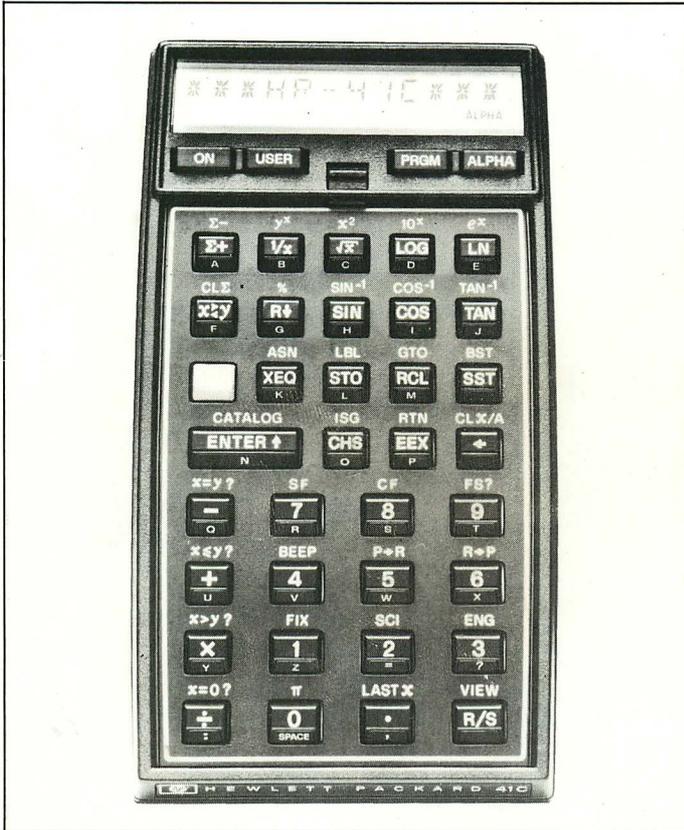


Best of

PRISMA



CCD - Computerclub Deutschland e.V.

Best of

PRISMA

1. Auflage 0 - 2600 · 1986

Copyright by CCD - Computerclub Deutschland e.V.
Postfach 2129 · 6242 Kronberg 2
Bestelladresse: CCD-Computerclub Deutschland e.V.
Schwalbacher Straße 50 · 6000 Frankfurt (M)
Preis: 32,- DM

Inhalt

Vorwort	1
1. Tips & Tricks	2
S?	9
ACXR	14
2. Barcode - Programme	15
Umbauanleitung Halbschrittdrucker	16
Programmbarcode	20
Sonstige Barcodes	24
3. Weitere Anwendungen des Druckerumbaus	26
Querschrift	27
Snoopy	29
Catalog	30
4. Synthetik	32
Rechnerorganisation	33
Byte Tabelle	35
Synthetische Programmierung	38
Registeraufbau	44
Der F7 - Wolf	49
Synthetische Alpha Labels	50
Anwendung von Private	51
Register d	53
Register c	56
Programmierung im Alpha Register	57
128 Töne	59
Der programmierbare Byte Jumper	61
Flags und der Drucker	69
Anmerkungen zu Flag 55	71
LB - Load Byte	73
MK - Make Keys	76
5. Mathematik	86
Universal-Basisumwandler	87
Gleichungen 4. Grades	89
Primzahlen	92
Vektorrechnung	93
Bruchrechnung	99
Lottozahlen	99
Kürzen von Brüchen	102
Primfaktorzerlegung	103

Inhalt

10-Punkte Gauß-Quadratur	119
Zufallszahlengenerator	122
Eulersche Gammafunktion	126
Differentialgleichungen	128
...nach Runge-Kutta	130
Aussagenlogische Ausdrücke	136
Diskrete Fourier-Transformation	138
Inverse Fourier-Transformation	145
Statistik	150
Matrizenrechnung	151
Lineare Gleichungssysteme	159
Lineare Regression	162
Interpolation	163
6. Plotprogramme	171
Plotten einer Funktion	172
Donald Duck & Goofy	173
High-Resolution-Plotter für 2 Funktionen	175
Multiplotter (bis zu 6 Funktionen)	177
Derby (Pferderennen)	179
7. Elektrotechnik	182
Rauschoptimierung	184
Stern-Dreieck-Umwandlung	186
Betriebsdämpfung von Vierpolen	187
Impedanz-Anpassung	195
Hochpaßfilter/Tiefpaßfilter	196
Widerstandsthermometer	198
Addition von zwei Sinusschwingungen	200
8. Spiele	201
Minischach	202
LOGICC	208
9. Sonstiges	212
Bubble-Sort	213
Monatslohnsteuer	214
DIN/ASA Umrechnung	222
Urlaub	223
Tanken	224
Entfernungsberechnung mit Koordinaten	230
Flugnavigation	234
Astronomie	237
Rechner-Bugs	240
Querschriftalphabet	243

V o r w o r t

Liebe Mitglieder,

lange habt Ihr auf dieses Werk warten müssen. Sicher habt Ihr Euch gefragt, warum wir - die Mitglieder der Ortsgruppe Köln - uns so lange Zeit gelassen haben. Nun, wenn Ihr dieses Heft durchblättert und es mit den Ausgaben der Jahre 80/81 vergleicht, werdet Ihr die Antwort finden: Wir wollten nämlich nicht nur die alten PRISMA's auseinanderschneiden, um sie dann - nachdem 17 von 18 Size-Routinen oder 12 von 13 Primfaktorenzerlegungen gestrichen wurden - anders zusammenzuleimen.

Es sollte vielmehr etwas neues, eigenständiges und besseres entstehen, was für Euch einen höheren Gebrauchswert hat. Um das zu erreichen, haben wir

- uns bedeutungslos erscheinende Programme gestrichen,
- brauchbare Unikate optimiert, sowohl Dokumentation als auch Programmiertechniken
- bei mehreren Programmen zum gleichen Problem das nach unserem Ermessen beste herausgesucht, und falls möglich, optimiert
- und vor allem neues Wissen eingebracht, welches wir als wichtig erachten, jedoch bislang nicht oder nur wenig in den bisherigen PRISMA-Ausgaben gewürdigt wurde.

Ein Beispiel ist z.B. das ABARP+ Programm zum Barcode-Plotten von W. Maschke. Sicher erklärt sich eben aus dieser Janusköpfigkeit dieses Werkes, zum einen altes Material wieder zu veröffentlichen, aber andererseits dessen Schwächen mit neuem zu optimieren.

Mitgearbeitet haben an dieser Ausgabe im einzelnen (alphabetisch):

	Friedhelm Birk	
	Achim B. Gemein	
	Henry Heimbach	
	Uwe Hommelsheim	
	Wilfried Kötz	
	Winfried Maschke	
	Ralf Pfeifer	
	Werner Ratzki	
	Andreas Trögel	

Viel Spaß wünscht Euch Eure Ortsgruppe Köln

Fehlermeldungen

Viele Programme lassen nicht alle reellen Zahlen als Eingangsparameter zu, z.B. erlauben Primfaktorenzerlegungen nur positiv, ganzzahlige Eingaben. Um nun falsche Eingaben mit einer Fehlermeldung zu quittieren, benutzt man zweckmäßigerweise Standardfunktionen:

Positive Ganzzahlen mit 0: FRC, FACT
Negative Ganzzahlen mit 0: CHS, FRC, FACT
Positive Ganzzahlen ohne 0: X=0?, 1/X, FRC, FACT
Negative Ganzzahlen ohne 0: X=0?, 1/X, CHS, FRC, FACT
Alle Ganzzahlen: FRC, OCT
Positive reelle Zahlen ohne 0: LN oder LOG
Positive reelle Zahlen mit 0: X≠0?, LN
Negative reelle Zahlen ohne 0: CHS, LN oder CHS, LOG
Negative reelle Zahlen mit 0: CHS, X≠0?, LN
Reelle Zahlen zwischen -1 und 1 (mit -1, 1): ASIN oder ACOS
Reelle Zahlen zwischen -1 und 1 (ohne -1, 1): ABS, CHS, LN1+X
Reelle Zahlen zwischen -10 und 10 (ohne -10, 10): CF 26, TONE IND X

Dezimal-Oktal-Umwandlungen für alle Zahlen lassen sich auf dem HP-41C so durchführen: LBL "ROCT", ENTER, INT, OCT, X↔Y, FRC, 8, X^2, 5, Y^X, *, INT, OCT, 10, 10^X, /, +, verwandelt reelle Dezimalzahlen in reelle Oktalzahlen, und: LBL "RDEC", ENTER, INT, DEC, X<>Y, FRC, 10, 10^X, *, INT, DEC, 8, X^2, 5, Y^X, /, + rechnet genau umgekehrt. Beide Routinen belegen 28 Bytes.

Rechner-Aus. Um den Rechner gegen Einschalten während des Transportes oder gegen unerwünschte Anwender zu sichern, verwendet man am besten folgende synthetische Routine: RCL b, SF 11, OFF, STO b. Wer Synthetik nicht mag, kann statt RCL b auch LBL 00, und GTO 00 statt STO b verwenden. Um den Rechner wieder in Betrieb zu nehmen, schaltet man ihn über ON bei gedrückter R/S-Taste wieder ein. Statt der R/S-Taste kann natürlich auch die Taste verwendet werden, jedoch läßt sich dann von den gespeicherten Programmen und Daten nicht mehr ganz so effizienter Gebrauch machen.

Doppelter Durchlauf. Um ein Programmteil exakt zweimal zu durchlaufen, wählt man zweckmäßigerweise folgende Anordnung: SF xy, LBL 00 (Programmteil), FS?C xy, GTO 00.

MOD. Um mit der MOD-Funktion $x - (y \text{ mod } x)$ zu berechnen, kann man ganz einfach auch $-y \text{ mod } x$ berechnen. Das liegt am Fehler in der MOD-Funktion (s. Anhang Rechner-Bugs).

(Nach-)Kommastellen. Soll eine Zahl aufgespalten werden, z.B. in Vor- und Nachkommateil, so verwendet man INT und FRC. Um jedoch bei einer Zahl z.B. folgenden Formats: aa,bbccc zu bestimmen, wie ccc aussieht, gibt man ein: ,01, MOD und evtl. LASTX, /;. In anderen Fällen muß statt ,01 eine andere Zahl gewählt werden, nämlich die Zehnerpotenz der Ziffer, die unmittelbar vor dem ersten c steht. Wegen des Fehlers der MOD-Funktion müssen beide Zahlen positiv sein.

Letztes RTN. Ein letztes RTN oder das END des letzten Programms sind überflüssig, denn während des Programmlaufs verhalten sich RTN, END und .END. absolut gleich.

ALPHA-LABELS. Die globalen ALPHA-LBL's sollten höchstens 6 Zeichen haben, da sie sich sonst nicht mehr indirekt adressieren lassen.

Endgültiges STOP. Damit innerhalb eines Programms angehalten wird und die Ausführung nicht mit R/S fortgesetzt werden kann, kann man entweder: LBL 00, STOP, GTO 00 oder STOP, CF 30 (=NONEXISTENT, jedes R/S nach dem Programmstop führt zu dieser Meldung) programmieren.

Falsche Zeilennummerierung ist das Ergebnis, wenn im TRACE-Modus mit SST eine LBL-Anweisung ausgeführt wird. Die vom Drucker ausgegebene Zeilennummer ist um 1 höher als die im Programm.

Formatierung. Wer über das X-FUNCTION-Modul, aber nicht über den TL-Drucker verfügt, kann mit folgender Routine den Ausdruck formatieren: LBL "FORM", ACA, ARCL X, ALENG, 24, -, SKPCHR, CLA, ARCL Y, ACA, ADV. Der Text wird dabei in ALPHA, und die Zahl in X übergeben. ACX akkumuliert den X-Wert so, daß er beim Ausdruck linksbündig erscheint. Um jedoch mehrere Zahlen rechts untereinander zu drucken, muß folgende Routine benutzt werden, wenn X-FUNCTION nicht verfügbar ist: LBL "ACXR", RND, STO Z, X=O?, GTO O1, ABS, LOG, INT, X<=O?, GTO O1, ST-Y 3, /, INT, FS? 29, ST- Y, LBL O1, RDN, X<O?, CLX, SKPCHR, RDN, ACX, END. In Y wird die Anzahl der Stellen zwischen rechtem Rand und Dezimalpunkt minus 2, und in X die zu formatierende Zahl übergeben.

SIZE läßt sich ohne synthetische Befehle so ermitteln: LBL "SIZE", CLST, 511, FIX O, LBL OO, INT, X=Y?, RTN, STO Z, RCL Y, SF 25, LBL O1, +, 2, /, ARCL IND X, FC? 25, GTO OO, RND, STO Y, R, GTO O1.

Kettenrechnungen mit konstanter Größe lassen sich in UPN einfach realisieren:

- Addition: Konstante eintasten, ENTER, ENTER, ENTER, 2. Operanden eintasten. Mit jedem + wird die Konstante zum Inhalt von X addiert.
- Subtraktion: Wie Addition, bevor die Konstante geENTERt wird, noch schnell CHS betätigen.
- Multiplikation: Konstante eintasten, ENTER, ENTER, ENTER, 2. Operanden eingeben. Nach jedem * wird der Inhalt von X mit der Konstanten multipliziert.
- Division: Wie Multiplikation, jedoch vor dem ENTERn schnell noch 1/X betätigen.

Eine mögliche Anwendung wäre z.B.: Ein Sparer legt zu 7 % 6 Jahre 2385,15 DM an. Wir berechnen den Zinsfaktor q. Der ist nämlich Zins durch 100 plus 1, hier also 1,07, nun 3 x ENTER und 2385,15 eintasten. Nach dem ersten * steht in der Anzeige, was die Bank nach einem Jahr auf's Konto schreibt: 2552,11. Nach jedem weiteren * steht in der Anzeige der neue Betrag (mit Zins und Zinseszins).

WALL-Karten führen zu Memory-Lost, wenn der Lesevorgang abgebrochen wird. Um das zu verhindern, entfernt man kurz das Batteriepaket, bevor der Lesevorgang beendet wird.

Zufallszahlen lassen sich einfach mit folgendem Generator erzeugen: LBL 00, RCL 00, R-D, FRC, STO 00. Als Startzahl dürfen allerdings weder 0 noch ganzzahlige Vielfache von Pi verwendet werden.

Rundung. Statt mit FIX 0, RND auf ganze Zahlen zu runden, und dabei das Anzeigeformat zu verändern, verwendet man ,5; +; INT (für positive x) oder ,5; -; INT; (für negative x) oder ENTER; SIGN; ,5; *; +; INT (alle x).

Kartenleser anschließen. Nach dem Aufsetzen des Kartenlesers hat der Rechner einen erhöhten Stromverbrauch. Um diesen wieder auf das normale Niveau abzusenken, betätigt man 2 x ON.

Statuskarten. Wer seinen Status von Magnetkarten einliest, sollte beachten, daß beim Aufzeichnen mit gesetztem Flag 14 dieses auf der Karte gelöscht ist. Wer den Drucker angeschlossen hat, muß evtl. SF21 ausführen, um diesen zu korrekter Arbeitsweise zu bewegen.

Zwei Zahlen sollten nicht in zwei aufeinander unmittelbar folgende Programmzeilen geschrieben werden (z.B. 06 33,6; 07 3 E8), denn zwischen beiden steht ein (unsichtbares) NULL-Byte, das sie trennt und auch nicht durch PACK zu entfernen ist. Zweckmäßigerweise baut man statt dessen hier trennende Befehle ein, die sowieso im Programm vorkommen, z.B. SF 27, RAD, CLA, AON, TONE, FIX usw., oder die erste durch mathematische Operationen erzeugen; statt 25 12321; 26 ,978 besser: 24 111; 25 X^2; 26 ,978.

Der Status, den ein Programm braucht, läßt sich wie beim guten alten HP-67/97 auch auf dem HP-41C auf die Magnetkarte aufzeichnen: Statusbefehle (z.B. SF 00, RAD usw.) in die ersten Zeilen des Programms schreiben, und dann mit SF 11 aufzeichnen. Wer so jedoch arbeitet, braucht im allgemeinen keinen Permanentpeicher.

256 Zeichen mit BLDSPEC. Die Zeichen 0-127 lassen sich mit BLDSPEC ganz einfach so erzeugen: 0, ENTER, xyz, BLDSPEC, wenn xyz die Nummer des Zeichens ist (82143A-Handbuch, s.S. 39). Die Zeichen 128 - 255 müssen dagegen so erzeugt werden: 0, ENTER, 1, BLDSPEC, abc, BLDSPEC, wobei abc die Nummer des Zeichens minus 128 ist.

VIEW & AVIEW - Drucker betriebsbereit? Programme, die VIEW und AVIEW-Befehle benutzen, laufen nur dann richtig, wenn Flag 21 richtig gesetzt ist: Setzt man es ohne Drucker, halten die Befehle das Programm an, ebenso bei angeschlossenem und ausgeschaltetem Drucker. Ist Flag 21 gelöscht, läuft das Programm immer durch, doch der eingeschaltete Drucker arbeitet nicht. Mit den folgenden Schritten läßt sich das Flag 21 immer richtig einstellen: CF 21, SF 25, CLX, SKPCOL, FS?C 25, SF 21.

Wer seinen Status synthetisch mit einer Textkette, RCL M, STO d einstellt, sollte dabei Flag 25 und 55 setzen, und Flag 21 löschen.

CATALOG-Ausdruck. Die 3 CAT's lassen sich nur im TRACE-Modus auflisten. Dabei wird jedoch immer auch CAT und die Nummer ausgedruckt. Um das zu vermeiden, drückt man im MAN-Modus nach dem Start mit CAT sofort R/S (wer zu langsam war, muß mit BST wieder an den Anfang). Zum Ausdruck nun in den TRACE-Modus und BST, R/S ausführen.

Druckerlistings sollten nicht mit SF 12 im NORM-Modus ausgedruckt werden, da sich im Listing sonst nicht die Anzahl der Leerstellen abzählen läßt. Beispiel:

Belegung des Druckbuffers. Wer nicht über den IL-Drucker verfügt und viel mit Sonderzeichen arbeiten will, sollte Leerspalten/-zeichen nicht mit O ACCOL/SPACE ACA erzeugen, sondern SKPCOL und SKPCHR benutzen: SKPCOL 0-7 und SKPCHR 0-23 belegen im Druckbuffer nur eine Zelle. SKPCOL 8-167 wird in SKPCHR 0-23 und SKPCOL 0-7 zerlegt, benötigt also nur 2 Zellen.

Überlauf des Buffers. Wer mit ACA, ACSPEC, ACX, ACCHR, ACCOL, SKPCOL oder SKPCHR den Druckbuffer auffüllt und überlaufen läßt, kann mit PRBUF oder ADV vorher bestimmen, ob rechts- oder linksbündig der überlaufende Inhalt gedruckt wird.

MRG. Mit MRG lassen sich nur an nicht geschützte Programme andere anhängen. In einem PRIVATE-PRGM führt MRG zur Meldung PRIVATE. Umgekehrt lassen sich jedoch an nicht geschützte Programme solche, die mit PRIVATE geschützt sind, ohne weiteres anhängen.

END. Listet man mit CAT 1 im TRACE-Modus die Bytezahlen der Programme im RAM auf, dann ist das END jedes Programms mit 3 Bytes eingerechnet (wenn am Ende des Programms .END. steht, dann werden noch mehr Bytes dazugerechnet). Beim Aufzeichnen auf Magnetkarte ist dieses END bedeutungslos, da es sowieso übernommen wird. So kann ein Programm mit 115 Bytes auf eine Kartenseite, mit 227 Bytes auf eine Karte, mit 339 auf 1 1/2 Karten usw., ohne daß Informationen verlorengehen.

PRIVATE oder nicht? Mit der Sequenz: E 8, X<>d, PSE, CLA, FC? 50, "KEIN", X<>d, "PRIVATE" zeigt der Rechner an, ob ein PRGM geschützt ist.

CLD löscht die Anzeige nach VIEW oder AVIEW-Befehlen.

NOP. Als NOP (No Operation)-Befehle lassen sich die Befehle X<>X oder T † oder CLD verwenden. Als synthetischer 1-Byte-Befehl empfiehlt sich FO (240).

Verdoppeln einer Zahl in X kann auf 3 Arten geschehen:

1. einfach mit 2 multiplizieren. Dabei geht T verloren, und in L steht 2.
 2. ENTER↑, +; T geht verloren, und X steht in L
 3. ST+ X; X wird verdoppelt, sonst geht nichts verloren und L wird nicht verändert.
-

Bytesparen mit Zifferneingaben. Es gibt verschiedene Möglichkeiten, bei der Zifferneingabe im Programm die wertvollen Bytes zu verschwenden. Um das zu vermeiden, sollte man folgendes beachten:

1. Führende Nullen sind unnötig: 0,1 und ,1 bewirken den gleichen X-Register-Inhalt, doch die 0 kann gespart werden.
2. Nachfolgende Nullen sollten durch EEX ersetzt werden: Statt 123000 sollte 123 E3 (aber nicht 1,23 E5) verwendet werden.
3. Bei Zahlen zwischen 10^{10} und 10^{20} lassen sich durch Kommaverschiebung um eine Ziffer im Exponenten kürzen: Statt 1,6438766 E12 oder 4,77779213 E18 lassen sich mit 1643,8766 E9 oder 477779213 E9 ein bzw. 2 Bytes sparen.
4. Auch bei Zahlen um 10^{-10} läßt sich so sparen: Statt 6.45 E-10 spart ,645 E-9 oder auch 645 E-12 ein Byte.
5. Von Synthetik-Freaks längst zum Standard erhoben ist: E3 statt 1 E3. Wer jedoch auf Synthetik verzichten will, kann auf dem HP-67/97 alle Exponenten zwischen -99 und +99 erzeugen, und in den 41C(V) als Programm einlesen - auch hier fehlen die Einsen. Einfacher geht es jedoch so: $3;10^x$ für E3, $-26;10^x$ für E-26, $88;10^x$; CHS statt-E88 oder $-7;10^x$; CHS statt -E-7. Die Bytezahl ist die gleiche, nur verändert die Funktion 10^x das LASTX-Register und dauert wesentlich länger als eine Zifferneingabe.
6. Statt Ziffern explizit einzugeben, sollte man versuchen, diese zu programmieren, als Folge von mathematischen Operationen: Statt ,3333333333 besser 3; 1/X.
7. Versuchen, mit der Gegenoperation und dem Gegenelement Bytes zu sparen: Statt mit 2,5 zu multiplizieren, durch den Kehrwert von 2,5 (also ,4) dividieren. Statt -3 zu addieren, 3 subtrahieren.
8. Oft wird bei der Schleifensteuerung ein solcher Wert eingegeben: 16,020, obwohl 16,02 genügt. Im Falle von DSE und ISG braucht die Schrittweite 1 nicht angegeben zu werden: Statt 489,60001 genügt 489,6;

Bytesparen mit Tests. Oft lassen sich 2 Tests mit 2 Sprungbefehlen zu 2 Tests und einem Sprungbefehl vereinigen. Das ist nämlich immer dann der Fall, wenn diese Tests ein logisches ODER darstellen, wenn also nur eine der beiden Bedingungen erfüllt sein muß, damit verzweigt wird. In der Praxis wird da oft so programmiert: TEST 1, SPRUNG zu xyz, TEST 2, SPRUNG zu xyz, also z.B.: X=0?; GTO O3; FC? O7; GTO O3. Um das nun zu vereinfachen, ersetzt man die erste Testbedingung durch ihre Umkehrung (z.B. X=0? durch X≠0?) und löscht den ersten Sprungbefehl. Das Ergebnis sieht dann so aus: X≠0?; FC? O7; GTO O3. Dabei gehören folgende Tests zusammen, d.h. folgende Tests sind ihre gegenseitige Umkehrung: X=0? und X≠0?, X=Y? und X≠Y?, X>Y? und X≤Y?, X>O? und X≤O?, FS? und FC?. Zu X<Y? und X<O? fehlen auf dem HP41 die Umkehrungen X>Y? und X>O?; Wer genau nachdenkt, wird merken, daß unser oberes Beispiel genau-sogut funktioniert, wenn es FC? O7; GTO O3; X=0?; GTO O3 lautet. Aber kann jetzt auch noch zusammengefaßt werden? Ja, nämlich zu: FS? O7; X=0?; GTO O3. Mit diesem Wissen können wir nun auch dem Rechner die Tests programmieren, die nicht vorhanden sind: X>Y? und X>O? lassen sich so also durch X≠Y?; X>Y? bzw. X≠Y?; X=Y? und X≠O?; X>O? bzw. X≤O?; X=O? ersetzen.

Noch ein Hinweis: Taucht in einem solchen Testfall einmal der Test FS?C oder FC?C auf, so muß er seine Position als erster oder zweiter Test behalten. Würde nämlich die Sequenz: FS?C 23; GTO A; FC? 25; GTO A statt zu FC?C 23; FC? 25; GTO A einfach in FS? 25; FS?C 23; GTO A umgewandelt, würde das Programm vielleicht Unfug produzieren: Bei der ersten Version erreicht das Programm immer den Test FC?C 23, der Flag 23 immer löscht. Bei der zweiten Version erreicht das Programm diesen kombinierten Test-/Löschbefehl nur, wenn Flag 25 gesetzt ist - also Vorsicht!

Ein Crash legt den Rechner lahm, wenn mit 64 BLDSPEC ein Sonderzeichen erzeugt wird, und dann XEQ IND X ausgeführt wird.

Die Funktionen fehlender Peripherie-Einheiten lassen sich mit den Barcode-Tabellen im Anhang einfach programmieren. Es geht aber auch anders: Entweder man gibt z.B. XEQ "PRX" ein, oder man adressiert indirekt, z.B. "MRG", ASTO X, XEQ IND X. Doch Achtung: die nicht programmierbaren Funktionen (LIST, PRP, VER, WALL, WPRV und NEWM) lassen sich so nicht ausführen. Außerdem suchen diese Befehle erst im CAT 1, und dann in CAT 2 nach dem Programmlabel oder der Funktion. Und schließlich braucht ein XEQ "WNDDTX" 8 Bytes, während die XROM-Nummer nur 2 benötigt.

Das Einlesen von Programmen mit Kartenleser oder Wand löscht immer das, was zwischen dem letzten END (falls überhaupt eines vorhanden ist) und der eingebauten .END.-Anweisung. Das gilt nicht für programmiertes Laden von Programmen: RSUB, WNDSUB und WNDLNK hängen erst noch ein END an den bisherigen Speicherinhalt an und überschreiben das, was nach diesem END steht.

Unterprogrammrückspungadressen werden von folgenden Operationen gelöscht: SIZE (aber nicht PSIZE!), GTO .000, GTO .., RTN über die Tastatur, PACK, CAT 1, DEL, , Einfügen von Zeilen und XEQ-Befehle, die nicht zur Fehlermeldung NONEXISTENT führen.

Energie-Test. Besonders in Programmen, die auf frisierten (schnellen) Rechnern laufen, sollten im Programm die Sequenz FS? 49, OFF enthalten, da gerade die schnellen Rechner sonst die gespeicherten Programme und Daten modifizieren oder gleich ganz löschen, wenn die Batterien/Akkus nachlassen.

Startverzögerungen nach R/S. Oft kommt es nach dem Drücken von R/S zu einer Verzögerung des Programmstarts. Wer die Taste nämlich zu lange drückt, veranlaßt den Rechner, die Zeilennummer und den Zeileninhalt herauszusuchen, es ist also der gleiche Vorgang, der BST verzögert, wenn es am Ende eines langen Programms ausgeführt wird.

Ersatz für manche Formelzeile findet man in vielen Standardfunktionen, z.B. statt der Schrittfolge x , 100, /, geht es mit % viel besser. Andere Beispiele (x = Inhalt des X-Registers, y = Inhalt des Y-Registers): $x \cdot y / 100 = \%$; $y \cdot \sin(x)$ nach P-R in Y, $y \cdot \cos(x)$ nach P-R in X:

ENTER, 1, R-P = $\sqrt{1+x^2}$	ASIN, COS = $\sqrt{1-x^2}$	ATAN, COS = $1/\sqrt{1+x^2}$
ATAN, SIN = $x/\sqrt{1+x^2}$	ASIN, TAN = $x/\sqrt{1-x^2}$	ACOS, TAN = $\sqrt{1-x^2}/x$
D - R = $\frac{\pi}{180} \cdot x$	R - D = $\frac{180}{\pi} \cdot x$	X \uparrow 2, LOG = $2 \cdot \log(x)$
\sqrt{x} , LOG = $1/2 \log(x)$		
R-P: $X = \sqrt{x^2 + y^2}$	Y = arctan ($\frac{y}{x}$)	%CH = $(\frac{x}{y} - 1) \cdot 100$

Zeit und Bytes spart, wer statt folgender Routine: LBL "ABC", (Programm), GTO "ABC" diese nimmt: LBL "ABC", LBL 00,(Programm),GTO 00. Bei Sprüngen über mehr als 112 Bytes: LBL "ABC", LBL 15,(Programm), GTO 15.

SIGN. Die Funktion SIGN weicht von der mathematischen Signumfunktion an der Stelle 0 ab: sign (0) =0, aber 0, SIGN ergibt 1. Um das zu umgehen, verwendet man: X≠0?, SIGN oder, wenn auch LASTX den richtigen Inhalt haben soll: STO L, X≠0?, SIGN.

Flag-Umkehr. Um ein gesetztes Flag zu löschen oder zu setzen, falls es gelöscht war, bedient man sich der Sequenz: FC?C nn, SF nn.

Anzeigenwechsel. Soll eine VIEW/AVIEW-Meldung so lange in der Anzeige bleiben, bis sie der Anwender gelesen hat, dann lassen sich folgende Routinen anwenden: LBL 00, PSE, FC? 47, GTO 00. Wenn SHIFT gedrückt wird, setzt der Rechner das PRGM fort. Statt FC? 47 kann auch FC? 27 verwendet werden, dann muß allerdings die USER-Taste betätigt werden oder auch FC? 48, dann muß allerdings die ALPHA-Taste betätigt werden.

Dateneingabe. Die Schleife CF 22, LBL 00, PSE, FC?C 22, GTO 00 bzw. AON, CF 23, LBL 00, PSE, FC?C 23, GTO 00 läßt den Rechner so lange warten, bis die Daten eingegeben werden. Hat man einen Teil der Eingabe vergessen, drückt man die letzte Zahl/Alphazeichen so lange nieder, bis eine Erleuchtung die vergessenen Informationen wiederbringt.

SIZE-Test

..... Wenngleich inzwischen schon viele SIZE-Routinen veröffentlicht wurden, so möchte ich doch meine noch anbieten, obwohl sie vom Speicherbedarf mit 41 Bytes etwas zu dick geraten ist. Der Vorteil liegt aber eindeutig in den nahezu konstanten Laufzeiten, egal ob nun SIZE 000 oder SIZE 310 eingestellt sind (jeweils 3,5 bis 3,8 sec.).

SIZE-ROUTINE
VON FRIEDR. HILLEBRANDT

Friedrich Hillebrandt

```
01+LBL "S?"
02 CLST
03 511
04 FIX 0
05+LBL 00
06 INT
07 X=Y?
08 RTN
09 STO Z
10 RCL Y
11 SF 25
12+LBL 01
13 +
14 2
15 /
16 ARCL IND
X
17 FC? 25
18 GTO 00
19 RND
20 STO Y
21 R↑
22 GTO 01
23 END
```

+	+	
-	-	
*	*	
/	:	
1/X	1/x	
10^X	10 ^x	
ABS	x	- absolute value
ACOS	arc cos	- arc cosine
ADV	Papiervorschub	- advance
AOFF	α aus	- alpha mode off
AON	α ein	- alpha mode on
ARCL	zurückrufen	- alpha recall
ASHF	Löscht 6 α Zeichen	- alpha shift
ASIN	arc sin	- arc sine
ASN	Fkt. Zuordnung	- assign
ASTO	α speichern	- alpha store
ATAN	arc tg	- arc tangent
AVIEW	α betrachten	- alpha view
BEEP	Akust. Signale	- beep
BST	1 Schritt zurück	- backstep
CAT	Catalog	- catalog
CF	Flag löschen	- clear flag
CHS	Vorzeichenwechsel	- change sign
CLA	Cl. Register	- clear alpha register
CLD	Cl. Anzeige	- clear display
CLP	Cl. Programm	- clear program
CLRG	Cl. Datenspeicher	- clear registers
CL [Cl. der 6 Statistik R.	- clear statistical registers
CLST	Cl. Stack	- clear stack
CLX	Cl. x Register	- clear x
COPY	Kopieren oder Um- speichern von Progr.	- copy
COS	cos	- cosine
D-R	Altgrad → Bogenmaß	- degrees to radians
DEC	Octal → Dezimal	- octal → decimal
DEG	Altgrad-Modus	- degrees

DEL	Cl. von n Progr. Zeilen	- delete
DSE	Decrement (Sp. wenn =)	- decrement and skip if equal
END	Programm Ende	- end
ENG	Tech. Anz. Format	- engineering notation display
ENTER↑	Kopieren X in Y-Reg.	
E↑X	e^x	
E↑X-1	e^{x-1} (für Argumente nahe 0)	
FACT	$x!$	- factorial
FC ?	Flag gelöscht ?	- flag clear test
FC ? C	Flag gelöscht ? + Cl.	- flag clear test and clear
FIX	Festkomma Format	- fixed point display
FRC	Dezimal Anteil e. Zahl	- fraction
FS ?	Flag gesetzt ?	- flag set test
FS ? C	Flag gesetzt ? + Cl.	- flag set test and clear
GRAD	Neugrad-Modus	- grads
GTO	Sprungbefehl	- goto
HMS	Stunden → h, mmss	- decimal hours to hours, minutes, seconds
HMS +	Addition von h, mmss	- add hours, minutes, seconds
HMS -	Subtrak. von h, mmss	- subtract hours, minutes, seconds
HR	h, mmss → Stunden	- hours, minutes, seconds to decimal hours
INT	Ganzzahl. Anteil	- integer
ISG	Inkrement (Sprung w.)	- increment and skip if greater
LASTX	vorletzte Zahl in X	- last x
LBL	Programm Marke	- label
LN	ln	- natural log
LN1+X	ln (1+x) (f. Arg. nahe 1)	- natural log for arguments close to
LOG	log	- common log one
MEAN	arithmet. Mittel	- mean
MOD	Modulo Funktion	- modulo
OCT	Dezimal Oktal (DEC)	- decimal to octal
OFF	Ausschaltung	
ON	Einschaltung	
P-R	Polar rechth. K.	- polar to rectangular
PACK	Packen des Prgm. Sp.	
%	%	

%CH	Proz. Unterschied	- percent of change
PI		- pi
PROMPT	Textausgabe	- prompt
PSE	Programm Pause	- pause
R↑	R	- roll up
R-D	Bogenm. Altgrad	- radians to degrees
R-P	Rechtw. Polarkoord.	- rectangular to polar
RAD	Bogenmaß Modus	- radians
RCL	Zurückrufen aus Sp. R.	- recall
RDN	R	- roll down
RND	Runden	- round
RTN	Return	
	Unterprogramm Ende	- return
SDEV	Standard Abweichung	- standard deviation
SCI	Wissensch. Format	- scientific notation display
SF	Flag setzen	- set flag
Σ+	+ in mehreren Reg.	
Σ-	- in mehreren Reg.	
ΣREG	Definieren des Statistikblocks	
SIN	sin	- sin
SIGN	Vorzeichen von x	- sign
SIZE	Anzahl der Datenreg.	- size of data register allocation
SORT	\sqrt{x}	- square root
SST	1 Schritt vorwärts	- single step
ST+	Datenregister +	
ST-	Datenregister -	
STx	Datenregister x	
ST/	Datenregister :	
STO	Speichern im Dar. Reg.	- store
STOP	R/S	- run/stop
TAN	tg	- tangent
TONE	1 akust. Signal	- tone
VIEW	Betrachten d. Reg. Inh.	- view

X=O?	x = 0?	} Bei Nein: Sprung
X≠O?	x ≠ 0?	
X(O?	x < 0?	
X(=O?	x ← 0?	
X)O?	x > 0?	
X=Y?	x = y?	
X≠Y?	x ≠ y?	
X(Y?	x < y?	
X(=Y?	x ← y?	
X)Y?	x > y?	
X()	Austausch X mit Y.Z,	
X()Y	T,L oder Daten Reg.	
X*Y	x * y	
XEQ	Aufruf einer Fkt.	- execute
X^2	x ²	
Y^X	y ^x	
EEEX	Exponent von 10	- enter exponent of x
←	Cl. letztes Z. der Eing.	
	Cl. Anzeige (ohne _)	

Routinen

Die ACXR-Routine von K.W. Hoenow ist sehr gut, auch ich habe mich mit dem Problem beschäftigt, bin aber auf längere Laufzeiten gekommen. Das Programm funktioniert allerdings nur, wenn Fig.29 gelöscht ist. Ich habe das Programm etwas geändert, so daß große Zahlen auch mit Trennzeichen richtig untereinander geschrieben werden.

```
01+LBL "ACXR"
02 RND
03 STO 2
04 X=0?
05 GTO 01
06 ABS
07 LOC
08 INT
09 X=0?
10 GTO 01
11 ST- Y
12 J
13 /
14 INT
15 FS? 29

16 ST- Y
17+LBL 01
18 RND
19 X=0?
20 CLX
21 SUPCHR
22 RND
23 ACX
24 END

LBL"ACXR"
END . 41 BYTES
RY=ANZ. STELLEN VOR KOMA
EINSCHL. VORZ. -2
RX=DRUCKWERT
```

56 Walter Kropf

Wollt Ihr ein Teil-Listing der Hex-Code Tabelle ? Falls ja, so schließt den Drucker an den 41c, Trace Modus und drückt folgende Tasten: Ø STO d 1,080002 EEX 20 STO d R/S.

