EX
  D1=C
  R3=C
  C=R2
  C=C+1 A
  R2=C
  A=C A
  GOSBVL #ED2C
  B=C A
  GOSBVL #ED2C
  ?A A
  GOYES LI4
  ?AC A
  GOYES ZZE
  GOSUB ZEIG
  GOSUB ZZEIG
  GOSUB CN
  ?ST=0 3
  GOYES LI4
  D1=D1+ 14
  GOSUBL JHI
  GOSUB ZZEIG
LI4 GOSUB NXA
  GONC LI3
ZZEIG GOSUBL CRLF
  GOSBVL #76AD
  C=R3
  CD1EX
  RTNC
  C=RSTK
ZZE GOSUBL W5RA
  XM=0
  RTNCC
FTY ABEX A
  LC(5) ATYP
  ?A=C A
  GOYES FTY1
  RTNSXM
FTY1 GOSUB FTY2
  BSS 3
  CON(2) 5
  NIBASC 'APPT '
  CON(1) 1
  CON(4) ATYP
  NIBHEX FF
FTY2 C=RSTK
  CD1EX
  A=0 S
  A=A+1 S
  RTNCC
ST2 D0=(5) #2F901
  GOLONG ST1
P1 GOSUB BUL
  GOC P1J
P1E RTNSXM
P1J D1=D1- 5
  D1=D1- 12
  C=DAT1 A
  ST=C
  ?ST=1 2
  GOYES P1E
  GOSUB ST2
  D0=(2) #41
  P= 1
C5 C=RSTK
  DAT0=C A
  D0=D0+ 16
  P=P-1
  GONC C5
  C=D A
  DAT0=C A
LBA GOSUB SNA3
  B=C W
  BSR W
  BSR W
  BSRB
  GOSUB B2
LB0 CD1EX
  D1=C
  ?C=D A
  GOYES PE
  GOSUB CN
  ?ST=0 1
  GOYES LB2X
  GOSUB NXA
  GONC LB0
LB2X C=0 W
  C=DAT1 12
  CSR W
  CSR W
  ?B W
  GOYES LB3X
  GOSUB SAN
  GOLONG W8
SAN LCHEX C4
SAN1 GOVLNG #135FA
W8Z D1=D1+ 13
  C=DAT1 A
  ST=C
  ST=1 0
  ST=1 1
  C=ST
  DAT1=C A
  D1=D1- 13
  ?ST=0 3
  GOYES LB5X
  ?ST=1 2
  GOYES LB5X
  GOSUB AGO
LB5X GOTO LBA
LB3X BSL W
  BSL W
  B=B+B W
  GOSUB SNA
PE GOSUB DSX
  GOSUB ST2
  D0=(2) #61
  C=DAT0 A
  D=C A
  P= 1
C4 D0=D0- 16
  C=DAT0 A
  RSTK=C
  P=P-1
  GONC C4
  RTNSXM
W4E GOTO WEND
W4 GOSUB WIPE
  GOSUB B2
  GONC W4E
PRA1 CD1EX
  ?C=D A
  GOYES W4E
PRN CD1EX
  A=0 A
  A=DAT1 B
  GOSUB CN
  ?ST=1 0
  GOYES PRAG
  GOSUB NXL
  GONC PRA1
PRAG D1=D1+ 13
  ST=0 0
  C=ST
  DAT1=C P
  D1=D1+ 1
  A=A+A B
  AD0EX
  D0=D0- 14
  ?ST=0 3
  GOYES PRANR
  D1=D1+ 10
  D0=D0- 10
PRANR A=DAT1 B
  LCASC '!'
  ?A#C B
  GOYES PRA3
  D1=D1+ 2
  D0=D0- 2
  A=0 W
  AD0EX
  ASRB
  P= 2
  GOSBVL #9723
  GOSBVL #2296
  GOSUB VNT
  LCHEX 20000
PRA4 C=C-1 A
  GOC PRA5
  ?ST=0 12
  GOYES PRA4
PRA5 GOSUB IRT
PRAA GOTO W4E
PRA3 AD1EX
  CD0EX
  B=C A
  D0=(5) #2F480
  CD0EX
  GOSBVL #1308
  CD0EX
  A=DAT0 B
  LCASC ''
  ?A#C B
  GOYES PRA6
  LCASC ' '
  DAT0=C B
PRA6 GOSBVL #14C8A
  GOVLNG #2620
VNT GOSUB VNT1
  INTOFF
  ST=1 12
  C=0 A
I1 C=C+1 X
  GONC I1
  INTON
  GOVLNG #DB
VNT1 ST=0 12
  D0=(5) #2F43C
  C=DAT0 A
  D0=(4) #F971
  DAT0=C A
  C=RSTK
  GOTO IRT1
IRT D0=(5) #2F971
  C=DAT0 A
  ST=1 12
IRT1 D0=(4) #F43C
  DAT0=C A
  P= 0
  RTN
CN2 C=R3
CN3 CD1EX
CN D1=D1+ 13
  C=DAT1 A
  D1=D1- 13
  ST=C
  RTN
DS C=D A
  R2=C
DSX GOSUB B2
  GONC DSE
DS1 CD1EX
  ?C=D A
  GOYES DSE1
  CD1EX
  C=DAT1 A
  ST=C
  ?ST=1 7
  GOYES DS2
  GOSUB CN
  ?ST=1 1
  GOYES DS3
DS2 GOSUB NXL
  GONC DS1
DSE1 LCHEX C4
  GOSBVL #13601
  GONC DSE
DS3 GOSUB SAN
DSE C=R2
  D=C A
Y7E RTNSXM
Y7 C=D A
  R2=C
  R3=A
  GOSUB P1
  GOSUB BUL
  GONC DSE
  D1=D1- 5
  D1=D1- 12
  C=DAT1 B
  ST=C
  ?ST=1 2
  GOYES DSE
  GOSUB B2
Y71 CD1EX
  ?C=D A
  GOYES P
  GOSUB CN3
  ?ST=1 0
  GOYES Y72
  GOSUB NXL
  GONC Y71
Y72 AR3EX
  A=A+1 A
  A=A+1 B
  GOC UA
  A=0 W
  LCHEX 35
  A=C B
  GOSBVL #12917
P GOTO DSE
UA C=RSTK
  R3=C
  LCHEX CD
  GOSBVL #13601
  LCHEX CE
  GOSUB SAN1
  C=0 A
  LCHEX E
  B=C A
  LCHEX 809
  GOSBVL #1197D
  GONC UA1
  LCHEX
0D4358454D4C41
  DAT1=C 14
UA1 C=R3
  RSTK=C
  GOTO P
CLEAR GOSUB BUL
  D1=D1- 16
  D1=D1- 16
  D1=D1- 5
  A=R3
  D0=A
  C=0 A
  C=DAT0 B
  C=C+C B
  A=A+C A
  C=-C A
  B=C A
  CD1EX
  RTN
ACFN GOLONG AFN1
AC GOSBVL #F186
  CD1EX
  R0=C
  R1=A
  GOSUB B2
  GONC ACFN
  R3=C
  CD1EX
  R2=C
  C=R0
  CD1EX
  C=DAT1 S
  A=R1
  C=C+1 S
  GOC AY
  GOTO AM
NX1 C=R3
NX2 CD1EX
NXL GOLONG NXA
BUL GOLONG BU
AY GOSBVL #BD31
  R1=A
  CD1EX
  R0=C
  C=R3
  D=C A
  C=R2
  CD1EX
AD C=R0
  CD0EX
  CD1EX
  R3=C
  D1=C
  ?C=D A
  GOYES Q
  GOSUB CN
  D1=D1+ 14
  ?ST=0 3
  GOYES AD1
  D1=D1+ 10
AD1 C=R1
  AD0EX
  A=A+C A
  AD0EX
  AD1EX
  A=A+C A
  AD1EX
  GOSBVL #1B1C7
  GOC AK
NES GOSUB CLEAR
  GOSBVL #1AF01
  CON(5) #133C
  GOSUB AFI
  GOTO PRAA
AK GOSUB NX1
  GONC AD
Q P= 0
  LCHEX  0036
  P= 9
  GOSBVL #93BC
WEND GOVLNG #8A48
AM GOSBVL #1B223
  B=A A
  C=R3
  D=C A
  C=R2
  CD1EX
AM1 CD1EX
  R3=C
  ?C=D A
  GOYES Q
  GOSUB NX2
  B=B-1 A
  ?B#0 A
  GOYES AM1
  GONC NES
TIMH D0=(5) #2F8C5
  P= 0
  LCASC '00.00.00'
  GOSUB XSPC
  LCASC ':00:00'
XSPC DAT0=C W
  D0=D0+ 16
```

```
SPC LCASC ' '
  DAT0=C W
  D0=D0+ 2
  RTN
U4 D0=(5) #2F8E5
U3 CDEX W
  GOSUB U2
  C=B A
  GOSUB U2
  C=A A
  GOSUB U2
  CDEX W
  RTN
U2 DAT0=C 1
  D0=D0- 2
  CSR W
  DAT0=C 1
  D0=D0- 4
  RTN
WIPE P= 0
  D1=(5) #2F480
  C=0 A
  LCHEX D8
  GOVLNG #1B0AF
ZEIG GOSUB TIMH
  GOSUB CN2
  ?ST=0 0
  GOYES ZEI1
  LCASC 'x'
  D0=D0- 2
  DAT0=C B
  D0=D0+ 2
ZEI1 LCASC '  '
  ?ST=0 1
  GOYES ZEI2
  LCASC 'P'
  A=DAT1 B
  A=A+A B
  GONC ZEI2
  LCASC 'a'
ZEI2 DAT0=C A
  D0=D0+ 4
  C=0 A
  P= 3
  CPEX 3
  D1=D1+ 12
  C=DAT1 XS
  LCASC 'N'
  ?ST=0 3
  GOYES ZEI4
  LCASC 'R'
  ?ST=0 2
  GOYES ZEI4
  LCASC 'A'
ZEI4 DAT0=C A
  D0=D0+ 4
  C=0 A
  GOSUB SPC
  C=R3
  CD1EX
  C=0 W
  D1=D1+ 2
  C=DAT1 10
  GOSBVL #130E5
  GOSUB U4
  GOSBVL #1B418
  A=C B
  GOSUBL CS
  D=C B
  GOSUB RNSU
  C=R3
  CD1EX
  C=0 A
  C=DAT1 B
  C=C+C B
  CD0EX
  D1=D1+ 13
  D0=D0- 14
  C=DAT1 A
  ST=C
  D1=D1+ 1
  ?ST=0 3
  GOYES ZEI5
  D1=D1+ 10
  D0=D0- 10
ZEI5 A=0 W
  AD0EX
  ASRB
  GOTO ZSP
SE1? CD1EX
SE? C=R2
  GOSBVL #ED2C
  D=C W
  CD1EX
  D1=C
  ?C A
  RTNYES
  RTN
W5RA1 P= 0
  LCHEX 4
  D1=(5) #2F6C1
  DAT1=C 1
W5RA GOSUBL BU
  RTNNC
  D1=D1- 12
  D1=D1- 5
  C=DAT1 S
  CD1EX
  D1=(5) #2F6C1
  A=DAT1 S
  DAT1=C S
  CD1EX
  DAT1=A S
  RTNSC
QUIT GOSUB W5RA
  GOSBVL #14C8A
WOUT GOLONG COE
W5 P= 0
  GOSBVL #18534
  GOSUB W5RA1
  GONC QUIT
  GOSUBL B2
  CD1EX
  R3=C
  ?C=D A
  GOYES QUIT
  CDEX A
  GOSBVL #ED3D
  CDEX A
  R2=C
  CD1EX
W5B GOSUB SE?
  GONC W51
  AD1EX
  D1=A
  GOSUB CN
  ?ST=0 1
  GOYES W51
  R3=A
  GOSUB NXL
  GONC W5B
W51 GOSUB ZEIG
W52 GOSUB CLN
W53 GOSBVL #CF7
  GOSBVL #152BA
  GONC TA1
  LCHEX 32
  ?B=C B
  GOYES TA2
  C=C+1 A
  ?B=C B
  GOYES TA2
TA1 GOSBVL #212E
  GOSBVL #10EE
  GOC W53
TA2 A=B A
  ST=0 0
  GOSBVL #23E3
  CON(2) 51
  REL(3) DWN
  CON(2) 163
  REL(3) FDWN
  CON(2) 50
  REL(3) UP
  CON(2) 162
  REL(3) FUP
  CON(2) 99
  REL(3) QUIT
  CON(2) 43
  REL(3) QUIT
  CON(2) 5
  REL(3) TIM
  CON(2) 4
  REL(3) REPI
  CON(2) 86
  REL(3) XRU
  CON(2) 91
  REL(3) DEL
  CON(2) 71
  REL(3) AGOW
  CON(2) 73
  REL(3) RST
  CON(2) 46
  REL(3) RUN
  CON(2) 0
DWN1 GOTO W51
FDWN ST=1 0
DWN GOSUB NX1
  GOSUB SE?
  GONC DWN1
  CD1EX
  R3=C
  ?ST=0 0
  GOYES DWN1
  GONC FDWN
UP C=R2
  A=R3
  B=A A
UP1 ?C=B A
  GOYES DWN1
  R3=C
  GOSUB NX2
  CD1EX
  GONC UP1
FUP C=R2
  R3=C
FUP1 GOTO W51
TIM GOSUB TIMH
  C=0 W
  DAT0=C W
  GOSBVL #130DB
  GOSUB U4
  GOSBVL #1B418
  A=C B
  GOSUBL CS
  D=C B
  GOSUB U3
  P= 0
  D1=(5) #2F8C5
  GOSBVL #15147
TIM1 GOTO W51
REPI GOSUB CN2
  D1=D1+ 14
  ?ST=0 3
  GOYES TIM1
  GOSUB JHI
  GOTO W52
JHI P= 0
  D0=(5) #2F8C5
  LCASC '0+00    '
  DAT0=C W
  D0=D0+ 16
  LCASC ':00:00+0'
  DAT0=C W
  D0=D0+ 16
  LCHEX 003030
  DAT0=C 6
  A=0 W
  A=DAT1 B
  ?A#0 B
  GOYES RN3
  GOTO RN2
RN3 ASR W
  GOSUB RN1
  NIBASC 'SaSoMoDi'
  NIBASC 'MiDoFrWe'
RN1 C=RSTK
  A=A-1 A
  A=A+A A
  A=A+A A
  A=A+C A
  AD0EX
  C=DAT0 A
  D0=(5) #2F8C5
  DAT0=C A
  A=0 A
  A=DAT1 1
  GOSUB RN4
  NIBASC ' 12345+-'
RN4 LCHEX 6
  ?A P
  GOYES RN5
  A=A-1 A
  A=A-1 A
RN5 A=A+A A
  C=RSTK
  C=C+A A
  CD0EX
  C=DAT0 B
  D0=(5) #2F8C9
  DAT0=C B
RN2 D0=(5) #2F8CF
  D1=D1+ 2
  C=DAT1 A
  GOSUB U2
  D1=D1+ 2
  C=0 W
  C=DAT1 6
  SETHEX
  GOSBVL #13229
  D=C W
  GOSBVL #ECAF
  SETHEX
  D0=(5) #2F8D5
  GOSUB U2
  CDEX W
  GOSBVL #13252
  D0=(5) #2F8E7
RNSU GOSUB U3
  P= 0
  D1=(5) #2F8C5
  A=0 A
ZSP2 D1=D1+ 2
  A=A+1 A
  C=DAT1 B
  ?C#0 B
  GOYES ZSP2
  D1=(2) #C5
ZSP D0=A
  P= 0
  GOSBVL #18542
  AD0EX
  ST=1 4
  C=R2
  R0=C
  C=RSTK
  CD0EX
  GOSBVL #17E15
  C=R0
  R2=C
  CD0EX
  RSTK=C
  RTN
CLN D0=(5) #2F478
  C=DAT0 A
  CSTEX
  ST=1 9
  CSTEX
  DAT0=C A
CRLF C=R2
  CD0EX
  GOSBVL #17DC1
  C=R2
  CD0EX
  R2=C
  RTN
XRU GOSUB CN2
  D1=D1+ 13
  ?ST=0 0
  GOYES XRU1
  ST=0 0
  GONC XRU2
XRU1 ST=1 0
XRU2 C=ST
  DAT1=C A
  GOTO W51
DEL C=R2
  R4=C
  GOSUB CLEAR
  GOSBVL #1AF01
  CON(5) #133C
  GOSUBL AFI
  C=R2
  CD1EX
  D1=D1+ 16
  D1=D1+ 16
  C=DAT1 A
  AD1EX
  D1=A
  C=C+A A
  GOSBVL #ED3D
  D1=D1+ 5
  CD1EX
  A=R4
  C=A S
  R2=C
  GOSUB SE1?
  GOC DEL1
  GOTO QUIT
DEL1 C=R3
  GOSUB SE1?
  GOC RST2
  C=R2
  R3=C
  GOTO FDWN
AGOW GOSUB CN2
  ?ST=0 1
  GOYES RST2
  GOSUBL AGO
  GOSUB SE?
  GONC RST2
  CD1EX
  GOC RST3
RST GOSUB CN2
  D1=D1+ 13
  ?ST=0 3
  GOYES RST2
  ST=1 1
  C=ST
  DAT1=C A
  D1=D1- 13
  GOSUBL AGO
  CD0EX
RST3 R3=C
RST2 GOTO W51
RUN ?ST=1 13
  GOYES RST2
  GOSUB CN2
  ?ST#1 0
  GOYES RST2
  GOSUB W5RA
  GOSUB WIPE
  C=R3
  GOLONG PRN
W8 D1=D1+ 12
  P= 0
  A=0 A
  A=DAT1 P
  D1=D1- 12
  LCHEX 9
  ?CA P
  GOYES OUT
  R1=A
  GOSUBL VNT
W81 C=R1
  GOSBVL #2426
  REL(3) OUT
  REL(3) T1
  REL(3) T2
  REL(3) T3
  REL(3) T4
  REL(3) T5
  REL(3) T6
  REL(3) T2
  REL(3) T4
  REL(3) T6
T1 GOSBVL #EC5A
OUT GOSUBL IRT
  GOLONG W8Z
T2 LCHEX 100
  D=C A
  LCHEX 800
  GOSUB TUT
  GOTO WT
T31 P= 0
  LCHEX 400
  GOSUB T33
  P= 0
  LCHEX 300
T33 D=C A
  LCHEX 200
TUT GOSUB TUTE
  RTNC
  C=0 M
  D=0 M
  GOVLNG #EB40
T3 GOSUB T31
  GOSUB T31
  GOSUB T31
T32 GOTO OUT
T4 LCHEX 40
T41 ST=C
  GOSBVL #EC5A
  GOSUB TUTE
  GOC T32
  C=ST
  C=C-1 B
  GONC T41
  GOTO WT
T5 LCHEX 0B0
  D=C A
  LCHEX E00
  GOSUB TUT
  P= 0
  LCHEX A00
  D=C A
  LCHEX E00
  GOSUB TUT
T59 GOTO OUT
T6 LCHEX 4
T61 ST=C
  C=0 W
  LCHEX 60
  R0=C
  ST=0 11
T62 B=0 W
  B=B-1 P
  C=R0
  GOSBVL #EBEB
  P= 0
  GOSUB TUTE
  GOC T59
  A=R0
  ?ST=1 11
  GOYES T63
  A=A-1 A
  LCHEX 10
  GONC T64
T63 A=A+1 A
  LCHEX 60
T64 R0=A
  ?A#C B
  GOYES T62
  ?ST=1 11
  GOYES T65
  ST=1 11
  GONC T62
T65 C=ST
  C=C-1 P
  GONC T61
WT P= 0
  A=R1
  LCHEX 10007
  ?A P
  GOYES T59
WT4 GOSUB TUTE
  GOC T59
  C=C-1 A
  GONC WT4
  GOTO W81
TUTE ACEX W
  D0=(5) #2E3FF
  C=DAT0 A
  CSTEX
  ?ST=1 2
  GOYES TUTE1
TUTE1 CSTEX
  ACEX W
  RTNC
  ?ST=1 12
  RTNYES
  RTN
```

Wulf-Thorsten Gerdts
Bunzlauer Weg 18
3002 Wedemark 1

CAL

Zeile 1 von CAL (1-5) CCD-Barcodes Gnter Schapka

Zeile 2 von CAL (6-12) CCD-Barcodes Gnter Schapka

Zeile 3 von CAL (13-15) CCD-Barcodes Gnter Schapka

Zeile 4 von CAL (16-21) CCD-Barcodes Gnter Schapka

Zeile 5 von CAL (22-28) CCD-Barcodes Gnter Schapka

Zeile 6 von CAL (29-32) CCD-Barcodes Gnter Schapka

Zeile 7 von CAL (33-34) CCD-Barcodes Gnter Schapka

Zeile 8 von CAL (35-42) CCD-Barcodes Gnter Schapka

Zeile 9 von CAL (43-49) CCD-Barcodes Gnter Schapka

Zeile 10 von CAL (50-53) CCD-Barcodes Gnter Schapka

Zeile 11 von CAL (54-62) CCD-Barcodes Gnter Schapka

Zeile 12 von CAL (63-69) CCD-Barcodes Gnter Schapka

Zeile 13 von CAL (70-73) CCD-Barcodes Gnter Schapka

Zeile 14 von CAL (74-81) CCD-Barcodes Gnter Schapka

Zeile 15 von CAL (82-88) CCD-Barcodes Gnter Schapka

Zeile 16 von CAL (89-96) CCD-Barcodes Gnter Schapka

Zeile 17 von CAL (97-105) CCD-Barcodes Gnter Schapka

Zeile 18 von CAL (106-112) CCD-Barcodes Gnter Schapka

Zeile 19 von CAL (113-115) CCD-Barcodes Gnter Schapka

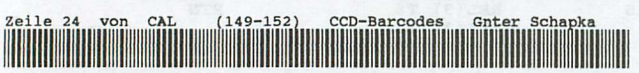
Zeile 20 von CAL (116-125) CCD-Barcodes Gnter Schapka

Zeile 21 von CAL (126-136) CCD-Barcodes Gnter Schapka

Zeile 22 von CAL (137-142) CCD-Barcodes Gnter Schapka

Zeile 23 von CAL (143-148) CCD-Barcodes Gnter Schapka

Zeile 24 von CAL (149-152) CCD-Barcodes Gnter Schapka



Zeile 25 von CAL (153-157) CCD-Barcodes Gnter Schapka

Zeile 26 von CAL (158-164) CCD-Barcodes Gnter Schapka

Zeile 27 von CAL (165-166) CCD-Barcodes Gnter Schapka

Zeile 28 von CAL (167-174) CCD-Barcodes Gnter Schapka

Zeile 29 von CAL (175-175) CCD-Barcodes Gnter Schapka

UMLAM

Zeile 1 von UMLAM (1-3) CCD-Barcodes Gnter Schapka

Zeile 2 von UMLAM (4-5) CCD-Barcodes Gnter Schapka

Zeile 3 von UMLAM (6-13) CCD-Barcodes Gnter Schapka

Zeile 4 von UMLAM (14-18) CCD-Barcodes Gnter Schapka

Zeile 5 von UMLAM (19-23) CCD-Barcodes Gnter Schapka

Zeile 6 von UMLAM (24-31) CCD-Barcodes Gnter Schapka

Zeile 7 von UMLAM (32-37) CCD-Barcodes Gnter Schapka

Zeile 8 von UMLAM (38-43) CCD-Barcodes Gnter Schapka

Zeile 9 von UMLAM (44-50) CCD-Barcodes Gnter Schapka

Zeile 10 von UMLAM (51-57) CCD-Barcodes Gnter Schapka

Zeile 11 von UMLAM (58-61) CCD-Barcodes Gnter Schapka

Zeile 12 von UMLAM (62-66) CCD-Barcodes Gnter Schapka

SUSAL

Zeile 1 von SUSAL (1-2) CCD-Barcodes Gnter Schapka

Zeile 2 von SUSAL (3-8) CCD-Barcodes Gnter Schapka

Zeile 3 von SUSAL (9-9) CCD-Barcodes Gnter Schapka

UMDAK

Zeile 1 von UMDAK (1-4) CCD-Barcodes Gnter Schapka

Zeile 2 von UMDAK (5-8) CCD-Barcodes Gnter Schapka

Zeile 3 von UMDAK (9-14) CCD-Barcodes Gnter Schapka

Zeile 4 von UMDAK (15-19) CCD-Barcodes Gnter Schapka

Zeile 5 von UMDAK (20-28) CCD-Barcodes Gnter Schapka

Zeile 6 von UMDAK (29-35) CCD-Barcodes Gnter Schapka

Zeile 7 von UMDAK (36-42) CCD-Barcodes Gnter Schapka

Zeile 8 von UMDAK (43-49) CCD-Barcodes Gnter Schapka

Zeile 9 von UMDAK (50-52) CCD-Barcodes Gnter Schapka

Zeile 10 von UMDAK (53-60) CCD-Barcodes Gnter Schapka

Zeile 11 von UMDAK (61-67) CCD-Barcodes Gnter Schapka

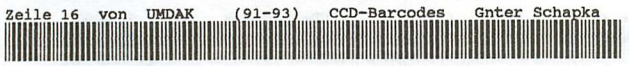
Zeile 12 von UMDAK (68-71) CCD-Barcodes Gnter Schapka

Zeile 13 von UMDAK (72-79) CCD-Barcodes Gnter Schapka

Zeile 14 von UMDAK (80-83) CCD-Barcodes Gnter Schapka

Zeile 15 von UMDAK (84-90) CCD-Barcodes Gnter Schapka

Zeile 16 von UMDAK (91-93) CCD-Barcodes Gnter Schapka



Zeile 17 von UMDAK (94-98) CCD-Barcodes Gnter Schapka

Zeile 18 von UMDAK (99-102) CCD-Barcodes Gnter Schapka

Zeile 19 von UMDAK (103-107) CCD-Barcodes Gnter Schapka

Zeile 20 von UMDAK (108-112) CCD-Barcodes Gnter Schapka

Zeile 21 von UMDAK (113-119) CCD-Barcodes Gnter Schapka

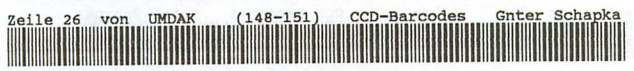
Zeile 22 von UMDAK (120-126) CCD-Barcodes Gnter Schapka

Zeile 23 von UMDAK (127-134) CCD-Barcodes Gnter Schapka

Zeile 24 von UMDAK (135-141) CCD-Barcodes Gnter Schapka

Zeile 25 von UMDAK (142-147) CCD-Barcodes Gnter Schapka

Zeile 26 von UMDAK (148-151) CCD-Barcodes Gnter Schapka



Zeile 27 von UMDAK (152-153) CCD-Barcodes Gnter Schapka

PG=

Zeile 1 von PGGLEICH (1-4) CCD-Barcodes Gnter Schapka

Zeile 2 von PGGLEICH (5-8) CCD-Barcodes Gnter Schapka

Zeile 3 von PGGLEICH (9-12) CCD-Barcodes Gnter Schapka

Zeile 4 von PGGLEICH (13-21) CCD-Barcodes Gnter Schapka

Zeile 5 von PGGLEICH (22-28) CCD-Barcodes Gnter Schapka

Zeile 6 von PGGLEICH (29-34) CCD-Barcodes Gnter Schapka

Zeile 7 von PGGLEICH (35-40) CCD-Barcodes Gnter Schapka

Zeile 8 von PGGLEICH (41-46) CCD-Barcodes Gnter Schapka

Zeile 9 von PGGLEICH (47-49) CCD-Barcodes Gnter Schapka

Zeile 10 von PGGLEICH (50-56) CCD-Barcodes Gnter Schapka

Zeile 11 von PGGLEICH (57-60) CCD-Barcodes Gnter Schapka

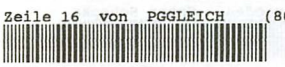
Zeile 12 von PGGLEICH (61-64) CCD-Barcodes Gnter Schapka

Zeile 13 von PGGLEICH (65-69) CCD-Barcodes Gnter Schapka

Zeile 14 von PGGLEICH (70-74) CCD-Barcodes Gnter Schapka

Zeile 15 von PGGLEICH (75-79) CCD-Barcodes Gnter Schapka

Zeile 16 von PGGLEICH (80-80) CCD-Barcodes Gnter Schapka



KAL

Zeile 1 von KAL (1-3) CCD-Barcodes Gnter Schapka

Zeile 2 von KAL (4-5) CCD-Barcodes Gnter Schapka

Zeile 3 von KAL (6-9) CCD-Barcodes Gnter Schapka

Zeile 4 von KAL (10-17) CCD-Barcodes Gnter Schapka

Zeile 5 von KAL (18-26) CCD-Barcodes Gnter Schapka

Zeile 6 von KAL (27-33) CCD-Barcodes Gnter Schapka

Zeile 7 von KAL (34-41) CCD-Barcodes Gnter Schapka

Zeile 8 von KAL (42-48) CCD-Barcodes Gnter Schapka

Zeile 9 von KAL (49-56) CCD-Barcodes Gnter Schapka

Zeile 10 von KAL (57-64) CCD-Barcodes Gnter Schapka

Zeile 11 von KAL (65-68) CCD-Barcodes Gnter Schapka


Zeile 12 von KAL (69-74) CCD-Barcodes Gnter Schapka


Zeile 13 von KAL (75-77) CCD-Barcodes Gnter Schapka


Zeile 14 von KAL (78-80) CCD-Barcodes Gnter Schapka


Zeile 15 von KAL (81-88) CCD-Barcodes Gnter Schapka


Zeile 16 von KAL (89-95) CCD-Barcodes Gnter Schapka


Zeile 17 von KAL (96-104) CCD-Barcodes Gnter Schapka