Happy Programming!

Norbert Weber (44)

Das folgende Programm zeigt die 10-stellige Mantisse m der in X stehenden Zahl z an, wobei $z = s m 10^t$ mit s = Vorzeichen von z und o kleiner gleich m kleiner 10 ist und t eine ganze Zahl. Der Stack bleibt erhalten, ebenso das Anzeigeformat. Der Inhalt von L und A (und Q) geht verloren (z steht in L). Die Mantisse steht in A und in der Anzeige (mit Clear löschen): LBL "MANT" ABS STO Q CLX RCL d SCI 9 " " ARCL Q STO d ASTO X ASHF ASTO Q CLA ARCL X ARCL Q CLX LAST X AVIEW END

Johannes Schu (129)

Die Statuskarte als Programmanfangskarte!

Als erste Karte eines Programms kann man immer eine Statuskarte erstellen.

Die Vorteile:

Schaltet zum Beispiel den USER-Modus ein. Dadurch können alle Tastenfeldzuordnungen mit eingelesen werden (Vergessen ist menschlich). Man braucht kein SIZE mehr auszuführen. In der Anzeige kann sofort die erste Programminformation stehen. Im Alpharegister steht die erste Druckzeile. In den Registern Y; Z; T; L können Werte oder Alphazeichen stehen, die man sofort verwenden kann.

Als Nachteile sind anzuführen:

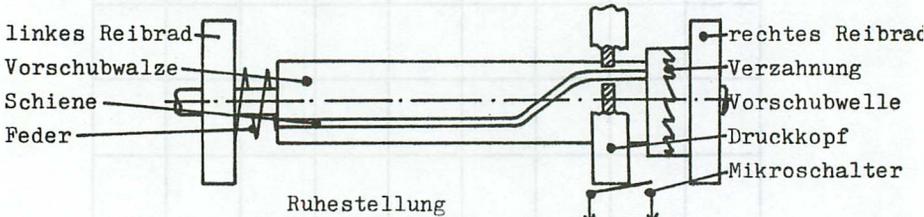
Es muß immer eine Karte mehr eingelesen werden. Beim zweiten Programmstart muß erst immer wieder die Statuskarte eingelesen werden, um gleiche Anfangsbedingungen zu haben.

	<p>- 2. -</p> <p>Barcode-Programme</p>							

Bar-Codes und Graphik mit dem HP-41C / Umbauanleitung

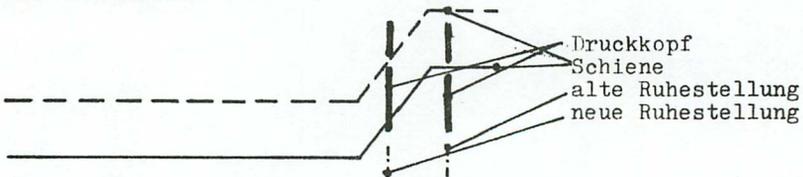
Der normale Druckervorschub

Der Drucker HP 82143A hat einen mechanischen Vorschub, der eine stufenlose Regelung unmöglich macht. Gesteuert wird er von einer auf der Vorschubwalze befindlichen Schiene und einem Mikroschalter, der die Ruhestellung festlegt. Während sich der Druckkopf nach links bewegt, wird die Zeile ausgedruckt, die Vorschubwalze dreht sich nach oben und überspringt 2 Zähne in der Verzahnung. Die beiden durch die Vorschubwelle miteinander verbundenen Reibräder stehen währenddessen still und sorgen dafür, daß sich der Papierstreifen nicht bewegt. Nach Ausdruck der Zeile bewegt sich der Druckkopf wieder nach rechts und dreht die Vorschubwalze über die Schiene wieder nach unten, bis er den Kontakt im Mikroschalter schließt und der CPU damit mitteilt, daß er sich in Ruhestellung befindet. Der Motor wird abgeschaltet, der Druckvorgang für diese Zeile ist beendet. Die Drehung der Vorschubwalze wurde über die Verzahnung auf die Vorschubwelle und damit auf die Reibräder übertragen, die wiederum den Papierstreifen um ca. 4 mm nach oben bewegen.



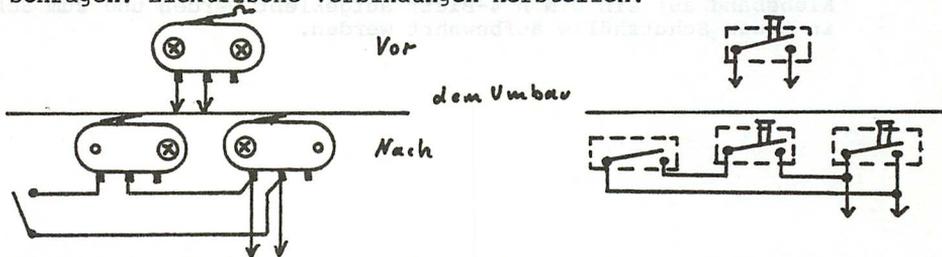
Der veränderte Druckervorschub

Verschiebt man den Mikroschalter nach links, so verschiebt sich auch die Ruhestellung und damit die ganze Druckzeile nach links. Wenn man die Ruhestellung weit genug nach links verschiebt, kann nur noch ein Zahn in der Verzahnung übersprungen werden, der Vorschub wird halbiert. Nun schlägt jedoch der Druckkopf bei mehr als 133 Druckspalten am Gehäuse an und eine eingebaute Überlastungssicherung schaltet den Motor aus. Nach Aus- und Einschalten des Druckers wird die Sperre aufgehoben. Um nun alle Möglichkeiten des Druckers zu erhalten, bauen wir zwei Mikroschalter ein, einen in der alten Ruhestellung für normalen Vorschub und einen in der neuen Ruhestellung für halben Vorschub (= Graphik). Ein weiterer, von außen zugänglicher Schalter erlaubt die Umschaltung des Vorschubes.



Umbaubeschreibung:

- : Wenn der Druckkopf festhängt, Drucker aus- und einschalten. :
 - : Sollte dies nicht helfen, schaltet entweder einer der Mikro- :
 - : schalter nicht durch oder eine Verbindungsleitung wurde ver- :
 - : gessen. :
1. 5 Gehäuseschrauben auf der Druckerunterseite lösen (eine in der Mitte, 4 unter den Gummifüßen) und das Gehäuse vorsichtig öffnen.
 2. Anschlüsse der zum Druckwerk führenden Drähte notieren und diese von der Platine abziehen.
 3. 3 Befestigungsschrauben des Druckwerks lösen und es vorsichtig herausheben. Druckwerk so drehen, daß Mikroschalter und Fenster nach oben weisen. Befestigungsschrauben des Mikroschalters lösen.
 4. Alten und neuen Mikroschalter nach Abbildung in den Befestigungslöchern des alten Mikroschalters festschrauben. Kanten des Druckkopfes an den Stellen, wo er von den Zungen der Mikroschalter berührt wird, mit der Schlüsselfeile ein wenig absträgen. Elektrische Verbindungen herstellen.



5. Verbindungskabel des Druckwerks wieder mit der Platine verbinden, Schalter auf Normal (= Aus), Papierrolle einlegen und einlaufen lassen.
6. Zunge des rechten Mikroschalters so verbiegen, daß er durchschaltet, bevor der Druckkopf an der rechten Seite anschlägt. Probelauf mit wiederholtem "ADV".
7. Schalter auf Graphik (= Ein), Zunge des linken Mikroschalters so verbiegen, daß Vorschub halbiert wird. Probelauf mit Druckermodus "MAN" und dem Programm FIX Ø, Ø, LBL ØØ, PRX, GTO ØØ. Stellen sich Unregelmäßigkeiten in den Zeilenabständen ein, muß die Zunge des Mikroschalters nachgebogen werden. Nicht die Nerven verlieren, es dauert etwas, bis die richtige Einstellung gefunden ist.
8. Vorschubschalter an geeigneter Stelle am Gehäuse anbringen und beschriften.
9. Druckwerk wieder einbauen, Gehäuse schließen, Probelauf wiederholen, fertig? - Hoffentlich nicht nur mit den Nerven!

Anwendung:

Wir haben nun 133 Spalten Schmalschrift = 19 Zeichen pro Zeile zur Verfügung. Mögliche Befehle in der Stellung Graphik sind: "ACCOL, BLDSPEC, ACSPEC, SKPCOL, SKPCHR, ACA, ACX" und zum Ausdruck der Zeile "ADV" (nicht "PRBUF"!). Es ist darauf zu achten, daß die oben angegebene maximale Spaltenzahl nicht überschritten wird. Alle anderen Druckerbefehle sollten nur in der Stellung Normal ausgeführt werden.

Hinweis

Ich übernehme keinerlei Haftung für Schäden, die durch diesen Umbau am Drucker auftreten. Sollten Sie Schwierigkeiten mit dem Umbau haben, stehe ich Ihnen gerne zur Verfügung. Falls Sie doch wünschen, daß ich den Umbau für Sie ausführe, senden Sie mir mit dem Drucker diese Anleitung zu. Ich berechne dann nur noch DM 65,- für den Umbau.

Papierauswahl

Nicht jedes Thermopapier ist zum Druck von Bar-Codes geeignet. Nicht verwendbar ist z.B. das blaue druckende Papier von HP und kontrastschwaches schwarzdruckendes Papier. Mir ist zur Zeit nur eine Sorte bekannt, die direkt eingelesen werden kann (lag dem Info als Originalausdruck bei, gekauft bei Firma Schüngel Datentechnik, Bonn), andere Sorten sind jedoch als Fotokopie gut lesbar (ausprobieren). Erkennen kann man die Papierqualität daran, ob die breiten Striche des Bar-Codes als 3 einzelne oder ein einziger Strich erscheinen. Ferner sollte das Papier wischfest sein und nicht verblässen. Die einzelnen Streifen können dann mit doppelseitigem Klebeband auf ein DIN A 4-Blatt aufgeklebt werden und zum Schutz in einer Schutzhülle aufbewahrt werden.

Bar-Code-Typen

Typ	Bezeichnung	Aufbau
1	Program Code	<ul style="list-style-type: none">- Byte 1 = Checksum = Summe aller Bytes außer Byte 1 (Rest der Division durch 256 + ganzzahliges Ergebnis der Division)- Byte 2 = Typ und Sequence Number = 16 + Nummer der Zeile - 1 (Rest der Division durch 16)- Byte 3 = Partial Funktion Code Information = Anzahl der Bytes, die zu einer in der letzten Zeile begonnenen Mehrbytefunktion gehören und in dieser Zeile stehen * 16 + Anzahl der Bytes, die zu einer in der nächsten Zeile endenden Mehrbytefunktion gehören und in dieser Zeile stehen- Byte 4 bis 16 = Programm
2	Private Progr. Code	<ul style="list-style-type: none">- Byte 1 wie Typ 1- Byte 2 wie Typ 1 + 16- Byte 3 wie Typ 1- Byte 4 bis 16 = Private Programm
4	Direct Execution	<ul style="list-style-type: none">- Byte 1 wie Typ 1- Byte 2 = 64- Byte 3 bis 11 = Befehl (z.B. XEQ ABC oder STO 25)
6	Numeric Data Code	<ul style="list-style-type: none">- Byte 1 wie Typ 1- Byte 2 = 96 + erstes Digit- Byte 3 bis 16 = Digits der Zahl (jede Ziffer wird durch ein Halbbyte dargestellt) Ø bis 9 = Ø bis 9, Dezimalpunkt = 11, - = 13, E = 14
7	Alpha Data Code	<ul style="list-style-type: none">- Byte 1 wie Typ 1- Byte 2 = 112 + Länge der Alpha-Kette- Byte 3 bis 16 = Textzeichen
8	Alpha Append Code	<ul style="list-style-type: none">- Byte 1 wie Typ 1- Byte 2 = 128 + Länge der Alpha-Kette- Byte 3 bis 16 = Textzeichen

Andere Typen lassen sich zwar auch erzeugen, werden jedoch nicht gelesen ("TYPE ERR").

Automatischer Ausdruck von Programm-Barcode mit Reihenbeschriftung

Schließen Sie an Ihren Rechner den umgebauten Drucker und das X-Function-Modul an. Ferner benötigen Sie die in der untenstehenden Tabelle angegebene Anzahl von Speichererweiterungs-Modulen (die Angaben beziehen sich auf SIZE 000).

Nun lesen Sie das zu druckende Programm in den Rechner ein. Dies kann auf beliebige Art und Weise geschehen, es ist jedoch zu beachten, daß dieses Programm als erstes im Rechnerspeicher stehen muß. Direkt hinter Ihr Programm muß das Programm ABARP geladen werden. Wenn Sie dabei mit WNDSUB oder RSUB und nicht mit GTO.. arbeiten, bleiben die in den Sprungbefehlen enthaltenen Sprungweiten auch im Barcode erhalten, ferner werden auch die Tasten- und Ordnungen mit übernommen. Vermeiden Sie es auch, nach dem Einlesen den Rechner zu packen. Setzen Sie Flag 10, falls der Barcode mit PRIVATE geschützt werden soll. Schalten Sie Ihren Drucker ein, auf MAN und auf halben Zeilenvorschub und rufen Sie das Programm ABARP mit XEQ ALPHA ABARP aLPHA auf. Nach kurzer Zeit wird die Länge des Programms in Byte ausgedruckt und der Rechner beginnt mit dem Druck des Barcodes. Der Programmablauf sollte nun nicht mehr unterbrochen werden.

Bedienung in Stichworten:

1. Peripherie anschließen
2. MEMORY LOST verursachen
3. SIZE 000
4. zu druckendes Programm einlesen
5. ABARP einlesen
6. Drucker: ON, MAN, halber Zeilenvorschub
7. SF 10 (PRIVATE) oder CF 10 (nicht PRIVATE)
8. XEQ ABARP

Tabelle: Maximale Länge des zu druckenden Programms

Länge des Programms in Byte	Anzahl der Memory- Module (41 C)	Anzahl der X- Memory-Module
549 Byte	2 Memory	--
840 Byte	3 Memory	--
997 Byte	3 Memory	1 X-Memory
1445 Byte	4 Memory	1 X-Memory

LISTING ZU PROGRAMM ABARP+ <792 BYTE>

+ + + + +

```

06*LBL "ABARP"
CLX PSIZE RCL b STO I
-2 AROT 1773 ATOX 16
/ INT - 7 ATOX
STO I * - RCL c
X<> I [-3 AROT RDN
253 - ATOX 16 *
X<>Y ST+ Y X<>Y LASTX
* ATOX + 256 / XTOA
LASTX * LASTX MOD
XTOA SIGN AROT X<> I
STO c RDN CHS ">"
SF 25 PURFL CRFLD
SAVER 30 PSIZE CLRG
Rt ADV FIX 0 CF 29
SF 12 ARCL X ACA 2 -
STO 05 Rt STO 06 7
X<>F SIGN STO 01 CLA
XEQ 11 GTO 08
    
```

```

02*LBL 15
ISG 03 GTO 02 XEQ 05
    
```

```

06*LBL 08
RCL 11 RCL 01 2 -
SIGN X>0? CLX FS? 03
CLX - STO 07 16.028
STO 03 INT RCL 01
DSE X "" * STO 15
CF 04 FC? 00 SF 04
    
```

```

109*LBL 02
DSE 05 GTO 02 "+/" 3
STO 04 STO 05 FS?C 01
GTO 08 FS?C 02 XEQ 05
BEEP RTN
    
```

```

122*LBL 02
DSE 04 GTO 08 7
STO 04 DSE 06 ""
RCL 06 SEEKPT GETX
STO I
    
```

```

133*LBL 08
SF 02 ALENG RCL 04
X=Y? 0 X=Y? ATOX
ENTERt STO IND 03
ST+ 13 DSE 01 GTO 15
FS?C 03 GTO 02 2 /
RND 15 X=Y? SF 03
RCL Z LASTX 8 / INT
1 X=Y? SF 00 X=Y?
GTO 08 FS?C 04 DSE 07
"" FS?C 00 ISG 11
    
```

```

169*LBL 08
RDN 12 X=Y? SF 03
RCL Z Rt 2 / INT.
103 X=Y? CF 03 RCL Z
FC? 00 GTO 02 X=0?
ISG 11
    
```

```

187*LBL 02
CF 04 143 - X<=0? 97
64 - X<=0? 34 32 -
X<=0? 3 STO 01 STO 09
GTO 15
    
```

```

204*LBL 05
CF 09 CF 02 RCL 08 16
MOD LASTX FS? 10
ST+ X + ST+ 13 STO 14
ISG 08 "" RCL 09
RCL 01 DSE X - RCL 15
+ STO 15 RCL 13 +
255 MOD X=0? LASTX
STO 13 .ASTO 29
"REIHE " ARCL 08 "- (-"
ARCL 07 "- (-" ARCL 11
"-)" RCL 03 INT 12999
+ E3 / STO 03 .09
STO 12 SIGN STO 09
CHS STO 07 CLX STO 10
X<>F STO 02 XEQ 12
    
```

```

258*LBL 13
7 CHS STO 07
RCL IND 03 X<>F XEQ 12
ISG 03 GTO 13 1 CHS
STO 07 2 X<>F XEQ 12
XEQ 10 RCL 02 X<>F
    
```

```

276*LBL 11
ALENG X=0? XEQ 09
X=0? GTO 11 XEQ 07
"-----" ACA 8 CHS
STO 12
    
```

```

288*LBL 07
ADV ISG 12 GTO 07 CLA
ARCL 29 RTN
    
```

```

295*LBL 12
7 RCL 09 FS? IND 07 *
ST+ 10 4 FS? IND 07
X+2 ST* 09 16 RCL 09
X<=Y? GTO 14 X<>Y
ST+ X / STO 09
    
```

```

313*LBL 10
CLX X<> 10 BLDSPC
LASTX BLDSPC LASTX
BLDSPC LASTX BLDSPC
LASTX BLDSPC LASTX
BLDSPC LASTX BLDSPC
ACSPC ACSPC ACSPC
ACSPC 2 SKPCOL
    
```

```

335*LBL 09
CLX FC? 09 ATOX
XEQ IND X CLX X<> I
ACSPC -7 AROT ADV
FC?C 09 SF 09 ISG 12
    
```

```

349*LBL 14
ISG 07 GTO 12 RTN
    
```

```

353*LBL 32
354*LBL 00
"+0+++++" RTN
    
```

```

357*LBL 48
"+0eY4p " RTN
    
```

```

360*LBL 49
"+0e0XT"A" RTN
    
```

```

363*LBL 50
"+050 " RTN
    
```

```

366*LBL 51
"+0uA I-" RTN
    
```

```

369*LBL 52
"+00Ev0" RTN
    
```

```

372*LBL 53
"+0vA2" RTN
    
```

```

375*LBL 54
"+0v0
A<" RTN
    
```

```

378*LBL 55
"+0A A I-" RTN
    
```

```

381*LBL 56
"+0v0" " RTN
    
```

```

384*LBL 57
"+0yA2" " RTN
    
```

```

387*LBL 69
"+0Aa
e" RTN
    
```

```

390*LBL 40
"+0==x0e0" RTN
    
```

```

393*LBL 41
"+0a0e==" RTN
    
```

```

396*LBL 02
"+0Bv0" " RTN
    
```

```

399*LBL 73
"+0e0xae" RTN
    
```

```

402*LBL 72
"+0I"vE" " RTN
    
```

```

405*LBL 45
"+0+0+2
++" END
LBL"ABARP
    
```

END 792 BYTES

Automatischer Ausdruck von Programm-Barcode

Schließen Sie Ihren Rechner an den umgebauten Drucker und das X-Function-Modul an. Ferner benötigen Sie die in der untenstehenden Tabelle angegebene Anzahl von Speichererweiterungsmodulen (die Angaben beziehen sich auf SIZE 000).

Legen Sie das zu druckende Programm und das Programm ABARP so in den Rechner ein, daß Ihr Programm an erster Stelle und das Programm ABARP an zweiter Stelle im Programmspeicher steht. Schalten Sie den Drucker ein und auf MAN und halben Zeilenvorschub. Setzen Sie Flag 10, wenn das Programm mit PRIVATE geschützt werden soll. Rufen Sie das Programm ABARP mit XEQ ALPHA ABARP ALPHA auf. Nach kurzer Wartezeit beginnt der Ausdruck des Barcodes. Der Programmablauf sollte nicht unterbrochen werden.

Bedienung in Stichworten:

1. Peripherie anschließen
2. Vorhandene Programme löschen
3. SIZE 000
4. zu druckendes Programm einlesen
5. ABARP einlesen
6. Drucker: ON, MAN, halber Zeilenvorschub
7. SF 10 (PRIVATE) oder CF 10 (nicht PRIVATE)
8. XEQ ABARP

Tabelle: Maximale Länge des zu druckenden Programms

Länge des Programms in Byte	Anzahl der Memory- Module (41 C)	Anzahl der X- Memory-Module
446 Byte	1 Memory	--
840 Byte	2 Memory	--
914 Byte	2 Memory	1 X-Memory
1462 Byte	3 Memory	1 X-Memory
1910 Byte	4 Memory	1 X-Memory

LISTING ZU PROGRAMM ABARP <426 BYTE>

+++++

```
06*LBL "ABARP"
CLX PSIZE RCL b STO I
-2 AROT 1773 ATOX 16
/ INT - 7 ATOX
STO I * - RCL c
X<> I -3 AROT RDM
253 - ATOX 16 *
X<>Y ST+ Y X<>Y LASTX
* ATOX + 256 / XTOA
LASTX * LASTX MOD
XTOA SIGN AROT X<> I
STO c RDM CHS "-"
SF 25 PURFL CRFLD
SAVER 30 PSIZE CLRG
R↑ FIX 0 CF 29 SF 12
2 - STO 05 R↑ STO 06
7 X<>F SIGN STO 01
CLA XEQ 11 GTO 08
```

```
79*LBL 09
ISG 03 GTO 02 XEQ 05
```

```
83*LBL 08
16.028 STO 03 INT
RCL 01 DSE X ** *
STO 15
```

```
92*LBL 02
DSE 05 GTO 02 "+/" 3
STO 04 STO 05 FS?C 01
GTO 08 FS?C 02 XEQ 05
BEPP RTN
```

```
105*LBL 02
DSE 04 GTO 08 7
STO 04 DSE 06 **
RCL 06 SEEKPT GETX
STO I
```

```
116*LBL 08
SF 02 ALENG RCL 04
X*Y? 0 X=Y? ATOX
ENTER↑ STO IND 03
ST+ 13 DSE 01 GTO 09
143 - X<=0? 97 64 -
X<=0? 34 32 - X<=0?
3 STO 01 STO 09
GTO 09
```

```
144*LBL 05
CF 09 CF 02 RCL 08 16
MOD LASTX FS? 10
ST+ X + ST+ 13 STO 14
ISG 08 ** RCL 09
RCL 01 DSE X - RCL 15
+ STO 15 RCL 13 +
255 MOD X=0? LASTX
STO 13 ASTO 29 RCL 03
INT 12999 + E3 /
STO 03 .09 STO 12
SIGN STO 09 CHS
STO 07 CLX STO 10
X<>F STO 02 XEQ 12
```

```
191*LBL 13
7 CHS STO 07
RCL IND 03 X<>F XEQ 12
ISG 03 GTO 13 1 CHS
STO 07 2 X<>F XEQ 12
XEQ 10 RCL 02 X<>F
```

```
209*LBL 11
XEQ 07 "----" ACA 8
CHS STO 12
```

```
216*LBL 07
ADV ISG 12 GTO 07 CLA
ARCL 29 RTN
```

```
223*LBL 12
7 RCL 09 FS? IND 07 *
ST+ 10 4 FS? IND 07
X↑2 ST* 09 16 RCL 09
X<=Y? GTO 14 X<>Y
ST+ X / STO 09
```

```
241*LBL 10
CLX X<> 10 BLDSPEC
LASTX BLDSPEC LASTX
BLDSPEC LASTX BLDSPEC
LASTX BLDSPEC LASTX
BLDSPEC LASTX BLDSPEC
ACSPEC ACSPEC ACSPEC
ACSPEC ADV ISG 12
```

```
263*LBL 14
ISG 07 GTO 12 END
LBL"ABARP"
END 426 BYTES
```

Barcodes der übrigen Funktionen

Dieses Programmpaket entstand nach den von Winfried Maschke erforschten Grundlagen über den Aufbau der Barcodes; es braucht außerdem das Programm "BP" als Unterprogramm, welches ebenfalls von Winfried stammt.

Dieses Programmpaket ist außerdem die Verbesserung des in PRISMA 82.5/6.13 abgedruckten Programms, welches zuweilen fehlerhaft arbeitete.

Mit diesem Programmpaket kann z.B. die Anwenderdokumentation verbessert werden. Vor der Anwendung dieses Paketes muß geprüft werden, ob sich der Barcodeleser im CAT 2 auch mit -WAND 1F- vorstellt; sollte jedoch im Display -WAND 1E- erscheinen, muß in Zeile 53 des Programmpaketes die 256 in eine 255 geändert werden.

Hier nun die Erklärung der Funktionen:

BARA - Vor dem Aufruf des Programms können bis zu 13 Zeichen ins ALPHA-Register geladen werden, die dann vom Drucker als Barcode ausgeworfen werden. Wird außerdem vorher das Flag 13 gesetzt, wird ein ALPHA-Append-Code gedruckt, so daß beim Lesen mit dem WAND das ALPHA-Register vorher nicht gelöscht wird, sondern die max. 13 Zeichen angehängt werden.

BARXEQ - BARGTO - Vor dem Aufruf des Programms wird ein Label-Name ins ALPHA-Register gegeben, der dann ausgedruckt wird.

BARASN - Ins ALPHA-Register wird der Name der Funktion eingegeben, und ins X-Register die Nummer der Taste im üblichen Reihen-Spalte-Code, und mit einem Minus für geshiftete Funktionen. Der Rechner akzeptiert so auch die Tastencodes 31 und -31 (im Gegensatz zu PASN!), die Funktionen müssen jedoch legal sein (also kein RCL M). Hier zeigt sich auch der Unterschied zwischen dem -WAND 1E- und -WAND 1F-: Will man mit den beiden WAND's und einem Barcode den geshifteten Funktionen der untersten Tastenreihe eine Funktion zuweisen, werden verschiedene Tasten belegt! Um ein Assignment mit dem Barcodeleser zu löschen, einfach CLA statt eines Textes ins ALPHA-Register geben.

BARSIZE - BARDEL - BARGTOL - Um diese Funktionen als Barcode darzustellen, ist die Anzahl der Register/die Zahl der zu löschenden Zeilen/die Nummer der Zeile in das X-Register einzugeben; mit 4095, XEQ "BARGTOL" läßt sich außerdem der Befehl GTO.. erzeugen.

BARDE - Mit dieser Funktion können alle 2-Byte-Befehle (bei 2 1-Byte-Befehlen wird nur eine ausgeführt) erzeugt werden, wenn sie legal sind (also nicht STO b); Funktionsbyte (in der Form aaa, dezimal) und Adressbyte (in der Form bbb, dezimal) werden, durch Dezimalpunkt getrennt, ins X-Register eingegeben.

Ralf Pfeifer

	<p>— 3. —</p> <p>Weitere Anwendungen des Druckerumbaus</p>							

Autor: Wilfried Kötz
Programm: Winfried Maschke

NAME: Querschrift
LBL Q5X SIZE 150 Peripherie: X-Funktion,
Bytes 187 Drucker, Kartenleser

Wer hat sich nicht schon über die für manche Zwecke etwas kleine Schrift oder über die schmale Papierbreite des Druckers geärgert? Hier nun zur Linderung dieses Mankos ein Querschriftprogramm, welches es erlaubt, mit dem HP-Peripheral Printer Querschrift mit nur durch die Papierrolle begrenzter Länge zu erzeugen.

Pro Zeile kann man in einem Durchgang maximal 24 Zeichen eingeben. Diese können entweder über die Tastatur oder mit dem Lesestift eingegeben werden. Die zweite Möglichkeit hat den Vorteil, daß alle 127 Standardzeichen zur Verfügung stehen.

Den benötigten SIZE erstellt sich das Programm selbst (Zeile 2-6). Weiterhin überprüft es stellvertretend für alle 127 Datenregister das Register 02 und stellt so fest, ob die Datenkarten schon eingelesen wurden oder nicht. Dies ist vorteilhaft, wenn das Programm z.B. für Briefaufkleber als Unterprogramm aufgerufen wird. Als Datenregister benötigt das Programm die Register 1 bis 127.

Hier einige Beispiele für den Ausdruck:

HERRN
EDUART ZIMMERHAMM
RUBENSTRASSE 21
5000 KOELN 80
=DEUTSCHLAND=

Beispiel_1:

Die Querschriftzeichen wurden hier über die Tastatur eingegeben.

Beispiel_2:

Für geschäftliche Briefaufkleber ruft das Absender-Programm das Querschrift-Programm auf. Eingabe erfolgt mittels Lesestift.



Herrn
Walter Maier
Ottostr. 5
5000 Köln 1

Die Datenregisterinhalte für das Querschriftprogramm sind nachstehend aufgelistet. Die entsprechende dezimale Byte-Aufschlüsselung ist ebenfalls aufgeführt, so daß man sich die Registerinhalte über das Alpha-Register mit der XTOA-Funktion notfalls selbst erstellen kann.

R01= "AB"	1 65 5 0 0 0	R44= "0A0"	16 193 120 0 0 0	R87= "↓Y0'A"	7 29 89 48 96 193
R02= "Δ Xα>"	136 160 130 136 128 62	R45= "10"	7 192 0 0 0 0	R88= "00A00A"	5 17 65 5 17 65
R03= "000A"	16 23 240 65 0 0	R46= "0"	48 96 0 0 0 0	R89= "0=00A"	32 64 129 5 17 65
R04= "αδδ"	113 18 36 87 0 0	R47= "α0A000"	4 16 65 4 16 64	R90= "Δ0A00"	252 16 65 4 16 127
R05= "σTH0'<"	9 212 104 79 161 60	R48= "zeY4p"	250 28 89 52 112 190	R91= "↑0=α00"	96 64 129 2 4 24
R06= "αΔ0 '↓"	4 8 16 32 96 255	R49= "00=αΓΔ"	112 64 129 2 6 8	R92= "*****"	1 1 1 1 1 1
R07= " b'ΓαΔ"	32 226 161 2 4 8	R50= "Δ0F0 >"	252 24 198 16 32 190	R93= "00=α00"	48 64 129 2 4 12
R08= "Σα1α00"	254 20 73 20 48 64	R51= "zυΓΔ ↓"	250 12 6 8 32 127	R94= "0=αΓΔ"	32 64 129 10 142 8
R09= "αδδ."	113 18 36 95 0 0	R52= "↑Δ1α0"	129 7 244 73 20 48	R95= "Δ"	252 0 0 0 0 0
R10= " cooΓΔ"	32 227 239 239 142 8	R53= "zυΔΓ0↑"	250 12 8 15 192 255	R96= "αΔ)"	2 4 62 0 0 0
R11= "00A=xxx"	5 17 65 1 1 1	R54= "zυ0'ΓΔυ"	250 12 23 224 65 12	R97= "zυ0?+"	250 12 24 63 128 0
R12= "σdH0"	9 228 72 145 0 0	R55= "0 Aα0↑"	16 32 65 4 16 255	R98= "Eυ0/0="	254 12 24 47 192 12
R13= "σΓΓΔ "	253 9 33 130 8 32	R56= "zυ0P'?"	250 12 23 208 96 190	R99= "xΔ0?+"	248 8 16 63 128 0
R14= "00P"	32 64 151 208 0 0	R57= "ααP'?"	97 4 15 208 96 190	R100= "zυ0? 0"	250 12 24 63 160 64
R15= "xCi/α)"	248 67 233 47 132 62	R58= "0"	48 96 3 6 0 0	R101= "xΔx/α"	248 11 248 47 128 0
R16= "zυ00P'?"	250 12 31 240 96 190	R59= "0A"	16 193 128 12 24 0	R102= "0=αΓα0"	32 64 131 130 20 16
R17= "α(0'?"	220 162 40 48 96 190	R60= "000Aα0"	64 64 64 65 4 16	R103= "rΔh?+"	242 7 232 63 128 0
R18= "zυ00E0"	250 12 23 194 2 24	R61= "↓p00"	7 240 31 192 0 0	R104= "Γυ000="	6 12 24 111 64 129
R19= "Γξx/αΔ"	6 15 248 47 128 8	R62= "0Aαααα"	16 65 4 4 4 4	R105= "p0e+"	112 64 129 3 8 0
R20= "yα/αΔ"	121 10 20 47 128 8	R63= " "	32 134 16 32 190 0	R106= "zυα0"	250 12 8 16 64 0
R21= "Γξx/αΔ"	6 15 248 47 128 20	R64= "x0'?"	248 15 153 62 96 190	R107= "αHs00="	132 200 115 56 64 12
R22= "zυ0?+?"	250 12 24 63 128 36	R65= "Γξx0'?"	6 15 248 48 96 190	R108= "p0=α0υ"	112 64 129 2 4 12
R23= "zυ00.α"	250 12 24 48 95 20	R66= "Eυ00'?"	254 12 23 240 96 191	R109= "Γυ0(α"	6 12 154 168 128 0
R24= "zυ0/α"	250 12 24 47 128 34	R67= "zΔ0 '?"	250 8 16 32 96 190	R110= "Γυ000"	6 12 24 111 64 0
R25= "zυ00'α"	250 12 24 48 96 148	R68= "Σα(P!?"	254 20 40 80 161 63	R111= "zυ0/α"	250 12 24 47 128 0
R26= "zυ000"	250 12 24 48 64 34	R69= "αΔ0'0"	252 8 23 224 64 255	R112= "0Δ0/0"	5 252 24 47 192 0
R27= "t(αδΣ"	244 40 95 225 66 254	R70= "αΔ0'0↑"	4 8 23 224 64 255	R113= "αΓΔ/α"	129 242 20 47 128 0
R28= "h'X.α"	232 47 216 174 128 0	R71= "zα0 '?"	250 14 16 32 96 190	R114= "αα000"	4 8 24 111 64 0
R29= "G000"	71 241 31 196 0 0	R72= "Γυ00'A"	6 12 31 240 96 193	R115= "Σx'?"	254 3 224 63 128 0
R30= "Σ0 '!<"	254 16 32 224 161 60	R73= "10 64 129 2 4 28	252 8 23 224 64 255	R116= "0=αΓα"	32 160 64 131 130 4
R31= "HHHHH"	255 255 255 255 255 255	R74= "zυα0 0"	250 12 8 16 32 64	R117= "zυ000"	250 12 24 48 64 0
R32= " "	0 0 0 0 0 0	R75= "αΔ00H1"	132 136 144 226 72 161	R118= "0=00"	32 162 40 48 64 0
R33= " "	32 130 133 10 8 0	R76= "Δα 0="	252 8 16 32 64 129	R119= "α00"	138 172 152 48 64 0
R34= "α"	10 20 0 0 0 0	R77= "Γυ05αA"	6 12 25 53 113 193	R120= "υ α00"	148 160 130 152 192
R35= "αxZH?""	139 250 36 72 191 162	R78= "Δα01αA"	7 13 25 49 97 193	R121= "0α00"	32 64 130 152 192 0
R36= "!Δα0B'α"	33 252 135 194 127 8	R79= "zυ00'?"	250 12 24 48 96 190	R122= "αD000"	252 17 196 31 192 0
R37= "Δ0Aα0C"	135 16 65 4 17 195	R80= "αΔ00'?"	4 8 23 240 96 191	R123= "P!HP"	80 161 87 208 0 0
R38= "yΔ0DH'α"	121 13 17 196 72 142	R81= "αΔ00'?"	251 13 24 48 96 190	R124= "0=αΓα"	32 64 129 2 4 8
R39= "υ0"	12 24 0 0 0 0	R82= "0000'?"	5 9 23 240 96 191	R125= "Altα"	65 7 244 4 0 0
R40= "0000=00"	64 64 64 129 4 16	R83= "zυα0'?"	250 12 7 192 96 190	R126= "E0A'!"	254 16 65 1 33 127
R41= "00Aααα"	16 65 2 4 4 4	R84= "0=αΓα"	32 64 129 2 4 127	R127= "αΔE'0="	4 8 31 224 64 129
R42= "00α"	66 163 138 132 0 0	R85= "zυ00'A"	250 12 24 48 96 193		
R43= "0Δ0α"	64 135 194 4 0 0	R86= " !DH α"	32 161 68 72 160 193		

Programmlisting:

```

01*LBL 05X
SIZE? 150 X<Y?
GTO 04 PSIZE

07*LBL 06
"DATENKARTEN" PROMPT
RDTA

11*LBL 04
RCL 01 "AB00" ASTO Y
X=Y? GTO 06 ADV FIX 0
CF 29

20*LBL 00
1,005 130,133 AON

24*LBL 03
CF 23 "ZEILE " ARCL Y
PROMPT FC? 23 CLA

31*LBL 02
ASTO IND X ASHF ISG X
GTO 02 ,004 + ISG Y
GTO 03 SF 12 ,005
STO 00 129 ,003
STO IND Y X<Y RDN
146,12904

49*LBL 01
CLA ARCL IND X ATOX
ASTO IND Y 32 X<Y
X=0? X<Y RDN
RCL IND X ACSPEC 6
SKPCOL RT DSE X
GTO 01 20 + PRBUF
ISG 00 GTO 01 1,001 +
RCL 00 FRC STO 00 RDN
129 X<Y ISG IND Y
GTO 01 GTO 00 END

```

Autor: Wilfried Kötz
 Programm: Winfried Maschke

NAME: SNOOPY Peripherie enthält synth.
 LBL SNOOPY SIZE 000 umgeb. Drucker Textzeilen

Dieses Programm ist ein Beispiel dafür, daß man mit einem auf den halben Zeilenvorschub umgebauten Drucker außer BAR CODES auch richtige Graphik machen kann.