Zeile 18 von KAL (105-108) CCD-Barcodes Gnter Schapka


Zeile 19 von KAL (109-117) CCD-Barcodes Gnter Schapka


Zeile 20 von KAL (118-119) CCD-Barcodes Gnter Schapka


Zeile 21 von KAL (120-126) CCD-Barcodes Gnter Schapka


Zeile 22 von KAL (127-137) CCD-Barcodes Gnter Schapka


Zeile 23 von KAL (138-143) CCD-Barcodes Gnter Schapka


Zeile 24 von KAL (144-149) CCD-Barcodes Gnter Schapka


Zeile 25 von KAL (150-155) CCD-Barcodes Gnter Schapka


Zeile 26 von KAL (156-163) CCD-Barcodes Gnter Schapka


Zeile 27 von KAL (164-170) CCD-Barcodes Gnter Schapka


Zeile 28 von KAL (171-177) CCD-Barcodes Gnter Schapka

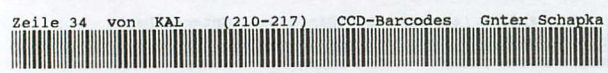

Zeile 29 von KAL (178-185) CCD-Barcodes Gnter Schapka


Zeile 30 von KAL (186-191) CCD-Barcodes Gnter Schapka


Zeile 31 von KAL (192-197) CCD-Barcodes Gnter Schapka


Zeile 32 von KAL (198-202) CCD-Barcodes Gnter Schapka


Zeile 33 von KAL (203-209) CCD-Barcodes Gnter Schapka


Zeile 34 von KAL (210-217) CCD-Barcodes Gnter Schapka


Zeile 35 von KAL (218-225) CCD-Barcodes Gnter Schapka


Zeile 36 von KAL (226-231) CCD-Barcodes Gnter Schapka

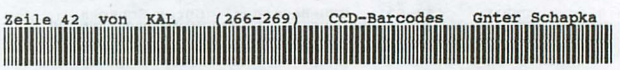

Zeile 37 von KAL (232-239) CCD-Barcodes Gnter Schapka


Zeile 38 von KAL (240-248) CCD-Barcodes Gnter Schapka


Zeile 39 von KAL (249-255) CCD-Barcodes Gnter Schapka


Zeile 40 von KAL (256-259) CCD-Barcodes Gnter Schapka


Zeile 41 von KAL (260-265) CCD-Barcodes Gnter Schapka


Zeile 42 von KAL (266-269) CCD-Barcodes Gnter Schapka


Zeile 43 von KAL (270-276) CCD-Barcodes Gnter Schapka


Zeile 44 von KAL (277-283) CCD-Barcodes Gnter Schapka


Zeile 45 von KAL (284-293) CCD-Barcodes Gnter Schapka


Zeile 46 von KAL (294-297) CCD-Barcodes Gnter Schapka


Zeile 47 von KAL (298-304) CCD-Barcodes Gnter Schapka


Zeile 48 von KAL (305-311) CCD-Barcodes Gnter Schapka


Zeile 49 von KAL (312-318) CCD-Barcodes Gnter Schapka


Zeile 50 von KAL (319-327) CCD-Barcodes Gnter Schapka


Zeile 51 von KAL (328-334) CCD-Barcodes Gnter Schapka

Zeile 52 von KAL (335-341) CCD-Barcodes Gnter Schapka

Zeile 53 von KAL (342-348) CCD-Barcodes Gnter Schapka

Zeile 54 von KAL (349-355) CCD-Barcodes Gnter Schapka

Zeile 55 von KAL (356-359) CCD-Barcodes Gnter Schapka

Zeile 56 von KAL (360-366) CCD-Barcodes Gnter Schapka

Zeile 57 von KAL (367-373) CCD-Barcodes Gnter Schapka

Zeile 58 von KAL (374-381) CCD-Barcodes Gnter Schapka

Zeile 59 von KAL (382-388) CCD-Barcodes Gnter Schapka

Zeile 60 von KAL (389-392) CCD-Barcodes Gnter Schapka

Zeile 61 von KAL (393-399) CCD-Barcodes Gnter Schapka

Zeile 62 von KAL (400-403) CCD-Barcodes Gnter Schapka

Zeile 63 von KAL (404-412) CCD-Barcodes Gnter Schapka

Zeile 64 von KAL (413-419) CCD-Barcodes Gnter Schapka

Zeile 65 von KAL (420-423) CCD-Barcodes Gnter Schapka

Zeile 66 von KAL (424-431) CCD-Barcodes Gnter Schapka

Zeile 67 von KAL (432-436) CCD-Barcodes Gnter Schapka

Zeile 68 von KAL (437-446) CCD-Barcodes Gnter Schapka

Zeile 69 von KAL (447-455) CCD-Barcodes Gnter Schapka

Zeile 70 von KAL (456-462) CCD-Barcodes Gnter Schapka

Zeile 71 von KAL (463-468) CCD-Barcodes Gnter Schapka

Zeile 72 von KAL (469-473) CCD-Barcodes Gnter Schapka

W1

Zeile 1 von W1 (1-4) CCD-Barcodes Gnter Schapka

Zeile 2 von W1 (5-8) CCD-Barcodes Gnter Schapka

Zeile 3 von W1 (9-12) CCD-Barcodes Gnter Schapka

Zeile 4 von W1 (13-21) CCD-Barcodes Gnter Schapka

Zeile 5 von W1 (22-28) CCD-Barcodes Gnter Schapka

Zeile 6 von W1 (29-34) CCD-Barcodes Gnter Schapka

Zeile 7 von W1 (35-40) CCD-Barcodes Gnter Schapka

Zeile 8 von W1 (41-46) CCD-Barcodes Gnter Schapka

Zeile 9 von W1 (47-49) CCD-Barcodes Gnter Schapka

Zeile 10 von W1 (50-56) CCD-Barcodes Gnter Schapka

Zeile 11 von W1 (57-60) CCD-Barcodes Gnter Schapka

Zeile 12 von W1 (61-64) CCD-Barcodes Gnter Schapka

Zeile 13 von W1 (65-69) CCD-Barcodes Gnter Schapka

Zeile 14 von W1 (70-74) CCD-Barcodes Gnter Schapka

Zeile 15 von W1 (75-79) CCD-Barcodes Gnter Schapka

Zeile 16 von W1 (80-80) CCD-Barcodes Gnter Schapka

BEN

Zeile 1 von BEN (1-4) CCD-Barcodes Gnter Schapka

Zeile 2 von BEN (5-8) CCD-Barcodes Gnter Schapka

Zeile 3 von BEN (9-12) CCD-Barcodes Gnter Schapka

Zeile 4 von BEN (13-21) CCD-Barcodes Gnter Schapka

Zeile 5 von BEN (22-28) CCD-Barcodes Gnter Schapka

Zeile 6 von BEN (29-34) CCD-Barcodes Gnter Schapka

Zeile 7 von BEN (35-40) CCD-Barcodes Gnter Schapka

Zeile 8 von BEN (41-46) CCD-Barcodes Gnter Schapka

Zeile 9 von BEN (47-49) CCD-Barcodes Gnter Schapka

Zeile 10 von BEN (50-56) CCD-Barcodes Gnter Schapka

Zeile 11 von BEN (57-60) CCD-Barcodes Gnter Schapka

Zeile 12 von BEN (61-64) CCD-Barcodes Gnter Schapka

Zeile 13 von BEN (65-69) CCD-Barcodes Gnter Schapka

Zeile 14 von BEN (70-74) CCD-Barcodes Gnter Schapka

Zeile 15 von BEN (75-79) CCD-Barcodes Gnter Schapka

Zeile 16 von BEN (80-80) CCD-Barcodes Gnter Schapka

SIOUX

Zeile 1 von SIOUX (1-2) CCD-Barcodes Wolfgang Teegler

Zeile 2 von SIOUX (3-6) CCD-Barcodes Wolfgang Teegler

Zeile 3 von SIOUX (7-11) CCD-Barcodes Wolfgang Teegler

Zeile 4 von SIOUX (12-15) CCD-Barcodes Wolfgang Teegler

Zeile 5 von SIOUX (16-19) CCD-Barcodes Wolfgang Teegler

Zeile 6 von SIOUX (20-28) CCD-Barcodes Wolfgang Teegler

Zeile 7 von SIOUX (29-36) CCD-Barcodes Wolfgang Teegler

Zeile 8 von SIOUX (37-44) CCD-Barcodes Wolfgang Teegler

Barcodes


Zeile 9 von SIOUX (45-52) CCD-Barcodes Wolfgang Teegler


Zeile 10 von SIOUX (53-60) CCD-Barcodes Wolfgang Teegler


Zeile 11 von SIOUX (61-68) CCD-Barcodes Wolfgang Teegler


Zeile 12 von SIOUX (69-76) CCD-Barcodes Wolfgang Teegler

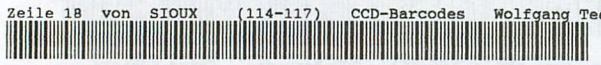

Zeile 13 von SIOUX (77-83) CCD-Barcodes Wolfgang Teegler


Zeile 14 von SIOUX (84-91) CCD-Barcodes Wolfgang Teegler


Zeile 15 von SIOUX (92-99) CCD-Barcodes Wolfgang Teegler

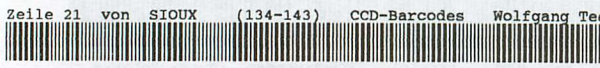

Zeile 16 von SIOUX (100-106) CCD-Barcodes Wolfgang Teegler

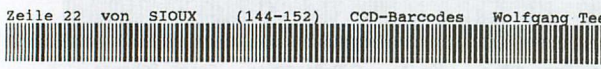

Zeile 17 von SIOUX (107-113) CCD-Barcodes Wolfgang Teegler


Zeile 18 von SIOUX (114-117) CCD-Barcodes Wolfgang Teegler


Zeile 19 von SIOUX (118-125) CCD-Barcodes Wolfgang Teegler


Zeile 20 von SIOUX (126-133) CCD-Barcodes Wolfgang Teegler


Zeile 21 von SIOUX (134-143) CCD-Barcodes Wolfgang Teegler


Zeile 22 von SIOUX (144-152) CCD-Barcodes Wolfgang Teegler


Zeile 23 von SIOUX (153-161) CCD-Barcodes Wolfgang Teegler


Zeile 24 von SIOUX (162-170) CCD-Barcodes Wolfgang Teegler


Zeile 25 von SIOUX (171-177) CCD-Barcodes Wolfgang Teegler


Zeile 26 von SIOUX (178-186) CCD-Barcodes Wolfgang Teegler


Zeile 27 von SIOUX (187-194) CCD-Barcodes Wolfgang Teegler

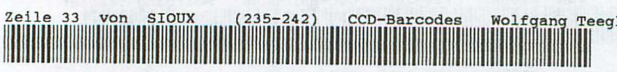

Zeile 28 von SIOUX (195-202) CCD-Barcodes Wolfgang Teegler


Zeile 29 von SIOUX (203-210) CCD-Barcodes Wolfgang Teegler


Zeile 30 von SIOUX (211-217) CCD-Barcodes Wolfgang Teegler


Zeile 31 von SIOUX (218-226) CCD-Barcodes Wolfgang Teegler


Zeile 32 von SIOUX (227-234) CCD-Barcodes Wolfgang Teegler


Zeile 33 von SIOUX (235-242) CCD-Barcodes Wolfgang Teegler


Zeile 34 von SIOUX (243-250) CCD-Barcodes Wolfgang Teegler


Zeile 35 von SIOUX (251-257) CCD-Barcodes Wolfgang Teegler


Zeile 36 von SIOUX (258-266) CCD-Barcodes Wolfgang Teegler


Zeile 37 von SIOUX (267-274) CCD-Barcodes Wolfgang Teegler


Zeile 38 von SIOUX (275-282) CCD-Barcodes Wolfgang Teegler


Zeile 39 von SIOUX (283-290) CCD-Barcodes Wolfgang Teegler


Zeile 40 von SIOUX (291-297) CCD-Barcodes Wolfgang Teegler


Zeile 41 von SIOUX (298-306) CCD-Barcodes Wolfgang Teegler


Zeile 42 von SIOUX (307-314) CCD-Barcodes Wolfgang Teegler


Zeile 43 von SIOUX (315-322) CCD-Barcodes Wolfgang Teegler


Zeile 44 von SIOUX (323-330) CCD-Barcodes Wolfgang Teegler


Zeile 45 von SIOUX (331-337) CCD-Barcodes Wolfgang Teegler


Zeile 46 von SIOUX (338-346) CCD-Barcodes Wolfgang Teegler


Zeile 47 von SIOUX (347-354) CCD-Barcodes Wolfgang Teegler


Zeile 48 von SIOUX (355-362) CCD-Barcodes Wolfgang Teegler

Zeile 49 von SIOUX (363-370) CCD-Barcodes Wolfgang Teegler

Zeile 50 von SIOUX (371-377) CCD-Barcodes Wolfgang Teegler

Zeile 51 von SIOUX (378-386) CCD-Barcodes Wolfgang Teegler

Zeile 52 von SIOUX (387-394) CCD-Barcodes Wolfgang Teegler

Zeile 53 von SIOUX (395-402) CCD-Barcodes Wolfgang Teegler

Zeile 54 von SIOUX (403-410) CCD-Barcodes Wolfgang Teegler

Zeile 55 von SIOUX (411-417) CCD-Barcodes Wolfgang Teegler

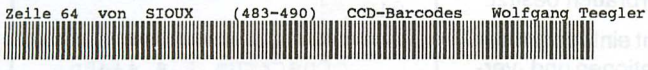
Zeile 56 von SIOUX (418-426) CCD-Barcodes Wolfgang Teegler

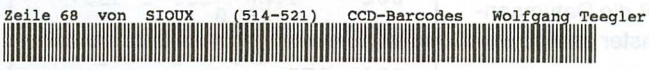
Zeile 57 von SIOUX (427-434) CCD-Barcodes Wolfgang Teegler

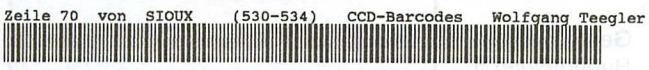
Zeile 58 von SIOUX (435-442) CCD-Barcodes Wolfgang Teegler

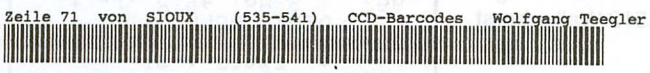
Zeile 59 von SIOUX (443-450) CCD-Barcodes Wolfgang Teegler

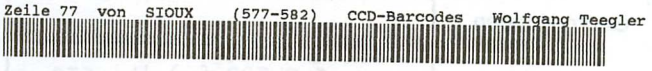
Zeile 60 von SIOUX (451-457) CCD-Barcodes Wolfgang Teegler

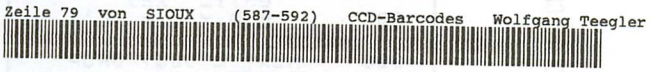


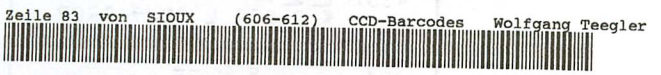




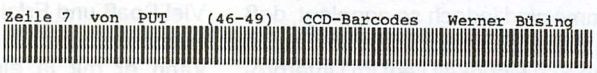








PUT



Schiffstrimmung

Fortsetzung von Seite 30

Einige Grunddaten der "M/T INDIO"

Länge über alles	115,84 m
Länge zwischen den Senkrechten	106,49 m
Breite	15,86 m
Tiefe bis zum Hauptdeck	8,00 m
Tiefe bis zum Achterdeck	9,30 m
Tiefgang beladen	ca. 7,30 m
Höhe der Mastspitze über Achterdeck	19,20 m
Reisegeschwindigkeit	ca. 13,5 kn
Verbrauch der Maschinen	ca. 10,5 mt
Verbrauch der Boiler	bis zu ca. 2,0 mt
Bunkerkapazität: Diesel 380 cSt.	342,0 mt
Gas	81,4 mt
Separater Ballasttank für Wasser	3.309,0 mt
Tonnage	
Leergewicht Sommertiefgang	6.400,0 mt
Leergewicht Wintertiefgang	6.160,0 mt
GRT	3.556,9
NRT	2.277,5

Folgende Programmzeilen müssen bei einem anderen Schiffstyp geändert bzw. angepaßt werden:

81, 114, 116, 544, 556, 606, 609, 616, 619

Wolfgang Teggler
MT "INDIO"
c/o Atlantic Rhederei
F.u.W. Joch
Johannisbollwerk 20
2000 Hamburg 11

Lottozahlengenerator

Für das Programm SORT aus Prisma 89.6.38 habe ich mir das folgende Spiel einfallen lassen - der persönliche Erfolg erstreckt sich damit auf drei Richtige pro Ziehung.....

Die Programmiertechnik läßt sich sicher erheblich verbessern, doch ich bin bewußt einen Weg gegangen, in dem nachvollzogen werden kann, was wo passiert. Durch die modulare Programmierung ist es auch hier möglich, die einzelnen Bausteine beliebig zu kombinieren und weitere Anwendungen zu konstruieren.

In ZL wird zunächst das Unterprogramm LZB (Lottozahl bestimmen) aufgerufen, dieses wiederum ruft ZB (Zufallszahl bestimmen) auf. Auf den ersten Blick ist dies ziemlich verwirrend, die einzelnen Unterprogramme sind jedoch so angelegt, daß sie sich praktisch selbst erklären. Die eigentliche Schwierigkeit liegt im Unterprogramm LZP (Lottozahl prüfen), da hier sichergestellt werden muß, daß keine Zahl doppelt in der Liste L gespeichert wird. Dies leisten die Zeilen 004 - 010 in LZP, der Rechner arbeitet eben so lange, bis er mal eine Zahl erwischt, die er noch nicht hatte... aber keine Angst, innerhalb von ca. 6 Sekunden zeigt er das Ergebnis, wobei das Sortierprogramm den Löwenanteil an dem Zeitverbrauch besitzt.

Die Programmidee beruht einfach auf der Arbeit mit Listenmanipulationen und -vergleichen.

Das Programm STR hat nur die Aufgabe, die endgültige Liste optisch aufzubereiten, damit man die Zahlen besser vergleichen kann ...

Das Programm ZE1 ist eine Variation des bekannten ZE-Programmes, hier wird lediglich sichergestellt, daß die Datumsanzeige über dem Stackfenster erfolgt.

Viel Spaß und Erfolg mit dem Programm (wenn jemand damit mal gewinnen sollte, kann er mir ja eine Speichererweiterungskarte spendieren...).

Georg Hoppen
Hubertusring 5
4512 Wallenhorst

```
48SX LOTTO
     « CLEAR { } 'L' STO
002  1 6 ZL L SORT 'L1'
     STO 1 'Z' L1 OBJ→
004  DROP 1 6 STR 1 5
       START +
006    NEXT x
     "Zusatzzahl" →TAG {
008  L1 x L } PURGE
     "Lottozahlen vom
010  "
     ZE1
012  » Bytes : 181,5
     Checksum : # 444Bh

48SX ZE1
     « DATE TIME TSTR +
002  1 DISP 1 FREEZE
     » Bytes : 30
004  Checksum : # F6C7h

48SX ZL
     «
002    START LZB L OBJ→
         IF 0 ≠
004      THEN LZP
         END x L + 'L'
006  STO
       NEXT
008  » Bytes : 71
     Checksum : # DB53h

48SX STR
     «
002    START →STR ":" +
     DEPTH ROLLD
004    NEXT
     » Bytes : 31
006  Checksum : # 4A7Eh

48SX ZB
     « RAND 100 * IP 'x'
002  STO
     » Bytes : 40
004  Checksum : # D5C0h

48SX LZB
     «
002    DO ZB
       UNTIL 'x≤49 AND x
004  ≠0'
       END
006  » Bytes : 57,5
     Checksum : # 711h


48SX SP77
     « CLEAR { } 'L' STO
002  1 7
       START
004      DO ZB
         UNTIL 'x≤9'
006      END x L + 'L'
     STO
008    NEXT L OBJ→ DROP
     1 7 STO 1 6
010    START +
       NEXT { x L }
012  PURGE
     "Spiel 77 vom
014  " ZE1
     » Bytes : 165
016  Checksum : # D7B8h


48SX LZP
     « CLEAR L OBJ→ DROP
002  DEPTH 1 SWAP
       START
004      IF x ==
         THEN 10 SF
006      END
       NEXT
008    IF 10 FS?C
       THEN LZB LZP
010    END
     » Bytes : 105,5
012  Checksum : # F9D7h
```

Speichererweiterung 48SX

Georg Hoppen teilt uns folgende Bezugsquelle mit:

EBM Ernst Süring GmbH
Dornröschenweg 19 -21
3000 Hannover 1
0511/604 32 48

Der Preis von ca. DM 300.- plus MwSt ist für die 128 KByte-Karte bestätigt worden, Ansprechpartner ist Herr Bremer (altes CCD-Mitglied, synthetische Programmierung, Statistik-Modul usw.), neuerdings "macht" er in Mailbox (HBB) und firmiert unter Tolo@HBB.UUCP. Die o.a. Mailbox versucht übrigens ebenfalls den HP48 zu unterstützen, einige Teilnehmer sind ganz wild entschlossen, dem HP auch noch die letzten Geheimnisse zu entreißen. Lassen wir uns mal überraschen, ich halte weiterhin losen Kontakt zu dieser Gruppe und werde gegebenenfalls Interessantes berichten.

Die Lieferzeit für die Karte soll zwischen 8 und 12 Wochenliegen.

Georg Hoppen

Alarmverwaltung Fortsetzung von Seite 36