Eine Programmbeschreibung erübrigt sich. Bitte Drucker auf "Graphik" stellen, Programm einlesen und laufen lassen!



Programmlisting:

Die Textzeilen sind am Rand dezimal entschlüsselt. Es handelt sich durchweg um Text-6 Zeilen (Dez. 246).

01*LBL "SNOOPY"		43 "00 Aa"	16 16 32 32 65 4	87 " ' **"	128 192 96 32 0 0
02 SF 12		44 XEQ 01		88 XEQ 01	
03 "xXx"	128 129 1 2 2 4	45 " +Δ0 "	32 130 0 16 32 0	89 34	
04 XEQ 01		46 XEQ 01		90 SKPCOL	
05 "0 AαΔ"	16 32 65 2 4 8	47 11		91 "α1++"	4 49 140 0 0 0
06 XEQ 01		48 SKPCOL		92 XEQ 01	
07 "AAL"	160 65 65 76 128 133	49 "0 α"	48 146 166 192 145 4	93 1	
08 XEQ 01		50 XEQ 01		94 XEQ 02	
09 "αΔ Δ "	4 8 32 131 8 32	51 "Qα++ "	145 17 20 191 0 0	95 127	
10 XEQ 01		52 XEQ 01		96 ACCOL	
11 1		53 15		97 1	
12 XEQ 02		54 XEQ 02		98 SKPCOL	
13 "+++±z"	15 1 128 0 0 0	55 "++αLü"	4 230 76 26 0 0	99 "SNOOPY"	
14 XEQ 01		56 XEQ 01		100 ACA	
15 "++++αv"	4 12 0 0 0 0	57 "Δ\$GZ"	96 185 18 36 71 2	101 1	
16 XEQ 01		58 XEQ 01		102 SKPCOL	
17 "p++"	112 224 128 129 0 0	59 "Δ00αv"	8 16 64 130 4 12	103 126	
18 XEQ 01		60 XEQ 01		104 ACCOL	
19 "000+++"		61 "M0="	77 129 208 64 194 1	105 3	
20 XEQ 01	16 48 0 0 0 0	62 XEQ 01		106 SKPCOL	
21 "XfΔ↓	128 2 6 8 7 176	63 "0Δ0"	128 240 16 7 222 48	107 ADV	
		64 XEQ 01		108 7	
22 XEQ 01		65 15		109 ACCOL	
23 ADV		66 XEQ 02		110 8	
24 "++++Δ"	194 8 0 0 0 0	67 32		111 ACCOL	
25 XEQ 01		68 ACCOL		112 "-----"	
26 "ABv08"	65 66 229 12 24 16	69 19		113 ACA	
27 XEQ 01		70 ACCOL		114 8	
28 "Axxα&"	65 129 2 12 4 38	71 12		115 ACCOL	
29 XEQ 01		72 ACCOL		116 7	
30 "p+α↓+9"	112 4 7 1 57 0	73 ACCOL		117 ACCOL	
31 XEQ 01		74 "0"αΔ"	48 96 193 131 4 8	118 3	
32 "±"		75 XEQ 01		119 SKPCOL	
33 XEQ 01	241 251 247 227 208 31	76 " 0xαΔ"	32 64 129 2 4 8	120 ADV	
34 ADV		77 XEQ 01		121 ADV	
35 "μ:μ	P	78 " ' "	32 96 178 4 8 16	122 STOP	
		αΔ0"		123*LBL 01	
36 XEQ 01	12 58 119 168 80 177	79 XEQ 01		124 ASTO X	
37 " AΔ"		80 "0xαΔ0"	64 129 2 4 8 16	125 ACSPEC	
38 XEQ 01	32 32 32 65 196 8	81 XEQ 01		126 RTN	
39 "0xxx"		82 "Δ 0v "	124 161 66 133 12 32	127*LBL 02	
40 XEQ 01	64 128 129 1 1 1	83 XEQ 01		128 SKPCOL	
41 1		84 1		129 ADV	
42 XEQ 02		85 SKPCHR		130 END	
		86 ADV			

Autor: Wilfried Kötz
Programm: Wilfried Kötz

NAME: CATALOG Peripherie synthetisch
LBL CAT SIZE 000 x-Funktion

Sinn und Zweck dieses Programms ist es, die bei viel angeschlossener Peripherie sehr zeitraubende CAT-2-Funktion zu verkürzen.

Zum Beispiel mußte man bisher, um den Katalog des Kartenlesers zu sehen, den gesamten Katalog aller angeschlossenen Peripherieeinheiten mitlesen. Dies ist jetzt nicht mehr nötig!

Dadurch, daß man mit Hilfe der synthetischen Programmierung das CAT-Listing teilweise überspringen kann, ergänzt das CATALOG-Programm die rechner eigenen drei CAT-Funktionen (CAT 1-3) um drei weitere CAT-Befehle (CAT 4-6) und ist darüber hinaus noch beliebig ausbaufähig.

Beispiel: Angeschlossene Peripherie: Time Modul,
X-Funktion Modul,
PPC-ROM,
Kartenleser.

USER-MODE an, LBL CAT auf geschiftete ENTER-Taste legen.
CAT 4 ... R/S = Listing beginnt mit X-Funktion Modul
CAT 5 ... R/S = Listing beginnt mit PPC-ROM
CAT 6 ... R/S = Listing beginnt mit Kartenleser
CAT 1-3 ... R/S = "normale" CAT-Funktion

Bei weiterer Peripherie kann das Programm z.B. noch auf CAT 7 mit LBL 52, X, GTO 01 erweitert werden, wobei X die Anzahl der zu überspringenden CAT-Zeilen ist. Je nach angeschlossener Peripherie muß dies auch in den Zeilen 08, 11 und 14 des nachfolgenden Programm-Listings geändert werden.

01*LBL "CAT"	13*LBL 64	25 CLA	37 STO ↑	17 F110
02 CF 21	14 201	26 GTO 02	38 "↑ R/S"	28 F120 <SPACE>
03 "CAT _"	15 GTO 01	27*LBL 01	39 AVIEW	32 F71FF00000237048
04 AVIEW	16*LBL 72	28 " "	40 STOP	
05 GETKEY	17 "0"	29*LBL 02	41 END	
06 XEQ IND X	18 CLX	30 XTOA		
07*LBL 62	19 GTO 02	31 RCL [
08 30	20*LBL 73	32 "##*#pH"	LBL "CAT	
09 GTO 01	21 CLX	33 RCL [END	93 BYTES
10*LBL 63	22 GTO 01	34 STOFLAG		
11 78	23*LBL 74	35 RDN		
12 GTO 01	24 CLX	36 CLA		

Programmanalyse:

- Zeilen 1-6: Abfrage, welches CAT-Listing gewählt wird. Hierbei sind bei Zeile 5 "GETKEY" nur die Ziffern 1-6 zulässig, alle anderen Tasten bewirken "NON EXISTENT".
- Zeilen 7-15: Die Anzahl der zu überspringenden CAT-Zeilen wird zur weiteren Verarbeitung bei LBL 02 ins X-Register gebracht.
- Zeilen 16-31: Die Katalogadresse wird hergestellt und ins X-Register gebracht.

Zeilen 32-34: Setzen des Flag 30. Hierbei wird von einem "FLAG-SET"-Status der Flags 22, 26, 27, 29, 30, 31, 37, 40 und 50 ausgegangen.

Zeilen 35-41: Die CAT-Adresse wird ins Register P gebracht, welches den CAT-Ablauf steuert. Hierbei ist das erste Halbbyte dieses Registers die CAT-Nummer (1-3), und die nachfolgenden drei Halbbytes sind die Position des CAT-Pointers.

Da alle CAT-Eingaben außer 1 und 2 den CAT-3 bewirken, wurde in Zeile 25 CLA ausgeführt, welches erreicht, daß das erste Halbbyte im Register P später Null ist, was aber auch CAT-3 bewirkt.

Zu bemerken bleibt noch, daß das Programm natürlich auch bei geschlossenem Drucker funktioniert. Allerdings sollte man bei einem CAT-Listing erst nach der R/S-Aufforderung den Drucker in den TRACE-MODE schalten.

	<p>– 4. –</p> <p>Synthetik</p>							

Rechner-Organisation

Wer früher einen Hp-67/97 hatte, der wußte immer ganz genau, wieviel Bytes oder besser: Programmzeilen noch zur Verfügung standen. Der Rechner hatte ca. 250 verschiedene Befehle (wobei GTO A und SF O ebenfalls komplette Befehle waren), die in jede Programmzeile eingespeichert werden konnten. Entsprechend beschränkt waren da auch die Programmiermöglichkeiten.

Der 41C ist dagegen wesentlich großzügiger gestaltet: In jede Zeile lassen sich Alphatexte mit bis zu 15 Zeichen schreiben, oder 10-stellige Zahlen mit 2-stelligem Exponenten, oder 210 verschiedene STO-Befehle (STO OO-STO 99, STO X-STO Z und die indirekten Befehle).

Um eine solche Vielfalt zu ermöglichen, mußte der Rechner die Fähigkeit haben, mehrere Bytes in eine Programmzeile zu bringen. Beim HP-67/97 war eine PRGM-Zeile = ein Byte. Das Byte ist die kleinste Einheit, in der das Programm abgespeichert werden kann. Egal ob in einer PRGM-Zeile des 67ers nun ein einfaches % oder ein komplizierter ST+ i-Befehl steht, beide brauchen nur ein Byte.

Beim 41er dagegen erhielten die einfachen Funktionen weiterhin ein Byte: z.B. SDEV, %, + benötigen pro Zeile ein Byte.

Aber die komplizierteren Befehle brauchen mehr Bytes: STO IND 58 benötigt 2, GTO "ABC" belegt 5, und "ABCDEFGHJKLMNO" oder -2,345678901 E-78 brauchen sogar 16 Bytes und belegen trotzdem nur eine Programmzeile. Fazit: Die Nummer der Programmzeile hat überhaupt nichts mit dem Speicherbedarf zu tun.

Ein Programmspeicher besteht aus 7 Bytes. Mit einem GTO .OOO kann man in jedem Programm sehen, wieviele Register noch frei sind. Für den normalen Gebrauch ist es einfacher, mit den (Programm-) Registern zu arbeiten, da so einfach ermittelt werden kann, wieviel Speicherraum sich noch in Datenregister verwandeln läßt oder für wieviele Tastenbelegungen noch Platz ist.

Beschäftigen wir uns nun noch einmal mit dem Byte: Wieviele Bytes gibt es eigentlich? Nun, es sind 256 verschiedene. Der Anwender hat aber nicht alle zur Verfügung. Jedes Byte stellt eine Funktion dar. Bei manchen weiß der Rechner, daß sie nicht alleine stehen, sondern daß ein oder mehrere andere folgen. Findet er z.B. das Byte 144, weiß er, daß es sich um ein RCL handelt und daß da noch ein zweites folgen muß. Hat das 2. Byte nun die Nummer 65, weiß der HP-41 auch, daß er nun nicht die Operation - durchführen muß, sondern daß es sich hier um die Adresse des RCL-Befehls handelt, nämlich Register 65.

Fazit: Manche Bytes bewirken, daß nachfolgende Bytes eine andere Bedeutung erhalten. Steht die 65 alleine, bedeutet sie -, steht sie nach STO, RCL, VIEW usw., bedeutet sie: Register Nummer 65. In einer Textkette hat unser Byte schon wieder eine andere Bedeutung: Hier stellt es den Buchstaben A dar. Nun noch etwas: Auch die Stack-Register, Datenregister und das ALPHA-Register sind aus Bytes aufgebaut. Am Ende eines normalen Rechenregisters (also Stack- und Datenspeicher) steht der Exponent der Zahl, die in einem solchen Register versteckt ist. Unser Byte 65 würde hier als 41 erscheinen!

Was nun welches Byte bewirkt, haben einige Anwender einmal zusammengestellt. Das Ding, was dabei herauskam, nannten sie Byte-

Tabelle - in Zukunft kürzen wir dieses Wort einfach mit BT ab.
Auch eine Tastenbelegung nennen wir zukünftig nur noch KA (von
engl. Key-Assignment).

Wie ziemlich einfach zu erkennen ist, besteht unsere BT aus 16x16
Feldern, sieht also aus wie 4 Schachbretter. Der räumlichen Orien-
tierung folgend nennen wir die oberen beiden Schachbretter "obere
Hälfte" und die verbleibenden "untere Hälfte".

Die Byte-Tabelle

Die nun folgende BT ist besonders für das nächste Kapitel "Synthetische Programmierung" wichtig.

Zunächst ist einmal die Numerierung der Zeilen und Spalten (am Rande der BT) wichtig: Sie beginnt jeweils mit 0 und endet mit F - 0 bis 9 und A-F sind nämlich die 16 Ziffern, aus denen man im Hexadezimalsystem (16er-System) Zahlen bildet. Wird nun die Angabe gemacht, das Kästchen Nr. 4D zu suchen, gehen wir mit dem Finger zunächst links an der BT herunter, bis wir bei der Ziffer 4 angekommen sind, dann bewegen wir den Finger in waagerechter Richtung, bis wir das Kästchen gefunden haben, über dem wir oben an der BT das D entdeckt haben. In einem solchen quadratischen Kästchen finden wir oben links erst einmal den Namen der Funktion, die das Byte erhält, wenn es ganz allein im Speicher steht, im Beispiel also %CH.

Mitte links: Folgt das Byte einem STO, VIEW, SF, ISG usw. Befehl, erscheint nach diesen Befehlen unser Byte als Adresse, hier also 77.

Mitte rechts: Steht unser Byte in einer Textkette oder im ALPHA-Register, so erscheint es in der Anzeige wie hier dargestellt, im Beispiel als M.

Unten links: Das ist einfach die dezimale (im 10er-System) Nummer des Bytes.

Unten rechts: Oft unterscheiden sich die Zeichen, die der Drucker zu Papier bringt, erheblich von den angezeigten, deshalb sind die Druckerzeichen extra aufgeführt.

In der unteren Hälfte der BT stehen nur noch die IND-Adressen in der mittleren Zeile. Grund: Ein Byte aus dieser Hälfte, welches sich ins ALPHA-Register oder in den Programmspeicher verirrt hat, wird in der Rechneranzeige wie die meisten Zeichen aus der 1. oder 7. Zeile dargestellt, so daß also alle Segmente sichtbar sind.

Die BT ist grob gegliedert in die 1,2,3 und Mehrbyte-Funktionen: So sind die Zeilen 0-8 (ausgenommen GTO alpha, XEQ alpha und der sinnlose Befehl W alpha) nur mit 1-Byte-Befehlen belegt.

2-Byte-Befehle finden sich in den Zeilen 9-11 und CE sowie CF (also X und LBL).

Die Zeilen C (ohne CE und CF), D und E enthalten 3-Byte-Funktionen, und die Zeile F enthält die Textbefehle, mit bis zu 16 Bytes.

Im einzelnen sind die Zeilen so gegliedert:

Zeile 0: Die NULL-Funktion wird vom Rechner nach DEL oder statt der gelöschten Programmzeilen (und damit der Bytes) eingefügt. Sonst befinden sich hier die Kurzformlabels.

Zeile 1: Auch die Ziffern sind 1-Byte-Befehle, auch wenn mehrere in einer Programmzeile stehen. Wenn nämlich eine im Speicher steht, heißt das noch nicht, daß auch eine zweite Ziffer ihr direkt folgen muß. Anders ist das bei den GTO- und XEQ-Befehlen: Sie werden nur eingefügt, wenn sie zu einem ALPHA-LBL verzweigen sollen, das in CAT 1 steht. Sofort auf das GTO oder XEQ folgt im Speicher ein Byte aus Zeile F, welches die Anzahl der Zeichen angibt. Das W ist zwar im Aufbau eines Befehls gleich, hat aber im Rechner keine Wirkung - jedenfalls konnte keine vernünftige festgestellt werden.

Zeile 2, 3: Viele Befehle brauchen eine Adresse, auch STO und RCL. Andererseits werden gerade diese oft in Programmen benutzt. Um nun Platz im Speicher zu sparen, wurden manche Befehle direkt in die

Byte-Tabelle aufgenommen, so daß sie nur noch ein Byte brauchen. So hat man also die LBL 00 - LBL 14 und STO/RCL 00-15 in die Byte-Tabelle als komplette Befehle aufgenommen.

Zeilen 4 - 8: Bunt gemischt finden sich hier die mathematischen Operationen, Tests, Lösch- und Ausgabefunktionen.

Zeilen 9, A8 - AF, CE - CF: Dieses sind die Zweibyte-Funktionen des Rechners. Den Befehl (also die Instruktionen, die hier aufgelistet sind) nennt man auch Präfix (engl.: Prefix) und die Adresse Postfix.

Zeilen A0 - A7: Dies sind die XROM-Nummern, mit denen der Rechner Module und Peripherie-Geräte ansteuert. Die Befehle haben bei angeschlossener Einheit einen Namen mit max. 7 Zeichen; falls das Zubehörteil jedoch fehlt, zeigt der Rechner einen Befehl der Form: XROM aa,bb an. aa ist eine Gerätenummer und bb die Funktionsnummer. Meist ist bb = 0 nicht verfügbar. Es handelt sich dann um den Namen, unter dem sich das Gerät in CAT 2 vorstellt.

Zeile B: Zu den Kurzformlabels (LBL 00 - LBL 14, O1 - OF) gehören auch kurze GTO's. Während die normalen GTO's 3 Bytes brauchen, begnügen sich diese mit 2. Eines speichert dabei die Funktion, und ein anderes die Entfernung zum nächsten LBL. Der Befehl BO wird im Programmspeicher als GTO 15 angezeigt, während des Programmlaufs bewirkt er aber nichts.

Zeilen CO - CD: Findet der Rechner ein solches Byte im Speicher, dann ist es entweder ein globales ALPHA-LBL oder ein END. Um zu entscheiden, wie er es deutet, betrachtet er das 3. Byte: steht hier eine Textkette, behandelt er es als LBL.

Zeilen D, E: Hier stehen die lokalen GTO- und XEQ-Befehle. Sie alle brauchen 3 Bytes.

Zeile F: Hier sind die Text-Befehle. Um Alphazeichen im Programm als solche kenntlich zu machen, benutzt der Rechner diese Befehle. TEXT13 bedeutet, daß diesem Byte (FD) noch 13 Alphazeichen folgen.

HP-41C COMBINED HEX/DECIMAL BYTE TABLE

HP-41C COMBINED HEX/DECIMAL BYTE TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NULL 00 0	LBL 00 01 x	LBL 01 02 y	LBL 02 03 z	LBL 03 04 α	LBL 04 05 β	LBL 05 06 γ	LBL 06 07 δ	LBL 07 08 ε	LBL 08 09 ζ	LBL 09 10 η	LBL 10 11 θ	LBL 11 12 ι	LBL 12 13 κ	LBL 13 14 λ	LBL 14 15 μ	0
1	0 16 ρ	1 17 σ	2 18 τ	3 19 υ	4 20 φ	5 21 χ	6 22 ψ	7 23 ω	8 24 δ	9 25 ϰ	.	EE 26 ϱ	NEG 27 ϲ	GTO 28 ϳ	XEQ 29 ϴ	W 30 ϵ	1
2	RCL 00 32 0	RCL 01 33 1	RCL 02 34 2	RCL 03 35 3	RCL 04 36 4	RCL 05 37 5	RCL 06 38 6	RCL 07 39 7	RCL 08 40 8	RCL 09 41 9	RCL 10 42 *	RCL 11 43 +	RCL 12 44 /	RCL 13 45 -	RCL 14 46 =	RCL 15 47 /	2
3	STO 00 48 0	STO 01 49 1	STO 02 50 2	STO 03 51 3	STO 04 52 4	STO 05 53 5	STO 06 54 6	STO 07 55 7	STO 08 56 8	STO 09 57 9	STO 10 58 0	STO 11 59 1	STO 12 60 2	STO 13 61 3	STO 14 62 4	STO 15 63 5	3
4	+ 64 ρ	- 65 ϱ	* 66 ϲ	/ 67 C	X<Y? 68 D	X>Y? 69 E	X≤Y? 70 F	Σ+ 71 G	Σ- 72 H	HMS+ 73 I	HMS- 74 J	MOD 75 K	% 76 L	%CH 77 M	P→R 78 N	R→P 79 O	4
5	LN 80 P	X↑2 81 Q	SQRT 82 R	Y↑X 83 S	CHS 84 T	E↑X 85 U	LOG 86 V	10↑X 87 W	E↑X-1 88 X	SIN 89 Y	COS 90 Z	TAN 91 0	ASIN 92 1	ACOS 93 2	ATAN 94 3	→DEC 95 -	5
6	1/X 96 0	ABS 97 a	FACT 98 b	X≠0? 99 c	X=0? 100 d	LN1+X 101 e	X<0? 102 f	X=0? 103 g	INT 104 h	FRC 105 i	D→R 106 j	R→D 107 k	→HMS 108 l	→HR 109 m	RND 110 n	→OCT 111 o	6
7	CLT 112 p	X<>Y 113 q	PI 114 r	CLST 115 s	R↑ 116 t	RDN 117 u	LASTX 118 v	CLX 119 w	X=Y? 120 x	X≠Y? 121 y	SIGN 122 z	X≤0? 123 0	MEAN 124 1	SDEV 125 2	AVIEW 126 3	CLD 127 4	7
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	

HP-41C COMBINED HEX/DECIMAL BYTE TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
8	DEG IND 00 128 0	RAD IND 01 129 x	GRAD IND 02 130 y	ENTER↑ IND 03 131 z	STOP IND 04 132 α	RTN IND 05 133 β	BEEP IND 06 134 γ	CLA IND 07 135 δ	ASHF IND 08 136 ε	PSE IND 09 137 ζ	CLRГ IND 10 138 η	AOFF IND 11 139 θ	AON IND 12 140 ι	OFF IND 13 141 κ	PROMPT IND 14 142 λ	ADV IND 15 143 μ	8	
9	RCL IND 16 144 ρ	STO IND 17 145 σ	ST+ IND 18 146 τ	ST- IND 19 147 υ	ST* IND 20 148 φ	ST/ IND 21 149 χ	ISG IND 22 150 ψ	DSE IND 23 151 ω	VIEW IND 24 152 δ	ΣREG IND 25 153 ϰ	ASTO IND 26 154 ϱ	ARCL IND 27 155 ϲ	FIX IND 28 156 ϳ	SCI IND 29 157 ϴ	ENG IND 30 158 ϵ	tone IND 31 159 ϶	9	
A	XR 0-3 IND 32 160 0	XR 4-7 IND 33 161 1	XR8-11 IND 34 162 2	X12-15 IND 35 163 3	X16-19 IND 36 164 4	X20-23 IND 37 165 5	X24-27 IND 38 166 6	X28-31 IND 39 167 7	CF IND 40 168 8	CF IND 41 169 9	FS?C IND 42 170 *	FS? IND 43 171 +	FS? IND 44 172 /	FC? IND 45 173 -	FC? IND 46 174 =	SPARE IND 47 175 0	A	
B	SPARE IND 48 176 ρ	GTO 00 IND 49 177 1	GTO 01 IND 50 178 2	GTO 02 IND 51 179 3	GTO 03 IND 52 180 4	GTO 04 IND 53 181 5	GTO 05 IND 54 182 6	GTO 06 IND 55 183 7	GTO 07 IND 56 184 8	GTO 08 IND 57 185 9	GTO 09 IND 58 186 *	GTO 10 IND 59 187 +	GTO 11 IND 60 188 /	GTO 12 IND 61 189 -	GTO 13 IND 62 190 =	GTO 14 IND 63 191 0	B	
C	GLOBAL IND 64 192 ρ	GLOBAL IND 65 193 σ	GLOBAL IND 66 194 τ	GLOBAL IND 67 195 υ	GLOBAL IND 68 196 φ	GLOBAL IND 69 197 χ	GLOBAL IND 70 198 ψ	GLOBAL IND 71 199 ω	GLOBAL IND 72 200 δ	GLOBAL IND 73 201 ϰ	GLOBAL IND 74 202 ϱ	GLOBAL IND 75 203 ϲ	GLOBAL IND 76 204 ϳ	GLOBAL IND 77 205 ϴ	GLOBAL IND 78 206 ϵ	X<>- IND 79 207 ϶	LBL - IND 80 208 Ϸ	C
D	GTO -- IND 80 208 P	GTO -- IND 81 209 Q	GTO -- IND 82 210 R	GTO -- IND 83 211 S	GTO -- IND 84 212 T	GTO -- IND 85 213 U	GTO -- IND 86 214 V	GTO -- IND 87 215 W	GTO -- IND 88 216 X	GTO -- IND 89 217 Y	GTO -- IND 90 218 Z	GTO -- IND 91 219 0	GTO -- IND 92 220 1	GTO -- IND 93 221 2	GTO -- IND 94 222 3	GTO -- IND 95 223 4	D	
E	XEQ -- IND 96 224 0	XEQ -- IND 97 225 a	XEQ -- IND 98 226 b	XEQ -- IND 99 227 c	XEQ -- IND 100 228 d	XEQ -- IND 101 229 e	XEQ -- IND 102 230 f	XEQ -- IND 103 231 g	XEQ -- IND 104 232 h	XEQ -- IND 105 233 i	XEQ -- IND 106 234 j	XEQ -- IND 107 235 k	XEQ -- IND 108 236 l	XEQ -- IND 109 237 m	XEQ -- IND 110 238 n	XEQ -- IND 111 239 o	E	
F	TEXT 0 IND 240 P	TEXT 1 IND 241 Q	TEXT 2 IND 242 R	TEXT 3 IND 243 S	TEXT 4 IND 244 T	TEXT 5 IND 245 U	TEXT 6 IND 246 V	TEXT 7 IND 247 W	TEXT 8 IND 248 X	TEXT 9 IND 249 Y	TEXT 10 IND 250 Z	TEXT 11 IND 251 0	TEXT 12 IND 252 1	TEXT 13 IND 253 2	TEXT 14 IND 254 3	TEXT 15 IND 255 4	F	
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111		

SYNTHETISCHE PROGRAMMIERUNG

Sicher werdet Ihr Euch fragen, was das ist. Es handelt sich dabei nicht um irgendwelche geniale Programmierertechniken, sondern um die Anwendung völlig neuer Funktionen und Befehle, die im Handbuch nicht beschrieben wurden und sich auch normalerweise nicht in den Programmspeicher befördern lassen. Durch bestimmte Tastenfolgen läßt sich der Rechner so manipulieren, daß solche Befehle doch möglich werden und man dem "normalen" Anwender weit überlegen ist. Denn mit synthetischer Programmierung lassen sich manche Programme wesentlich verkürzen und andere Vorhaben werden dadurch erst möglich. Dazu ein Beispiel: Ein Programm soll ein Sonderzeichen erzeugen, damit es z.B. in einen Text eingebaut werden kann. Wenn das Sonderzeichen 7 Spalten haben soll, muß also auch 7 mal BLDSPEC ausgeführt werden. Dazu muß dann jedes Mal eine ein- bis dreistellige Zahl eingegeben werden, was einen Bedarf von 21 - 35 Bytes bedeutet.

Normalerweise kann man nun einen beliebigen Text, z.B. "ABC" oder "HALLO", aber nicht diese scheinbar chaotischen Zeichen, wie sie BLDSPEC hinterläßt, im Programm einbauen. Mit der synthetischen Programmierung lassen sich nun aber genau diese Zeichen speichern. Ein Sonderzeichen braucht dann nur höchstens 10 Bytes und 2 Programmzeilen! Um alle 56 Flags - also auch die, die normalerweise nicht benutzt werden können - auf einmal in einen von Euch gewünschten Status zu bringen, braucht man 12 Bytes und 3 Programmzeilen!

Doch nun zum ersten synthetischen Schritt: Wie programmiere ich nichtprogrammierbare Funktionen der Peripheriegeräte (also z.B. WALL, WPRV, VER, LIST, PRP oder NEWM)? Ganz einfach:

1. Rechner aus. Drucker, Kartenleser oder Kassettenlaufwerk anschließen.
2. Rechner an. Mit ASN die Funktionen verschiedenen Tasten zuordnen
3. Rechner aus. Peripherie entfernen.
4. Rechner an, PRGM. Wenn nun die belegten Tasten gedrückt werden, landen die Befehle XROM 30,05 (VER), XROM 30,06 (WALL), XROM 30,09 (WPRV), XROM 29,07 (LIST), XROM 29,13 (PRP) und XROM 28,03 (NEWM) im Programmspeicher - USER-Modus nicht vergessen!
5. Werden jetzt die Peripheriegeräte angeschlossen, kann über die Funktionen per Programm verfügt werden. Dabei ist jedoch zu beachten, daß PRP immer auf NONEXISTENT führt, LIST das ganze Programm ab der auf diesen Befehl folgenden Zeile druckt und VER nie genug Karten überprüfen kann. NEWM führt immer NWEM 003 a

Bevor es nun losgeht, noch etwas für ganz ängstliche Besitzer: Durch synthetische Programmierung kann der Rechner keinen Schaden nehmen - schließlich verbiegen Programme, Daten und Key-Assignments (Tastenzuordnungen, in Zukunft mit KA abgekürzt) nichts im Rechner. Nur wenn Ihr etwas nicht hinbekommt, immer hektischer werdet, und schließlich das kleine Goldstück an die Wand pfeffert, können eventuell kleine Defekte auftreten.

Was aber durchaus möglich ist, ist, daß Ihr den Rechner in einen Zustand geistiger Umnachtung versetzt. Deshalb solltet Ihr gleich eine WALL-Karte fertigmachen. In solchen Fällen hilft nämlich oft nur noch ein MEMORY LOST, und wenn das über die Tastatur nicht funktioniert, sieht man ziemlich dumm aus. Doch nun erst einmal eine Liste der Rettungsmaßnahmen:

1. Der Rechner befindet sich an einer rätselhaften Stelle im Programmspeicher. Hier hilft ein RTN (außerhalb von PRGM) oder CAT 1.

2. Batterien herausnehmen und sofort wieder einsetzen.
3. Bei gedrückter  Taste den Rechner über ON einschalten: Es erfolgt die Meldung MEMORY LOST.
4. Falls vorhanden, X-Function-Modul entfernen. CAT 1 im PRGM-Modus ausführen und sofort R/S drücken. Nun ALPHA, , ALPHA und GTO .002. Nach DEL 999 erfolgt nach einiger Wartezeit MEMORY LOST.
5. Läßt sich der Rechner nicht einschalten, kurz die Batterien herausnehmen und statt ON die CLX-Taste zum Einschalten betätigen. Während die Batterien außerhalb des Rechners sind, schließt man noch schnell den Kartenleser an und kann, nachdem der Rechner eingeschaltet ist, die WALL-Karte einlesen. Wird nun R/S betätigt, erfolgt MEMORY LOST.
6. Hilft das alles nichts, müssen die Batterien entfernt werden. Jetzt muß sich der Kondensator, der den Permanentpeicher während eines Batteriewechsels mit Strom versorgt, entladen werden. Um diesen Vorgang zu beschleunigen, sollten möglichst viele stromfressende Peripheriegeräte angeschlossen werden: RAMS (QUAD-RAM, X-MEMORY) oder den Kartenleser (Magnetkarte bis zum Anschlag einschieben!) oder den WAND (mit Tesafilm und Streichhölzern den Schalter am Griffel eindrücken) anschließen.

Manchmal kommen solche Störungen auch durch den Anschluß von Modulen oder Peripheriegeräten zustande, weil Kontakte nicht richtig zustande kommen.

Bevor es nun mit der Programmierung losgeht, erst einmal einige Informationen zum Aufbau des Rechners.

Zunächst setzen wir einen HP-41CV oder den C mit QUAD-RAM oder vier Memory Module voraus - alles andere dürfte wohl aus der Mode gekommen sein.

Auf Skizze 1 ist der Speicher des HP-41CV wie eine abgerollte Klopapierrolle dargestellt. Links daneben finden sich einige Zahlen: 000, OOF, OCO und 1FF. Der Rechner hat seinen Speicher in Register aufgeteilt (zu je 7 Bytes) und diese hexadezimal (also im 16er-System) durchnumeriert. Die Register 000 (= 0 dezimal) bis OOF (= 16 dez.) sind die Statusregister (vgl. auch Skizze 2). Hier bewahrt der Rechner alles das auf, was der Kartenleser auf die erste Spur einer Statuskarte schreibt. Dann kommt eine große Lücke von 176 Registern im Speicher, und solange kein X-FUNCTION-Modul angeschlossen ist, kann hier nichts gespeichert werden.

Erst ab Register OCO (= 192 dez.) kann der Rechner wieder speichern. Hier stehen die KA's der Funktionen aus CAT 2 und CAT 3. Die ersten beiden kommen in Register OCO (= 192), die nächsten werden in Register OC1 (= 193) abgespeichert. Steigen wir dann weiter im Speicher auf, gelangen wir schließlich zu unserer eingebauten .END.-Marke. Jetzt kommt das letzte Anwenderprogramm, bis hinauf zum ersten, und dann beginnen die Datenregister: Zunächst Register OO, O1 usw.. Unser letztes Datenregister hat die Nummer 1FF (= 511), womit der Speicher auch schon zu Ende wäre.

Doch bleiben wir bei den Statusregistern: 000 - 004 ist der Stack, dessen Arbeitsweise hinreichend bekannt sein dürfte. Es folgt das ALPHA-Register. Um dieses zu verstehen, betrachten wir

einmal Skizze 4: Tastet man nach CLA einmal das Alphabet ein, so landet das A in Register M (005) ganz rechts. Wenn jetzt B angehängt wird, verschiebt der Rechner A auf die 2. Position von links, und B erscheint ganz rechts. Nach dem Buchstaben G ist das Register M voll, und A steht jetzt ganz links. Hängt man nun das H an, wird A in das Register N geschoben, B ist nun ganz links in M und H erscheint rechts in M. Dieses Spiel geht nun weiter, bis auch die Register O und die ersten drei Stellen von P gefüllt sind - nach dem Stack-Prinzip. Denn jedes neue Zeichen wird nicht irgendwo dazugestellt, sondern alle vorhergehenden Zeichen werden verschoben, und dann erst wird das letzte angehängt, wie bei der Zahleneingabe in den Stack mit ENTER.