```
 62♦LBL 05
RDN  GETKEY  X≠0?
GTO 03  ISG Z  SK2  ""
SK3  RDN  STO b

 73♦LBL 03
TONE 2  TONE 2  TONE 2
OFF  Ge

 79♦LBL "DW"
FIX 2  CLRG  .  X<>F
CLST  8  PSIZE  "W7"
GTREG  "TG SMDIOFAX"
PMTK  X=Y?  SF 03  X-1
FS? 03  .006  STO 07

 97♦LBL 07
CLA  RCL 07  ARCLI  "├ "
RCL IND 07  X<0?  SF 01
X<0?  "├N-"  "├WECK "
ABS  ARCL X  AVIEW
TONE 9  PSE  FS? 03
GTO 05  FS? 01  CHS
"WECK "  FS? 01  "├JN"
FC?C 01  "├NJ"  PMTK
X<>Y  GTO IND Y

125♦LBL 05
ISG 07  GTO 07  GTO 06

129♦LBL 01
CHS

131♦LBL 02
ABSP  "├ZT: "  ARCL X
PROMPT  ENTER↑

137♦LBL 03
STO IND 07  7  PSIZE
"W7"  LDREG
143♦LBL 06
  X<>F CLST CLA CLD
  .  Ge END
```

Günter Schapka
Rebusgasse 11
6100 Darmstadt

CCD\Base

Universelles Datenbankprogramm

von Alf-Norman Tietze und Matthias Rabe

Mit dem Programm CCD\Base kann der HP-71 als mobiles Informationssystem für strukturierbare Daten eingesetzt werden. "Strukturierbar" bedeutet, daß alle Datensätze den gleichen Satzaufbau haben müssen - z.B. mit den Datenfeldern: Name, Ort, Straße, Telefon oder ähnliches. Die Datenfelder können allerdings "dynamisch" verwaltet werden, d.h. sie können von Datensatz zu Datensatz unterschiedlich lang sein, je nach Informationsinhalt. Das ist übrigens eine besondere und Speicherplatz sparende Eigenschaft - dBASE, Clipper und Konsorten (PC-Datenbankprogramme) können das nicht, weil sie festplattenorientiert arbeiten.

Die Datenbankstruktur ist beliebig festzulegen. Es können - abhängig vom verfügbaren Arbeitsspeicher - mehrere unterschiedliche Datenbanken im HP-71 verwaltet werden. CCD\Base beherrscht die folgenden Grundfunktionen:

- Definition einer Datenbank
- Eingeben von neuen Datensätzen
- Ändern von Datensätzen
- Suchen von Datensätzen
- Löschen von Datensätzen
- Sortieren von Datensätzen
- Goto Datensatz-Nr.
- Anzeigen/Drucken von Datensätzen bzw. Listen

und einige Spezialfunktionen:

- Datenbank kopieren zur Datensicherung.
- Summenkalkulation eines numerisch verwendeten Feldes.
- Unterbrechung zum BASIC-System und Programmfortsetzung.
- Auswahl einer Datenbank.
- Information zur geöffneten Datenbank.
- Anzeige der Datensatz-Nr.
- Titelnamen der Datenbankfelder ändern.
- Hilfe-Funktion mit Tastenbeschreibung.

Obwohl CCD\Base dadurch schon erstaunlich vielseitig ist, gibt es gegenüber den PC-Datenbankprogrammen jedoch einige Einschränkungen:

- Die Datenbank muß immer in den Arbeitsspeicher (RAM) passen und ist somit schnell in der maximalen Größe begrenzt.
- Es kann immer nur eine Datenbank gleichzeitig geöffnet sein. Datenbank-Relationen (Verknüpfungen) sind nicht möglich.
- Es gibt keine Indizes (Sortierschlüssel).
- Die Sortierreihenfolge entspricht der physikalischen Reihenfolge der Datensätze und ist auf Wunsch ausschließlich aufsteigend (mit Großbuchstaben vor Kleinbuchstaben). Die Sortierung erfolgt dann in der Hierarchie vom ersten bis zum letzen Feld der einzelnen Datensätze gemäß dem HP-71 Zeichensatz.
- Alle Datenfelder sind Strings (Zeichenketten). Es gibt keine unterschiedlichen Feldtypen. Selbstverständlich können die Felder trotzdem unterschiedlich benutzt werden, z.B. als Zeichen-, Datums- oder Zahlenfeld.
- Die dynamische Länge eines einzelnen Datenfeldes ist auf maximal 96 Zeichen begrenzt, was der Zeilenlänge im Display des HP-71 entspricht.

Interne Datenbankorganisation

Intern ist die Datenbank als Text-File organisiert. Jeder Record (bei Text-Files mit unterschiedlicher Länge möglich) entspricht einem kompletten Datensatz (Set). Im ersten Record des Text-Files müssen die Feldbezeichnungen (Titles) stehen. Als Feldtrenner wird CHR$(3) verwendet, der ebenfalls als Abschlußzeichen nach dem letzten Datenfeld eines jeden Records angefügt ist. Die Kennung "DB71" ist im ersten Datensatz (der mit den Feldbezeichnern) hinter dem letzten CHR$(3) angefügt. Nur durch diese Kennung identifiziert CCD\Base eine Datenbank. Diese interne Organisation wird jedoch vom Programm automatisch verwaltet, so daß der Anwender davon nichts merkt.

Programm-Philosophie

Funktional ist die Datenbank wie ein Notizbuch organisiert, mit je einem kompletten Datensatz pro "Seite". Die Datenfelder eines Datensatzes stehen zeilenweise (!) auf selbiger "Seite" untereinander. Sinnvolle Bezeichnung und Reihenfolge für Datenfelder einer Adressdatenbank sind z.B. "Name", "Ort", "Str.", "Tel." und "Bem" (für Bemerkung). Bei einer Sortierung stehen dann die Namen in alphabetischer Reihenfolge gefolgt von PLZ/Ortsname und dem Rest.

Das Wechseln des Datensatzes erfolgt also durch rückwärts- oder vorwärtsblättern der "Seiten" im Notizbuch - entsprechend der Cursortasten [links] und [rechts]. Das Wechseln der einzelnen Datenfelder erfolgt deshalb durch aufwärts- oder abwärtsrollen der einzelnen "Zeilen" einer "Seite" im Notizbuch - entsprechend der Cursortasten [auf] und [ab]. Werden diese vier Cursortasten g-geshiftet, so entspricht das jeweils dem ersten bzw. letzten Datensatz oder Datenfeld. Wenn das angezeigte Datenfeld länger als 22 Zeichen ist, kann man nicht mehr mit [links] und [rechts] den Datensatz wechseln, weil dann diese Cursor-Tasten zum "Rollen" benötigt werden. In diesen Fällen helfen die f-geshifteten [links] /[rechts]-Tasten weiter.

Bei Abfragen im Display ist anhand der Position des Fragezeichens "?" zwischen Eingabefrage und Ja/Nein-Frage zu unterscheiden. Steht das "?" am Anfang, so ist es eine Eingabeaufforderung für Dateinamen, Devicenamen, Suchstrings etc. Wenn es nicht am Anfang steht, handelt es sich um eine Ja/Nein-Frage, die mit den Tasten "Y" (yes) für JA und "N" (no) für NEIN beantwortet werden muß.

CCD\Base erkennt übrigens automatisch, ob ein Printer oder ein Display vorhanden sind und richtet danach entsprechende Anzeigeparameter ein. Ohne Display muß bei einigen Anzeigen die [ENDLINE]-Taste gedrückt werden, um mit der Anzeige fortzufahren.

Programm starten

Das Programm wird mit RUN CCDBASE oder CALL BASE('filename') gestartet. Im ersten Fall wird der Dateiname der Datenbank abgefragt. Bei der Erstanwendung ist hier der Name der neuen Datenbank einzugeben. Danach werden die einzelnen Feldbezeichnungen (als 'Title' benannt) der Datenbank abgefragt. Nach Eingabe der letzen Feldbezeichnung wird die darauf folgende einfach ohne Eingabe einer Bezeichnung direkt mit [ENDLINE] quittiert. Anschließend erfolgt sofort die Abfrage nach dem Inhalt des ersten Datensatzes.

Das Menüsystem

Nach einem regulären Programmstart wird immer das erste Feld des ersten Datensatzes angezeigt. Ein "unsichtba-

res" Tastaturmenü ist nun aktiviert - mit folgender Tastaturbelegung:

[ATTN] = Quit (Ende)
H = Help (Hilfe zur Tastaturbelegung)
A = Append Dataset (Datensatz anfügen)
C = Copy File to ... (Datenbank kopieren nach ...)
c = [gC] Calculate Sum (Summenkalkulation auf Feld-Nr.)
= System Level (BASIC-System, Rückkehr mit [RUN])
D = Delete Dataset (Datensatz löschen)
E = Edit Title (Datenfeld editieren)
F = Select File (andere/neue Datenbank öffnen)
G = Goto Dataset-No. (Gehezu Datensatz-Nr.)
I = File-Informations (Datenbank-Information)
L = List all Datasets (ALLE Datensätze listen)
N = Disp Dataset-No. (Datensatz-Nr. anzeigen)
P = Print Dataset (Datensatz anzeigen/drucken)
S = Search Data (nach Daten suchen)
s = [gS] Sort Data (Datenbank sortieren)
T = Title-Names (Feldnamen editieren)
t = [gT] Time & Date (Zeit + Datum anzeigen)
0-9 = Title Quick-Keys (Schnelltasten für Datenfelder)

Nach Daten suchen

Mit "S" (Search Data) erhält man die Suchabfrage. Es wird strikt zwischen Groß- und Kleinschreibung unterschieden. Innerhalb der Eingabe des Suchstrings sind zwei weitere Tasten von besonderer Bedeutung: [f][+] (= "|" im Display) und [f][-] (= "|_" im Display) für eine UND- bzw. UND-NICHT-Verknüpfung von Suchargumenten.

Beispiele:

? Search: 6000 Frankfurt|CCD

... sucht alle Frankfurter, die auch das Kennwort CCD eingetragen haben.

? Search: 6000 Frankfurt|_CCD

... sucht alle Frankfurter, die nicht das Kennwort CCD eingetragen haben.

Hinweise

CCD\Base aktiviert das KEY-File BASEKEYS (falls vorhanden), wodurch man die Möglichkeit hat, eigene Tastaturbelegungen für die Dateneingabe (z.B. Umlaute etc.) zu definieren. Das KEY-File BASEKEYS muß man selbst erstellen. CCD\Base belegt selbst nur drei Tasten währen des Programmlaufs: [f][+], [f][-], [RUN]. Nach ordentlichem Programmende wird die alte Benutzertastatur wieder hergestellt.

Die Subroutine SUB KEY(T$,K$,K) ab Programmzeile 5000 ist aus dem Programm XLIB (Extension Library) entnommen und kann, wenn XLIB im HP-71 vorhanden ist entfernt werden.

CCD\Base benötigt das HPIL-ROM sowie die LEX-Files EDLEX, STRINGLX, und CCDUTIL. Es ist mit allen dazugehörigen Dateien in der HP-71 Programmbibliothek verfügbar. Auch das BASIC-File XLIB ist dort erhältlich.

Alf-Norman Tietze (1909)

```
1000 ! CCD\Base 2.01 (c) 1990 ant.target Frankfurt am Main
1010 ! by M.RABE & A.-N.TIETZE/ant.target
1020 ! requires: Lex HPIL,EDLEX,CCDUTIL,STRINGLX; evtl. KEYWAIT
1030 !
1040 CALL BASE('') @ SUB BASE(F$) @ F0$=STS$(67)
1050 ATTN OFF @ E$=CHR$(27) @ WIDTH 96 @ PWIDTH INF @ IF F$#'' THEN PUT '#38'
1060 WINDOW 1 @ DELAY 0,8 @ DISP E$;'E';E$;'<CCD\Base 2.01 (c) 1990'
1070 OPTION BASE 0 @ IF MEM<4000 THEN DISP MSG$(24) @ GOTO 3570
1080 DIM T(16),T0(16),I$[96],K$[66],S$[80],S1$[80],T$[1500],T0$[1500],L$[80],X$[96]
1090 IF POS(VER$,'HPIL') AND POS(VER$,'EDT') AND POS(VER$,'CCD') AND POS(VER$,'STR')
     THEN 1120
1100 DISP 'no HPIL/EDLEX/CCDGP/STRINGLX' @ GOTO 3570
1110 !
1120 P=PRT+1 @ D=DSP+1 @ IF D THEN DISP 'for the HP-71'
1130 CFLAG ALL @ USER ON @ STD @ SFLAG -1
1140 K$='1234567890,ACcDEFGILNPSsTtH#47#48#103#104#159#160#50#51#162#163#43'
1150 E0$='.End.' @ L$=RPT$('-',79) @ IF D THEN DISP L$ @ DISP @ DISP
1160 ON ERROR GOTO 1170 @ PURGE USERKEYS @ RENAME KEYS TO BASEKEYZ
1170 ON ERROR GOTO 1180 @ MERGE BASEKEYS
1180 DEF KEY 'f+',CHR$(124); @ DEF KEY 'f-',CHR$(124)&CHR$(95); @ DEF KEY '#46','CONT'
1190 IF FILE$('LASTBASE')[1,8]#'LASTBASE' THEN CREATE DATA LASTBASE,1,100
1200 ON ERROR GOTO 'E' @ ASSIGN #2 TO LASTBASE
1210 ON ERROR GOTO 1220 @ IF NOT LEN(F$) THEN READ #2;F$,S$ ELSE READ #2;S$,S$
1220 LINPUT '? Filename: ',UPRC$(F$);F$ @ F$=UPRC$(F$)
1230 IF NOT LEN(F$) THEN 'J'
1240 ON ERROR GOTO 'E' @ IF FILE$(F$)[1,LEN(TRIM$(F$))]=TRIM$(UPRC$(F$)) THEN 'I'
1250 CALL KEY('new File ?','',K) @ IF K THEN CREATE TEXT F$ @ T0$='' @ GOTO 'I'
1260 LINPUT '? Device: ',':TAPE(1)';D$ @ IF NOT LEN(D$) THEN 1220
1270 ON ERROR GOTO 'E' @ DISP MSG$(240010) @ COPY F$&D$ @ F$=F$[1,8]
1280 !
1290 'I': ASSIGN #1 TO F$ @ F=MAX(0,FILESZR(F$)-1) @ IF D THEN DISP
1292 ! IF UPRC$(F$)='LOGBUCH' THEN M0=F ELSE M0=0
1300 ON ERROR GOTO 1310 @ READ #1,0;T0$ @ GOTO 1350
1310 IF ERRN=63 THEN 'E'
1320 ON ERROR GOTO 'E' @ FOR I=1 TO 32 @ DISP '? Title';I;': '; @ LINPUT '';I$
1330 IF NOT LEN(I$) THEN I=32 ELSE T0$=T0$&I$&CHR$(3)
1340 NEXT I @ T0$=T0$&'DB71' @ RESTORE #1 @ PRINT #1;T0$
1350 T1=0 @ J=0
```