Wenn jetzt ein 25. Zeichen hinzukommt, wird immer noch verschoben, auch im P-Register. Das A landet dann in der 3. Stelle von links in P, ist aber jetzt nicht mehr normal verfügbar. Es wird dort wohl auch nicht lange bleiben, denn dieser Rest von P wird vom Rechner zur Durchführung von CAT und bei Zifferneingaben benutzt.

Wenn eine Funktion über die Anzeige ausgeführt wird, speichert der Rechner den Text im Register Q (009) ab. Allerdings wird der Text nicht wie in den Registern M, N, O und P abgespeichert, sondern entgegen der Leserichtung, also von rechts nach links. Q wird dabei nur von Funktionen benutzt, die gerade nach einem Namen aus CAT 1, 2 oder 3 suchen, z.B. XEQ, XEQ IND, GTO, GTO IND, COPY, CLP, ASN usw..

Wenn im USER-Modus eine Taste gedrückt wird, muß der Rechner feststellen, ob diese Taste irgendwann belegt wurde. Nun kann es ziemlich lange dauern, wenn viele Belegungen gemacht wurden, bis der Rechner das herausgefunden hat. Er muß dazu nämlich den ganzen CAT 1 durchgehen, um festzustellen, ob der Taste ein LBL zugewiesen wurde, und für Funktionen aus CAT 2 und 3 muß er die KA-Register durchsuchen. Um die Bedienung zu beschleunigen, hat der Rechner in den Registern t und e selbst so etwas wie Flags.

Jede Taste hat nun 2 Flags: Eines in Register t für die normale Tastenfunktion, und eines in Register e für die Zweitfunktion, die mit SHIFT angesprochen werden kann. Der Rechner sieht also im entsprechenden Register nach, ob eine Tastenbelegung vorliegt, und macht sich dann eventuell auf die Suche. Allerdings kann eine Tastenbelegung auch durch die lokalen Labels A-J und a-e vorliegen. Diese werden nicht in den beiden KA-Flag-Registern vermerkt. Umgekehrt wird nach solchen lokalen Labels auch nicht mehr gesucht, wenn in diesen Registern eine Belegung verzeichnet ist. Auch die gelbe SHIFT-Taste hat hier ein Flag, und die ENTER-Taste sogar zwei - eines wird jedoch nie benutzt, da der Rechner für diese Taste immer das andere abfragt.

Numerieren wir diese KA-Flags von links nach rechts durch, dann hat eine Taste in der Spalte A und der Zeile B das Flag: Fl.Nr. = 44 - 8·A - B. In den Registern a und b speichert der Rechner die Unterprogrammebenen, die belegt sind. Außerdem steht in Register b noch die gegenwärtige Adresse im Speicher. Auch hier wird nach dem Stackprinzip (LIFO-Speicher, Last-In-First-Out) gearbeitet: Ein XEQ-Befehl bewirkt, daß die augenblickliche Adresse nach links geschoben wird (sie wird dabei auch verändert!), um beim nächsten RTN oder END wieder zurückzukommen.

Die Register a und b werden nach SIZE, CAT, PACK, GTO., RTN über die Tastatur, sowie beim Einfügen und Löschen von Programmen oder Befehlen gelöscht. In Register c bewahrt der Rechner die Position

von .END., Register 00 und der Statistikregister auf. Allerdings muß man, um SIZE zu finden, erst die hexadezimale Position von R 00 in eine Dezimalzahl umwandeln. Das Ergebnis muß dann noch von 512 abgezogen werden, dann hat man erst die Speichergröße. Hat man außerdem kein QUAD-RAM oder den CV, muß man noch die Anzahl der Register, die zum CV fehlen, abziehen (pro Memory Modul also 64).

Um die Position der Statistikregister zu erhalten, muß man erst die in c abgelegte Adresse dieser Register in eine Dezimalzahl umwandeln und dann davon die Adresse von R 00 abziehen.

Außerdem befindet sich in Register c die Konstante 169: Diese wird vom Rechner immer wieder überprüft. Findet der Rechner irgendwann einmal diese Zahl nicht mehr vor, so löscht er den ganzen Speicher, weil er annimmt, daß nicht nur diese Zahl, sondern auch Programme und Daten verändert wurden. In der Anzeige erscheint dann MEMORY LOST - so kann man das dann später auch programmieren.

Register d enthält die 56 Flags des Rechners. Später wird noch erklärt, wie man z.B. BAT und SHIFT im Programm setzt (z.B. zur Demonstration) oder dem Rechner vorgaukelt, daß der angeschlossene und eingeschaltete Drucker doch nicht da ist (das bringt einen Geschwindigkeitsvorteil), und ähnliches.

Doch nun zu den KA's. Wir wollen nun ein bißchen experimentieren, und dazu sind folgende Vorbereitungen nötig:

1. MEMORY LOST
2. USER, PRGM, ASN "+" 11 (Σ +), ASN "MOD" 12, SIZE maximal wählen
3. CAT 1, sofort R/S, ALPHA, , ALPHA, GTO. 001

Wir stehen nun am Anfang eines KA-Registers, dessen schematischer Aufbau in Skizze 3 dargestellt ist: Zunächst der Befehl 240, er zeigt sich als kleines hochgestelltes T, wie bei den Textketten. Mit SST sehen wir dann LBL 03. Ein weiteres SST zeigt dann die Funktion: Es ist MOD, und erneutes SST zeigt in Form einer 1 die Taste an. Die nächsten 3 SST's zeigen LBL 03, +, LBL 00, das KA von + auf Taste 11. Jetzt noch ein letztes SST, und es dauert etwas, bis $\text{CL}\Sigma$ erscheint; Grund: Der Rechner überspringt die Lücke im Speicher und landet dann im Register e, welches er als Programm auffaßt, und eben $\text{CL}\Sigma$ anzeigt.

Sicher wolltet Ihr schon immer wissen, ob sich SIZE, ASN, COPY usw. nicht doch programmieren lassen. Wie Ihr gesehen habt, wurden die Funktionen + und MOD als Programmschritte angezeigt - was passiert dann bei SIZE? Probieren geht über studieren, heißt es, und darum führen wir schnell STO .004 und ASN "" 12 aus, wobei unsere 1 als Code für die Taste verschwindet und statt dessen das LBL 03 der nächsten Zuordnung (also 9) erscheint. Ein BST zeigt uns immer noch MOD an; Fazit: Bestehende KA's werden dadurch entfernt, daß der Rechner die Tastenbelegung löscht (hier also die 1). Nun führen wir ASN "SIZE" 12 aus, und es erscheint - oh herbe Enttäuschung - LBL 05 statt des ersehnten SIZE-Befehls!

Basteln wir jetzt einmal anders herum: Statt mit ASN die Befehle im KA-Register zu ändern, löschen wir zunächst einmal mit  das LBL 05 und betätigen 1/X, um zu sehen, welches KA wir nun haben: Es ist CAT.

Jetzt fügen wir LBL 00 ein und halten die Taste 1/X gedrückt, bis NULL erscheint; das Ergebnis ist ein ominöses \mathcal{C} (?).

Mit \leftarrow löschen wir LBL 00 und probieren alle Kurzformlabels auf diese Art durch. Es bietet sich folgendes Bild:

NULL	=	CAT	LBL 00	=	???	LBL 01	=	DEL
LBL 02	=	COPY	LBL 03	=	CLP	LBL 04	=	R/S
LBL 05	=	SIZE	LBL 06	=	BST	LBL 07	=	SST
LBL 08	=	ON	LBL 09	=	PACK	LBL 10	=	DEL 001
LBL 11	=	ALPHA/PRGM/USER	LBL 12	=	???	LBL 13	=	SHIFT
LBL 14	=	ASN						

Zu dieser Tabelle sind noch einige Anmerkungen zu machen: NULL ist ein Befehl, der nichts bewirkt, und der vom Rechner nicht angezeigt wird (nicht zu verwechseln mit dem NULL, das der HP-41C anzeigt, wenn eine Taste zu lange gedrückt wird). Wenn im Programm Zeilen gelöscht werden (mit DEL oder \leftarrow), ersetzt sie der Rechner durch NULLen.

LBL 00 und LBL 12 führen zu rätselhaften Anzeigen und sind bisher kaum erforscht. LBL 04 hat die gleichen Wirkungen wie R/S, jedoch kann kein laufendes Programm damit gestoppt werden. Dies geht nur mit ON und der R/S-Taste. LBL 10 entspricht zwar \leftarrow , doch hier wird immer die Programmzeile gelöscht, auf der der Rechner steht - auch außerhalb von PRGM. LBL 11 hat ähnliche Funktionen wie die Tasten USER, PRGM und ALPHA - welche es genau ist, hängt von der belegten Taste ab. Reihe 1 und 5 entsprechen ALPHA (man kann allerdings nur einschalten), Reihe 2 und 6 bewirken PRGM, und mit den übrigen Tasten aus den Reihen 3, 4, 7 und 8 kann man sich aus dem USER-Modus ausschalten.

LBL 13 entspricht zwar SHIFT, um jedoch aus einem STO ein STO IND zu machen, braucht man immer noch die gelbe Taste.

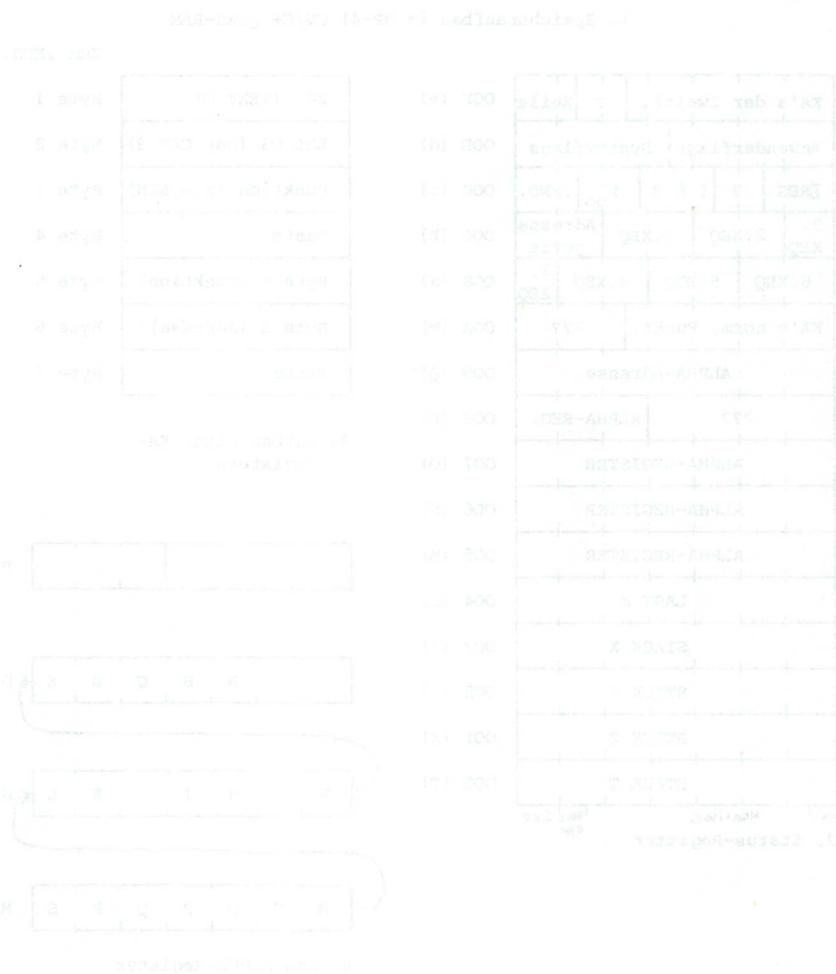
Auf diese Art und Weise kann man auch die Kurzform-STO und RCL-Befehle ausprobieren (also STO 00 - STO 15 und RCL 00 - RCL 15).

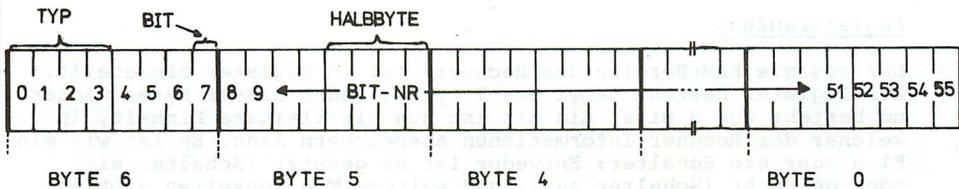
Wenn noch eines der Kurzformlabels in unserem KA-Register steht, mit dem wir eben experimentiert haben, dann löschen wir dieses jetzt, und ebenfalls das LBL 03 davor. Jetzt tasten wir SF 00 ein, dann SST, SST, DEL 002, CF 00. Auf den Tasten haben wir dann diese beiden Befehle, und durch abwechselndes Drücken können wir dann auch den Erfolg an dem Statusindikator für Flag 00, unten in der Anzeige, erkennen. Auf diesem Wege lassen sich dann auch andere 2-Byte-Befehle zuweisen, dabei darf man sich allerdings nicht daran stören, daß SF 00 als XROM 32,00 und CF 00 als XROM 36,00 angezeigt werden, wenn die jeweilige Taste niedergedrückt wird.

Nach diesen einfachen Experimenten müssen wir erst einmal erkunden, was der HP-41C überhaupt für Möglichkeiten bietet. Was ist z.B. ein Byte? Ein Byte besteht aus acht kleinen Schaltern, die entweder an- oder ausgeknipst werden können. Wir können nun diese 8 Bits als Binärzahl ansehen: Jedes dieser Bits ist eine Stelle. Wir können dann in ein Bit Zahlen zwischen 00000000 und 11111111 schreiben (Achtung: In Binärzahlen kommen nur die Ziffern 0 und 1 vor), das sind dann 256 verschiedene Möglichkeiten. Die

Vorgänger des HP-41C, nämlich HP-67/97, hatten etwas weniger als 256 verschiedene Befehle, und so belegte jeder Befehl - egal, ob ein einfaches % oder GTO A oder ST+ 0 - ein Byte = 1 Zeile. Überlegen wir jedoch, daß der HP-41C in jede Zeile eine zehnstellige Zahl und außerdem noch einen zweistelligen Exponenten mit Vorzeichen unterbringen kann, dann geht das bestimmt nicht mit einem Byte alleine. Hier kombiniert der Rechner vielmehr viele Bytes und zeigt sie als eine Zeile an.

Auch ST+, GTO, XEQ, VIEW oder FIX bestehen aus mehreren Bytes, um genau zu sein, aus exakt zweien.





REGISTERAUFBAU

	6	5	4	3	2	1	0		
e	SHIFT KA's				?	ZEILE		00F	
d	ANWENDER			FLAGS		SYSTEM		00E	
c	ZREG	?	1	6	9	REG 00	.END.	00D	
b	3. RTN	2.RTN		1.RTN		PRGM ZEIGER		00C	
a	6.RTN		5.RTN		4.RTN		3. RTN	00B	
t	NORMAL KA's			?				00A	
q	?								009
p	?			ALPHA (22-24)				008	
o	ALPHA (15-21)								007
n	ALPHA (8-14)								006
m	ALPHA (1-7)								005
l	LASTX								004
x	X								003
y	Y								002
z	Z								001
t	T								000
	± MANTISSE						± EXP		

Registeraufbau

Der gesamte RAM-Bereich des Rechners ist in Register eingeteilt. Ein Register besteht dabei aus 7 Bytes. Jedes dieser Bytes wiederum besteht aus 8 Bits. Ein Bit ist nun die kleinste Einheit, in welcher der Rechner Informationen abspeichern kann. Es ist wie ein Flag oder ein Schalter: Entweder ist es gesetzt (Schalter ein) oder gelöscht (Schalter aus), und weitere Möglichkeiten gibt es nicht. Damit können wir unser Bit als eine einstellige Binärzahl verstehen: Entweder 0 (Flag gelöscht) oder 1 (Flag gesetzt). Nun faßt man so 4 Bits zusammen und notiert sie so: 0000 oder 1011 oder 1111. Es gibt dabei 16 verschiedene Muster, die die kleinen Bits darstellen können. Da - wie oben erwähnt - ein Register 56 (nämlich 7 x 8 Bits) hat, können wir diese nicht immer ausschreiben. Daher ziehen wir nun das Hexadezimalsystem (16er-System) heran und machen aus einer 4-stelligen Binärzahl, z.B. 1010, eine 1-stellige Hex-Zahl, hier also A.

Den genauen Zusammenhang von 4-stelliger Binärzahl, 1-stelliger Hex-Zahl und deren dezimaler Wert ist in folgender Tabelle zusammengestellt:

Binär	Hex	Dez	Binär	Hex	Dez
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

Diese Binärdarstellung kann man im Prinzip vergessen. Wir werden allerdings bei einigen Registern (insbesondere d) noch einmal darauf zurückgreifen. Dennoch wollen wir der Ordnung halber die Bits in Abb. 2.1 numerieren: Das erste Bit von links bezeichnen wir als Nummer 0, und das erste Bit von rechts als 55.

Die Bytes werden dagegen genau umgekehrt nummeriert: Nummer 0 ist das erste von rechts und Nummer 6 das erste von links.

Jedes Byte können wir als 2-stellige Hex-Zahl darstellen. Denn jedes Byte hat 256 Möglichkeiten, und es gibt 16 verschiedene Hex-Ziffern. Kombiniert man nun zwei Hex-Ziffern, dann ergeben sich $16 \times 16 = 256$ verschiedene Zahlen, also ganz zufällig genauso viele, wie ein Byte Möglichkeiten hat!

Interessant wird für uns ein Register erst, wenn es ein normales Datenregister ist. Denn in so einen Datenspeicher kann man ja einen ganzen Haufen verschiedener Sachen ablegen: Bis zu 6 Alphazeichen freier Wahl, Sonderzeichen (mit BLDSPEC erzeugt), den Rechnerstatus (mit RCLFLAG) und ganz, ganz viele Zahlen. Aber wie merkt sich ein kleiner Datenspeicher das alles in 7 Bytes? Und wie findet der Rechner heraus, was sich der Datenspeicher gemerkt hat? Im Prinzip ist das alles ganz einfach: Einen Hinweis gibt schon die Tatsache, daß nur 6 Alphazeichen in ein Register mit 7 Bytes passen. Der Rechner speichert nämlich intern in das Datenregister eine Information, die nur für ihn bestimmt ist, und die nicht an den Anwender weitergegeben wird. In der linken Hälfte des 6. Bytes steht der Typ des Datenregisters. Hier findet der Rechner eine 0, wenn im Datenregister eine 0 oder positive Zahl

enthalten ist. Eine 9 bedeutet dagegen eine negative Zahl, und eine 1 bedeutet, daß es sich um irgendetwas Alphanumerisches handelt, z.B. ein Text, ein Sonderzeichen oder der Rechnerstatus. Der Rechner trifft dabei keine weitere Unterscheidung zwischen diesen 3 Formen der Alphazeichen.

Doch befassen wir uns zunächst mit den Zahlen. Wie allgemein bekannt, rechnet der HP-41 mit 10-stelliger Genauigkeit. Da er außerdem im Gleitkommaformat arbeitet, braucht er noch 3 Stellen für den Exponenten, eine davon für das Vorzeichen. Gleitkommaformat bedeutet dabei, daß intern die Zahlen immer so gespeichert und verarbeitet werden, wie sie durch SCI 9 dargestellt werden: Wenn die Zahl nicht gerade selbst die Null ist, ist die erste Ziffer ungleich Null, gefolgt vom Komma und 9 weiteren Stellen, und jetzt kommt das Vorzeichen des Exponenten und dann dessen beide Stellen.

In dieser Reihenfolge sind die Zahlen auch intern im Datenregister angeordnet. Byte 6 enthält die erste Ziffer in seiner rechten Hälfte, die zweite befindet sich in der linken Hälfte des Byte 5 bis zum Vorzeichen des Exponenten (Byte 1, rechts) und dessen beiden Stellen (ganzes Byte 0). Ein Komma braucht der Rechner dabei nicht abzuspeichern. Die interne Darstellung ist ja immer wie bei SCI 9, auch wenn der Anwender ENG 3 oder FIX 0 gewählt hat, und da ergibt es sich dann von selbst, daß das Komma immer zwischen der ersten und zweiten Stelle liegt.

Jede Ziffer bekommt also ein halbes Byte = 4 Bits. Mit diesen 4 Bits kann man nun also alle 16 Hex-Ziffern, nämlich von 0-9 und A-F, darstellen. Da wir im 10er-System nur 10 Ziffern haben, können wir auf die zusätzlichen Ziffern A-F, die uns jedes Halbbyte bietet, verzichten. Enthält ein Datenspeicher numerische Werte, tauchen die Hex-Ziffern A-F nirgends auf! Wenn sie auftauchen würden, dann überstehen sie zwar das STO, welches sie in den Speicher bringt, aber jede andere Operation, die auf diesen Speicher zugreift, zerstört dann diese Zahl.

Der Grund ist die sogenannte oder auch Normalisierungsroutine. Die Vorgänger des HP-41, nämlich der HP-67/97, hatten so etwas nicht. Die bösen, bösen Anwender aber hatten nichts besseres zu tun, als das auszunutzen und dem Rechner so z.B. das vernünftige Plotten beizubringen (es blieb trotz allem eine Bauklötzchengrafik). Nun wollte man das beim HP-41C verhindern, zumal dieser ja Daten- und Programmkarten der Vorgänger übernehmen konnte, und baute also diese Normalisierung ein. Der Effekt ist dabei folgender: Mit ganz besonderem Eifer wacht der kleine Rechenzweig darüber, ob die Zahlen, die er aus den Datenspeichern holt, auch seinen Anforderungen entsprechen: Es dürfen nur die Hex-Ziffern 0-9 auftreten, und die allererste Ziffer ist ja in der SCI-Darstellung auch immer ungleich Null, weshalb der Rechner das auch in seinem Datenspeicher überprüft. Hat er nun an einer Zahl etwas zu bemängeln, so macht er daraus entweder eine andere, oder eine ALPHA-Kette, oder er setzt sie einfach Null. Das Ergebnis dieser Untersuchung wird dann nicht nur dem Anwender angeboten, sondern auch in das Datenregister geschrieben. Das alles passiert aber nur mit den normalen Datenspeichern! Zum Exponenten ist noch eine Besonderheit zu bemerken: Die niederwertige (rechte) Hälfte des Bytes 1 enthält das Vorzeichen. Eine 0 bedeutet wieder positiver Exponent, und eine 9 negativer Exponent. Bei negativen Exponenten gibt es noch eine Besonderheit, es wird nämlich nicht der Exponent, sondern 100 minus Exponent ins Byte 0 gespeichert. Die letzten beiden Bytes sehen nach EEX -93 so aus: 09 07. Wozu das gut sein soll, wissen wohl nur noch die Götter.

Beim Speichern von Alpha-Ketten sieht das Byte 6 normalerweise so aus: 0001 0000 oder in Hex: 10. Normalerweise heißt dies in diesem Fall, daß es leider auch einige Serien gibt, bei denen ASTO dazu führt, daß von einem evtl. vorhandenen 7 Alphazeichen, das direkt auf die 6 folgt, die abgespeichert werden, die höherwertige (linke) Hälfte genommen wird und noch in die niederwertige (rechte) Hälfte des Bytes 6 dazugespeichert wird. Das kann dann zu Fehlern beim Vergleich mit X=Y? führen (Näheres s. Anhang: Rechner-Bugs). Führt man z.B. ABCDEF, ASTO X oder ASTO 00 aus, so befindet sich das A in Byte 5 und das F in Byte 0. Werden weniger als 6 Zeichen eingespeichert, z.B. nur ABCD, dann steht das A in Byte 3 und das D in Byte 0. Die Bytes 4 und 5 werden mit Hex 00 belegt. In einem anderen Abschnitt dieses Kapitels steht dann Genaueres über die NULL (Hex 00) als Alphazeichen.

Die Funktionen BLDSPEC und RCLFLAG (gehören zum Drucker als X-Funktion) belegen dagegen das Byte 6 ganz regulär:

RCLFLAG ruft den Status der Flags 0-43 ab und formt daraus eine ALPHA-Kette. Damit man später mit STOFAG nicht jeden beliebigen Alphatext in das Flagregister speichern kann, enthält die niederwertige Hälfte von Byte 6 und die höherwertige Hälfte des Bytes 5 jeweils Hex F. In der Hex-Darstellung sieht also der Inhalt eines Registers so aus: 1F Fa bc de fg hi jk. Dabei ist in a der Zustand der Flags 0-3, in b der von 4-7, in k von 40-43 abgespeichert. Man kann das übrigens dazu nutzen, eine Verschiebung von Byteinhalten um 4 Bits vorzunehmen.

Eine andere Funktion, die von der niederwertigen Hälfte des 6. Bytes Gebrauch macht, ist BLDSPEC. Das kommt folgendermaßen: Die Punkte der 7x7 Matrix des Sonderzeichens werden durch einzelne Bits repräsentiert. 7 Spalten mit 7 Punkten brauchen 49 Bits. Die Bytes 0-5 stellen aber nur 48 zur Verfügung. Daher muß ein Punkt noch im Byte 6 untergebracht werden, und das ist der Punkt, der ganz unten in der ersten Spalte von links, also der 7. Spalte von rechts, erscheint.

Bei beiden Zeichenketten ist folgendes zu beachten: Ein ARCL holt nur die Bytes 0-5, nicht aber den Inhalt des 6. Bytes. Im Falle des Sonderzeichens führt die Sequenz CLA, ARCL X, ASTO X dazu, daß der oben genannte Punkt auf jeden Fall gelöscht wird. Bei der mit RCLFLAG erzeugten Textkette führt das dazu, daß im ersten Byte statt 1F die 10 steht. Der Versuch, diese Textkette wieder mit STOFAG zurückzuspeichern, wird dann mit DATA ERROR beantwortet. Die Normalisierungsroutine interessiert sich für ALPHA-Ketten nicht. Sobald die erste Hälfte des Bytes 6 als 1, also Typ ALPHA, erkannt ist, läßt der Rechner seine Finger von diesem Register - Alpha-Ketten haben also Narrenfreiheit, ebenso die 16 Statusregister, bei denen ja auch die RCL-Befehle und X -Anweisungen nicht zur Normalisierung führen.

Der Name Normalisierung kommt wahrscheinlich daher, daß die Zahlen, die nicht den Normen des Rechners entsprechen, "nicht normalisierte Zahlen" genannt werden. Sie können nur synthetisch erzeugt werden, da die Normalisierung auch bei der Zifferneingabe gegenwärtig ist. Auch wenn es in der Anzeige mit z.B. 0,003 E67 etwas chaotisch aussieht, hat der Rechner schon dafür gesorgt, daß alles mit normalisierten Zahlen zugeht.

Der F7-Wolf

Um synthetische Befehle schnell über die Tastatur zu erzeugen, benutzen wir nun den F7-Wolf, der sich recht einfach auf jedem Rechner als Tastenbelegung erzeugen läßt. Dazu wird der CAT-BUG (s. Anhang) benutzt.

1. MEMORY LOST herbeiführen
2. ASN "+" -51 (Funktion X=Y? im NORMAL-Modus)
3. ASN "-" -74 (Funktion ENG im NORMAL-Modus)
4. USER, PRGM, CAT 1 und sofort R/S drücken
5. ALPHA, , ALPHA, GTO .002, DEL 006 (etwas warten!)
6. ALPHA, W?  K  AA, PRGM, RTN

Das Zeichen  ist der Winkel (Zweitfunktion des O); ist das X-FUNCTION-Modul angeschlossen, erscheinen die beiden A's in Schritt 6 normal, sonst erscheinen sie als waagerechte Striche oben in der Anzeige (als NULLen). Auf der Taste -51 liegt nun die MOD-Funktion, die ersetzt werden kann (am besten durch PACK, da man dieses sehr oft braucht, wenn mit dem Wolf gearbeitet werden soll. Wenn der WOLF nun im PRGM-Modus eingefügt wird, um ein Byte zu fressen, wird das in den folgenden Beschreibungen immer mit WOLF gekennzeichnet (nicht: Wolf).

Bei angeschlossenem Drucker erscheint in der Anzeige SKPCOL statt XROM 29,23, und auch PRKEYS weist diese Funktion aus. Tatsächlich hat sich aber nichts an der Arbeitsweise dieses Befehls geändert, und schon Hildegard Knief sang: "... denn WOLF bleibt WOLF ...".

Erscheint in der Anzeige \uparrow W-----

nachdem der Wolf eingefügt wurde, dann hat er nichts bewirkt: Entweder wurde er dann in einem gepackten Programm zwischen zwei Zahlen eingefügt, oder das Programm war noch gar nicht gepackt.

Nun noch einige praktische Anwendungen des Wolfes. Oft liegen bei Messen und Ausstellungen einige 41er achtlos herum. Um sie zum Objekt interessierter Erheiterung der umstehenden Kunden zu machen, geht man am besten wie folgt vor:

1. Wie oben beschrieben den Wolf zuweisen.
2. PRGM (ein), ALPHA,  DEFECT, AVIEW, ALPHA
3. VIEW IND Z, +, BST,  BST, WOLF, , SST, ASTO X, XEQ IND X
4. PRGM, RTN, R/S.

Nun zeigt der Rechner DEFECT an und läßt sich über die Tastatur nicht mehr bedienen. Nur ein kurzes Herausnehmen der Batterien hilft hier.

Der Text in Schritt 2 läßt sich natürlich beliebig variieren; sollte allerdings innerhalb der 12 Zeichen der Anzeige bleiben - Punkt, Komma und Doppelpunkt lassen sich dann auch noch zwischen die übrigen Zeichen setzen. Als weitere Alternativen bieten sich an: ZU TEUER, KOMM ZUM CCD, LIEBER TI-59, VW KAEFER, FREIBIER HER usw.

Die Rechnerstarre läßt sich auch durch 64, BLDSPEC, XEQ IND X erzeugen (s. Anhang: Rechner-Bugs).

Synthetische Alphalabels

Als erstes großes Anwendungsbeispiel für den WOLF nehmen wir die Alpha-Labels: Auf dem HP-41 C(V) lassen sich die globalen Labels A-J und a-e nicht erzeugen, weil der Rechner daraus lokale LBLs macht. Um das nun zu erreichen, gehen wir so vor:

1. GTO .. , ENTER ↗ (oder irgend einen anderen Befehl)
2. VIEW IND 64, ENTER ↗, ALPHA, APPEND
3. A
4. ALPHA, BST, BST, BST, WOLF,
5. PACK (oder GTO ..)

Fertig! Schritt 1 hat dabei im Prinzip keine Bedeutung. Die nachfolgenden Schritte können auch mitten in einem Programm erfolgen.

Statt des Buchstaben A in Zeile 3 kann jeder beliebige andere Buchstabe/Zeichen/Text (auch :,.) verwendet werden. Nimmt man jedoch mehr als 7 Zeichen, kann das LBL nicht mehr adressiert werden, sondern nur noch als Informationszeile in CAT 1 dienen.

In Schritt 5 muß ein PACKING ausgelöst werden, weil sonst das LBL nicht in CAT 1 erscheint - also nicht adressiert werden kann - und außerdem auch nicht gelöscht wird. Stattdessen gibt es nämlich einen Crash, der sich nur durch Entfernen der Batterien beheben läßt.

Außerdem sollte man nicht an Alphalabels herumdoctorn, da sonst MEMORY LOST erscheinen könnte.

Zu unseren LBLs brauchen wir aber auch XEQ und GTO-Befehle. Bei den LBLs A-J und a-e lassen sich die XEQ's recht einfach erzeugen: Mit ASN das LBL irgendeiner Taste zuweisen, und dann im USER und PRGM-Modus diese Taste betätigen. Andere LBLs lassen sich vielleicht nicht einer Taste zuweisen, deshalb verfährt man am besten nach diesem Schema:

1. GTO .. , ENTER ↗
2. VIEW 29 (für GTO) oder VIEW 30 (für XEQ)
3. ALPHA, A, ALPHA
4. BST, BST, WOLF,

Auch hier ist Schritt 1 prinzipiell bedeutungslos. Der Text in Zeile 3 ist beliebig austauschbar, aber mehr als 7 Zeichen sind vollkommen sinnlos. Ein ganz besonderes LBL ist das LBL Affenschwanz (LBL ↗). Wenn dieses LBL nämlich nicht im Programmspeicher auffindbar ist, führen die Befehle GTO/XEQ und GTO IND/XEQ IND alle zum Crash. Das LBL wird wie folgt erzeugt:

1. GTO .. , ENTER ↗
2. VIEW IND 64, ENTER ↗, VIEW IND Y, STO 64, BST, BST, WOLF, ,
PACK
3. BST, BST, WOLF, , PACK

Und die GTO/XEQ-Befehle werden so gebastelt:

1. GTO .. , ENTER ↗
2. VIEW 29 (für GTO) oder VIEW 30 (für XEQ)
3. VIEW IND Z, +, BST, BST, WOLF, , PACK
4. BST, WOLF,

Anwendung von PRIVATE

Gute Programme sind immer ein Grund zur Freude - bei den Anwendern und bei Software-Haien, die nach dem Motto "Lieber gut kopiert als schlecht kreiert" Raubkopien anfertigen und diese ohne weitere Arbeitsleistung teuer verkaufen.

Um solch unflätigem Verhalten erfolgreich entgegenzuwirken, kann man von allen, denen man ein solches Programm gibt, ein Weitergabeverbot unterschreiben lassen. Doch natürlich ist das keine "wasserdichte" Methode, und wenn das Programm doch weitergegeben wird, ist es meist schwer, den zu finden, der den Vertrag gebrochen hat.

Wesentlich einfacher ist es da, wenn man die Hardware (hier also unser HP-41C) veranlassen kann, den Anwender auf dem Pfad der Tugend lustwandeln zu lassen. So verfügt dann auch der Kartenleser über die Funktion WPRV, die es ermöglicht, Programme, die von solchen Magnetkarten stammen, gegen Einsichtnahme, Veränderung, Auflisten und erneutes Kopieren zu schützen. Auf alle diese Versuche antwortet der Rechner mit der Meldung PRIVATE. Es gibt jedoch noch andere Anwendungen für diesen Selbst-Schutz: Wird der Rechner z.B. zeitweise von minderqualifizierten Benutzern traktiert, kann man sich so gegen zweifelhafte Modifikationen der im RAM gespeicherten Programme schützen - Lehrer, Saboteure und infantile Gemüter haben dann keine Chance mehr.

Die zu Anfang einfachste Methode ist, zunächst ein Programm einzulesen, dann mit WPRV aufzuzeichnen, wieder einzulesen, und GTO ..; jedoch zeugt dies nicht gerade von umwerfendem Rechnerverständnis. Einfacher ist es nämlich so:

1. GTO .., WPRV.
2. Jetzt das erste eigene Programm einlesen, RTN, BST, MRG und die unter 1. aufgezeichnete Karte einlesen, GTO ..

Synthetisch läßt sich das jedoch ganz ohne den Kartenleser bewältigen; da GTO .., WPRV offenbar ausreicht, um die PRIVATE-Information zu übertragen, vermuten wir, daß diese Information im END oder .END. steckt. Und tatsächlich ist das so, allerdings wird auf der Magnetkarte noch zusätzlich diese Information aufgezeichnet, sonst würde man nur die letzte Seite nicht einlesen, um ein nichtgeschütztes Programm zu erhalten. Wir erzeugen also unser END wie die ALPHA-Labels, und für unsere END-Befehle gelten auch die gleichen Vorsichtsmaßnahmen!

1. Zum Programmende gehen (z.B. GTO .001, BST)
2. RCL IND 77, RCL IND 77, PACK, BST, BST, WOLF, , PACK oder GTO ..

Nach dem PACKING erscheint in der Anzeige dann PRIVATE.

Mit dieser Methode kann man auch einfach ein Programm mit den globalen LBLs SIZE und CLP erstellen. So können diese Funktionen nicht mehr benutzt werden, es sei denn, man benutzt eine Statuskarte mit den KA's oder liest diese Funktionen mit dem Lesestift auf dem WAND PAPER KEYBOARD.

Außerdem gibt es eine Möglichkeit, das Programmieren auf dem HP-41C zu verhindern: GTO .., WPRV, und dann die Karte wieder einlesen. Damit enthält das .END. die PRIVATE-Information, und sooft auch GTO .. ausgeführt wird, es läßt sich kein Befehl einfügen. Lediglich mit CLP läßt sich diese .END. Marke löschen, oder durch Einlesen eines Programms (WAND oder Kartenleser). Benutzt man COPY, steht vor dem kopierten Programm ein einames, PRIVATE-geschütztes END, und nachdem das kopierte ROM-Programm gelöscht ist, kann auch wieder programmiert werden.

Register d

Das wohl interessanteste Register im Rechner ist das Register d, in welchem der Zustand der gesamten Flags abgespeichert ist. Durch den Zugriff auf dieses Register werden umfangreiche Manipulationen möglich, z.B. den gesamten Rechnerstatus mit 12 Bytes festzulegen, wenn man bloß RCL M und STO d ausführt. Doch ist beim Umgang mit dem Flag-Register Vorsicht geboten:

- So selten wie möglich STO d verwenden! Besser ist X<>d, damit kann nämlich der alte Status später zurückgespeichert werden.
- Enthält ein Programm Teile, in denen zwischen 2 X<>d-Befehlen gearbeitet wird, sollte das Programm nicht angehalten werden. Es kann z.B. sein, daß in dieser Sequenz das Flag 55 zwischen- durch gelöscht wird. Jedes Anhalten würde es aber wieder setzen, wenn der Drucker angeschlossen ist. Das kann später aber zu falschen Ergebnissen führen. SST setzt außerdem noch Flag 51, wenn das Programm durchgetastet wird.
- Register d sollte nie mit unkontrollierten Werten gefüttert werden (z.B. wenn es als Zwischenspeicher benutzt wird). Es kann z.B. sein, daß das PRGM-Modus Flag gesetzt wird. Jede Zifferneingabe führt dann dazu, daß der 41C seinen Programmspeicher mit Zahlen füllt. Außerdem wird das Flag 53 ständig gelöscht. Wer z.B. EEX 44, X<>d, X<>d programmiert - was eigentlich nichts bewirken sollte, außer daß 1,0000000 44 in der Anzeige steht -, erhält nach diesen 3 Schritten nur 1,0000000 40, da ja Flag 53 gelöscht wurde.

Soweit die Vorsichtsmaßnahmen. Wie schon erwähnt, stehen im Register d alle Flags. Das Bit 0 wird dabei als Flag 0 gebraucht, und Bit 55 als Flag 55. Es werden also alle 7 Bytes x 8 Bits = 56 Bits für die Flags genutzt. Der Rechner schreibt also in Byte 6 nicht, um was für ein Register es sich handelt, denn dieses Register bleibt - egal, wie kurios SIZE gewählt wird - immer an seinem Platz (Hexadezimal: 00E) und enthält immer die Flags. Da nun jedes Flag ein Bit ist und Flag-Nummer sowie Bit-Nummer so auffallend übereinstimmen, können wir damit ALPHA-Zeichen ganz einfach entschlüsseln: Wir praktizieren sie ins d-Register und analysieren mit FS? oder FC? das Bitmuster des Alphazeichens und erhalten so ein numerisches Äquivalent. Nehmen wir doch einmal ein simples M, das wir ins ALPHA-Register praktizieren. Nach RCL M steht in X (bei FIX 9): 0,0000000 4=; In der Byte-Tabelle finden wir, daß die 4= aus der sensationellen Anzeige dem Buchstaben M entspricht. Spötter werden nun sagen, daß das alles zu einem früheren Zeitpunkt schon deutlich abzusehen war. Dennoch ist das ein wichtiger Schritt, wie uns das Programm CD (Character-to-Decimal) beweist. Unser ALPHA-Zeichen ist nämlich als 2-stellige Hexadezimalzahl in Byte 0 untergebracht. In unserer Byte-Tabelle finden wir die erste Stelle (die 4) am linken Rand, und das = (es soll das D sein) am oberen.

Nach unserem RCL M steht also im X-Register:

	00	00	00	00	00	00	4D	
Byte:	6	5	4	3	2	1	0	(Hexadezimal)

Es wäre für unsere Betrachtung allerdings wesentlich günstiger, wenn das 4D in ein anderes Byte käme. Das wird durch die Zeile O2 im PRGM CD erreicht. Nach einem RCL M (bzw. dort wird X<M verwendet) sieht das X-Register (und nach X<d auch das d-Register) so aus:

	00	4D	00	00	00	00	O2	
Byte:	6	5	4	3	2	1	0	(Hexadezimal)

oder in der Bit-(Binären-)Schreibweise:

0000	0000	0100	1101	0000	0000	0000	0000	0000	0000
Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0			

0000	0000	0000	0010
Byte 1	Byte 0		

Jetzt folgt dem X<d-Befehl eine ganze Reihe von Flagtests, die zunächst sinnlos erscheinen, in Wirklichkeit aber ein Geniestreich sind.

Durch diese Tests wird nämlich aus der Hex-Zahl eine Oktal-Zahl gemacht. Im Oktalsystem (8er-System) gibt es nur die Ziffern 0-7. Schreibt man nun diese Ziffern im Dual- oder Binärsystem (2er-System) auf, so findet man, daß unsere Oktalzahl nur 3 Bits benötigt, denn 7 ist 111 und 0 ist 000. Unsere Hex-Zahl dagegen benötigt 4 Bits, nämlich von 0000 bis 1111. Was wäre da einfacher, als die beiden Bits, die da übrig bleiben, wenn man aus einer Hex-Zahl eine Oktalzahl machen will, auf andere Register zu verteilen? Unsere Bytes 5 und 6 sehen mit den beiden Hex-Ziffern so aus:

0000	0000	0100	1101	Hexadezimal
		↙	↘	
0000	0001	0001	0101	Oktal

Die untere Darstellung erreichen wir dadurch, daß wir mit den Flagtests die Werte der Bits verschieben. Nun brauchen wir nur noch DEC auf diesen Registerinhalt anzuwenden, und wir erhalten den dezimalen Wert unseres Alphazeichens.

Jetzt läßt sich auch erklären, warum das letzte Byte im Text der Zeile O2 ein Hex O2 war: Damit wurde das Komma (in der FIX 9 Darstellung) zwischen Byte 4 und 5 gesetzt. So war das Ergebnis später immer ganzzahlig (da DEC ja keine gebrochenen Eingaben akzeptiert). Was aber vorkommen könnte, wäre, daß z.B. aus dem Zeichen ≠ die Oktalzahl 035 wird. Das ist aber eine nicht-normalisierte Zahl. Die Normalisierung kann nicht z.B. durch STO 00 RCL 00 durchgeführt werden, da der Rechner daraus eine 0 macht. Glücklicherweise führen auch mathematische Funktionen z.T. die Normalisierung durch. Doch Vorsicht: Die meisten Rechenroutinen sind auf normalisierte Zahlen angewiesen. Gibt man z.B. 0,0001 EOO ein (die Zahl muß dieses Format auch bei SCI 9 beibehalten, sonst ist es eine normalisierte Zahl), dann muß man mit LOG bis zu 45 sek. auf das Ergebnis warten!

In unserem Fall führt DEC die Normalisierung durch.

Der Rückweg, also die Umwandlung einer Dezimalzahl in ein Alpha-
zeichen, verläuft im Prinzip genau umgekehrt zum eben beschriebenen
Prozeß. Aber nur im Prinzip, denn der Teufel steckt ja bekanntlich
im Detail. Die eingegebene Zahl wird zunächst vom Programm
DC (Decimal-to-Character; frei nach Wickes) in eine Oktalzahl um-
gewandelt und im Register d wieder zu einer Hexzahl zusammenge-
schoben. Doch die Sache wird durch folgenden Haken verkompliziert:
Bei der Bitschieberei im d-Register sollte das Zeichen ≠ (Code 29)
so aussehen:

00 35 00 00 00 00 00 (29 Dez. = 35 Okt.)

Dummerweise normalisiert der Rechner ja. Wenn wir nämlich - so wie
es das Programm braucht - O29 eintasten, dann hat der Rechner
daraus schon 2,9 E1 gemacht, und mit OCT wird daraus 3,5 E1, so
daß unser Register dann so aussieht:

03 50 00 00 00 00 01

Das Problem ist dabei nun folgendes: Ob wir als Zahlencode 1, 10
oder 100 eingeben, die ersten beiden Bytes sehen so aus: 01 00;
wo das Dezimalkomma liegt, wird einzig durch den Exponenten be-
stimmt. Um das ganze nun nicht zu schwer zu machen, wird zur Oktal-
zahl deshalb 10000 addiert, damit die führenden Nullen eingeklemmt
werden. Die ersten 3 Bytes sehen dann für 1, 10 und 100 so aus:
01 00 01, 01 00 10 und 01 01 00.

Nun haben wir also unsere Bits wieder an der Stelle, wo sie CD
zurückgelassen hat. Da die 1 von der 10000, die wir dazuaddiert
haben, nun durch Bit 7 nach dem X<>d repräsentiert wird, löschen
wir einfach Flag 7 und sind sie so los. Im Exponenten steht nun
eine 4, welche durch das Bit 53 gespeichert wird. Flag 53 wird
zwar immer gelöscht, und damit haben wir am Ende des Registers d
(also in Byte 0) keine 4 mehr, und dafür eine NULL; doch im
Kapitel über NULLen wird erklärt, warum uns das nicht weiterhilft.

Nachdem wir die Bits also zurückverschoben haben, bringen wir
unser Zeichen ins Register M und isolieren es von einem Ratten-
schwanz, der ihm da noch anhängt - fertig!

In etwas modifizierter Form wird dieser Trick auch beim Programm
"Σ+S" verwendet (SIZE und ΣREG-Bestimmung).

Wenden wir uns nun dem Programm B3 zu: Mit diesem Programm kann
der Zustand jedes Flags geändert werden. Dazu gibt man dessen
Nummer ein und führt einfach XEQ "B3" aus.

Register c

In Register c befinden sich eine Reihe von interessanten Daten: So kann man mit den Informationen in c die Speichergröße und die Position der Statistikregister bestimmen. Der Rechner speichert dazu hexadezimal die absolute Adresse des Registers 00 und des ersten Statistik-Registers. Um gerade den Ort der Statistik-Register zu bestimmen, bestimmt man einfach die Position des ersten Registers und zieht das von der Adresse des REG 00 ab - fertig. Dabei können diese 6 Register ganz oder teilweise außerhalb des Speichers liegen, was z.B. nach der Tastenfolge SIZE 319, 311, EREG IND X, SIZE 026 der Fall ist, und der Rechner reagiert auf $\Sigma+$, $\Sigma-$ und CL Σ mit NONEXISTENT. Etwas komplizierter ist es da schon, die gegenwärtige Speichergröße zu bestimmen. Wählen wir auf einem einfachen C SIZE 024, dann haben wir 24 Datenspeicher (REG 00 - REG 23) zur Verfügung. Schieben wir nun ein Memory Modul ein, sind es 64 Register mehr, also 88. Jetzt hat aber der Rechner nirgends abgespeichert, ob ein, zwei, drei Module oder ein Quad-RAM (= CV) eingeschoben sind. Daher muß ein Programm, welches so die Speichergröße bestimmt, in einer Schleife erproben, wie viele Datenregister nun tatsächlich vorhanden sind - allerdings muß bei richtigem Aufbau eine solche Schleife höchstens viermal durchlaufen werden.

Das Programm " $\Sigma+S$ " bestimmt diese Positionen der Statistik-Register und die Speichergröße. Nach dem Programmstart mit XEQ " $\Sigma+S$ " steht (nach ca. 2 sek.) in X die Speichergröße und in Y die Position der Statistik-Register.

Eine weitere Informationsquelle kann die Position der .END.-Marke sein. Zieht man von REG 00 diese Adresse ab, weiß man, wieviel Platz insgesamt verbraucht wurde.

Als letztes wollen wir uns noch der ominösen 169 widmen: An dieser Zahl überprüft der Rechner ja, ob das RAM in Ordnung ist oder ob evtl. Veränderungen eintraten - falls er die 169 nicht mehr findet, ist mit den Speicherinhalten auch etwas passiert, und darum zeigt der Rechner MEMORY LOST. Diese Gesamtlöschung kann nun einfach mit \emptyset , STO c erreicht werden, und um jemanden zu ärgern, zeichnet man am besten eine Programmkarte mit SF 11 auf, denn sobald er die Karte einliest, piepst der Rechner und löscht alles. Etwas anders wirkt X<>c: Mit \emptyset , X<>c löscht der Rechner noch nicht. Erst wenn das Programm anhält oder mit R/S unterbrochen wird, erfolgt MEMORY LOST. Speichert man später allerdings den Inhalt von c zurück, passiert nichts. So kann man ein PRGM z.B. vor Unterbrechung schützen: Zwischen zwei X<>c-Befehlen wird ein eingegebener Text mit einem geheimen Wort verglichen. Will der Anwender dieses Wort sehen, muß er mit R/S stoppen - und dann löscht der Rechner den ganzen Speicher.

Eine weitere Anwendung ist z.B. X<>c, WSTS, X<>c. Hier wird nur der Inhalt der Statusregister aufgezeichnet, nicht aber die KA's, egal, ob welche gemacht wurden oder nicht.

Programmierung im ALPHA-Register

Die piepsenden Programme haben gezeigt, was es wert ist, wenn ein Programm selbst programmieren kann. Doch nun zur Erklärung, wie es geht. Wir betrachten dazu das Programm T3. Der Befehl XTOA stammt aus dem X-FUNCTION Modul, und er verwandelt eine Zahl zwischen 0 und 255 in ein Alphazeichen, welches an das letzte Zeichen im ALPHA-Register angehängt wird.

In Zeile 2 steht der Textbefehl F1 9F, und wie durch einen wunderbaren Zufall gefügt, handelt es sich bei 9F um das Byte, welches im Programmspeicher als TONE gelesen wird (wie das bloß kommt?). Mit XTOA hängt das Programm nun die Adresse an den TONE-Befehl an. Dabei ist zu beachten, daß XTOA auch Werte zwischen 128 und 255 akzeptiert, daß jedoch später daraus TONE IND Anweisungen werden - diese überprüfen allerdings, ob der entsprechende Registerinhalt zwischen 0 und 9 liegt!

Nun werden in Zeile 04 mit F3 7F die Befehle CE 7C = X<>b angehängt. Sie dienen dazu, daß der Rechner aus dem ALPHA-Register, welches er als Programm abarbeitet, wieder zum alten Programm zurückkehrt.

Die Zahl E6 in Zeile 5 ist die Adresse, die ins b-Register gespeichert wird. Nach X<>b in der folgenden Zeile sieht das Register b dann nämlich so aus:

01	00 00	00 00	00 06
3.RTN	2.RTN	1.RTN	Adresse jetzt

Die 1 in der 3. Unterprogrammebene (3.RTN), die daher kommt, daß der Rechner immer 1 E6 ins X-Register schreibt, auch wenn im Programm nur E6 steht, ist für die Programmierung unbedeutend.

Nur daß die "Adresse jetzt" in 00 06 verwandelt wurde, macht den Sprung ins ALPHA-Register aus: Die erste 0 war ja die Nummer des Bytes in einem Register, und die letzten 3 geben das Speicherregister an, in welchem der Rechner gerade steht.

Da der Rechner mit X<>b auf das letzte Byte im Register 6 gesetzt wird, landet er im Register N, wo er eben dieses Byte (es ist ein NULL-Byte) abarbeitet, und dann in Register M springt, wo er unseren TONE-Befehl und den X<>b-Befehl findet, welcher in wieder ins alte Programm bringt.

Dabei sind allerdings einige Punkte zu beachten:

- Tastet man z.B. T3 mit SST durch, erscheint in Zeile 8 der TONE-Befehl, und in Zeile 9 ein X<>b-Befehl. Der + Befehl in Zeile 8 des Listings wandert deshalb in Zeile 11 und SIGN in Zeile 10. Dies hat jedoch keine tiefgreifende Bedeutung, solange man nicht versucht, mit BST zur Zeile 1 zu kommen.
- Tritt eine Fehlermeldung durch Operationen im ALPHA-Register auf, so dauert es sehr lange, bis der Rechner wieder benutzbar ist. Dies ist genau dann der Fall, wenn der PRGM-Indikator verschwindet.
- Wer im ALPHA-Register steckenblieb und wieder ins normale Programm zurück will, führt einfach CAT 1 durch.

Mit E8 lassen sich dann bis zu 22 Bytes im ALPHA-Register abarbeiten. Es ist nicht empfehlenswert, noch mehr Bytes im P oder Q-Register abzuarbeiten, da P von jeder Zifferneingabe (z.B. E9) und Q von ALPHA-Eingaben verändert wird.

Ein weiteres Programm zur Demonstration ist APRGM.

Die Bedienung ist denkbar einfach: Um einen 2-Byte-Befehl oder 2 1-Byte-Befehle abzuarbeiten, gibt man deren dezimales Äquivalent (aus der Byte-Tabelle zu entnehmen) ein (Befehl, ENTER, Adresse) und führt XEQ "APRGM" aus. Das Programm kann auch STO, RCL und X<>-Befehle durchführen; für STO und X<> gibt man zuerst die Zahl oder den Text ins X-Register ein. Dann ENTER, Befehl, ENTER, Adresse, XEQ "APRGM". Nach Programmende steht das, was einen RCL oder X<>-Befehl geholt hat, in X.

128 Töne

Wenden wir uns nun einmal der Funktion TONE auf dem HP-41 zu. Mit ihr können 10 verschiedene Töne direkt erzeugt werden. TONE ist eine 2-Byte-Funktion, im ersten Byte steht natürlich der Befehl TONE, im zweiten die Adresse. Von den 256 Möglichkeiten des 2. Bytes werden 128 für die indirekten Operationen benötigt (TONE IND), 10 für die verfügbaren - und die restlichen 118? Nun, es lassen sich insgesamt 16 verschiedene Frequenzen verfügen, die jeweils in 8 verschiedenen Zeitintervallen aufgebaut sind. Um die neuen Töne einmal vorzuführen, bedient man sich zweckmäßigerweise der folgenden Programme: BACHdM - Toccata in d-Moll von Johann Sebastian Bach.

- T1 - Programm von A. Marktscheffel. Nach Eingabe einer Zahl zwischen 0 und 127 piepst der Rechner. Fortwährendes R/S läßt die Töne nacheinander erklingen.
- T2 - Wie T1, da die Druckerfunktion BLDSPEC verwendet wird, kürzer und schneller. Der Drucker muß angeschlossen werden, braucht aber nicht eingeschaltet zu sein.
- T3 - Wie T1, benötigt aber die Funktion XTOA aus dem X-FUNCTION-Modul. Im folgenden dient es auch der Erklärung zur Programmierung im Alpha-Register.

01*LBL "CD"
 02 "I++++X"
 03 X<> [
 04 X<> d
 05 FS?C 08
 06 SF 06
 07 FS?C 09
 08 SF 07
 09 FS?C 10
 10 SF 09
 11 FS?C 11
 12 SF 10
 13 FS?C 12
 14 SF 11
 15 X<> d
 16 DEC
 17 END

ZEILE 02 :
 F6 7F 00 00 00 00 02

 REQ "K"
 NONEXISTENT

 PLNG "CD"
 43 BYTES

01*LBL "DC"
 02 OCT
 03 1 E4
 04 +
 05 X<> d
 06 CF 07
 07 FS?C 19
 08 SF 20
 09 FS?C 18
 10 SF 19
 11 FS?C 17
 12 SF 18
 13 FS?C 15
 14 SF 17
 15 FS?C 14
 16 SF 16
 17 X<> d
 18 STO [
 19 "FABC"
 20 X<> \
 21 CLA
 22 X<> [
 23 AVIEW
 24 END

 PLNG "DC"
 53 BYTES

01*LBL "DEMO"
 02 "0+0+ +rA"
 03 RCL \
 04 X<> d
 05 VIEW [
 06 TONE 7
 07 STOP
 08 X<> d
 09 CLST
 10 CLA
 11 END

LBL'DEMO
 END 39 BYTES

ZEILE 02 :
 FE F8 10 00 30 0A A3 CB
 0A 0E EE EC CC C0 13

01*LBL "T3"
 02 F1 9F
 03 XTOA
 04 F3 7F CE 7C
 05 E6
 06 X<> b
 07 SIGN
 08 +
 09 .END.

LBL'T3
 END 23 BYTES

01*LBL "T1"
 02 ABS
 03 128
 04 MOD
 05 ENTER†
 06 F2 91 7C
 07 LASTX
 08 +
 09 INT
 10 OCT
 11 CHS
 12 X<> d
 13 FS?C 11
 14 SF 12
 15 FS?C 10
 16 SF 11
 17 FS?C 09
 18 SF 10
 19 FS? 07
 20 SF 09
 21 SF 07
 22 SF 05
 23 SF 04
 24 X<> d
 25 STO \
 26 CLX
 27 E7
 28 X<> b
 29 SIGN
 30 +
 31 .END.

LBL'T1
 END 58 BYTES

01*LBL "T2"
 02 F2 01 3E
 03 ASTO Y
 04 BLDSPC
 05 STO [
 06 "I-I"
 07 E6
 08 X<> b
 09 LASTX
 10 1
 11 +
 12 .END.

LBL'T2
 END 29 BYTES

01*LBL "Σ+5"
 02 CLA
 03 RCL c
 04 X<> [
 05 "HAAR"
 06 X<> \
 07 X<> [
 08 REQ 00
 09 "AA"
 10 RCL c
 11 X<> [
 12 X<> \
 13 ASHF
 14 REQ 00
 15 -
 16 LASTX
 17 CHS
 18 64
 19 MOD
 20 SF 25
 21*LBL 01
 22 RCL IND X
 23 FC? 25
 24 RTN
 25 CLX
 26 LASTX
 27 +
 28 GTO 01
 29*LBL 00
 30 "I++++"
 31 X<> [
 32 X<> d
 33 FS?C 10
 34 SF 07
 35 FS?C 11
 36 SF 09
 37 FS?C 12
 38 SF 10
 39 FS?C 13
 40 SF 11
 41 FS?C 14
 42 SF 13
 43 FS?C 15
 44 SF 14
 45 FS?C 16
 46 SF 15
 47 X<> d
 48 INT
 49 DEC
 50 .END.

LBL'Σ+S
 END 101 BYTES

ZEILE 30: F4 7F 00 00 03

LBL'B3
 END 56 BYTES

Der programmierbare Byte-Jumper

Beim programmierbaren Byte-Jumper handelt es sich um eine Entdeckung von Rolf Mach. Kombiniert man nämlich ein GTO mit einem der Kurzformlabels OO-14, dann überspringt der Rechner eine - vom LBL abhängige - Anzahl von Bytes. Normalerweise sieht ein GTO für globale ALPHA-Labels ja so aus:

```
1D Fn XX XX XX XX XX
```

Dabei ist 1D der GTO-Befehl, Fn ist der Textbefehl, der angibt, wieviele Zeichen folgen (es sind genau n Zeichen), und XX sind die Hexcodes der folgenden Alphazeichen.

Beim programmierbaren Bytejumper sieht es dagegen so aus:

```
1D Oi XX YY ZZ
```

Oi ist dabei das entsprechende Kurzformlabel (i = 1 ist LBL OO, i = F ist LBL 14), und i gibt die Anzahl der zu überspringenden Bytes an.

Im folgenden sind die Artikel von Rolf Mach, Sven Beierstorf und Hagen Klemp abgedruckt. Besondere Bedeutung kommt dann der abschließenden Würdigung von Walter Kropf zu, denn hier wird die Bedeutungslosigkeit des PBJ klar.

In diesem Artikel möchte ich auf eine, von mir entdeckte, neue Befehlsgruppe eingehen. Ich nenne sie programmierbarer Byte-Jumper; kurz PBJ. Es handelt sich dabei um die 2-Byte-Befehle 1D, 1E und 1F, die mit 01 bis 0F als zweites Byte gekoppelt sind. Laut Hextabelle sind dies GTOa, XEQa und Spare, in Verbindung mit LBL 00 bis LBL 14. Im Display erscheinen sie als GTO^T, XEQ^T und W^T. Die Länge des Textes, der in diesen Befehlen steht, hängt vom zweiten Digit des zweiten Bytes ab.

Beispiele: 1D 03 im Programmspeicher ergibt in der Anzeige
GTO^TXYZ
1F 07 ergibt W^TTUVWXYZ

Dabei steht TUVWXYZ für irgendeine Alpha-Kette, da dieser Text ein Abbild der folgenden Bytes ist. Beispielsweise wird hinter GTO^TXYZ LN, X² und X geschrieben. Das ergibt dann GTO^TPQR, da PQR in der Hextabelle die zu den Befehlen gehörigen Alphazeichen sind. Um die Haupteigenschaft des PBJ zu verstehen, sollte das folgende Programm eingegeben werden:

```
01 SF 25
02 SF 26
03 GTO "o"
04 TONE 9
05 STOP
06 BEEP
07 .END.
```

```
Zeile 03 ist Hex 1D 03 und
steht in der Anzeige als
GTOTo:
Packen ist notwendig, damit
keine Null-Bytes (erkennbar
an "-") zwischen PBJ und dem
nächsten Befehl stehen.
```

Beim Ablauf hört man jedoch nicht TONE 9, sondern nur BEEP. Wird das Programm im SST-Modus durchgegangen, so verharret der Rechner einen kurzen Moment auf Zeile 03 (er sucht LBL^To, wobei ^To = Hex 9F 09 84; nämlich die Hexcodes für TONE 9 und STOP), um danach mit BEEP fortzufahren. Die drei Bytes TONE 9 und STOP werden also übersprungen. Das SF 25 in 02 ist notwendig, damit das Programm nicht in 03 mit NONEXISTENT anhält.

Hier sieht man zwei Nachteile des PBJ mit 1D und 1E:

- a) Existiert zufällig ein Alphalabel mit dem passenden Text, so wird zu ihm verzweigt.
- b) Es muß Flag 25 gesetzt sein.

Doch was ist mit 1F?

Dieser PBJ arbeitet nach seltsamen Gesetzmäßigkeiten. Die Auswirkungen reichen von Ausschalten bis Aufhängen, je nachdem, welche Peripherie an den HP-41C angeschlossen ist. Bleibt noch das Problem, diese Befehle zu erzeugen. Es geht am besten mit Bar Codes oder dem Byte Jumper (STO 29-31, LBL 00-14); KA kann es nicht.

* INTEGRAL NACH WEDDLE *

DECODIERTE VERSION, SVEN

Zur Anwendung des PBJ
ein Beispielprogramm von
Michael Fehse:

Seit der Beseitigung des PRIVATE schien das "geschützte" Programm ausstorben. Michael fand einen Ausweg. Er ließ sein Programm unter der "Tarnkappe" der Textbytes zum Alphatext werden. Diese Textbytes werden mittels PBJ übersprungen, sobald das Programm ausgeführt wird.

Links die so getarnte Version (numerisches Integral nach Weddle), rechts die von mir wieder normalisierte.

Benutzung: (F programmiert)

```

01*LBL "AB"
02 SF 25
03 XEQ ""
04 "
05*LBL A
06 "F ?"
07 AON
08 STOP
09 ASTO 00
10 AOFF
11*LBL 07
12 CF 22
13 "<a,b> ?"
14 PROMPT
15 FC?C 22
16 GTO A
17 SF 25
18 XEQ ""
19 "uA+üCv+ü00ä"
20 STO 01
21*LBL 08
22 SF 25
23 GTO ""
24 "uv!h00
0

25 DEC
26 GTO IND 01
27*LBL 00
28*LBL 06
29 41
30 GTO 10
31*LBL 05
32*LBL 01
33 216
34 GTO 10
35*LBL 02
36*LBL 04
37 27
38 GTO 10
39*LBL 03
40 272
41 GTO "A"
42*LBL 10
43 SF 25
44 GTO ""
45 "Bw"
46 GTO 08
47 RCL 1
48 840
49 /
50 RCL \
51 6
52 *
53 *
54 TONE 9
55 STOP
56 GTO 07
57 END

01*LBL "AB"
02 FS? 51
03 STO c
04 ADV
05*LBL A
06 "F ?"
07 AON
08 STOP
09 ASTO 00
10 AOFF
11*LBL 07
12 CF 22
13 "<a,b> ?"
"
14 PROMPT
15 FC?C 22
16 GTO A
17 STO IND
c
18 STO I
19 -
20 6
21 /
22 STO \
23 .006
24 STO 01
25*LBL 08
26 RCL I
27 RCL \
28 RCL 01
29 INT
30 *
31 +
32 SF 25
33 XEQ IND
00
34 GTO IND
01
35*LBL 00
36*LBL 06
37 41
38 GTO 10
39*LBL 05
40*LBL 01
41 216
42 GTO 10
43*LBL 02
44*LBL 04
45 27
46 GTO 10
47*LBL 03
48 272
49 GTO "A"
50*LBL 10
51 *
52 ST+ 1
53 ISG 01

```

```

XEQ "AB"
F ?
X
<a,b> ?
RUN
3.0000 ENTER↑
0.0000 RUN
VIEW X
4.5000

```

$$\int_0^3 x dx = 4,5$$

```

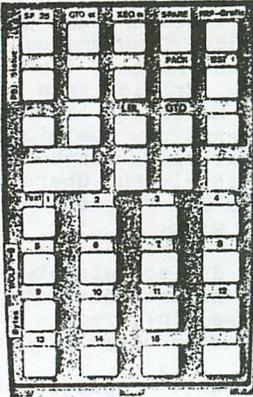
54 GTO 08
55 RCL 1
56 840
57 /
58 RCL \
59 6
60 *
61 *
62 TONE 9
63 STOP
64 GTO 07
65 END

```

← Sprung übr. "LBL 10"
(kein ?...)

Sven

Ich habe eine PBJ-Statuskarte (programmierbare Byte-Jumper) entworfen:



Erklärung des Status:

SF 25: Normale Standardfunktion

GTO α
 XEQ α
 SPARE } : Zur Erzeugung des PBJ

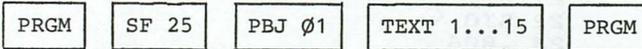
Bytes 1...15: RCL IND Z

⋮
 RCL IND e
 Zur Verschlüsselung der Daten

TEXT : RCL IND Z zur Verschlüsselung von Texten

WOLF 1...9: Der WOLF "frißt" 1-9 Bytes auf. Der so erzeugte String wird mit gelöscht.

Verschlüsselung eines Programms:



Der PBJ steht im Programm. Das SF 25 verhindert ein NONEXISTENT. Dann kommt der PBJ für ein Byte (PBJ 01) und ein Text-Byte, das die folgenden 1...15 Bytes zu Text macht. Dieses Textbyte wird in der Programmausführung vom PBJ übersprungen, so daß der Rechner trotz Textbyte die Bytes als Programmschritte liest. Im Listing springt der Rechner nicht, so daß die Bytes als Text dargestellt werden. Das Programm ist somit verschlüsselt.

Eingabe des PBJ:

Beispiel:

	Hex
.	
⋮	
54 Ø	1 Ø
55 STO d	91 7E
56 ENTER↑	83
⋮	
.	

Das STO d in Zeile 55 soll verschlüsselt werden. Das Programm ist vorher in den Programmspeicher geladen worden.

```
54 Ø
55 STO d
56 ENTER↑
```

Eingabe:

	$\emptyset, \emptyset\emptyset\emptyset\emptyset$	Hex	
GTC .054			
PRGM on	54 \emptyset		
SF 25 ($\Sigma+$)	55 SF 25		Fehlerflag setzen
GTO (1/X)	56 $\text{T}^{\text{--}} \text{L}$	F2 $\emptyset\emptyset$ 29	3.Byte = 1.Byte PBJ
LBL $\emptyset\emptyset$	57 \blacklozenge LBL $\emptyset\emptyset$	\emptyset 1	= 2.Byte PBJ
Bytes 2 (7)	58 RCL IND Y	9 \emptyset F2	2.Byte = Textbyte
BST	57 LBL $\emptyset\emptyset$		
BST	56 $\text{T}^{\text{--}} \text{L}$		
BST	55 SF 25		
WOLF 2 (■ 7)	56 $\text{T}^{\text{--}} \text{2} \text{---} \text{---} \text{---} \text{---}$		2 Byte (F2 $\emptyset\emptyset$) auf- fressen
-	55 SF 25		
SST	56 GTO $\text{T}^{\text{--}} \text{---}$		
PACK (COS)	56 GTO $\text{T}^{\text{--}} \text{---}$		1 Byte (9 \emptyset) auffressen
WOLF 1 (■ -)	57 $\text{T}^{\text{--}} \text{W} \text{---} \text{---} \text{---} \text{---}$		1 Byte (9 \emptyset) auffressen
←	56 GTO $\text{T}^{\text{--}}$		
PACK (COS)	56 GTO $\text{T}^{\text{--}} \text{---}$		

---FERTIG---

PRGM:

	Hex	
54 \emptyset	1 \emptyset	54 \emptyset
55 SF 25	A8 19	55 SF 25
56 GTO $\text{T}^{\text{--}} \text{---}$	29 \emptyset 1	56 GTO $\text{T}^{\text{--}}$
57 $\text{T}^{\text{--}} \text{---}$	F2 91 7E	57 $\text{T}^{\text{--}}$
58 ENTER \uparrow	83	58 ENTER \uparrow
⋮		

Entschlüsselung der PBJ-geschützten Programme:

Die Programme, die bereits mit dem PBJ geschützt sind, lassen sich auf synthetischem Wege entschlüsseln.

Beispiel:

	Hex
54 Ø	1Ø
55 SF 25	A8 19
56 GTO T	29 Ø1
57 T	F2 91 7E
58 ENTER↑	83
⋮	

Beim Entschlüsseln werden einfach die 5 Bytes A8 19 29 Ø1 F2 entfernt.

```
54 Ø
55 SF 25
56 GTO ""
57 "Σ"
58 ENTER↑
```

Eingabe:

```
GTO .Ø54 (vor SF 25)
PACK (COS-Taste)
WOLF 5 (■ +-Taste)
←
---FERTIG---
```

Mit dieser Befehlsfolge werden die nächsten 5 Byte beginnend mit Zeile 5 vom WOLF 5 "aufgefressen". Diese 5 Bytes stehen dann als ALPHA-Zeichen im WOLF-String. Dieser wird dann mit ← gelöscht. Das Programm steht dann in der Ursprungsversion zur Verfügung, kann geändert werden, und bei Bedarf kann man es später wieder verschlüsseln. Der PBJ mit XEQ (und SPARE ???) funktioniert ähnlich.

Sprünge mit dem PBJ:

Mit dem PBJ kann man relative Sprünge um 1...15 Bytes programmieren. Die Methode ist dieselbe wie bei der Verschlüsselung mit dem Unterschied, daß das 2. Byte im Byte-Jumper (bei der Verschlüsselung Ø1, erzeugt durch LBL ØØ) verändert wird (LBL ØØ...LBL 14).

Eingabe:

```
SF 25 (Σ +)
GTO (1/X)
LBL ØØ...14 ←
BST
BST
WOLF 2 (■ 7)
←
PACK (COS)
```

Abhängig von der Sprungweite:
 LBL ØØ ≙ 1 Byte
 LBL Ø1 ≙ 2 Byte
 ⋮
 LBL 14 ≙ 15 Byte

Programm:

```
⋮
SF 25
PBJ ØX
⋮
⋮
⋮
```

Sprung über X Bytes

Erzeugung des PBJ-Status über das Key-Assignment-Programm:

Achtung!!! Genügend freie Register schaffen. Wenn KA im PRGM-Speicher ist, müssen nach GTO .. noch mindestens 15 freie Register sein.

KEY	1.Byte	2.Byte	Taste
1	168	25	11
2	242	29	12
3	242	30	13
4	242	31	14
5	4	10	24
6	4	7	25
7	144	241	51
8	144	242	52
9	144	243	53
10	144	244	54
11	144	245	61
12	144	246	62
13	144	247	63
14	144	248	64
15	144	249	71
16	144	250	72
17	144	251	73
18	144	252	74
19	144	253	81
20	144	254	82
21	144	255	83
22	247	87	-51
23	248	50	-52
24	249	51	-53
25	250	52	-54
26	251	53	-61
27	252	54	-62
28	253	55	-63
29	254	56	-64
30	255	57	-71



POWER-LETTER
© COPYRIGHT BY H.KLEMP

Happy Programmjumping

Hagen

Ich habe ein Programm geschrieben, mit dem man Großschrift schreiben kann. Das Programm heißt POWER-LETTER und ist beim Magnetkartenservice zu erhalten. Es füllt 18 Kartenseiten. Es ist der komplette Zeichensatz und zusätzlich Grafic möglich. Tastenzuordnungen sind möglich (synthetisch). Wer Interesse an Statuskarten und Programm hat, schicke bitte 11 Magnetkarten an den Magnetkartenservice. Bedingt durch die Länge des Programms ist ein HP-41CV oder ein HP-41C mit 4 Memory-Modulen notwendig.