```
1360 J=POS(T0$,CHR$(3),J+1) @ IF J THEN T1=T1+1 @ T0(T1)=J @ GOTO 1360
1370 DIM T(T1),T0(T1) @ R=1 @ IF NOT FLAG(8) THEN GOSUB 'V'
1380 Z0=0 @ FOR I=1 TO T1 @ Z0=MAX(Z0,T0(I)-T0(I-1)) @ NEXT I @ Z=Z+3
1390 'RD': ON ERROR GOTO 'A' @ READ #1,R;T$ @ J=0 @ ON ERROR GOTO 'E'
1400 FOR I=1 TO T1 @ J=POS(T$,CHR$(3),J+1) @ T(I)=J @ NEXT I
1410 I=1 @ IF FLAG(7,0) THEN RETURN ELSE IF D THEN DISP
1420 IF R THEN DISP R
1430 'M': CFLAG 0 @ ON ERROR GOTO 'E' @ IF I>T1 THEN I=T1 @ DISP 'max Title = ';
     T1 @ WAIT D=0
1435 IF T$[LEN(T$)-1]="$$" THEN DISP "* ";
1440 DISP CUT$(T$,'←',I-1) @ IF LEN(CUT$(T$,'←',I-1))>22 THEN SCROLL 1
1450 CALL KEY('',K$,K) @ IF K<11 THEN I=K @ GOTO 'M'
1460 IF K<23 THEN ON K-10 GOTO 'CA','A','CF','CS','DL','ED','FL','G','IN','LST','N','P'
1470 ON K-22 GOTO 'S','SO','TIT','TD','H','L','R','L','R','F','LS','U','D','T','B','Q'
1480 !
1490 'A': RESTORE #1,F+1 @ T$='' @ I$='' @ IF D THEN DISP
1500 FOR I=1 TO T1 @ DISP CUT$(T0$,'←',I-1); @ LINPUT ': ';I$
1510 T$=T$&I$&CHR$(3) @ NEXT I @ ON ERROR GOTO 1540
1520 PRINT #1;T$ @ F=F+1 @ R=F @ IF D THEN DISP
1530 GOTO 'RD'
1540 GOSUB 'UN' @ RESTORE #1,F+1 @ GOTO 1520
1550 'CF': IF D THEN DISP E$;'EFilename: ';F$
1560 INPUT '? COPY to ',':TAPE(1)';D$
1570 IF LEN(D$) THEN 1580 ELSE IF D THEN 'IN' ELSE 'M'
1580 COPY F$ TO D$ @ IF D THEN DISP E$;'E'
1590 IF D THEN 'IN' ELSE 'M'
1600 'DL': ON ERROR GOTO 1650 @ IF D THEN DISP
1610 DISP 'Delete ? ';CUT$(T$,'←',0)
1620 CALL KEY('','',K) @ IF NOT K THEN 'RD' ELSE IF D THEN DISP
1630 DELETE #1,R @ F=F-1 @ IF R>1 THEN R=R-1
1640 GOTO 'RD'
1650 GOSUB 'UN' @ GOTO 1610
1659 ! 'ED': IF R<=M0 THEN DISP '... kein Editieren !' @ GOTO 'rd'
1660 'ED': IF LEN(CUT$(T$,'←',I-1))>90 THEN DISP '... kein Editieren !' @ GOTO 'rd'
1662 ON ERROR GOTO 1700 @ DISP CUT$(T0$,'←',I-1);
1670 LINPUT ': ',CUT$(T$,'←',I-1);T$[T(I-1)+1,T(I)-1]
1680 REPLACE #1,R;T$ @ IF NOT R THEN T0$=T$
1690 GOTO 'RD'
1700 GOSUB 'UN' @ GOTO 1680
1710 'UN': DISP @ BEEP
1720 CALL KEY(MSG$(61)&':unsecure?','',K) @ IF NOT K THEN POP @ DISP @ GOTO 'RD'
1730 IF ERRN=61 THEN SFLAG 1 @ UNSECURE F$ @ ASSIGN #1 TO F$ @ RETURN
1740 POP @ GOTO 'E'
1750 'FL': IF FLAG(1,0) THEN SECURE F$
1760 IF D THEN DISP E$;'E'
1770 ASSIGN #1 TO * @ DIM T(16),T0(16) @ GOTO 1220
1780 'G': IF D THEN DISP
1790 INPUT '? Goto Dataset-No.: ','1';R @ IF R>F THEN BEEP @ DISP 'max No. = ';
     F @ WAIT D=0
1800 R=MIN(ABS(IP(R)),F) @ GOTO 'RD'
1810 'N': IF D THEN DISP
1820 DISP 'Dataset-No.';R @ IF KEYDOWN AND NOT D THEN 1820 ELSE 'M'
1830 'P': CFLAG 6 @ CALL KEY('? Print: Printer/Displ','PD#43',K)
1840 ON K GOTO 1860,1850,1910
1850 SWPRT OFF @ SFLAG 6
1860 PRINT @ FOR I=1 TO T1 @ IF CUT$(T0$,'←',I-1)[1,1]='*' THEN 1890
1870 PRINT CUT$(T0$,'←',I-1)&':';TAB(Z0+3);
1880 PRINT CUT$(T$,'←',I-1)
1890 NEXT I
1900 PRINT @ IF FLAG(6,0) THEN SWPRT ON
1910 IF FLAG(7,0) THEN RETURN ELSE GOTO 'RD'
1920 'IN': IF NOT D THEN DELAY 8,8 ELSE DISP E$;'ECCD\Base 2.01 (c) 1990'
1930 DISP 'Filename: ';F$
1940 DISP STR$(T1);' Titles on';F;'Sets'
1950 IF D THEN DISP L$ @ DISP ELSE DELAY 0,0
1960 GOTO 'RD'
1962 'MK': P=LEN(T$)
1964 IF POS(T$,CHR$(3)&"$$")=P-2 THEN T$=T$[1,P-1] ELSE T$=T$&"$$"
1966 REPLACE #1,R;T$ @ GOTO 'RD'
1970 'LST': CFLAG 5,6,7,8,9 @ IF D THEN DISP
1980 CALL KEY('List '&STR$(F)&' Sets OK ?','',V)
1990 IF NOT V THEN 2000 ELSE S$=CHR$(3) @ K=2 @ SFLAG 10 @ GOTO 2080
```

```
2000 IF D THEN DISP
2010 GOTO 'M'
2020 'S': S=R @ S0=R @ S3=R @ S9=F+1 @ CFLAG 0,5,6,7,8,9,10 @ V=0
2030 IF D AND NOT FLAG(11) THEN DISP
2040 LINPUT '? Search: ',S$;S$ @ IF NOT LEN(S$) THEN DISP @ GOTO 'M'
2050 IF NUM(S$)=124 THEN S$=S$[2]
2060 SFLAG 0 @ IF FLAG(11) AND NOT FLAG(12) THEN 2110 ELSE IF FLAG(11) THEN 2090
     ELSE CFLAG 0
2070 CALL KEY('? Search: All/Single','SA#43',K) @ IF K=3 THEN DISP @ GOTO 'M'
2080 IF K=2 THEN SFLAG 0 @ R=1 @ S=0 @ S0=0 @ S3=0
2090 IF FLAG(0) THEN CALL KEY('Complete Sets ?','',K) ELSE 2120
2100 IF K THEN SFLAG 8
2110 CALL KEY('? Print: Printer/Displ','DP#43',K) @ IF K=3 THEN DISP @ GOTO 'M'
2120 IF K=1 THEN SFLAG 6 @ SWPRT OFF ELSE SWPRT ON
2125 IF D THEN DISP CHR$(27);'E';
2130 IF D OR P THEN PRINT
2140 IF NOT FLAG(0) THEN 2180 ELSE IF FLAG(11) AND NOT FLAG(12) THEN 2180
2150 PRINT 'Filename:  ';F$ @ IF NOT FLAG(10) THEN PRINT 'Search:    ';S$
2160 IF D OR P THEN PRINT L$
2170 PRINT
2180 FOR U=0 TO 99 @ V=POS(S$,CHR$(124),V+1) @ IF NOT V THEN 2200
2190 NEXT U
2200 S1$=S$ @ IF U THEN V=POS(S$,CHR$(124)) @ S1$=S$[1,V-1]
2210 IF POS(S1$,CHR$(95))=1 THEN S1$=S1$[2] @ SFLAG 5
2220 S=SEARCH(S1$,1,S0+1,F,1) @ S=IP(S)
2230 IF FLAG(5) AND S=S0+1 THEN S0=MIN(F,S0+1) @ GOTO 2220
2240 IF FLAG(5,0) AND S#S0+1 THEN S=(S0+1<=F)*(S0+1)
2250 IF NOT FLAG(9) AND S THEN S9=S @ SFLAG 9 @ GOTO 2320
2260 IF S#S9 THEN 2320 ELSE IF FLAG(11) THEN 2740
2270 IF FLAG(0) THEN PRINT E0$ @ PRINT @ PRINT ELSE DISP E0$
2280 IF D OR P THEN PRINT
2290 CFLAG 0,7,8,9,10 @ IF FLAG(6,0) THEN SWPRT ON
2300 IF S$=CHR$(3) THEN S$=''
2310 GOTO 'RD'
2320 IF NOT S THEN S0=0 @ GOTO 2390 ELSE S0=S @ S1=V @ S2=V
2330 READ #1,S;T$ @ FOR J=1 TO U
2340 S2=POS(S$,CHR$(124),S2+1) @ IF NOT S2 THEN S2=LEN(S$)+1
2350 IF NUM(S$[S1+1])=95 THEN S1=S1+1 @ SFLAG 5
2360 S=POS(T$,S$[S1+1,S2-1]) @ IF FLAG(5,0) THEN S=NOT S
2370 IF NOT S THEN 2200 ELSE S1=S2
2380 NEXT J @ S=S0 @ GOTO 2580
2390 S1$=S$ @ IF U THEN V=POS(S$,CHR$(124)) @ S1$=S$[1,V-1]
2400 IF POS(S1$,CHR$(95))=1 THEN S1$=S1$[2] @ SFLAG 5
2410 S=SEARCH(S1$,1,S0+1,9999,1) @ S=IP(S)
2420 IF FLAG(5) AND S=S0+1 THEN S0=MIN(F,S0+1) @ GOTO 2410
2430 IF FLAG(5,0) AND S#S0+1 THEN S=(S0+1<=F)*(S0+1)
2440 IF NOT FLAG(9) AND S THEN S9=S @ SFLAG 9 @ GOTO 2510
2450 IF S#S9 THEN 2510 ELSE IF FLAG(11) THEN 2740
2460 IF FLAG(0) THEN PRINT E0$ @ PRINT @ PRINT ELSE DISP E0$
2470 IF D OR P THEN PRINT
2480 CFLAG 0,7,8,9,10,11,12 @ IF FLAG(6,0) THEN SWPRT ON
2490 IF S$=CHR$(3) THEN S$=''
2500 GOTO 'RD'
2510 IF NOT S THEN 2730 ELSE S0=S @ S1=V @ S2=V
2520 READ #1,S;T$ @ FOR J=1 TO U
2530 S2=POS(S$,CHR$(124),S2+1) @ IF NOT S2 THEN S2=LEN(S$)+1
2540 IF NUM(S$[S1+1])=95 THEN S1=S1+1 @ SFLAG 5
2550 S=POS(T$,S$[S1+1,S2-1]) @ IF FLAG(5,0) THEN S=NOT S
2560 IF NOT S THEN 2390 ELSE S1=S2
2570 NEXT J @ S=S0
2580 READ #1,S;T$ @ IF NOT FLAG(11) THEN 2600
2590 X$=CUT$(T$,CHR$(3),N) @ IF LEN(X$) THEN X=X+VAL(X$)
2600 IF NOT FLAG(0) THEN 2690
2610 IF FLAG(11) AND NOT FLAG(12) THEN 2680
2620 IF NOT FLAG(8) THEN PRINT CUT$(T$,'←',0) @ S3=S @ GOTO 2200
2630 FOR I=1 TO T1 @ IF CUT$(T0$,'←',I-1)[1,1]='*' THEN 2660
2640 PRINT CUT$(T0$,'←',I-1)&':';TAB(Z0+3);
2650 PRINT CUT$(T$,'←',I-1)
2660 NEXT I
2670 IF D OR P THEN PRINT
2680 S3=S @ GOTO 2200
```