Hagen Klemp (73)

Walter Kropf (56)

Nachstehend einige Betrachtungen zum "Programmierbaren Byte Jumper" und dessen Anwendung zur Verschlüsselung von Programmen. (Siehe prisma 41/81 und 113/81)

Die Idee, Programmbytes in Textzeilen zu verstecken, ist bestimmt eine effektive Methode zur Verschlüsselung von Programmen, welche dadurch kaum mehr lesbar sind. Leider hat die Methode einige Nachteile:

1. Die Programme werden durch den PBJ sehr langsam, da bei jedem GTO "" oder XEQ "" der ganze Katalog durchgesucht werden muß.
2. Jede Textzeile erfordert einen Mehraufwand von 5 Bytes (SF 25, GTO "", Fn).
3. Wenn das Programm im SST abgefahren wird, sind alle Befehle im Klartext zu lesen! FS? 51, STO c, wie im Programm Seite 113 angewendet, hilft auch nicht, wenn nach den ersten Zeilen mit SST begonnen wird. Es müßte am Beginn jeder Textzeile FC? 51 eingegeben werden, doch ist dies wieder ein Mehraufwand von 2 Bytes je Textzeile.
4. Beim Programmablauf im TRACE-Modus werden alle Programmschritte im Klartext ausgedruckt, damit ist die ganze Verschlüsselung umsonst. Ich habe dagegen noch kein Mittel gefunden, obwohl der Rechner den TRACE-Modus irgendwie erkennen muß, bei PRIVATE ist die Ausgabe im TRACE-Modus gesperrt. Wer weiß mehr?

Für diejenigen, die auf diesem Gebiet weiterarbeiten, ein Hinweis: GTO und XEQ-Befehle sind in den Textzeilen möglich, doch darf nach dem ersten Programmablauf keine Änderung im Programm mehr erfolgen (vor Programmablauf packen!), da die Sprungweiten wohl richtig berechnet und gespeichert werden, bei einer Editierung aber diese Bytes, wenn sie in Textzeilen stehen, nicht mehr auf 0 gesetzt werden. Am besten ist es, man zeichnet das Programm nach Packen vor einem Probelauf auf Magnetkarten auf, dann hat man immer das jungfräuliche Programm mit undefinierten Sprungweiten zur Verfügung.

Flags und der Drucker

<u>Funktion:</u>	55 clear: 21 set	55 clear: 21 clear
ACA	X	/
ACCHR	X	/
ACCOL	X	/
ACSPEC	X	/
ACX	X	/
ADV	/	/
BLDSPEC	X	X
PRA	4	/
PRAXIS	1	/
PRBUF	/	/
PRFLAGS	2	/
PRKEYS	2	/
PRPLOT	1	/
PRREG	3	/
PRREGX	3	/
PR Σ	3	/
PRSTK	3	/
PRX	4	/
REGPLOT	5	/
SKPCHR	X	/
SKPCOL	X	/
STKPLOT	5	/

Erprobt wurde diese Tabelle nur auf einem Drucker Serial No. 2037S40054. Änderungen bei anderen Modellen sind also durchaus möglich.

Rolf Mach
Thorwaldsenstraße 3
6090 Rüsselsheim

Die in der Tabelle "Flags und der Drucker" verwendeten Symbole haben folgende Bedeutung:

- X Die Funktion arbeitet normal.
- / Die Funktion hat keine Wirkung.
- 1 Aufgrund falscher Formatierung ist der Ausdruck unbrauchbar.
- 2 Im Buffer stehende Zeichen werden ausgedruckt. Nach einiger "Bedenkzeit" passiert schließlich nichts.
- 3 Der Ausdruck ist in einem gestauchten Format. Siehe Bsp.
- 4 Im Buffer stehende Zeichen werden ausgedruckt. Danach wird die Funktion ACA/ACX ausgeführt.
- 5 An den bisherigen Bufferinhalt werden nnn (Spaltenbreite des Plots; siehe Handbuch Seite 59 unten) leere Spalten angehängt. Bei Überlauf wird eine Zeile ausgedruckt.

Beispiel zu 3:

Im Buffer stehende Zeichen werden ausgedruckt, dann wird er mit den betreffenden Zahlen und Zeichen vollgeschrieben.

```
R00= 3.1416
R01= 3.0000      Normal, groß
R02= 28.500
0
R03= "**REG3*"
"
R04= -523.22
24
R05= -0.0023
R06= 4.0000
```

```
R 3.1416R 3,
0000R 28.500      Gestaut, groß
0R**REG3*R-52
3,2224R-0.00
23R 4.0000
```

```
P00= 3.1416
P01= 3.0000      Normal, klein
P02= 28.5000
P03= "**REG3*"
P04= -523.2224
P05= -0.0023
P06= 4.0000
```

```
P 3.1416R 3.0000P 28.500
0R**REG3*R-523,2224R-0.00      Gestaut, klein
23R 4.0000
```

Hans-Günter Uphues
Bahnhofstraße 20
4401 Altenberge, 12.11.1980
Tel.: 02505/2149

Steigerung der Rechengeschwindigkeit des HP-41C bei Verwendung
des Druckers HP-82143A:

Vielen Benutzern des Taschenrechners HP-41C in Verbindung mit dem Thermodrucker HP-82143 wird es schon aufgefallen sein, daß der 41C bei angeschlossenem und eingeschaltetem Drucker um bis zu 25 % langsamer rechnet (je nach Befehl unterschiedlich; im Mittel ca. 15 %). - Ist der Drucker angeschlossen, aber ausgeschaltet, rechnet der Rechner so schnell wie ohne angeschlossenen Drucker. - Diese Verlangsamung der Rechengeschwindigkeit hat mich sehr gestört, weil ich den Drucker hauptsächlich in Programmen verwende, die eine längere Rechenzeit benötigen.

Im Mai dieses Jahres ist in dem PPC-Calculator-Journal ein Bug-3-Simulator-Programm von Jan Doig veröffentlicht worden, mit dem jeder HP-41C-Besitzer in die Lage versetzt wurde, den Zustand eines beliebigen Flags zu ändern. Mit diesem Programm habe ich nun bei angeschlossenem und eingeschaltetem Drucker (Flag 55 also automatisch gesetzt) das Druckeranwesenheits-Flag 55 gelöscht. Durch diesen Trick liefen meine Programme so schnell, als wäre der Drucker nicht eingeschaltet, also eine Rechenzeitersparnis von bis zu 20 bis 25% maximal. Leider mußte ich aber feststellen, daß man die eingebauten Funktionen des Druckers nicht mehr vollständig verwenden kann. Die Befehle ADV, MTKPLOT, REGPLOT, PRREG(X), PRBUF und PR können nicht mehr verwendet werden, da sie entweder nicht ausgeführt werden oder der Druck hintereinander erfolgt; es wird kein NONEXISTENT angezeigt.

Andere Befehle wie PRX und PRA können weiterhin verwendet werden, werden aber in einer etwas abgewandelten Form gedruckt. Funktionen (Befehle), die den Buffer des Druckers laden, bleiben aktiv, und der Buffer wird auch automatisch zeilenweise ausgedruckt, wenn er "überläuft".

Glücklicherweise ist es aber dennoch möglich, den Buffer gezielt auf anderem Wege ausdrucken zu lassen: nämlich durch die Befehle PRA,X. Bei diesen beiden Befehlen wird dann der Inhalt des X- bzw. Alpha-Registers in den Buffer übernommen, aber noch nicht gleich gedruckt. Erst beim nächsten PRX,A wird der vorherige X- bzw. Alpha-Inhalt gedruckt und der jetzige Wert in den Buffer übernommen. Soweit ist also noch ein PRA, PRX und PRBUF realisierbar. Eine Leerzeile (ADV) läßt sich durch die Befehlsfolge "(space)" PRA realisieren.

```
01+LBL -B3"  
02 ENTER↑  
03 CLA  
04 8  
05 ST/ Z  
06 MOD  
07 X<>Y  
08 INT  
09 1  
10 -  
11 10↑X  
12 RCL d  
13 STO \  
14 FIX 0  
15 CF 29  
16 ARCL Y  
17 RCL \  
18 STO d  
19 FC7C IND  
20 SF IND T  
21 RCL d  
22 STO \  
23 5  
24 LASTX  
25 -  
26 10↑X  
27 FIX 0  
28 CF 29  
29 ARCL X  
30 RCL J  
31 STO d  
32 END
```

Etwas heikel wird die Angelegenheit, wenn man den Buffer durch Befehle wie ACX, ACA und ähnlichen laden möchte: Hierbei werden diese Befehle wohl ausgeführt, aber wenn vorher ein PRX,A-Befehl abgehandelt wurde, steht diese alte Information noch im Buffer, und die neue Information wird hinten angesetzt und beim nächsten Druckvorgang gemeinsam ausgedruckt. Um dies zu verhindern, muß man durch die Befehlsfolge CLA PRA den Buffer ausdrucken, bevor man ihn mit AC-Befehlen neu lädt.

Die VIEW-Befehle führen bei gesetztem Flag 21 zur Programmunterbrechung, wodurch dann das Flag 55, wie auch nach den Befehlen R/S, PROMPT und PSE? wieder gesetzt wird. Durch den VIEW-Befehl ist die Information aber in den Buffer geladen und kann somit anschließend ausgedruckt werden.

War ein ADV-Befehl vor dem Löschen des Flags 55 ausgeführt worden, erfolgt der Bufferausdruck immer rechtsbündig. Um linksbündigen Druck zu erreichen, sollte vor dem Löschen des Flags 55 ein PRBUF-Befehl erfolgen.

Insgesamt gesehen ergeben sich einige vielleicht für spezielle Anwendungsfälle unangenehme Einschränkungen, so daß sich die Verwendung dieses beschriebenen Tricks nur für Programme mit langen Rechenzeiten von mindestens mehreren Minuten eignet.

Achtung: Selbstverständlich muß das Löschen des Flags 55 im Programm erfolgen!! (Unterprogrammaufruf von B3 oder durch synthetische Programmierung)

Happy programming

HP-41C ROM-Version, BYTE-LOADER- und KEY-ASSIGNMENT-Programm

Zu den am meisten benötigten Hilfsmitteln in der synthetischen Programmierung gehören das LB- (Load Bytes) und das MK- (Make Key Assignments) Programm. Mit dem Programm LB können wir jede gewünschte Sequenz Bytes direkt in den Programm-Speicher laden, ohne Byte-Jumping oder andere Tricks anwenden zu müssen. Dieses Programm beinhaltet 2 non-prompting (aufforderungslose) Versionen (L- und -B), die es uns gestatten, unter der Kontrolle eines eigenen Programmes, Bytes zu laden. Mit dem MK können wir jede gewünschte Funktion einer beliebigen Taste zuordnen. In diesem Programm sind ebenfalls 2 Versionen (1K und +K) enthalten, die die Zuordnung unter Programm-Kontrolle gestatten.

Beide Programme enthalten Sicherheitsvorkehrungen, damit nicht belegte Teile des Speichers überschrieben werden, und informieren den Benutzer, wie viele Register verfügbar sind. LB erlaubt Fehlerkorrekturen; MK schützt vorhandene Zuordnungen, überprüft auf nichtexistierende oder benutzte Tasten usw.

Die Programme sind in 3 Gruppen aufgeteilt; jede mit einem eigenen END. Die 1. Gruppe (DC-Gruppe) enthält Unterprogramm-Routinen (DC, 2D, OM und VA), die von beiden, von LB und MK, verwendet werden. Die 2. Gruppe (LB-Gruppe) enthält die Routinen LB, L-, und -B zusammen mit den Unterroutinen XD und QR, die von LB, nicht aber von MK benötigt werden. Die 3. Gruppe (MK-Gruppe) enthält die Routinen MK, 1K, +K und die Unterroutinen LF und E?. Es können alle 3 Gruppen geladen werden. Sollen LB oder MK allein verwendet werden, muß DC auf jeden Fall mit geladen werden. Wichtig! Nach dem Laden jeder Gruppe muß GTO.. ausgeführt werden, damit am Ende jeder Gruppe ein END gebildet wird, und vor LB oder MK muß auch ein END stehen. Sollen die Gruppen in einen leeren Speicher geladen werden, ist anzuraten, DC als 1. Gruppe zu laden.

Benutzeranweisungen für LB:

1. Vor LB muß ein END im Programm-Speicher stehen. Der SIZE muß 012 oder größer sein.
2. Wir gehen zu dem Punkt im Programm-Speicher, an dem die Bytes eingeführt werden sollen (muß nicht am Ende des Programm-Speichers sein) und geben das folgende im PRGM-Modus ein:

```
LBL "++" + + + ... + + + XEQ "LB"
```

(Die +'s bereiten den Raum im Speicher für das Einfügen der Bytes vor und werden außerdem zur Raumüberprüfung gebraucht, deshalb verwenden wir keine ENTER oder etwas anderes anstelle der + Instruktionen.) Damit genügend Raum für n Bytes vorhanden ist, sollte die Anzahl der +'s "n'+6" betragen, wobei "n" das kleinste Vielfache von 7 größer oder gleich "n" ist (z.B. 13 '+s" sind erforderlich zum Einfügen von 1-7 Bytes, 20 '+s" für 8-14 Bytes usw.) Man kann auch einfach für jedes Byte ein "+" plus 12 extra +'s programmieren. Überflüssige +'s werden automatisch durch Null-Bytes ersetzt.

- 2.1. Der Speicher braucht nicht gepacked zu werden.

3. Steht das Programm noch auf XEQ "LB", so verlassen wir den PRGM-Modus und drücken R/S. (Es kann von jedem Punkt aus zwischen LBL "++" und XEQ "LB" gestartet werden. Man kann LB auch vom Tastenfeld aus starten.) Nach ein paar Sekunden stoppt das Programm mit der Aufforderung "#1 OF m?", wobei "m" die maximale Anzahl der Bytes darstellt, die geladen werden können, aufgrund der Anzahl "+s", die programmiert wurden.
- 3.1. Ist die angezeigte Zahl "m" nicht groß genug, führen wir GTO "++" aus und fügen mehr "+s" ein. Haben wir von vornherein nicht genug "+s" programmiert, wird der Rechner "SST, MORE +s" auffordern. Nach Ausführung von SST sind wir bei LBL "++" und können mehr "+s" einfügen. Danach kann neu gestartet werden.
4. Wir geben das dezimale Equivalent des einzufügenden Bytes im Nicht-ALPHA-Modus ein, oder die Hex-Zahl des Bytes im ALPHA-Modus. Nach jeder Eingabe drücken wir R/S. Um das dezimale oder Hex Equivalent eines zu ladenden Bytes zu ermitteln, ziehen wir die Byte-Tabelle zu Rate. Dezimal oder Hex Eingaben können vermischt werden. Jede Eingabe wird nach ihrer Art richtig verarbeitet. Für die Hex-Eingabe geben wir aber nur 2 und wirklich nur 2 Ziffern ein.
5. Um das Byte-Loading zu beenden, drücken wir R/S ohne vorherige Eingabe. Wenn so viele Bytes geladen wurden, wie geladen werden konnten, endet der Prozeß automatisch. Jetzt erscheint die Aufforderung "SST, DEL OOp", "p" wird eine Zahl zwischen 1 und 7 sein. Wir führen SST aus, gehen in den PRGM-Modus (wir sehen LBL "++") und führen DEL OOp aus. Dies vernichtet LBL "++" und die restlichen "+s".
6. Hinter der kreierte Zeile stehen möglicherweise noch "+s", gefolgt von XEQ "LB". Sollen weitere Zeilen geschaffen werden, können sie stehen bleiben, ansonsten gelöscht werden. Im X-Register steht eine Zahl "p,OOq". "p" war die zu löschende Zeilenzahl vor der kreierte Zeile, "q" ist die zu löschende Zeilenzahl hinter der kreierte Zeile.
7. Wenn wir einen Fehler in der Eingabe bemerken und bereits R/S gedrückt haben, brauchen wir nur XEQ O3 auszuführen. Oder, wenn der Rechner im ALPHA-Modus steht, können wir irgendeinen Buchstaben eingeben und R/S drücken. Die jeweils letzte Eingabe wird damit rückgängig gemacht. So kann man alle Eingaben, eine nach der anderen, rückgängig machen. Dies gilt, solange LB noch nicht verlassen wurde. Eine negative Eingabe hat den gleichen Effekt. Dies ist wichtig für Benutzung von LB unter Programm-Kontrolle.
8. Sollten wir vergessen haben, welches die letzte Eingabe war, so werden wir nach XEQ O1 daran erinnert. Ist das Byte-Loading bereits beendet, haben aber LB durch SST noch nicht verlassen, so hat XEQ O1 die gleiche Wirkung wie XEQ O3.

Warnungen:

- a) Es liegt in der Verantwortung des Benutzers, dafür zu sorgen, daß genug Raum vorhanden ist, um eine Multi-Byte-Zeile beenden zu können. Wenn wir beginnen, eine Text-Zeile mit einer Länge von 15 Zeichen zu kreieren und der Vorgang wird beendet, ohne daß die letzten beiden Zeichen untergebracht werden

können, dann müssen wir damit rechnen, daß einiges aus einem anderen Programm, sogar ENDS, verschluckt werden.

- b) Während des Byte-Loadings darf weder gepacked noch der SIZE verändert werden.
- c) LB darf nicht verlassen werden, ohne die Beendigungs-Prozedur (Punkt 5.) durchgeführt zu haben, insbesondere; wenn Fehler korrigiert wurden.
- d) Während des Byte-Loading dürfen wir sowohl den Stack als auch die Register OO-05 benutzen, aber nicht die Register O6-11, weil die von LB intern verwendet werden.

Beispiel zum Gebrauch:

Nehmen wir an, wir wollen eine Text-Zeile kreieren, die so aussieht: `T-ÄH` (das sind 4 Zeichen, die wir anhängen (Append) wollen, mit den Hex-Codes OO, O1, OC und 4O), und diese Zeile soll eingefügt werden nach dem TONE 9 im folgenden Programm:

```
LBL "TEST" TONE 9 AVIEW END
```

Wir gehen davon aus, daß dieses Programm das 1. Programm-Speicher ist.

1. Wenn wir auf der Byte-Tabelle nachsehen, stellen wir fest, daß die Bytes folgende Kennzahlen haben:

F5	7F	OO	O1	OC	4O	(hex), oder
245	127	O	1	12	64	(dezimal).

2. Wir gehen zu TONE 9 und, im PRGM-Modus, tippen ein:

```
LBL "++" (13 +'s) XEQ "LB"
```

3. Wir verlassen den PRGM-Modus und drücken R/S. Nach ein paar Sekunden sehen wir die Aufforderung "`≠#1 OF 7?`".
4. Wir geben ein: 245, R/S, 127, R/S, O, R/S, 1, R/S, 12, R/S, 64, R/S (jetzt müßte "`≠#7 OF 7?`" zu sehen sein). Als Alternative könnten wir im ALPHA-Modus eingeben: F5, R/S, 7F, R/S, OO, R/S, O1, R/S, OC, R/S, 4O, R/S. Oder wir können die dezimalen und Hex-Eingaben mischen (nur darf der Modus nicht zwischen der Eingabe und R/S gewechselt werden). Die Hex-Eingaben werden langsamer verarbeitet, weil sie in das dezimale Equivalent umgerechnet werden müssen.
5. Wir drücken R/S in beiden Modi, ohne vorher eine Eingabe gemacht zu haben, um den Prozeß zu beenden. Wir sehen die Aufforderung "SST, DEL OO6". Wir drücken SST (die Zahl 6,002 in X verdient Beachtung), gehen in den PRGM-Modus und führen DEL OO6 aus. Nach Druck auf SST können wir die kreierte Text-Zeile sehen. Der Bruch-Anteil ,002 in X bedeutet, daß nach der Text-Zeile noch 2 Befehle zu löschen wären (+, XEQ "LB").

Anweisungen für L- und -B:

Diese Routinen gestatten das Byte-Loading unter Kontrolle eigener Programme. Beispiele sollen hier nicht gegeben werden, aber generale Regeln!

1. Wir programmieren XEQ "L-" in dem Programm, das das Byte-Loading kontrollieren soll. Dieser Befehl initialisiert den Prozeß und kehrt zum Kontroll-Programm zurück.

2. -B produziert die Bytes und orientiert sich an einer Zahl, die in X stehen muß. Es müssen dezimale und dürfen keine Hex-Zahlen sein. Es wird immer nur 1 Byte zur Zeit produziert.
3. Um den Prozeß zu beenden, muß die Befehlsfolge CF 09, XEQ "-B" im Programm stehen. Danach erscheint "SST, DEL Oop".
4. Bevor das Kontroll-Programm gestartet wird, überprüfen wir auf SIZE 012 und schaffen dort Raum im Speicher, wo die Bytes geladen werden sollen, in dem wir wie üblich programmieren: LBL "++", eine Reihe von "+s", XEQ "LB".
5. Wir verlassen den PRGM-Modus und starten das eigene Programm, anstatt durch R/S den Byte-Loader zu starten.
6. Die Ausführung wird wie gehabt beendet mit der Aufforderung: "SST, DEL OOp", so daß das weitere wie beim normalen Byte-Loading abläuft.
7. Soll unser Programm ein falsches oder unerwünschtes Byte korrigieren, so geschieht es, indem eine negative Zahl nach X gegeben und "-B" ausgeführt wird.

Warnung:

"-B" darf nicht aufgerufen werden, ohne daß vorher erst "L-" aufgerufen wurde. Einige Flag und andere Sicherheitsvorkehrungen werden durch "L-" aktiviert, weil "-B" allein für sich ausgeführt zu MEMORY LOST führen kann, oder zumindest andere Programme beschädigt oder vernichtet.

Anweisungen für MK:

1. Wie bei LB muß vor MK ein END im Programm-Speicher stehen, und der Size muß 012 oder größer sein.
2. Wir starten durch XEQ "MK". Das Programm überprüft das Key-Assignment-Register von oben bis unten und zeigt in einer PSE an, wieviele Register noch frei sind. Wenn wir diese Zahl verdoppeln, wissen wir die mögliche Anzahl von Zuordnungen.
- 2.1. Wenn "NO ROOM" angezeigt wird, müßten wir entweder einige Zuordnungen löschen oder den SIZE reduzieren. Anschließend kann durch R/S neu gestartet werden. Es gäbe auch noch die Möglichkeit, andere Programme zu löschen oder das ROM-Programm PK aufzurufen, um das Key-Assignment-Register zu packen.
3. Nachdem die Anzahl der freien Register genannt wurde, fordert MK "PRE#POST#KEY" auf. Wir geben das dezimale Equivalent für das 1. Byte (Prefix) ein, drücken Enter, das 2. Byte (Postfix), Enter, und den Tasten-Code. Dann starten wir die Zuordnung mit R/S (z.B. 159, Enter, 26, Enter, -81, R/S; um TONE 26 auf die geschiftete Taste in der Zeile 8, Spalte 1, zu legen). Wenn die Belegung richtig vollzogen wurde, fordert MK zur nächsten Eingabe auf (die Zuordnungen werden nicht gezählt, müssen also nicht paarweise eingegeben werden, wie bei anderen KA-Programmen). Der Stack wird vor Eingabe-Aufforderung gelöscht, damit der Prefix Null ist, falls versehentlich nur Postfix und Tasten-Code eingegeben werden.
4. Wenn eine Null für den Tasten-Code eingegeben wird, oder R/S ohne Eingabe gedrückt wird, zeigt der Rechner die Anzahl der noch freien Register an, um dann wieder zur Eingabe aufzufordern.

5. Wenn wir keine Zuordnungen mehr machen wollen, können wir entweder den Rechner ausschalten oder an irgendeinen anderen Punkt des Programm-Speichers gehen. Bei MK ist keine Beendigungs-Prozedur notwendig.
6. Die Anzeige "KEY TAKEN", gefolgt von dem PROMPT "KEYCODE?", bedeutet, daß die gewählte Taste bereits belegt ist. Entweder löschen wir die vorhandene Belegung, oder wir wählen einen anderen Tasten-Code und drücken R/S, um die neue Belegung zu vollziehen. "NO SUCH KEY", gefolgt von "KEYCODE?", bedeutet, daß versucht wurde, eine nicht vorhandene Taste zu belegen.
Nach jeder Fehler-Anzeige steht der ursprüngliche, falsche Tasten-Code in X, so daß er wieder verwendet wird, wenn R/S ohne Neueingabe gedrückt wird. Bei einer neuen, richtigen Eingabe wird die Belegung vollzogen. Bei einer Null als Eingabe werden wieder die noch freien Register angezeigt und anschließend zur neuen Gesamt-Eingabe aufgefordert.
7. Die Anzeige "DONE, NO MORE" bedeutet, daß die letzte Belegung getätigt wurde und jetzt kein Raum mehr für neue Belegungen vorhanden ist. Falls trotzdem neue Belegungen gemacht werden sollen, müssen wir wie bei Punkt 2.1. handeln.
8. Nach jedem Stop wegen einer Fehler-Anzeige (Punkt 2.1., 6. oder 7.), oder, falls der Rechner aus- und wieder eingeschaltet wurde, überprüft er das Key-Assignment-Register, um sicherzustellen, daß eine neue Belegung ohne Überlappung oder Lücke geschieht (durch den Test von Flag 20 wird entschieden, ob die Register überprüft werden müssen. Ist das Flag gelöscht, werden die Register überprüft).

Warnung:

- a) Falls zwischen 2 Tasten-Belegungen gepackt oder der SIZE geändert werden muß, schalten wir den Rechner aus und wieder ein (oder löschen Flag 20), um dem Programm zu signalisieren, daß die Register überprüft werden müssen.
- b) Es darf nichts in die Daten-Register 09, 10 oder 11 abgespeichert oder der Zustand der Flags 07, 09, 10 oder 20 zwischen 2 Belegungen gewechselt werden. Wie bei LB werden von MK die Register 06-11 intern benutzt. R 06, R 07 und R 08 enthalten den Prefix, Postfix und Tasten-Code für die zuletzt eingegebene Belegung; R 09 enthält den Index für die indirekte Speicherung der nächsten Belegung, und R 10 enthält den Inhalt des c-Registers. Wenn Flag 10 vom Benutzer gesetzt wurde, enthält R 11 die 1. Belegung eines Tasten-Belegungs-Paares.

Anweisungen für 1K und +K:

Das folgende ermöglicht es, Tasten-Belegungen unter Programm-Kontrolle zu machen.

1. Der SIZE muß 012 oder größer sein.
2. Wenn wir unter Programm-Kontrolle eine einzige Belegung machen wollen, müssen wir dafür sorgen, daß der Prefix in Z, der Postfix in Y und der Tasten-Code in X steht, bevor "1K" als Unterprogramm (XEQ "1K") aufgerufen wird. Wenn die Belegungseingabe einen Fehler enthält, wird das Programm mit einer

Fehler-Anzeige anhalten, vorausgesetzt, wir haben nicht Flag 25 gesetzt, bevor "1K" aufgerufen wurde. Wurde Flag 25 gesetzt, wird bei einem Eingabe-Fehler die Belegung nicht durchgeführt. Im ALPHA-Register kann nachgelesen werden, um welchen Fehler es sich gehandelt hat.

3. Sollen mehrere Belegungen durchgeführt werden, wird für die 1. "1K" und für jede weitere "+K" aufgerufen. Vor jedem Aufruf müssen die entsprechenden Daten in Z, Y und X stehen. "+K" tut das gleiche wie "1K", aber es überprüft nicht die Register (wenn Flag 20 gesetzt ist); dadurch erfolgen die Belegungen wesentlich schneller. Ist jedoch Flag 20 gelöscht, überprüft "+K" genau wie "1K" die Register (nach einer Register-Überprüfung ist Flag 20 jedoch stets gesetzt). Ist das Kontroll-Programm so umfangreich, daß es unbequem wird, "1K" für die 1. und "+K" für jede weitere Belegung aufzurufen, dann kann "+K" für alle Belegungen verwendet werden. Es müssen nur jedesmal Flag 07 gesetzt und Flag 20 gelöscht werden, bevor "+K" aufgerufen wird (Flag 07 verhindert das PROMPT).
4. "1K" und "+K" können auch vom Tastenfeld aus gestartet werden, nachdem die entsprechenden Daten in den Stack gebracht wurden. Wird "+K" ausgeführt, wird es die PROMPTing-Version verwendet, falls vorher "MK" ausgeführt wurde (Flag 20 ist gelöscht), oder die non-PROMPTing-Version, falls "1K" vorher ausgeführt wurde (Flag 20 ist gesetzt).

Warnung:

Wir dürfen "+K" nicht allein benutzen (ohne daß vorher "MK" oder "1K" ausgeführt wurde), ohne sicher zu sein, daß Flag 20 gelöscht ist, und wir müssen den Zustand von Flag 07 überprüfen, um sicher zu sein, daß die richtige Version abläuft. Die gleiche Warnung (Flag 20 muß gelöscht sein) gilt, wenn der SIZE geändert wurde, Tasten-Belegungen manuell ausgeführt oder gelöscht wurden, das Key-Assignment-Register gepacked wurde oder die Inhalte der Register 09-11 verändert wurden.

Verwendete Unterprogramme:

Das folgende ist eine kurze Beschreibung der Routinen, die von "LB" und "MK" verwendet werden.

- DC (Decimal to Character): Wandelt das in X stehende, dezimale Equivalent eines Bytes in das entsprechende ALPHA-Zeichen um und hängt es im ALPHA-Register an (APPEND).
- 2D (2 Bytes to Decimal): Ermittelt das dezimale Equivalent der letzten beiden Bytes eines ALPHA-Strings in X. Das dezimale Equivalent des vorletzten Bytes verbleibt in X; das des letzten Bytes kommt in M zu stehen.
- OM (Open Memory): Erzeugt einen neuen Byte-String und tauscht den ursprünglichen Inhalt des c-Registers gegen diesen aus. Dadurch wird der Curtain (Grenze zwischen Daten- und Programm-Register) auf die Adresse 010 (hex) bzw. 16 (dezimal) gesetzt und die absolute Adresse des Z-Registers auf 1FF (hex). Die Adresse des .END. wird nicht verändert. Dies erlaubt, Werte direkt in das Key-Assignment-Register und die Programm-Register abzuspeichern. Ein Programm kann während dieses Zustands

ruhig gestoppt werden, ohne daß MEMORY LOST geschieht; globale Label sind weiterhin aufrufbar. Der in X stehende ursprüngliche Inhalt des c-Registers kann wie ALPHA-Werte abgespeichert und wieder abgerufen werden.

Warnung: Dieser Wert darf nicht verloren gehen, da er benötigt wird, um, durch das Rückspeichern ins c-Register, den Curtain wieder auf die alte Adresse zurückzubringen. Wenn OM allein für sich verwendet werden soll, muß irgendwo über diesem Programm ein END im Speicher stehen, damit nach der Ausführung dieser Routine man durch zweimal BST zu der X<>c-Instruktion zurückkommen kann, durch die der Curtain auf die alte Adresse zurückgebracht werden kann. OM ist eine von den sehr machtvollen Routinen, die Vorsicht und Sorgfalt in der Handhabung verlangen.

- VA (View Alpha): Zeigt den Inhalt des Alpha-Registers an, genau wie AVIEW. Wenn der Drucker angeschlossen und eingeschaltet ist und Flag 21 gesetzt ist, wird der Inhalt auch ausgedruckt. Ungleich zu AVIEW, hält VA das Programm niemals an.
- XD (Hex to Decimal): Wandelt einen 2-Digit Hex-Wert (in Form von 2 Zeichen in Alpha) in eine dezimale Zahl um, die in X zu stehen kommt. Benutzt QR als Unterroutine.
- QR (Quotient and Remainder): Ersetzt die Werte aus Y und X durch $(Y - Y \text{ mod } X) / X$ (der Quotient) und $Y \text{ mod } X$ (der Rest). Verändert nicht den Inhalt des Alpha-Registers, solange dieser nicht länger als 14 Zeichen ist.
- LF (Locate Free Register): Überprüft das Key-Assignment-Register und gibt folgendes aus: X = aaa,eee, wobei aaa und eee die Adressen (relativ zur Curtain-Adresse 16 dezimal) sind, des Anfangs und des Endes des Blocks freier Register zwischen dem Key-Assignment und dem .END. im Programm-Speicher; Y = c-Register-Inhalt für Curtain auf 16 (erzeugt durch OM). Benutzt OM und E? als Unterroutinen.
- E? (.END. Finder): Ermittelt die absolute Adresse des Programm-Registers, welches das .END. enthält, und plaziert die Adresse (in dezimal) in X. Benutzt 2D als Unterroutine.

Synthetische Instruktionen, die in LB, MK und den Unterroutinen verwendet werden.

<u>Programm</u>	<u>Zeile</u>	<u>Synthetischer Code (hex)</u>
DC	74	Text: F5 1F FO 01 69 01
LB	80	3 Byte GTO 10: DO 00 OA
LB	111	TONE iu: 9F 87
LB	124	3 Byte GTO 03: DO 00 03
LB	164	3 Byte GTO 00: DO 00 00
LB	172	TONE 83: 9F 53
LB	173	3 Byte GTO 00: DO 00 00
LB	199	TONE 54: 9F 36
LB	219	3 Byte GTO: DO 00 08
LB	223	3 Byte GTO: DO 00 OA
MK	15	3 Byte GTO 07: DO 00 07
MK	62	3 Byte GTO 01: DO 00 01
MK	96	3 Byte GTO 08: DO 00 08
MK	109	3 Byte GTO 08: DO 00 08
MK	122	3 Byte GTO 09: DO 00 09
MK	135	Text: F3 2A 2A FO
MK	165	3 Byte GTO 03: DO 00 03
MK	178	TONE 83: 9F 53
MK	180	3 Byte GTO 01: DO 00 01
MK	193	TONE 83: 9F 53
MK	202	3 Byte GTO 01: DO 00 01
MK	212	Text: F8 2A 10 2A 00 00 2A 2A FO

Unterprogramm-Gruppe DC (Decimal to Character). Wird benötigt von der Programm-Gruppe LB (Load Bytes) und der Gruppe MK (Make Key Assignments). Erschienen im PPC Calculator Journal, Jahrgang 8 (1981), Heft 2 (März/April), Seite 34-38.

```

01♦LBL "DC"
02 INT
03 256
04 MOD
05 LASTX
06 +
07 OCT
08 X<> d
09 FS?C 11
10 SF 12
11 FS?C 10
12 SF 11
13 FS?C 09
14 SF 10
15 FS? 07
16 SF 09
17 FS? 06
18 SF 08
19 X<> d
20 X<> [
21 RCL \
22 "f*"
23 X<> ]
24 X<>Y
25 STO \
26 X<> ↑
27 "f*"
28 STO ↑
29 RDN
30 X<> ]
31 X<> \
32 STO [
33 RDN
34 RTN

35♦LBL "2D"
36 "-"
37 X<> [
38 X<> \
39 ASHF
40 "f♦♦♦♦♦♦♦♦♦♦"
41 X<> [
42 X<> \
43 X<> [
44 "f♦♦♦♦♦"
45 RCL [
46 INT
47 +
48 RCL \
49 *
50 ST+ [
51 X<> \
52 RCL ]
53 INT
54 HMS

55 *
56 RCL ]
57 +
58 E1
59 ST* [
60 *
61 X<> [
62 RTN

63♦LBL "OM"
64 RCL c
65 STO [
66 "f♦♦♦♦♦♦♦♦♦♦"
67 X<> [
68 X<> d
69 CF 00
70 CF 01
71 CF 02
72 CF 03
73 X<> d
74 "※i*"
75 X<> [
76 STO \
77 "f**"
78 X<> \
79 CLA
80 X<> c
81 RTN

82♦LBL "VA"
83 SF 25
84 PRA
85 SF 25
86 FS?C 21
87 CF 25
88 AVIEW
89 FC?C 25
90 SF 21
91 END

CAT 1
LBL*DC
LBL*2D
LBL*OM
LBL*VA
END
200 BYTES
40 =
F77F00070060
0000
44 =
F47F000005
66 =
F67F00000000
00
74 =
F51FF0016901

```

Programm-Gruppe LB (Load Bytes). Benötigt Unterprogramm-Gruppe DC (Decimal to Character). Erschienen im PPC Calculator Journal, Jahrgang 8 (1987), Heft 2 (März/April), Seite 34-49. Blatt 1 von 2.