```
2690 DISP STR$(S);' ? ';CUT$(T$,'←',0) @ SCROLL 1 @ S3=S
2700 CALL KEY('','YNP#43',K) @ ON K GOTO 2710,2200,2720,'RD'
2710 R=S @ GOTO 'RD'
2720 R=S @ SFLAG 7 @ GOSUB 'RD' @ SFLAG 7 @ GOSUB 'P' @ GOTO 2200
2730 DISP S$;MSG$(232) @ BEEP @ WAIT 1 @ GOTO 2460
2740 IF (D OR P) AND FLAG(12) THEN PRINT
2750 X$=CUT$(T0$,CHR$(3),N) @ PDELAY
2760 IF D OR P AND NOT FLAG(6) THEN PRINT USING '#,SDP3DP3DRDD';X ELSE PRINT X;
2770 PRINT ' :Sum of [';X$;']' @ DELAY 0,0
2780 GOTO 2480
2790 'CS': J=COUNT(T0$,CHR$(3))
2800 INPUT '? Sum of Title No.: ',STR$((N=0)+N+1);N @ N=N-1
2810 CALL KEY('View matching Sets ?','',K) @ IF K THEN SFLAG 12
2820 IF CLASS(K)=-1 THEN 'RD' ELSE SFLAG 11 @ X=0 @ GOTO 'S'
2830 'CA': IF D THEN DISP
2840 DISP 'System-Level: ... '; @ IF D THEN DISP 'Press [RUN] to continue' @ DISP
2850 PAUSE @ IF D THEN DISP
2860 GOTO 'RD'
2870 'SO': IF D THEN DISP
2880 CALL KEY('Insert-Sort?','',K) @ ON (CLASS(K)+3) DIV 2 GOTO 'M',2950,2890
2890 DISP MSG$(240010) @ SWDSP OFF @ ON ERROR GOTO 'C' @ FOR I=2 TO F @ DISP I
2900 J=I @ READ #1,I;T0$
2910 READ #1,J-1;T$ @ IF T$<T0$ THEN 2940
2920 IF T$=T0$ THEN DELETE #1,J @ GOTO 2900
2930 REPLACE #1,J-1;T0$ @ REPLACE #1,J;T$ @ IF J>2 THEN J=J-1 @ GOTO 2910
2940 NEXT I @ 'C': SWDSP ON @ GOTO 'I'
2950 DISP MSG$(240010) @ K=IP(F/2)
2960 FOR I=1 TO F-K
2970 READ #1,I;T0$ @ READ #1,I+K;T$
2980 IF T0$<T$ THEN 3020 ELSE J=I @ REPLACE #1,I+K;T0$
2990 IF J-K<1 THEN 3010 ELSE READ #1,J-K;T0$
3000 IF T$<T0$ THEN J=J-K @ REPLACE #1,I;T0$ @ GOTO 2990
3010 REPLACE #1,J;T$
3020 NEXT I @ K=IP(K/2) @ IF K>0 THEN 2960 ELSE 'I'
3030 !
3040 'TIT': R=0 @ GOTO 'RD'
3050 'TD': IF D THEN DISP ELSE DELAY 8,8
3060 W=MOD(IP(IP(DATE/1000)*365.25)-1+FP(DATE/1000)*1000,7)
3070 W$='MonTueWenThuFriSatSon'[3*W+1][1,3]
3080 DISP TIME$;'  ';W$;' ';DATE$ @ DDELAY @ GOTO 'RD'
3090 'L': IF R>1 THEN R=R-1
3100 IF KEYDOWN THEN DISP R @ GOTO 3090 ELSE 'RD'
3110 'R': IF R<F THEN R=R+1
3120 IF KEYDOWN THEN DISP R @ GOTO 3110 ELSE 'RD'
3130 'F': R=1 @ GOTO 'RD'
3140 'LS': R=F @ GOTO 'RD'
3150 'U': IF I>1 THEN I=I-1
3160 GOTO 'M'
3170 'D': IF I<T1 THEN I=I+1
3180 GOTO 'M'
3190 'T': I=1 @ GOTO 'M'
3200 'B': I=T1 @ GOTO 'M'
3210 !
3220 'H': Z=1 @ IF D THEN DISP
3230 CALL KEY('? Help: Printer/Displ','DP#43',K) @ ON K GOTO 3240,3250,'M'
3240 SWPRT OFF @ SFLAG 6 @ IF D THEN DISP E$;'E'; @ Z=11
3250 PDELAY @ IF NOT FLAG(6) AND P THEN PRINT
3260 PRINT TAB(Z);'CCD\Base Function-Keys' @ IF D OR P THEN PRINT TAB(Z);L$[1,22]
3270 PRINT TAB(Z);'A   Append Dataset' @ PRINT TAB(Z);'C   Copy File to ...'
3280 PRINT TAB(Z);'c   [gC] Calculate Sum' @ PRINT TAB(Z);',   System-Level'
3290 PRINT TAB(Z);'D   Delete Dataset'
3300 PRINT TAB(Z);'E   Edit Title' @ PRINT TAB(Z);'F   Select File'
3310 PRINT TAB(Z);'G   Goto Dataset-No.' @ PRINT TAB(Z);'I   File-Informations'
3320 PRINT TAB(Z);'L   List all Datasets'
3330 PRINT TAB(Z);'N   Disp Dataset-No.' @ PRINT TAB(Z);'P   Print Dataset'
3340 PRINT TAB(Z);'S   Search Data' @ PRINT TAB(Z);' |  [f+] AND Link'
3350 PRINT TAB(Z);' |_ [f-] AND NOT Link' @ PRINT TAB(Z);'s   [gS] Sort Data'
3360 PRINT TAB(Z);'T   Title-Names' @ PRINT TAB(Z);'t   [gT] Time & Date'
3370 PRINT TAB(Z);'0-9 Title Quick-Keys' @ IF D OR P THEN PRINT TAB(Z);L$[1,22]
3380 PRINT TAB(Z);'Cursorkeys,   H = Help' @ PRINT TAB(Z);'fLeft/fRight  for SST'
3390 PRINT TAB(Z);'ATTN  Quit Functn/Prgm' @ PRINT TAB(Z);'RUN   Continue at SUSP';
3400 IF NOT FLAG(6) AND P THEN PRINT TAB(Z);L$[1,22]
```

```
3410 IF NOT FLAG(6) THEN 3440
3420 SWDSP OFF @ DISP @ DISP '... press ENDLINE'
3430 IF KEY$#'#38' THEN 3430 ELSE SWDSP ON
3440 IF D OR P THEN PRINT @ PRINT
3450 IF FLAG(6,0) THEN SWPRT ON @ DISP E$;'E'
3460 DELAY 0,8 @ GOTO 'RD'
3470 'E': CALL ERR(ERRM$) @ WAIT 1 @ GOTO 'J'
3480 'V': SFLAG 8 @ IF T0$[T0(T1)]='←DB71' THEN RETURN
3490 POP @ DISP @ DISP 'Incompatible Dataset' @ WAIT 1 @ GOTO 3550
3500 'Q': IF D THEN DISP
3510 CALL KEY('Quit ?','',K) @ IF K THEN 'J'
3520 IF D THEN 'IN' ELSE 'RD'
3530 'J': IF D THEN DISP E$;'E.END. of CCD\Base  (c)' @ ON ERROR GOTO 3550
3540 PRINT #2,0;F$,S$ @ IF FLAG(1,0) THEN SECURE F$
3550 ON ERROR GOTO 3560 @ PURGE KEYS @ RENAME BASEKEYZ TO KEYS
3560 DISP 'Closed:  ';F$ @ IF D THEN DISP L$ @ DISP
3570 ASSIGN #1 TO * @ ASSIGN #2 TO *
3580 SETSTS F0$ ! @ IF D THEN PUT '#43'
3590 END
5000 !
5010 SUB KEY(T$,K$,K) @ L=FLAG(-15,0) @ Y=0
5020 DIM P0$[1],P$[4] @ P0$=PEEK$('2F441',1) @ P$=PEEK$('2F946',4)
5030 DELAY 0,0 @ POKE '2F441','F' @ DISP CHR$(27);'<';
5040 IF NOT LEN(K$) THEN K$='NY#43' @ Y=1
5050 IF LEN(T$) THEN DISP T$
5060 X$=KEYWAIT$ @ K=POS(K$,X$) @ IF K=0 OR X$='#' THEN 5060
5070 P=POS(K$,'#') @ IF P=0 OR K<=P THEN 5120
5080 IF X$[1,1]#'#' THEN 5060
5090 Q=P @ FOR I=1 TO 15 @ Q=POS(K$,'#',Q+1)
5100 IF K=Q THEN K=P+I @ GOTO 5120
5110 NEXT I
5120 IF NOT Y THEN 5130 ELSE K=K-1 @ IF K=2 THEN K=-0
5130 POKE '2F441',P0$ @ POKE '2F946',P$ @ L=FLAG(-15,L)
5140 END SUB
```

Alf-Norman Tietze
Matthias Rabe

Clipper-Entwickler-Konferenz 1990

Anfang Dezember 1990 fand in Köln die dreitägige Nantucket Clipper-Entwickler-Konferenz '90 statt. Clipper ist ein dBASE-ähnlicher Compiler zur Entwicklung von Datenbankanwendungen. Die Programmiersprache Clipper bietet allerdings zusätzliche Möglichkeiten, die weit über die von dBASE hinausgehen.

Die Teilnehmer hatten die Wahl, während der drei Tage an 8 von insgesamt 22 angebotenen Seminaren bzw. Vorträgen teilzunehmen und sich dort umfassend über einzelne Fachgebiete der Datenbankprogrammierung mit Clipper zu informieren.

Im Mittelpunkt Stand der Produktwechsel von Clipper Sommer '87 auf die neue Version Clipper 5.0, die mit einigen wesentlichen Erweiterungen und Verbesserungen ausgestattet ist. Gerade wegen dieser Neuerungen - die natürlich bei einem komplexen Produkt fast nie ohne Bugs (Fehler) sein können - war die Konferenz besonders sinnvoll und nützlich.

Die Themenbereiche reichten von konkreten Programmieranwendungen über Tips und Tricks sowie Anpassungshilfen an die neue Version bis hin zu grundlegenden Methoden und Theorien der Entwicklung von Datenbankanwendungen. Erstmalig waren diesmal auch amerikanische Referenten zu Gast, die die Konferenz mit ihren Vorträgen (trotz englischer Sprache) wesentlich bereicherten. Sowohl deren Themen waren interessant (Clipper Interna, Fehler-Behandlungs- und Verhütungsmethoden, modulare Programmentwicklung, fortgeschrittene Netzwerkprogrammierung und Einstieg in SQL) als auch die lockere Vortragsart, welche richtig erfrischend wirkte. Mit solchen Referenten macht das Lernen Spaß - Einschlafen aufgrund von Langeweile war ausgeschlossen.

Die besonderen Neuigkeiten von Clipper 5.0 sind folgende:

- ein Pre-Compiler, der eigene, persönlich definierte Schreibweisen im Programmlisting in Clipper-Code umwandelt. Zum Beispiel:

  `IF Esc` wird umgewandelt in

  `IF LASTKEY()=27`

  Schon an diesem kleinen Beispiel lassen sich die umfangreichen Möglichkeiten des Pre-Prozessors erahnen.
- ein neuer dynamischer Linker, der Overlaystrukturen automatisch erzeugt.
- ein neues BROWSE- und GET-System mit umfangreichen Möglichkeiten.
- Codeblöcke und "Pseudo"-Objektorientiertes Programmieren. Es ist durchaus von Vorteil, daß die objektorientierten Eigenschaften langsam und Schritt für Schritt in Clipper vollzogen werden, da man als Programmierer zum Teil erheblich umdenken muß, um diese Möglichkeiten zu nutzen.
- mehrdimensionale Arrays.
- Datenbanktreiber-Konzept zur Unterstützung unterschiedlicher Datenbankformate. dBASE und SQL sind bereits realisiert, weitere (z.B. Paradox) sollen folgen.

Wie bereits schon auf der Konferenz 1989 deutlich wurde, entwickelt sich Clipper zu einer mächtigen und eigenständigen Programmiersprache (weg von dBASE - weil besser) mit dem Schwerpunkt für Datenbankanwendungen. Man darf sich schon jetzt wieder auf die nächste Clipper-Entwickler-Konferenz freuen, die voraussichtlich im September 1991 in Köln stattfindet.

Alf-Norman Tietze

Modulares Programmieren mit Turbo- und Quickpascal

Erik Wischnewski

Reihe Praxis Wissen, 1990, 464 Seiten, zwei 5.25"-Disketten, DM 68.-, ISBN 3-8023-0428-4, Vogel Buchverlag Würzburg

Der Autor beschreibt sehr anschaulich die Vorteile einer strukturierten Programmierung. Mit Fallstudien wird dem Leser vom Phasenmodell über den Aufbau eines Standardprogramms, Flexibilität durch Parametrisierung, bis zu einem einfachen Programmsystem nahegebracht, wie sinnvoll die vorgeschlagene Vorgehensweise ist.

Besonders hervorzuheben ist, daß der Autor nicht nur beschreibt, wie Programme nach relativ festen Regeln und Prinzipien entwickelt, sondern auch erklärt, weshalb eine Regel wichtig ist und warum diese so und nicht anders aussehen sollte.

Dazu eine Leseprobe aus dem Kapitel "Allgemeine Variable": "Eigentlich ist es kein Geheimnis und auch nichts Neues, und dennoch wird es oftmals nicht bedacht: Die Parametrisierung von Größen. Ein Beispiel, das die Verlockung zeigt, nicht immer alle Größen als Variable zu deklarieren oder als Konstante vorab im Programm zu setzen, ist das folgende:

```
Write (Chr(201));
FOR I := 1 TO 30 DO Write
                        (Chr(205));
Write (Chr(209));
FOR I := 1 TO 25 DO Write
                        (Chr(205));
Write (Chr(187));
```

Dieses Beispiel zeichnet die obere Begrenzung einer Maske auf dem Bildschirm. Ich will nun nicht darauf hinaus, daß solch eine Routine, wenn sie mehrfach vorkommt, als standardisiertes Modul in ein Unit gehört, sondern ich will darauf hinaus, daß die Symbole 187, 201, 205 und 209, die hier verwendet wurden, als IBM/ASCII-Code direkt in die Programmzeilen geschrieben wurden. Wenn man nun die Situation hat, daß der IBM-Zeichensatz nicht unterstützt wird, dann müßte man alle Programme, Units und Module durchgehen und sämtliche Codes ändern. ....."

Das beschriebene Programmsystem verwendet vier Leistungsstarke "Units", die auf den Disketten in compilierter Form zu finden sind:

- "Global" stellt Funktionen zu Konstanten, Bildschirmfunktionen, Farbpalette, Tastatur und zur Programminitialisierung bereit.
- "EMS" macht über die enthaltenen Funktionen EMS-Speicher verfügbar.
- "Math" enthält mathematische Funktionen von Min, Max bis zur Matrixinvertierung.
- "Dienste" bringt von "AlternativAbfrage" über "EditierenFeld", "FehlerMeldung" bis zu "VorschlagSeite" in Programmen verwendbare Module.

Das Beispielprogrammsystem ist in weiten Bereichen konfigurierbar und überzeugt durch klare Darstellung.

Ein Wermutstropfen taucht beim Blättern in den einzelnen Auswahlpunkten auf: Die Fehlermeldung "Datei 0344.HLP nicht gefunden". Nach Rückfrage beim Autor bestätigt sich die Vermutung, einige Hilfedateien sind nicht mitgeliefert worden. Abhilfe ist einfach dadurch möglich, daß mit dem Namen der vermißten Datei eine Leerdatei angelegt wird. Die Fehlermeldung bleibt dann aus. Wer's ganz genau nehmen will, kann bei Autor oder Verlag den Text der fehlenden Meldungen anfordern. Und noch ein (kleines) Ärgernis: Die Farben werden beim Verlassen des Programmsystems nicht zurückgesetzt, überraschende Farbzusammenstellungen z.B. beim Norton Commander sind die Folge.

Trotzdem ist das Buch im Aufbau logisch und in der Darstellung klar gemacht.

Gesamturteil: Für den fortgeschrittenen Anfänger wie den semiprofessionellen Programmierer zu empfehlen. Grundkenntnisse im Programmieren in Turbo- oder Quickpascal sind vorausgesetzt, mindestens sollte der Zugriff auf entsprechende Handbücher gegeben sein.

Die vollständigen Quelltexte sind beim Autor für DM 198.- zu erhalten. Zwei 'Blaue' sind viel Geld, aber Autoren und Programmierer wollen halt auch leben. Lohnt sich nur, wenn jemand wirklich viel programmiert, dann dürfte es sich über die Zeitersparnis rechnen.

dw

Turbo Pascal

Eine Einführung in die Programmierung mit Beispielen und Übungen aus der Wirtschaftsinformatik

Franz Dürrschnabel

Reihe Software-Intensiv-Training, 1990, 646 Seiten, 5,25"-Diskette, DM 73.-.
ISBN 3-8023-0451-9, Vogel Buchverlag

Dieses Buch aus der Reihe Software-Intensiv-Training ist in der bewährten Form Lernabschnitt - Übungsaufgaben zur Vertiefung aufgebaut. Wie schon aus dem Untertitel hervorgeht sind die Beispiele aus Sachgebieten entnommen, die Wirtschaftler besonders interessieren. Die Beispiele sind auf der zugehörigen Diskette im Quelltext vorhanden.

Nach der Einführung in die integrierte Entwicklungsumgebung von Turbo-Pascal kommt der Autor zielstrebig zur Sache. Dem Anfänger werden die Grundzüge der Programmierung in Pascal gut vermittelt. Der sachlich knappe Stil gefällt, dabei sind notwendige Hintergrundinformationen aus der Wirtschaftsinformatik nicht untergegangen. Vektoren- und Matrizenrechnung werden ebenso angewendet wie lineare Gleichungssysteme. Sequentielle und indexsequentielle Dateiverarbeitung wird behandelt. Der objektorientierten Programmierung ist ein eigenes Kapitel gewidmet.

Das letzte Kapitel ist schließlich einer "Einführung in ausgewählte Verfahren des Operations Research" gewidmet. Hier werden von linearerer Optimierung über Netzplantechnik bis zu Simulationstechnik anhand von Programmbeispielen Programmieransätze aufgezeigt.

Lediglich im allgemeinen Teil sind dem Autor einige ärgerliche Ungereimtheiten unterlaufen:
So ist die Behauptung "Bei einem 16-bit-System bestehen Daten- und Adreßbus aus 16 paralellen Leitungen. Damit kann das System über den Adreßbus bis zu 640 KByte Arbeitsspeicher ansprechen." schlicht falsch! Mit 16 Adreßleitungen lassen sich nur 64 KByte ansprechen. Schon der einfache PC mit 8088-Prozessor verfügt über 20 Adreßleitungen, mit denen 1024 KByte adressiert werden können. Lediglich die von IBM eingeführte Systemarchitektur begrenzt den zusammenhängenden Hauptspeicher auf 640 KByte.

Zumindest für den Anfänger verwirrend ist die Darstellung der Systemdateien von MS-DOS. Der Autor sagt richtig, daß PC-DOS eine spezielle Version von MS-DOS für IBM darstellt, die nur in Kleinigkeiten abweicht. Umso wichtiger wäre es nach dieser Aussage, MS-DOS die Systemdateien MSDOS.SYS und IO.SYS und nicht IBMBIO.COM und IBMDOS.COM zuzuordnen, diese gehören zu PC-DOS.

Dennoch das Gesamturteil insgesamt empfehlenwert, besonders dann, wenn die behandelte Thematik sowieso interessiert.

dw

Postvertriebsstück
Gebühr bezahlt

D 2856 F

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

CCD
ISSN 0176-8735
PRISMA

September - Dezember 1990 Nr. 5/6

HP-71B Systemzubehör:
* Kartenleser für HP-71B mit Magnetkarten DM 220
* Kassettenlaufwerk mit Netzteil DM 480
* PAC Screen, einfach DM 120
* 12" Monitor bernst. passend f. PAC screen DM 150
* HP41 Translator ROM DM 120
* AC-Circuit Anal. Pac DM 120

Alle II-Geräte mit original Handbüchern und je 1 IL-Kabel.

Walter Becker, Buschkämpen 41, 2850 Bremerhaven, Tel. 0471/565 53 (abends).

Verkaufe

Microsoft Windows 3.00 deutsch, 5 1/4", 1.2 MB, Anleitungsbuch DM 210

HPIL Interface HP 82973 für IBM-Kompatible + 5 1/4" Diskette mit Software dafür DM 190

HPIL 3 1/2" Diskettenlaufwerk HP 9114 A mit Anleitung DM 490

HP-75 Zubehör Ramerweiterung um 8k DM 190

HP-75 Programmsammlung FIGForth 1 Kassette eigenes Betriebssystem DM 180

Alle Preise sind Ihre Endpreise incl. Porto und Verpackung.

Peter Habicht, Tel 06008/7235 ab 19.00 Uhr

Suche
für HP 82240A Printer und IR-Modul HP 82242 A ein deutsches Handbuch oder eine Kopie.

Angebote an:
Gunther Kemény, Am Schafgarten 3, 6724 Dudenhofen, Tel. 06262/9 22 57, Sa-So / 0631/2 54 36, Mo-Fr.

CCD-Modul für HP 41 zu verkaufen DM 150, Tel. 0211/229 10 01

HP DeskJet:

256 kB RAM-Cartridge DM 150

Univers Condensed Softfont DM 100

für DeskJet Plus zu verkaufen.

Martin Meyer, Kelkheimer Straße 20, 6232 Bad Soden 1

Verkaufe:

HP 41 CX inkl. X-Funktion- und Time-Module DM 220

HP 41 CX und HP-Card-Reader N82 104 A / Akten- und Netzteil, nur komplette Abgabe möglich DM 600

HP Super VGA-Board 256 k DM 480

HP 18 C Business Consultant alles inklusive Literatur DM 99

Edgar Mauntz, Niddatal 1, Parkstraße 16, Tel. 06034/7417 abends ab 18.00 Uhr.

Verkaufe:

HP 82162 A Thermodrucker DM 300

W&W CCD Modul DM 100

Plotter-Modul(41er) DM 50

FORTH-Assembler-Modul (71er) DM 150

Texteditor-Modul (71er) DM 100

HP 7470 A Plotter DM1100

ThinkJet DM 450

HP 82161 A Kassettenlaufwerk DM 425

Schneider Joice 8256 DM 350

Thorsten Bender, Erlenhof 3, 5309 Meckenheim, Tel. 02225/6763.

Literatur für HP 41
- Erweiterte Funktionen ...
- Tricks, Tips und Routinen
- Synthetische Programmierung

Zusammen DM 40.

Christoph Klug, Tel. 05121/324 78.

Suche dringend:

IL Multimeter HP 3468 bin auch für Hinweise dankbar, suche außerdem IL Video-Interface HP 82 163.

Karlheinz Jünemann (1072), Rugewisch 18, 2000 Hamburg 63, Tel. 040/532 11 11.

HP DeskJet mit PELIKAN-Tinte füllen! Kostenlose Anleitung gegen adressierten Freiumschlag (60 Pf) an W. Klos, Am Lohberg 11, 3553 Cölbe 3.

Verkaufe:

2 HP71 inkl. IL-Modul
4 32k-RAM Port Module
1 FORTH-Modul
1 Digitallaufwerk + Bänder
1 IL-ThinkJet-Drucker
1 5-fach Netzteil für HP-Geräte
1 IL-Kabel 5m

Alle Preise VHS

Ulrich Bunse, 0211/558 13 17

Verkaufe:

Masterface, Rambox 32k, ThinkJet, Diskettenlaufwerk, Drucker. Tel. 02204/602 00 (abends).

Compaq Deskpro 386/20: Suche Speicher Grundplatine 4 MB 113222-001 und 4 MB Speichermodul 113226-001.
Thor Gehrmann, Tel. 02339/3963.

Letzte Meldung * Letzte Meldung * Letz

Umtauschaktion HP48SX

Seit dem Erscheinen der Betriebssystemversion "E", in der die Fehler desselben ausgemerzt worden sind, kann ein Rechner mit einer alten Version kostenlos bei HP umgetauscht werden.

Ansprechpartner bei HP ist die Frau Reinke in Ratingen, Tel.Nr. 02102/499282.

mm

Letzte Meldung * Letzte Meldung * Letz

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:
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.

Urheberrecht:
Alle Rechte, auch Übersetzung, vorbehalten. Reproduktionen gleich welcher Art - auch ausschnittsweise - nur mit schriftlicher Zustimmung des CCD. Eine irgendwie geartete Gewährleistung kann nicht übernommen werden.