01♦LBL 00	52 RCL [106 ARCL 06
02 STOP	53 +	107 "F OF "
03 GTO "++"	54 7	108 ARCL 07
	55 *	109 "F?"
04♦LBL "LB"	56 +	110 XEQ "VA"
05 FS? 50	57 115	111 TONE 7
06 GTO 00	58 X<>Y	112 STOP
07 "DEC/HEX	59 -	113 FS? 48
INPT"	60 7	114 GTO 14
08 XEQ "VA"	61 XEQ "QR"	115 FC? 22
09 CF 08	62 ST- Z	116 GTO 19
10 GTO 13	63 X<>Y	117 GTO 08
	64 CHS	
11♦LBL "L-"	65 STO 09	118♦LBL 14
12 CLA	66 X<> Z	119 FC? 23
13 XEQ "VA"	67 LASTX	120 GTO 19
14 CF 08	68 XEQ "QR"	121 XEQ "XD"
15 RCL a	69 1,001	
16 STO [70 ST* 09	122♦LBL 08
17 RCL b	71 ST+ Y	123 X<0?
18 FS? 08	72 FRC	124 GTO 03
19 GTO 14	73 ST* T	125 ENTER↑
20 STO \	74 X<>Y	126 CLA
21 SF 08	75 R↑	127 ARCL 08
	76 +	128 XEQ "DC"
22♦LBL 13	77 *	129 RCL 06
23 RCL 11	78 X<>Y	130 X<=0?
24 CLST	79 X<=0?	131 GTO 10
25 STO 06	80 GTO 10	132 7
26 SIGN	81 ST+ 09	133 MOD
27 ENTER↑	82 7	134 X=0?
28 ENTER↑	83 *	135 GTO 07
29 R↑	84 +	136 X<>Y
30 GTO "++"	85 STO 07	137 RCL 09
	86 XEQ "OM"	138 RCL 10
31♦LBL 00	87 X<> c	139 X<> c
32 RCL b	88 STO 10	140 RCL [
33 FC? 08	89 CLST	141 STO IND
34 GTO 14		142 X<>Y
35 CLD	90♦LBL 06	143 X<> c
36 X<> [91 STO 11	144 R↑
37 STO a	92 CLA	145 RCL 08
38 X<> \		146 DSE 09
39 X<> b	93♦LBL 07	147 GTO 06
	94 ASTO 08	148 ISG 09
40♦LBL 14	95 X<>Y	
41 STO [96 ISG 06	149♦LBL 20
42 "F*****"		150 CF 09
43 X<> \	97♦LBL 15	151 CLX
44 XEQ "2D"	98 SF 09	152 X<>Y
45 2	99 FS? 08	153 RCL 07
46 /	100 RTN	154 FRC
47 INT	101 CF 22	155 E3
48 LASTX	102 CF 23	156 *
49 FRC	103 FIX 0	157 AOFF
50 512	104 CF 29	158 FIX 0
51 *	105 "#"	

Programm-Gruppe LB (Load Bytes). Benötigt Unterprogramm-Gruppe DC (Decimal to Character). Erschienen im PPC Calculator Journal, Jahrgang 8 (1987), Heft 2 (März/April), Seite 34-39. Blatt 2 von 2.

159 *SST, DE L 00"	207 ARCL 10	256 XEQ "QR"
160 ARCL X	208 CLX	257 ST
161 FIX 3	209 X<> \	258 ST- Z
162 XEQ "VA"	210 STO [259 -
163 BEEP	211 ASTO 08	260 ,9
164 GTO 00	212♦LBL 09	261 ST* Z
165♦LBL 01	213 RDN	262 *
166 RCL 06	214 GTO 15	263 INT
167 X>0?	215♦LBL "--B"	264 X<>Y
168 GTO 09	216 FC? 08	265 INT
169♦LBL 10	217 GTO 15	266 16
170 *SST, MO RE +:S"	218 FS? 09	267 *
171. XEQ "VA"	219 GTO 08	268 +
172 TONE 3	220♦LBL 19	269 RTN
173 GTO 00	221 RCL 06	270♦LBL "QR"
174♦LBL 03	222 X<=0?	271 X<>Y
175 *CORREC TION**	223 GTO 10	272 STO]
176 XEQ "VA"	224 CHS	273 X<>Y
177 TONE 6	225 ISG X	274 MOD
178 FC? 09	226 7	275 ST-]
179 GTO 01	227 MOD	276 LASTX
180 DSE 06	228 X=0?	277 ST/]
181 GTO 14	229 GTO 14	278 CLX
182 ISG 06	230 CLA	279 X<>]
183 GTO 10	231 ARCL 08	280 X<>Y
184♦LBL 14	232♦LBL 11	281 END
185 RCL 06	233 "I♦"	CAT 1
186 7	234 DSE X	LBL*LB
187 MOD	235 GTO 11	LBL*L-
188 X=0?	236 X<> [LBL*-B
189 ISG 09	237♦LBL 14	LBL*XD
190 GTO 14	238 RCL 09	LBL*QR
191 RCL 11	239 X<>Y	END
192 X=Y?	240 RCL 10	558 BYTES
193 GTO 13	241 X<> c	105 = <HEX>
194 STO 08	242 X<>Y	F123
195 RDN	243♦LBL 12	233 = <HEX
196 STO 11	244 STO IND	CLA
197 GTO 09	245 CLX	F27F00
198♦LBL 13	246 DSE Z	253 = <HEX>
199 TONE 4	247 GTO 12	F37F0008
200 6	248 RDN	
201 ST- 06	249 X<> c	
202 R↑	250 RDN	
203 GTO 15	251 GTO 20	
204♦LBL 14	252♦LBL "XD"	
205 CLA	253 "I♦Δ"	
206 ARCL 08	254 RCL [
	255 E2	

Programm-Gruppe MK (Make Key Assignments). Benötigt Unterprogramm-Gruppe DC (Decimal to Character). Erschienen im PPC Calculator Journal, Jahrgang 8 (1981), Heft 2 (März/April), Seite 34-40. Blatt 1 von 2.

01♦LBL "MK"	50♦LBL "+K"	105 +
02 CF 25	51♦LBL 14	106 36
03 RCL 11	52 STO 08	107 -
04 CF 07	53 RDN	108 X>0?
05 SF 09	54 STO 07	109 GTO 08
	55 RDN	110 FC? 09
06♦LBL 01	56 STO 06	111 RCL ^
07 XEQ "LF"	57 CF 09	112 FS? 09
08 STO 09	58 RCL 10	113 RCL e
09 E	59 SIGN	114 FC? 08
10 +	60 FS? 20	115 GTO 14
11 X<>Y	61 X≠0?	116 STO [
12 STO 10	62 GTO 01	117 "I*"
13 ASTO 11		118 X<> [
14 DSE Y	63♦LBL 13	
15 GTO 07	64 RCL 08	119♦LBL 14
16 SF 20	65 INT	120 X<> d
17 FC?C 09	66 X=0?	121 FS? IND
18 GTO 13	67 FS? 07	
	68 FC?C 20	122 GTO 09
19♦LBL 02	69 GTO 02	123 SF IND Y
20 RCL 09	70 X<0?	124 X<> d
21 INT	71 SF 09	125 FC? 08
22 LASTX	72 ABS	126 GTO 14
23 FRC	73 STO Z	127 STO [
24 E3	74 44	128 ARCL 10
25 *	75 -	129 X<> \
26 X<>Y	76 ABS	
27 ,5	77 2	130♦LBL 14
28 FC? 10	78 X<Y?	131 FC? 09
29 SIGN	79 DSE T	132 STO ^
30 -	80 R↑	133 FS?C 09
31 -	81 STO Y	134 STO e
32 "REG FRE	82 E1	135 "***"
E: "	83 ST/ Z	136 FS? 10
33 RCL d	84 MOD	137 ARCL 11
34 FIX 1	85 8	138 X<> Z
35 ARCL Y	86 *	139 RCL 07
36 STO d	87 ENTER↑	140 RCL 06
37 XEQ "VA"	88 CF 08	141 XEQ "DC"
38 TONE 6	89 LASTX	142 XEQ "DC"
39 PSE	90 FS? 09	143 XEQ "DC"
	91 ST+ Y	144 FS?C 10
40♦LBL 03	92 R↑	145 GTO 14
41 "PRE↑POS	93 INT	146 "I♦♦♦"
T↑KEY"	94 X≠0?	147 ASTO 11
42 CLST	95 X>Y?	148 SF 10
43 XEQ "VA"	96 GTO 08	
44 TONE 7	97 R↑	149♦LBL 14
45 STOP	98 +	150 RCL 09
46 GTO 14	99 ST+ Z	151 RCL 10
	100 X<>Y	152 X<> c
47♦LBL "1K"	101 X<=Y?	153 RCL [
48 CF 20	102 CLX	154 STO IND
49 SF 07	103 X≠0?	
	104 SF 08	155 X<>Y
		156 X<> c

Programm-Gruppe MK (Make Key Assignments). Benötigt Unterprogramm-Gruppe DC (Decimal to Character). Erschienen im PPC Calculator Journal, Jahrgang 8 (1981), Heft 2 (März/April), Seite 34-40. Blatt 2 von 2.

157 CLST			
158 CLA	203	LBL "LF"	CAT 1
159 FC? 10	204	XEQ "E?"	LBL*MK
160 ISG 09	205	17	LBL*1K
161 SF 20	206	-	LBL*+K
162 FS? 07	207	E3	LBL*LF
163 RTN	208	/	LBL*E?
164 FS? 20	209	177	END
165 GTO 03	210	+	535 BYTES
166 "DONE, N O MORE"	211	XEQ "OM"	135 =
167 SF 09	212	"*0*♦♦**"	F32A2AF0
168 GTO 14	213	,	146 =
	214	ENTER↑	F47F000000
169♦LBL 07	215	DSE T	212 =
170 "NO ROOM "	216	GTO 14	F72A102A0000
			2A2A
171 CF 20	217♦	LBL 00	
172 CLST	218	X<> IND	
			T
173♦LBL 14	219	X=Y?	
174 FS?C 25	220	GTO 14	
175 RTN	221	X<> [
176 XEQ "VA"	222	"f0"	
177 TONE 7	223	STO \	
178 TONE 3	224	ARCL X	
179 STOP	225	RDN	
180 GTO 01	226	RCL \	
	227	X<> IND	
			T
181♦LBL 08	228	ISG T	
182 "NO SUCH KEY"	229	GTO 00	
183 GTO 14	230♦	LBL 14	
184♦LBL 09	231	X<> [
185 X<> d	232	ARCL X	
186 "KEY TAK EN"	233	X<> \	
	234	SF 10	
	235	X=Y?	
187♦LBL 14	236	DSE T	
188 CF 09	237	CF 10	
189 CF 20	238	X<> Z	
190 FS?C 25	239	X<> c	
191 RTN	240	R↑	
192 XEQ "VA"	241	RTN	
193 TONE 3	242♦	LBL "E?"	
194 PSE	243	RCL c	
195 "KEYCODE ?"	244	XEQ "2D"	
	245	16	
196 CLST	246	MOD	
197 RCL 08	247	LASTX	
198 XEQ "VA"	248	X↑2	
199 TONE 7	249	*	
200 STOP	250	RCL [
201 STO 08	251	+	
202 GTO 01	252	CLA	
	253	END	

Programm: UNIVERSAL - BASISUMWANDLER

wandelt Zahlen aus einem beliebigen System in Zahlen eines anderer beliebigen Systems um.

Besondere Eigenschaften:

- beliebig viele Stellen und Nachkommastellen
- Genauigkeit der Nachkommastellen bis 10^{-8}
- beliebig große Basis (von der Rechnergenauigkeit begrenzt)
- hoher Ein- und Ausgabekomfort
- typ. Rechenzeit: 15 s, max: 40 s.

Anleitung:

Programm starten mit XEQ "BU"

Abfrage B1 / B2 beantworten mit Eingabe der Eingabebasis, ENTER, Ausgabebasis, R/S

"NR.?" Eingabe der umzuwandelnden Zahl.

Es kann unter 3 Eingabemodi gewählt werden:

1. Eingabe ins X-Register, R/S. Anwendbar für alle Zahlen ohne Ziffern größer als 9
2. Eingabe ins Alpha-Register. Auf ALPHA schalten, Zahl eintippen (bei Ziffern SHIFT nicht vergessen, die Buchstaben A-I können für 10-18 verwendet werden), R/S. Anwendbar für alle Zahlen ohne Ziffern größer als 18.
3. Eingabe einzelner Digits in Dezimalform ins X-Register. Nach Eingabe jeder Stelle A drücken. Bei Nachkommastellen: vor der ersten Nachkommastelle SF OO ausführen. Nach Beendigung der Eingabe: R/S. Anwendbar für alle Zahlen.

Bei Modus 1 können max. 10 Ziffern, bei Modus 2 max. 14 Zeichen eingegeben werden.

Zwei Ausgabemodi stehen zur Verfügung:

1. Normal- und Hex-codierte Form im Alpha-Register für Ausgabesysteme mit Basis 2 - 19 (Ausnahme: bei Ausgabebasis = 10 auch im X-Register). Maximal können 15 Stellen ausgegeben werden.
2. Einzelne Digits werden in Dezimalform durch Schrägstrich / getrennt über Alpha-Register ausgegeben, maximal 24 Zeichen.

Ausgabemodus 2 stellt sich automatisch ein, wenn

- Eingabemodus 3 gewählt wurde,
- Ausgabebasis größer als 19.

Sollte nach Eingabe 3 Ausgabe 1 gewünscht werden, muß am Ende der Eingabe CF O3 ausgeführt werden.

Für den Fall, daß bei Ausgabe 2 die 24 Zeichen des Alpha-Registers nicht ausreichen, kann man Programmschritt 140 durch STOP ersetzen (oder ACX mit Drucker) und die Stellen einzeln auslesen.

Viel Erfolg,

Michael Fiedler
Am Fußgraben 12
6257 Hünfelden

Nach erfolgter Ausgabe kann mit R/S zur nächsten Eingabe gegangen werden, wenn Ein- und Ausgabebasis gleich bleiben.

Anmerkung: einige synthetische Befehle werden verwendet.

01*LBL "BU"	58 GTO 09	115 RCL 02	172*LBL 06
02 CF 01	59*LBL 02	116 LN	173 RCL 00
03 CF 02	60 1,01	117 /	174 E
04 CF 29	61 STO J	118 FIX 8	175 X<Y?
05 "B1 ↑ B2 ?"	62 "↑"	119 RND	176 GTO 01
06 PROMPT	63*LBL 01	120 INT	177*LBL 03
07 STO 02	64 21	121 E	178 PROMPT
08 E1	65 RCL J	122 +	179 GTO 04
09 X=Y?	66 LOG	123 Y↑X	180*LBL 01
10 SF 02	67 RND	124 STO 00	181 "↑0"
11 RCL Z	68 X>Y?	125 FIX 0	182 FS? 03
12 STO 01	69 GTO 01	126*LBL 05	183 "↑/"
13 X=Y?	70 SF 00	127 RCL 00	184 RCL 02
14 SF 01	71 GTO 03	128 E	185 ST/ 00
15*LBL 04	72*LBL 01	129 X=Y?	186 GTO 06
16 CF 00	73 30	130 "↑."	187 END
17 ,	74 -	131 RCL 02	
18 STO 03	75 9	132 ST/ 00	PLNG "BU"
19 - E	76 X<>Y	133 RCL 03	313 BYTES
20 STO 00	77 X>Y?	134 RCL 00	
21 CF 03	78 DSE X	135 /	
22 CF 23	79*LBL 09	136 INT	R00= "used "
23 "NR.?"	80 FS? 00	137 FS? 03	R01= "B in "
24 PROMPT	81 GTO 01	138 ARCL X	R02= "B out "
25 AOFF	82 RCL 01	139 FS? 03	R03= "used "
26 FS? 23	83 ST* 03	140 "↑/"	
27 GTO 00	84 RDN	141 FS? 03	
28 FS? 01	85 ST+ 03	142 GTO 08	
29 STO 03	86 GTO 03	143 9	
30 FS? 01	87*LBL 01	144 X<>Y	
31 GTO 00	88 RCL 01	145 X>Y?	
32 ,	89 RCL 00	146 ISG X	
33 STO 00	90 Y↑X	147 AOFF	
34*LBL 07	91 *	148 130	
35 RDN	92 ST+ 03	149 +	
36 STO Y	93 E	150 X<> [
37 FIX IND 00	94 ST- 00	151 RCL \	
38 ISG 00	95*LBL 03	152 "↑0"	
39 AOFF	96 FS? 03	153 1,1	
40 RND	97 STOP	154 STO \	
41 X*Y?	98 RCL \	155 "↑0"	
42 GTO 07	99 SF 25	156 RDN	
43 CLA	100 X*0?	157 RCL \	
44 ARCL X	101 GTO 02	158 CLA	
45*LBL 00	102*LBL 08	159 RDN	
46 1,01	103 CF 25	160 STO \	
47 STO J	104 CLA	161 RDN	
48 "↑"	105 19	162 STO [
49 2	106 RCL 02	163 ARCL Z	
50 RCL J	107 X>Y?	164*LBL 08	
51 X<Y?	108 SF 03	165 RCL 03	
52 GTO 00	109 RCL 03	166 RCL 00	
53 - E	110 FS? 02	167 MOD	
54 STO 00	111 ARCL X	168 STO 03	
55 GTO 01	112 FS? 02	169 E-8	
56*LBL A	113 GTO 03	170 X<=Y?	
57 SF 03	114 LN	171 GTO 05	

Gleichungen 4. Grades

Das Programm löst algebraisch die Gleichung

$$ax^4+bx^3+cx^2+dx+e=0 \quad \text{mit} \quad a,b,c,d,e \in \mathbb{R} \\ x \in \mathbb{R} \text{ und } a \neq 0$$

Das Programm berechnet alle Lösungen (vier) incl. komplexe der Gleichung. Es sind weiterhin Gleichungen 2. und 3. Grades mit dem Programm zu berechnen. Die entsprechenden Gleichungen müssen dann mit x^2 bzw. x multipliziert werden. Die Lösungen (0;0) bzw. (0) sind nicht zu berücksichtigen. Es werden neun Speicher benötigt (SIZE 009).

<u>Schritt</u>	<u>Instruktionen</u>	<u>Eingabe</u>	<u>Anzeige</u>
1	Start	XEQ"ROOT"	a=?
2	Eingabe der Koeffizienten	a R/S	b=?
3	Eingabe der Koeffizienten	b R/S	c=?
4	Eingabe der Koeffizienten	c R/S	d=?
5	Eingabe der Koeffizienten	d R/S	e=?
6	Eingabe der Koeffizienten	e R/S	
7	Anzeige der 1. Lösung		X1=...
8	Anzeige der 2. Lösung	R/S	X2=...
9	Anzeige der 3. Lösung	R/S	X3=...
10	Anzeige der 4. Lösung	R/S	X4=...
11	Weiter bei 7	R/S	X1=...
12	FIX 9	CLX	Realteil
		X<>Y	Imaginärteil

Beispiele

$$x^4+x^4-\frac{25}{4}x^2-\frac{1}{4}x+1,5=0 \quad \text{mit} \quad \begin{array}{l} a=1 \\ b=1 \\ c=-6,25 \\ d=-0,25 \\ e=1,5 \end{array} \quad \text{Lsg.:} \quad \begin{array}{l} X1=-3 \\ X2=-0,5 \\ X3=0,5 \\ X4=2 \end{array}$$

$$x^3-8x^2+19x-12=0 \quad \text{mit} \quad \begin{array}{l} a=1 \\ b=8 \\ c=19 \\ d=-12 \\ e=0 \end{array} \quad \text{Lsg.:} \quad \begin{array}{l} X1=0 \text{ nicht zu ber-} \\ X2=1 \quad \text{ücks.} \\ X3=3 \\ X4=4 \end{array}$$

01	LBL"ROOT"	51	3	101	LBL 00
02	"a=?"	52	Y^X	102	SQRT
03	PROMPT	53	-	103	ST+ 05
04	STO 03 < 1/X	54	STO 05	104	CHS
05	STO 02	55	X^2	105	2
06	STO 01	56	RCL 07	106	x
07	STO 00	57	3	107	RCL 05
08	"b=?"	58	/	108	+
09	PROMPT	59	RCL 06	109	ENTER^
10	STx 00	60	X^2	110	SIGN
11	"c=?"	61	-	111	X<>Y
12	PROMPT	62	3	112	ABS
13	STx 01	63	Y/X	113	3
14	"d=?"	64	+	114	1/X
15	PROMPT	65	X>0?	115	Y^X
16	STx 02	66	GTO 00	116	x
17	"e=?"	67	CHS	117	RCL 05
18	PROMPT	68	SQRT	118	SIGN
19	STx 03	69	RCL 05	119	RCL 05
20	RCL 02	70	R-P	120	ABS
21	STO 07	71	3	121	3
22	X^2	72	1/X	122	1/X
23	RCL 00	73	Y^X	123	Y^X
24	STx 07	74	X<>Y	124	x
25	X^2	75	3	125	+
26	RCL 03	76	/	126	RCL 02
27	x	77	X<>Y	127	RCL 06
28	+	78	P-R	128	-
29	RCL 01	79	X<>Y	129	STO 07
30	CHS	80	3	130	2
31	STO 06	81	SQRT	131	/
32	RCL 03	82	x	132	STO 08
33	4	83	2	133	ENTER^
34	x	84	/	134	X^2
35	ST- 07	85	STO 07	135	RCL 03
36	x	86	CHS	136	-
37	+	87	STO 08	137	ENTER^
38	CHS	88	X<>Y	138	ABS
39	STO 08	89	ST- 08	139	+
40	RCL 06	90	ST- 07	140	2
41	3	91	2	141	/
42	/	92	x	142	SQRT
43	STO 06	93	RCL 08	143	ST- 08
44	RCL 07	94	X>Y?	144	+
45	x	95	X<>Y	145	STO 06
46	RCL 08	96	RDN	146	RCL 00
47	-	97	RCL 07	147	2
48	2	98	X<=Y?	148	/
49	/	99	X<>Y	149	STO 05
50	RCL 06	100	GTO 02	150	RCL 05

151	X^2	201	GTO 03
152	RCL 07	202	LBL 04
153	+	203	STO 04
154	RCL 01	204	RCL IND 04
155	-	205	2
156	SQRT	206	/
157	ST- 05	207	X^2
158	+	208	ISG 04
159	STO 07	209	X<>X
160	RCL 06	210	RCL IND 04
161	x	211	DSE 04
162	RCL 05	212	X<>X
163	RCL 08	213	-
164	x	214	0
165	+	215	X<>Y
166	RCL 02	216	X>0?
167	-	217	GTO 05
168	ABS	218	ABS
169	RCL 07	219	SQRT
170	RCL 08	220	X<>Y
171	x	221	LBL 05
172	RCL 05	222	SQRT
173	RCL 06	223	FS? 05
174	x	224	GTO 06
175	+	225	CHS
176	RCL 02	226	X<>Y
177	-	227	CHS
178	ABS	228	X<>Y
179	X>Y?	229	LBL 06
180	GTO 03	230	RCL IND 04
181	RCL 08	231	2
182	X<> 06	232	/
183	STO 08	233	-
184	LBL 03	234	FIX 2
185	CF 05	235	"I="
186	"X1"	236	ARCL X
187	7	237	X<>Y
188	XEQ 04	238	X=0?
189	SF 05	239	GTO 01
190	"X2"	240	X>0?
191	7	241	"I+"
192	XEQ 04	242	ARCL X
193	CF 05	243	"I"
194	"X3"	244	LBL 01
195	5	245	X<>Y
196	XEQ 04	246	FIX 9
197	SF 05	247	PROMPT
198	"X4"	248	RTN
199	5		
200	XEQ 04		

Ein Primzahlprogramm - mal wieder!

Zuerst waren es die Size-Routinen, vor denen sich weder Key-Notes noch prisma retten konnte. Jetzt sind es die Primzahlprogramme. In der Methode sind sie überwiegend gleich: Eine große Schleife, in der Teiler berechnet werden, die ihrerseits nicht durch 2, 3 oder 5 teilbar sind. Lediglich das Programm von F. Hillebrandt verwendet eine andere Methode, und es ist das schnellste und auch längste der veröffentlichten Programme.

Der Grund für das große Interesse an solchen Routinen dürfte der sein: ebenso wie bei Size läßt sich zwar immer eine Information aus dem Programmablauf herausziehen, aber der Zeitaufwand läßt sich nicht vom Anwender bestimmen, weil er nur von der eingegebenen Zahl abhängt.

Das folgende Programm ist nach Vorschlägen von F. Hillebrandt und eigenen Ideen aus dem Primzahlenprogramm von U. Davertzhofen und mir hervorgegangen. Im Beispiel 1 000 003 benötigt dieses Programm mit ca. 40 sec. (vorm. 1 min. 10 sec.) etwa 42 % weniger Zeit und geht auch mit den Bytes sparsamer um. Wird eine 0 eingegeben, folgt keine Endlosschleife, sondern ein DATA ERROR, ebenso bei negativen oder gebrochenen Eingaben. SIZE 006, 200 Bytes. Noch ein Hinweis: Die Schleife reizt zur Verwendung eines Unterprogramms, was aber für den Geschwindigkeitsvorteil absolut tödlich ist und außerdem kaum mehr als 2 oder 3 Register beschert.

Die Eingabe einer Zahl erfolgt über XEQ"PRF" oder XEQ"A", die Ausgabe der Teiler mit R/S, bis die Meldung ENDE erscheint.

Das Programm T gibt einen ziemlich genauen Hinweis auf die Laufzeit: Für Zahlen um 10^{10} weniger als 1 Std. 5 min.

```
PRP "PRF"
01*LBL "PRF"
02*LBL A
CLRg LN LASTX STO 00
FRC FACT 2 STO 01 4
STO 02 + STO 03
RCL 00 ENTER↑ ENTER↑
CF 00 RCL 01 MOD X=0?
XEQ IND 01 CLX 3 MOD
X=0? XEQ IND 01 CLX 5
MOD X=0? XEQ IND 01
SIGN SIGN

35*LBL 00
X<> L RCL 03 + MOD
X=0? XEQ IND 01 X<> L
RCL 02 + MOD X=0?
XEQ IND 01 X<> L
RCL 01 + MOD X=0?
XEQ IND 01 X<> L
RCL 02 + MOD X=0?
XEQ IND 01 X<> L
RCL 01 + MOD X=0?
XEQ IND 01 X<> L
RCL 02 + MOD X=0?
XEQ IND 01 X<> L
RCL 03 + MOD X=0?

XEQ IND 01 X<> L
RCL 01 + MOD X=0?
XEQ IND 01 X<> L
RCL 02 + MOD X=0?
XEQ IND 01 X<> L
RCL 01 + MOD X=0?
XEQ IND 01 X<> L
RCL 02 + MOD X=0?
XEQ IND 01 X<> L
RCL 03 + MOD X=0?
```

```
XEQ IND 01 X<> L
RCL 01 + MOD X=0?
XEQ IND 01 X<> L X+2
X<=Y? GTO 00 1 RCL 00
FC?C 00 GTO 01 X=Y?
XEQ IND 02 "ENDE"
PROMPT

96*LBL 01
"PRIM" PROMPT

99*LBL 02
FS? 00 GTO 03 RDH
LASTX X=Y? RTN SF 00

107*LBL 03
1 ST+ 05 RCL 00 LASTX
/ STO 00 LASTX MOD
X=0? GTO 03 LASTX

119*LBL 04
STO 04 FIX 0 CF 29
" " ARCL 04 CLX
X<> 05 1 X<Y? "+ ↑"
X<Y? ARCL Y FIX 2
SF 29 RCL 04 PROMPT
RCL 04 SIGN RCL 00
ENTER↑ ENTER↑ ENTER↑
END
```

```
PRP "T"
01*LBL "T"
02*LBL A
FIX 4 SORT 30 / INT
1 + 3086 / HMS END
```

Ralf Pfeifer
Rubensstr. 5
5000 Köln - 50
Tel. 35 20 34

ACHTUNG: Die Zahlen, die "T" berechnet, werden in der gleichen Form wie HMS ausgegeben!

Liebe Clubmitglieder!

Wie Ihr vielleicht im letzten Info gelesen habt, werde ich mich ab jetzt in unserem Club als "Projektleiter Mathematik" betätigen. Um mir diese Arbeit etwas zu erleichtern, möchte ich Euch bitten, nach Möglichkeit folgende Punkte zu beachten:

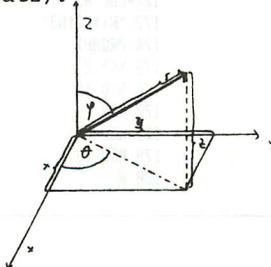
1. Bitte legt allen Programmen ein gedrucktes oder getipptes Listing bei.
2. Wer seine Programme, falls sie nicht veröffentlicht werden, wiederhaben will, der lege bitte Rückporto bei.
3. Wer mir einen ganz großen Gefallen tun will, füge beschriebene Magnetkarten bei (in diesem Fall das Rückporto nicht vergessen!).

Ansonsten freue ich mich auf Eure Post. Falls jemand Interesse an einem Mathe-Lösungsbuch hat, sollte er mir mal schreiben, welche Programme er gern darin vorfinden und was dafür anzulegen (grobe Preisvorstellung) er bereit wäre.

Nach diesem Gelabere möchte ich nun endlich in medias res gehen und Euch die nachfolgenden Programme etwas näher bringen:

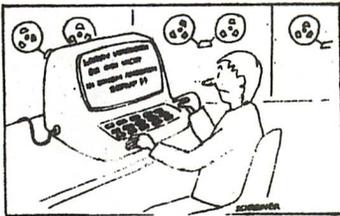
1. VEKTOR werden wohl hauptsächlich die streßgeplagten Schüler benutzen, um sich folgende Berechnungen durchführen zu lassen:
 1. Einheitsvektor V_0
 2. Betrag eines Vektors $|V|$
 3. a) Kreuzprodukt $V_1 \times V_2$
b) Spatprodukt SPAT
c) Skalarprodukt $V_1 \cdot V_2$
 4. Vektoraddition $V_1 + V_2$
 5. Vektorsubtraktion $V_1 - V_2$
 6. Volumen eines Vierflachs 4FLACH
 7. Winkel zwischen zwei Vektoren V_1 V_2
 8. Koordinatentransformation
 - a) Polar-Kartesisch P-C
 - b) Kartesisch-Polar C-P
2. Noch 'n PFZ-Programm für Leute, die es eilig haben, denen aber F. Hillebrandts Programm zu lang war. Sehr zu begrüßen ist das beiliegende Programm "T", das eine recht vernünftige Schätzung der Höchstrechenzeit liefert (die Zahl in Zeile 10 kann je. nach Modell etwas variieren).
3. Gleichungen 4. Grades kann man zwar auch mit dem Mathe-Modul Programm "POLY" berechnen, aber dabei wird eine Iteration verwendet, deren Rechenzeiten einen mitunter nicht gerade vom Hocker hauen, wohingegen das folgende Programm die Lösung direkt berechnet.

Nun noch eine Skizze zum VEKTOR-Programm (für weitere Formeln o.ä. fehlt mir der Platz):



01*LBL "VEKTOR"	61 R-P	121 RCL 21	181 GTO 03	241 LASTX
02*LBL "INPUT"	62 RTN	122 *	182*LBL "4FLACH"	242 COS
03*LBL 07	63*LBL "V1*V2"	123 -	183 XEQ 05	243 ST* 02
04 "N1X1Y1Z"	64 XEQ 06	124 CHS	184 6	244 RCL 02
05 PROMPT	65*LBL 03	125 RCL 00	185 /	245 RCL 01
06 X<> T	66 XEQ 09	126 X<>Y	186 RTN	246 RCL 00
07 X<>Y	67 STO 22	127 STO 00	187*LBL "z V1 V2"	247 RTN
08 X<> Z	68 RDN	128 RDN	188 XEQ 06	248*LBL 01
09 X<>Y	69 XEQ 09	129 RCL 19	189 STO 18	249 1
10*LBL a	70 RCL IND 23	130 *	190 X<>Y	250 ST+ 23
11 SF 05	71 XEQ 01	131 RCL 02	191 STO 19	251 RDN
12 STO 23	72 XEQ 01	132 RCL 20	192 XEQ 03	252 FS? 05
13 RDN	73 RCL 22	133 *	193 STO 00	253 RTH
14 X<> 23	74 STO 23	134 -	194 RCL 18	254 RCL IND
15 3	75 RDN	135 CHS	195 XEQ 00	255 RTN
16 *	76 RCL IND 23	136 STO 02	196 ST/ 00	256*LBL 02
17 X<> 23	77 R↑	137 RCL 01	197 RCL 19	257 1
18 STO IND 23	78 *	138 RCL 00	198 XEQ 00	258 ST- 23
19 XEQ 01	79 XEQ 01	139 RTN	199 ST/ 00	259 RDN
20 RDN	80 R↑	140*LBL "V1-V2"	200 RCL 00	260 RTN
21 STO IND 23	81 *	141 SF 00	201 ACOS	261*LBL 06
22 XEQ 01	82 +	142*LBL "V1+V2"	202 RTN	262 "N1+N2"
23 RDN	83 XEQ 01	143 XEQ 06	203*LBL "C-P"	263 PROMPT
24 STO IND 23	84 R↑	144 STO 22	204 "N"	264 X<>Y
25 CF 05	85 *	145 RDN	205 PROMPT	265 RTN
26 STOP	86 +	146 XEQ 09	206 STO 18	266*LBL 09
27 GTO 07	87 RTN	147 RCL IND 23	207 XEQ 00	267 3
28*LBL "V0"	88*LBL "V1XV2"	148 FS? 00	208 STO 00	268 *
29 "N"	89 XEQ 06	149 CHS	209 RCL 18	269 STO 23
30 PROMPT	90*LBL 04	150 STO 00	210 XEQ 09	270 END
31 STO 22	91 STO 22	151 XEQ 01	211 RCL IND 23	
32 XEQ 00	92 RDN	152 FS? 00	212 XEQ 01	
33 1/X	93 XEQ 09	153 CHS	213 X<>Y	
34 STO 21	94 RCL IND 23	154 STO 01	214 /	
35 RCL 22	95 STO 19	155 XEQ 01	215 ATAN	
36 XEQ 09	96 XEQ 01	156 FS?C 00	216 STO 01	
37 2	97 STO 20	157 CHS	217 XEQ 01	
38 +	98 XEQ 01	158 STO 02	218 RCL 00	
39 STO 23	99 STO 21	159 RCL 22	219 /	
40 RCL IND 23	100 RCL 22	160 XEQ 09	220 ACOS	
41 RCL 21	101 XEQ 09	161 RCL IND 23	221 RCL 01	
42 *	102 RCL IND 23	162 ST+ 00	222 RCL 00	
43 XEQ 02	103 STO 02	163 XEQ 01	223 RTN	
44 RCL IND 23	104 XEQ 01	164 ST+ 01	224*LBL "P-C"	
45 RCL 21	105 STO 00	165 XEQ 01	225 "PHI+01R"	
46 *	106 XEQ 01	166 ST+ 02	226 PROMPT	
47 XEQ 02	107 STO 01	167 RCL 02	227 STO 00	
48 RCL IND 23	108 RCL 19	168 RCL 01	228 STO 01	
49 RCL 21	109 *	169 RCL 00	229 STO 02	
50 *	110 RCL 02	170 RTN	230 RDN	
51 RTN	111 RCL 21	171*LBL "SPAT"	231 COS	
52*LBL "V/V"	112 *	172*LBL 05	232 ST* 00	
53 "N"	113 -	173 "N1+N2+N3"	233 LASTX	
54 PROMPT	114 RCL 01	174 PROMPT	234 SIN	
55*LBL 00	115 X<>Y	175 X<> 2	235 ST* 01	
56 XEQ 09	116 STO 01	176 STO 18	236 RDN	
57 RCL IND 23	117 RDN	177 RDN	237 RDN	
58 XEQ 01	118 RCL 20	178 XEQ 04	238 SIN	
59 R-P	119 *	179 RCL 18	239 ST* 00	
60 XEQ 01	120 RCL 00	180 0	240 ST+ 01	

LBL*VEKTOR		CAT*6
LBL*INPUT		*INPUT 11
LBL*V0		*V0 12
LBL*/V/		*V/ 13
LBL*V1+V2		*SPAT 14
LBL*V1XV2		*4FLACH 15
LBL*V1-V2		*V1 V2 23
LBL*V1+V2		*C-P 24
LBL*SPAT		*P-C 25
LBL*4FLACH		*V1-V2 51
LBL*V1 V2		*V1+V2 61
LBL*C-P		*V1*V2 71
LBL*P-C		*V1XV2 81
END	555 BYTES	



Folgende Beispielaufgaben sind zu lösen:

1. Die Richtung einer Ebene ist gegeben durch die Vektoren

$$a = \begin{pmatrix} 5 \\ 3 \\ -1 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} -7 \\ 1 \\ 4 \end{pmatrix}$$

Berechnen Sie den Lotvektor der Ebene und überprüfen Sie das Ergebnis, indem Sie

- a) das Skalarprodukt von jedem der Ausgangsvektoren mit dem Lotvektor bilden ($=0$),
b) die Winkel zwischen dem Lotvektor und den beiden Ausgangsvektoren bestimmen.

2. Wie lang ist der Lotvektor aus 1.?

3. Wie lautet der Einheitsvektor des Lotvektors aus 1.?

4. Bestimmen Sie die Winkel φ und ϑ des Vektors

$$a = \begin{pmatrix} 3 \\ 1 \\ -2 \end{pmatrix} \quad \text{Überprüfen Sie das Ergebnis mit der Funktion } \alpha \quad V_1 \quad V_2$$

5. Ein Vierflach ist gegeben durch die Punkte
 $P_1 (1/2/0)$, $P_2 (2/3/1)$, $P_3 (1/5/3)$, $P_4 (1/2/1)$
Berechnen Sie das Volumen.

6. Eine Ebene ist gegeben durch

$$\gamma_1 = \begin{pmatrix} 6 \\ 3 \\ 7 \end{pmatrix} + \mu \begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} + \nu \begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix}$$

eine Gerade ist gegeben durch

$$g : \gamma_2 = \begin{pmatrix} 3 \\ 7 \\ 9 \end{pmatrix} + \lambda \begin{pmatrix} -6 \\ 1 \\ -5 \end{pmatrix}$$

Bestimmen Sie den Schnittpunkt.

Das Programm:

Das Programm ist so aufgebaut, daß jede Funktion einer Taste zugeordnet ist (vgl. USER KEYS) und durch Drücken dieser Taste aufgerufen wird. Der oder die Vektoren müssen vorher über die Funktion INPUT eingegeben werden, es können bis zu fünf Vektoren gleichzeitig im Rechner gespeichert werden. Die Eingabe erfolgt in der Form $N^*X^*Y^*Z$, wobei N die Nummer des Vektors ist und X,Y,Z die entsprechenden Koordinaten sind.

Um Vektoren, die als Zwischenergebnisse abgespeichert werden sollen, einzugeben, gibt man die laufende Nummer des Vektors ein und drückt f A.

Führt man eine arithmetische Funktion (nicht INPUT) aus, fragt der Rechner nach der oder den Nummern der Vektoren und führt sie aus. Anschließend enthalten die Stackregister die entsprechenden Koordinaten, X in X, in Y, und Z in Z.

Lösungen der Beispielaufgaben:

①
 N1X1Y1Z XEQ "INPUT"
 1,00 ENTER↑
 5,00 ENTER↑
 3,00 ENTER↑
 -1,00 RUN
 XEQ "INPUT"
 N1X1Y1Z
 2,00 ENTER↑
 -7,00 ENTER↑
 1,00 ENTER↑
 4,00 RUN
 XEQ "V1XV2"
 N1↑N2
 1,00 ENTER↑
 2,00 RUN
 PRSTK

T = 3,00
 Z = 26,00
 Y = -13,00
 X = 13,00
 → $\begin{pmatrix} 13 \\ -13 \\ 26 \end{pmatrix}$

α
 3,00 XEQ a
 XEQ "V1*V2"
 N1↑N2
 1,00 ENTER↑
 3,00 RUN
 0,00 ***
 XEQ "V1*V2"

N1↑N2
 2,00 ENTER↑
 3,00 RUN
 0,00 ***
 XEQ "∠ V1 V2"
 β
 N1↑N2
 1,00 ENTER↑
 3,00 RUN
 90,00 ***
 XEQ "∠ V1 V2"

N1↑N2
 2,00 ENTER↑
 3,00 RUN
 90,00 ***
 XEQ "∠ V1 V2"

②
 N XEQ "V/"
 3,00 RUN
 31,04 ***
 XEQ "V0"

③
 N
 3,00 RUN
 PRSTK

T = 0,02
 Z = 0,02
 Y = -0,41
 X = 0,41
 → $\begin{pmatrix} 0,41 \\ -0,41 \\ 0,02 \end{pmatrix}$

④
 N1X1Y1Z XEQ "INPUT"
 1,00 ENTER↑
 3,00 ENTER↑
 1,00 ENTER↑
 -2,00 RUN
 XEQ "C-P"
 N
 1,00 RUN
 PRSTK
 T = 18,43
 Z = 122,31
 Y = 18,43
 X = 3,74

N1X1Y1Z XEQ "INPUT"
 2,00 ENTER↑
 0,00 ENTER↑
 0,00 ENTER↑
 1,00 RUN
 XEQ "∠ V1 V2"
 N1↑N2
 1,00 ENTER↑
 2,00 RUN
 122,31 ***

N1X1Y1Z XEQ "INPUT"
 3,00 ENTER↑
 3,00 ENTER↑
 1,00 ENTER↑
 0,00 RUN
 XEQ "INPUT"
 Projektion von y auf xy-Ebene

N1X1Y1Z
 4,00 ENTER↑
 1,00 ENTER↑
 0,00 ENTER↑
 0,00 RUN
 XEQ "∠ V1 V2"
 X-Achse

N1↑N2
 3,00 ENTER↑
 4,00 RUN
 18,43 ***

⑤
 N1X1Y1Z XEQ "INPUT"
 1,00 ENTER↑
 1,00 ENTER↑
 2,00 ENTER↑
 0,00 RUN
 XEQ "INPUT"
 N1X1Y1Z
 2,00 ENTER↑
 2,00 ENTER↑
 3,00 ENTER↑
 1,00 RUN
 XEQ "INPUT"

N1X1Y1Z
 3,00 ENTER↑
 1,00 ENTER↑
 5,00 ENTER↑
 3,00 RUN
 XEQ "INPUT"
 N1X1Y1Z
 4,00 ENTER↑
 1,00 ENTER↑
 2,00 ENTER↑
 1,00 RUN
 XEQ "V1-V2"

Berechnung d. Seitenvektoren

N1↑N2
 2,00 ENTER↑
 1,00 RUN
 2,00 XEQ a
 N1↑N2
 XEQ "V1-V2"

N1↑N2
 4,00 ENTER↑
 1,00 ENTER↑
 4,00 XEQ a
 N1↑N2
 XEQ "V1-V2"

N1↑N2
 4,00 ENTER↑
 1,00 ENTER↑
 4,00 XEQ a
 N1↑N2↑N3
 XEQ "4FLACH"
 2,00 ENTER↑
 3,00 ENTER↑
 4,00 RUN
 -0,50 ***

Lösung der 6. Aufgabe:

Durch Gleichsetzen entsteht folgendes Gleichungssystem:

$$\begin{pmatrix} 3 \\ 7 \\ 9 \end{pmatrix} + \lambda_5 \begin{pmatrix} -6 \\ 1 \\ -5 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \\ 7 \end{pmatrix} + \mu_1 \begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} + \nu_1 \begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix}$$

Daraus folgt

$$\begin{pmatrix} 3 \\ 7 \\ 9 \end{pmatrix} - \begin{pmatrix} 6 \\ 3 \\ 7 \end{pmatrix} = \mu_1 \begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} + \nu_1 \begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix} + \lambda_5 \begin{pmatrix} -6 \\ 1 \\ -5 \end{pmatrix}$$

$$\rightarrow \lambda_5 = \frac{\left| \begin{pmatrix} 3 \\ 7 \\ 9 \end{pmatrix} - \begin{pmatrix} 6 \\ 3 \\ 7 \end{pmatrix} \right| \cdot \begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix}}{\left| \begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix} \cdot \begin{pmatrix} -6 \\ 1 \\ -5 \end{pmatrix} \right|}$$

$$= \frac{\left[\left(\begin{pmatrix} 3 \\ 7 \\ 9 \end{pmatrix} - \begin{pmatrix} 6 \\ 3 \\ 7 \end{pmatrix} \right) \cdot \left(\begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} \right) \cdot \begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix} \right]}{\left[\left(\begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix} \right) \cdot \left(\begin{pmatrix} -1 \\ 3 \\ -16 \end{pmatrix} \right) \cdot \begin{pmatrix} -6 \\ 1 \\ -5 \end{pmatrix} \right]}$$

Ⓒ

N1X1Y1Z	XEQ "INPUT"	N1X1Y1Z	4,00 ENTER†
	1,00 ENTER†		6,00 ENTER†
	3,00 ENTER†		-1,00 ENTER†
	7,00 ENTER†		5,00 RUN
	9,00 RUN		
N1X1Y1Z	XEQ "INPUT"	XEQ "SPAT"	
	2,00 ENTER†	N11N21N3	
	6,00 ENTER†	Zähler	1,00 ENTER†
	3,00 ENTER†	determinante	2,00 ENTER†
	7,00 RUN		3,00 RUN
			20,00 ***
			STO 34
N11N2	XEQ "V1-V2"		
	1,00 ENTER†		
	2,00 RUN		
	1,00 XEQ a		
N1X1Y1Z	XEQ "INPUT"		
	2,00 ENTER†		
	2,00 ENTER†		
	-2,00 ENTER†		
	-4,00 RUN		
N1X1Y1Z	XEQ "INPUT"		
	3,00 ENTER†		
	-1,00 ENTER†		
	3,00 ENTER†		
	-16,00 RUN		
	XEQ "INPUT"		

Durch Einsetzen von $\lambda_5 = 0,08$ in die Grundgleichung erhält man:

$$\lambda_5 = \left(\frac{2,51}{7,08} \right)$$

Der Schmittwert liegt bei $s(2,51 / 7,08 / 0,53)$

282,283-81

λ_T

Niels Nöhren
Kielort 16
2000 Norderstedt

Liebe Clubfreunde!

Das Programm Bruchrechnung von Clemens Mirgel (vergl. prisma 26/9-80) hat mich richtig begeistert. Ich habe es gleich in meine Programmsammlung aufgenommen. Allerdings stellte ich fest, daß sich das Programm an einigen Stellen verbessern läßt. Ich stelle Euch hier eine optimierte Version vor. Die wesentlichen Veränderungen befinden sich in den Programmteilen =EB=, "KGV" und "K".

Weiterhin möchte ich Euch mein Programm "Lottozahlen" vorstellen. Es liefert in einem Durchgang je 6 verschiedene Zahlen zwischen 1 und 49. Zuerst wird eine Zahl nach der anderen erzeugt, auf Null und Gleichheit überprüft und abgespeichert. Anschließend wird eine nach der anderen angezeigt.

Das Programm wird mit XEQ "L" gestartet. Mit der Anzeige "QUELLZAHL ?" fragt der Rechner nach einer Ausgangszahl, die zwischen 0 und 1 liegen sollte. Mit TONE 0 wird die erste Zahl angekündigt. Mit TONE 9 macht der Rechner auf die 6. Zahl aufmerksam. Für jede weitere Zahlenreihe muß die Taste A gedrückt werden.

Außerdem stelle ich mein Kalenderprogramm vor. Es beruht zwar auf dem Kalenderprogramm der Standardsammlung, ist aber von mir in einigen Punkten verändert worden. Es arbeitet korrekt im gesamten Bereich des Gregorianischen Kalenders und überprüft jedes eingegebene Datum auf seine Legalität.

Im Grunde erfüllt das Programm den gleichen Zweck wie das Standardprogramm. Es werden aber nur 4 Register zur Zwischenspeicherung benötigt, alles andere spielt sich ausschließlich im Stack ab.

Alle notwendigen Angaben müssen gleich eingegeben werden, bevor das Programm gestartet wird. Also beispielsweise Datum A, ENTER, Datum B, XEQ "R" zur Berechnung der Tagesanzahl zwischen 2 Daten. Datum A, ENTER, Anzahl der Tage, XEQ "N" zur Ermittlung eines neuen Datums und schließlich: Datum, XEQ "Q" für den entsprechenden Wochentag.

Wird das Datum 29.02.1900 eingegeben, ein Datum, das es nicht gab, weil das Jahr 1900 kein Schaltjahr war, reagiert der Rechner mit der Anzeige "DATUMSFEHLER". Zur Erinnerung steht im X-Register noch einmal das unmögliche Datum.

Ich habe das Programm so ausgelegt, daß die bei uns übliche Datumsform DD,MMYYYY verwendet werden muß.

Für diesmal möchte ich schließen. Es grüßt Euch und wünscht Happy Programming

Euer

Niels Nöhren

Programm Bruchrechnung (optimiert):

01	LBL BR	56	ST- Ø1	111	STx Z
02	FIX Ø	57	GTO Ø2	112	XEQ GGT
03	CF 29	58	LBL x	113	/
04	Ø	59	RCL Ø4	114	RTN
05	GANZZAHL 1?	60	STx Ø1	115	LBL K
06	PROMPT	61	RCL Ø2	116	RCL Ø1
07	STO ØØ	62	RCL Ø5	117	RCL Ø8
08	"ZAEHLER 1?"	63	x	118	XEQ GGT
09	PROMPT	64	STO Ø8	119	ST/ Ø1
10	STO Ø1	65	GTO Ø2	120	ST/ Ø8
11	"NENNER 1?"	66	LBL /	121	RTN
12	PROMPT	67	RCL Ø5	122	LBL AS
13	sTO Ø2	68	STx Ø1	123	RCL Ø2
14	LBL Ø1	69	RCL Ø2	124	RCL Ø5
15	RCL ØØ	70	RCL Ø4	125	XEQ KGV
16	RCL Ø2	71	x	126	STO Ø8
17	x	72	STO Ø8	127	RCL Ø2
18	ST+ Ø1	73	GTO Ø2	128	/
19	Ø	74	LBL EB	129	STx Ø1
20	"GANZZAHL 2?"	75	Ø	130	RCL Ø8
21	PROMPT	76	STO ØØ	131	RCL Ø5
22	STO Ø3	77	RCL Ø1	132	/
23	"ZAEHLER 2?"	78	RCL Ø8	133	STx Ø4
24	PROMPT	79	X>Y?	134	RCL Ø4
25	STO Ø4	80	RTN	135	END
26	"NENNER 2?"	81	/		
27	PROMPT	82	INT		
28	STO Ø5	83	STO ØØ		
29	RCL Ø3	84	RCL Ø1		
30	x	85	RCL Ø8		
31	ST+ Ø4	86	MOD		
32	"- + x / ?"	87	STO Ø1		
33	PROMPT	88	RTN		
34	LBL Ø2	89	LBL ANZ		
35	XEQ EB	90	" "		
36	XEQ K	91	ARCL ØØ		
37	RCL Ø1	92	"- "		
38	X=Ø?	93	ARCL Ø1		
39	GTO Ø3	94	"-/"		
40	XEQ ANZ	95	ARCL Ø8		
41	LBL Ø3	96	AVIEW		
42	" "	97	STOP		
43	ARCL ØØ	98	GTO Ø4		
44	AVIEW	99	LBL GGT		
45	STOP	100	LBL Ø5		
46	LBL Ø4	101	MOD		
47	RCL Ø8	102	LASTX		
48	STO Ø2	103	X<>Y		
49	GTO Ø1	104	X≠Ø?		
50	LBL +	105	GTO Ø5		
51	XEQ AS	106	RDN		
52	ST+ Ø1	107	RTN		
53	GTO Ø2	108	LBL KGV		
54	LBL -	109	STO Z		
55	XEQ AS	110	X<>Y		

Programm Lottozahlengenerator:

01	LBL L	56	X<>Y
02	CF 27	57	X=Y?
03	FIX 0	58	TONE 9
04	CF 29	59	RCL IND X
05	1,006	60	STOP
06	STO 07	61	RDN
07	CLX	62	ISG X
08	"QUELLZAHL ?"	63	GTO 06
09	PROMPT	64	LBL CLR
10	SF 27	65	CF 27
11	X=0?	66	SF 29
12	GTO A	67	FIX 2
13	STO 00	68	CLA
14	LBL A	69	CLST
15	RCL 07	70	-
16	ENTER	71	OFF
17	CLX	72	GTO CLR
18	LBL 02	73	END
19	STO IND Y		
20	ISG Y		
21	GTO 02		
22	LBL 01		
23	RCL 00		
24	9821		
25	x		
26	0,211327		
27	+		
28	FRC		
29	STO 00		
30	50		
31	x		
32	INT		
33	X=0?		
34	GTO 01		
35	LBL 03		
36	RCL 07		
37	X<>Y		
38	LBL 04		
39	RCL IND Y		
40	X=0?		
41	GTO 05		
42	X=Y?		
43	GTO 01		
44	RDN		
45	ISG Y		
46	GTO 04		
47	LBL 05		
48	RDN		
49	STO IND Y		
50	ISG Y		
51	GTO 01		
52	RCL 07		
53	TONE 0		
54	LBL 06		
55	6,006		

Kürzen von Brüchen

Dieses Programm kürzt einen Bruch Z/N auf die einfachste Form.

Zuerst tippt man das Programm laut Auflistung ein. Danach startet man es mit GTO^TBK. Jetzt wird der ungekürzte Zähler Z_u mit R/S eingegeben, darauf der ungekürzte Nenner N_u.

Drückt man nun auf die Run-Taste, wird der gekürzte Bruch folgendermaßen angezeigt: Z_g / N_g

Listing 10 Register

```

LBLTBK
FIX 0
STO 07
STOP
STO 08
STO 09
LBL 08
RCL 09
-
X=0?
GTO 27
X>0?
GTO 08
RCL 09
+
X< 09
GTO 08
RCL 09
+
X< 09
GTO 08
LBL 27
RCL 07
XEQ A
STO 07
RCL 08
XEQ A
STO 08
CLA
ARCL 07
 $\frac{\uparrow}{\uparrow} \div \frac{\uparrow}{\uparrow}$ 
APPEND
ARCL 08
PROMPT
GTOTBK
LBL A
RCL 09
+
RTN
END.
    
```

Speicherbelegung:
 07: Z
 08: N
 09: ggT(Z,N)

Bei der Erstellung des weiter unten aufgelisteten Programms ging es in erster Linie um möglichst kurze Rechenzeiten bei Zahlen mit großen Faktoren; dies erklärt die Länge des Programms von 693 Bytes. Dieses Programm soll weiterhin ein erster Schritt in Richtung Faktorisierung von Zahlen $> 10^{10}$ in akzeptablen Zeiten (Tage?) sein. Dafür ist dann sicher zusätzlicher Aufwand, wie z.B. programmgesteuertes Einlesen von Daten oder Anschlussprogrammen, notwendig.

Ich habe nicht nur die Anregung, sondern auch viele Realisierungswege dem Werk von D.E. Knuth (The Art of Computer Programming, Vol. II, Ch. 4.5.4) entnommen.

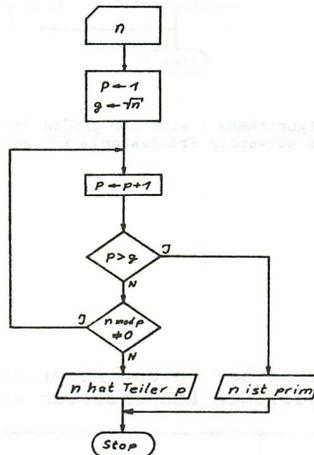
Das Programm kombiniert zwei unterschiedliche Algorithmen, die in ihrer Primitivform zunächst kurz vorgestellt werden sollen:

Algorithmus_E (Die bekannte konventionelle Teilertestmethode; ich vermute, daß sie auf Euklid zurückgeht (Die Elemente, neuntes Buch § 14); wäre aber für Hinweise auf andere Quellen dankbar)

Es wird nacheinander geprüft, ob 2, 3, 4, 5, 6, ... Teiler der gegebenen Zahl n sind. Der erste so erhaltene Teiler ist der kleinste Primteiler von n. Ist bis \sqrt{n} kein Teiler gefunden, so ist n eine Primzahl.

```

01+LBL "EUKLID"
02 STO 00
03 *IST PRIM"
04 VIEW 00
05 SGR
06 STO 02
07 1
08 STO 01
09+LBL 00
10 1
11 ST+ 01
12 RCL 02
13 RCL 01
14 X<Y?
15 GTO 01
16 RCL 00
17 RCL 01
18 MOD
19 X=0?
20 GTO 00
21 *HAT TEILER "
22 RCCL 01
23+LBL 01
24 BEEP
25 PROMPT
26 END
    
```



Algorithmus_F (Fermatsche Methode)

Sei n die zu faktorisierende Zahl. Ist $n = F_1 \cdot F_2$ (F_1, F_2), so kann man für F_1 und F_2 den Ansatz $F_1 = x - y$, $F_2 = x + y$ für geeignete (nicht notwendig ganze) Zahlen x, y machen. Dann ist $x = (F_1 + F_2) / 2$, $y = (F_2 - F_1) / 2$. Insbesondere sind x, y ganzzahlig, falls F_1 und F_2 beide ungerade sind.

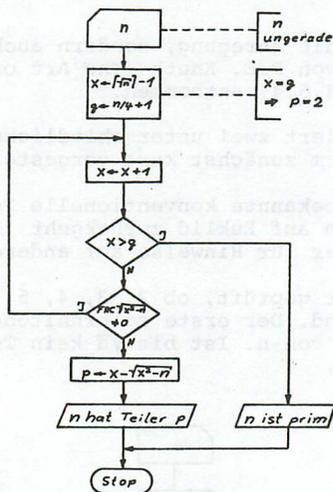
Mit dem Fermatschen Algorithmus werden für ungerades n diese Werte x und y gesucht: Ausgehend von $x = \lceil \sqrt{n} \rceil$ (kleinste ganze Zahl $\geq \sqrt{n}$) wird x erhöht und jeweils getestet, ob $x^2 - n$ eine Quadratzahl ist.

```

01+LBL *FEK
02 STO 00
03 *IST PRIM*
04 VIEW 00
05 2
06 MOD
07 1/X
08 PCL 00
09 SQRT
10 RCL X
11 CHS
12 1
13 MOD
14 +
15 1
16 -
17 STO 01
18 RCL 00
19 4
20 /
21 1
22 +
23 STO 02
24+LBL 00
25 1
26 ST+ 01
27 RCL 02
28 RCL 01
29 X*Y?
30 GTO 01
31 RCL 01
32 X*2
33 RCL 00
34 -
35 SQRT
36 FFC
37 X#0?
38 GTO 00
39 RCL 01
40 LASTX
41 -
42 *HAT TEILER *
43 RCCL X
44+LBL 01
45 BEEP
46 PROMPT
47 .END.

```

(Beachte: $n = x^2 - y^2 = (x-y)(x+y)$)



Mit Algorithmus F wird der größte Faktor (nicht notwendig Primfaktor) $\leq \sqrt{n}$ gefunden.

Zur Charakterisierung der beiden Algorithmen seien nur zwei Beispiele mit den jeweiligen Rechenzeiten angegeben:

	E	F
1001423 = 887 · 1129	4 min 17 sec	4,8 sec
1001293 = 113 · 8861	33,5 sec	24 min 12 sec

Ist nur die Alternative zwischen Algorithmus E oder F gegeben, so entscheidet man sich ohne Zweifel für Algorithmus E (ca. \sqrt{n} Tests gegenüber $n/4 - \sqrt{n}$ ($\gg \sqrt{n}$ bei großem n) Tests, zumal damit auf einfache Weise der Restfaktor weiter faktorisiert werden kann. Beim Auswerten der gemessenen Rechenzeiten fällt aber doch etwas auf: Der Abstand vom Startwert (\sqrt{n} bei F und 2 bei E) bis zum ersten

gefundenen Faktor ist bei beiden Beispielen ungefähr gleich: $\sqrt{1001423} - 887 = 113,7 \approx 113$; d.h., es war zu erwarten, daß Algorithmus F für 1001423 ungefähr die gleiche Zeit (oder mehr, da der jeweilige Test aufwendiger ist) wie Algorithmus E für 1001293 benötigte. Aber offensichtlich ist hier der Fermatsche Algorithmus nahezu 7 mal so schnell. Das verwundert nicht, wenn man beachtet, daß in F mit der ersten Erhöhung von $x = \lfloor \sqrt{n} \rfloor = \lfloor \sqrt{1001423} \rfloor = 1001$ um 1 der "Testfaktor" $p = x - y$ einen Sprung von $(x - \sqrt{x^2 - n}) - (x + 1 - \sqrt{(x+1)^2 - n}) = 25,8$ macht, während der Testfaktor im Algorithmus E konstant um 1 wächst. Die Erhöhung des Testfaktors $p = x - y$ wird mit wachsendem x allerdings immer geringer, was sich in der unverhältnismäßig hohen Rechenzeit beim Beispiel 1001293 ausdrückt. Man kann aber mit Recht erwarten, daß mit Algorithmus F große Faktoren wesentlich schneller gefunden werden als mit E.

Es liegt nahe, die beiden Verfahren zu kombinieren und den Übergang so zu wählen, daß eine optimale Rechengeschwindigkeit erreicht wird. Selbstverständlich wird zunächst Algorithmus E bis zu einem größten Testfaktor p . angewandt, da damit schnell kleine Faktoren gefunden werden und somit in der Regel nur noch ein Restfaktor weiterfaktoriert werden muß. Außerdem brauchen bei anschließender Anwendung von Algorithmus F nur solche x mit $x - y$ zwischen \sqrt{n} und p . getestet zu werden. Die Berechnung eines optimalen Wertes p . wird weiter hinten durchgeführt. Da dieser Wert immer größer als \sqrt{n} ist (wobei n der Restfaktor ist), kann n nur noch aus höchstens zwei Faktoren bestehen; somit liefert Algorithmus F dann auch in jedem Falle Primfaktoren.

Nun zur Realisierung der Algorithmen E und F im Programm. Analysiert man Algorithmus E, so fällt auf, daß es überflüssig ist, weiterhin gerade Faktoren p zu testen, wenn Teilbarkeit durch 2 nicht (mehr) gegeben ist. Damit wird die Anzahl der Tests praktisch halbiert; dementsprechend auch die Rechenzeit. Die entsprechende Überlegung gilt für den Testfaktor 3. Ist die zu faktorisierende Zahl nicht (mehr) durch 3 teilbar, so reicht es, den Testfaktor von 5 ausgehend abwechselnd um 2 oder 4 zu erhöhen. Damit werden alle Vielfachen von 2 und 3 übersprungen.

Eine Verminderung der Teilertests auf diese Weise läßt sich mit den Primfaktoren 5, 7, 11, ... fortführen. Jedoch zeigt die letzte Zeile der folgenden Tabelle, daß der zusätzliche Nutzen (Einsparung an überflüssigen Tests) immer geringer wird, während der Aufwand unverhältnismäßig ansteigt.

	Primitiv- version a)	b)	c)	d)	e)	f)	g)	allgem.
Testfaktoren enthalten keine Primfaktoren bis einschließl.		2	3	5	7	11	13	p
Bereich, den eine Schleife überdeckt	1	2	6	30	210	2310	30030	$\prod = 2 \cdot 3 \cdot 5 \dots p$
Anzahl der Tests pro Schleife (Programmlänge)	1	1	2	8	48	480	5760	$\varphi(\pi)$
Bereich, der mit einem Test überdeckt wird	1	2	3	3,75	4,38	4,81	5,21	$\pi / \varphi(\pi)$
Gewinn gegenüber vorherigem Fall		100%	50%	25%	16,8%	9,8%	8,3%	

(Dabei ist φ die Eulersche φ -Funktion: $\varphi(n)$ = Anzahl der zu n teilerfremden Zahlen kleiner n)

Da für jeden Test mindestens ein Unterprogrammaufruf mit Eingangsparameterübergabe notwendig ist, wären bei f) mindestens $480 \times 3 = 144$ Bytes nur für die Schleife notwendig (dabei wurde der relativ langsame indirekte Unterprogrammaufruf zugrunde gelegt). Bei e) sind mindestens 144 Bytes in der Schleife notwendig, was durchaus realisierbar ist. Für das folgende Programm wurde die Alternative e) zu d) untersucht, wobei allerdings in d) die Unterprogramme abgerollt wurden, d.h. nur im Erfolgsfall (Teiler gefunden) wird hier ein Unterprogramm aufgerufen (in diesem Fall ist der indirekte Unterprogrammaufruf nicht nur jeweils um ein Byte kürzer, sondern auch etwas schneller (d.h. das Überspringen ist schneller)).

Es zeigte sich, daß dieses Vorgehen nach d) die günstigsten Rechenzeiten lieferte (Ein Schleifendurchlauf Zeile 052 bis 110 dauert 1,160 sec).

Realisierung des Fermatschen Algorithmus.

Während alle mir bekannten Realisierungen des Algorithmus E mindestens die Inkrementierung nach b) der obenstehenden Tabelle benutzen, ist es nicht ganz so offensichtlich, daß man im Algorithmus F die Anzahl der Tests ebenfalls (sogar effektiver) drastisch reduzieren kann.

Dieser Algorithmus wurde von Fermat benutzt, und es ist nicht anzunehmen, daß dieser sich für ein Verfahren entschied, bei dem er wesentlich häufiger die Wurzel aus einer Zahl ziehen mußte, als im anderen Falle eine Division durchzuführen gewesen wäre (zumal Fermat bekanntlich nicht einmal einen TI-Rechner zur Verfügung hatte, geschweige denn eine wirklich leistungsfähige Hilfe wie den HP-41C).

Fermats Vorgehensweise bestand darin, daß er die beiden Endziffern des Radikanden $x^2 - n$ betrachtete, um so viele Fälle auszuschließen; er arbeitete modulo 100. Soll nämlich $x^2 - n$ ein Quadrat sein, so müssen die Endziffern 00, a1, a4, 25, b6 oder a9 sein, wobei a gerade und b ungerade ist. 100 wurde sicher deshalb als Modul gewählt, weil man $n \bmod 100$ bei dezimal dargestelltem n direkt ablesen kann. Für die Realisierung mit dem Rechner ist das kein Vorteil mehr; daher kann man einen günstigeren Modul suchen (Im Programm habe ich 144 gewählt).

Um zu veranschaulichen, wie diese Vorgehensweise im Algorithmus F benutzt werden kann, sei dieses zunächst mit dem Modul 9 verdeutlicht: Voraussetzung ist, daß N schon soweit reduziert wurde, daß 3 kein Teiler mehr ist.

In der folgenden Tabelle sind in der Kopfspalte alle in Frage kommenden Reste $n \bmod 9$ aufgeführt (das sind $\varphi(9) = 6$ Stück, denn 0, 3 und 6 scheiden aus, da n nicht durch 3 teilbar ist). In der Kopfreihe sind die Werte $x \bmod 9$ aufgeführt; das sind natürlich 9 Stück. Die Eintragungen in der Tabelle sind jeweils die Werte $x^2 - n \bmod 9$, wobei diese Werte mit doppelter Zeichen-

breite gedruckt wurden, falls sie ein Quadrat mod 9 darstellen (Als Quadrate mod 9 kommen nur die Zahlen 0, 1, 4 und 7 in Frage, wie man leicht nachrechnen kann).

```

-----
          *      X MOD 9
X12-N MOD 9 * 0  1  2  3  4  5  6  7  8
          *
*****
N MOD 9
1  *  8  0  3  8  6  6  8  3  0
   *
2  *  7  8  2  7  5  5  7  2  9
   *
4  *  5  6  0  5  3  3  5  0  6
   *
5  *  4  5  9  4  2  2  4  8  5
   *
7  *  2  3  6  2  0  0  2  6  3
   *
8  *  1  2  5  1  5  3  1  5  2
   *

```

Es ist im Algorithmus F also beispielsweise unsinnig, einen x-Wert zu testen, für den gilt: $x \bmod 9 = 3$ und gleichzeitig $n \bmod 9 = 4$, denn der zu untersuchende Radikand $x^2 - n$ ist keine Quadratzahl, da er bei Teilung durch 9 den Rest 5 ergibt, und 5 ist kein Quadrat mod 9. Unter Berücksichtigung der Ergebnisse dieser Tabelle ergeben sich folgende Inkremente für x:

- n mod 9 = 2, 5 oder 8:
 $x \leftarrow \lceil \sqrt{n/9} \rceil \cdot 9$ und dann jeweils um 3 erhöhen.
- n mod 9 = 1:
 $x \leftarrow \lceil \sqrt{n/9} \rceil \cdot 9 + 1$ und dann abwechselnd um 7 und 2 erhöhen.
- n mod 9 = 4:
 $x \leftarrow \lceil \sqrt{n/9} \rceil \cdot 9 + 2$ und dann abwechselnd um 5 und 4 erhöhen.
- n mod 9 = 7:
 $x \leftarrow \lceil \sqrt{n/9} \rceil \cdot 9 + 4$ und dann abwechselnd um 1 und 8 erhöhen.

Bei den ersten zwei (bzw. 3) Inkrementierungen muß zusätzlich abgefragt werden, ob $x^2 - n$ negativ ist.

Bei dieser Inkrementierung werden also mit einem Test mindestens 3, meistens sogar 4,5 x-Werte abgedeckt, im Gegensatz zu nur einem bei der Primitiv-Version.

Wie man sieht, sind jedoch leider die Inkremente abhängig vom Input (das ist beim Algorithmus E nicht der Fall). Die Berechnung der Inkremente erfolgt aber nur einmal und hat deshalb keine Rechenzeitverlängerung (im späteren Programm nur ca. 2 sec) zur Folge, sie verlängert das Programm jedoch erheblich.

Wie schon erwähnt, wurde im Programm eine Inkrementierung mod 144 gewählt. Die der oben entsprechende Tabelle ist fast 1 m^2 groß, so daß sie hier aus naheliegenden Gründen nicht abgedruckt werden kann. Die Auswertung dieser Tabelle (und die anderer Module zu Testzwecken) erfolgte selbstverständlich durch ein Programm, dessen Arbeitsweise hier kurz geschildert sei (mit der Hoffnung, daß jemand die Abhängigkeit der Inkremente vom Input und vom gewählten Modul besser durchschaut als ich; das entsprechende Programm stelle ich als Listing oder auf 2 Magnetkarten gerne zur Verfügung).

Eingegeben wird lediglich der zu untersuchende Modul (hier 144). Nach geraumer Rechenzeit werden dann alle Quadrate modulo 144 ausgedruckt, und anschließend für jede zum Modul teilerfremde Zahl ein "Zettel" der folgenden Form:

```

-----
      144      ***
R10=  0
R11=  1
R12=  4
R13=  9
R14= 16
R15= 25
R16= 36
R17= 49
R18= 52
R19= 64
R20= 73
R21= 81
R22= 97
R23= 100
R24= 112
R25= 121
-----

```

```

N MOD 144 = 189      N MOD 144 = 133
N MOD 2 = 1          N MOD 2 = 1
N MOD 72 = 37       N MOD 72 = 61
N MOD 3 = 1          N MOD 3 = 1
N MOD 48 = 13       N MOD 48 = 37
N MOD 4 = 1          N MOD 4 = 1
N MOD 36 = 1         N MOD 36 = 25
N MOD 6 = 1          N MOD 6 = 1
N MOD 24 = 13       N MOD 24 = 13
N MOD 8 = 5          N MOD 8 = 5
N MOD 18 = 1         N MOD 18 = 7
N MOD 9 = 1          N MOD 9 = 7
N MOD 16 = 13       N MOD 16 = 5
N MOD 12 = 1         N MOD 12 = 1

```

```

ANFANGSINCREMENT:      ANFANGSINCREMENT:
                        1      ***      5      **
SCHLEIFENINCREMENTE:  SCHLEIFENINCREMENTE:
                        16     ***      8      ***
                        38     ***      46     ***
                        16     ***      8      ***
                        2      ***      10     ***
                        16     ***      8      ***
                        38     ***      46     ***
                        16     ***      8      ***
                        2      ***      10     ***
-----

```

Diese 48 ($\varphi(144)$) "Zettel" wurden dann mühsam in die Programmsequenz 122 bis 234 (ohne 220 bis 229) umgesetzt, wobei ich sicher bin, daß man das eleganter machen kann.

An dieser Stelle sei eine zusätzliche Schwierigkeit genannt, die Algorithmus F bereitet: Da im Ausdruck $x^2 - n$ der x^2 -Wert bei großem n größer als 10^{10} wird, ist keine korrekte Berechnung mehr möglich, wenn, wie im obenstehenden Primitivprogramm, inkrementiert wird (Man versuche, damit $98947 \cdot 101063$ zu faktorisieren). Diese Schwierigkeit läßt sich aber umgehen, indem man ausnutzt, daß $(x + i)^2 - n = (x^2 - n) + (i^2 + 2ix)$ ist, wie das im späteren Programm realisiert wurde.

Im Gegensatz zu Algorithmus E, bei dem erhöhter Aufwand immer geringeren zusätzlichen Nutzen bietet, führt beim Algorithmus F die Wahl eines geeigneten größeren Moduls (mit mehr Primzahlpotenzen) zu wesentlich besseren Ergebnissen. Nach Knuth ist beispielsweise bei Berücksichtigung der ersten 30 Primzahlen (mit dem Taschenrechner natürlich nicht realisierbar) nur noch einer von 2^{30} Fällen zu testen.

Die Arbeitsweise von Programmen, die auch Zahlen 10^{10} faktorisieren, stelle ich mir so vor, daß beim Übergang zum Algorithmus F das Programm stoppt und dann nur noch anzeigt, welche von möglicherweise sehr vielen Datenkarten mit den richtigen Inkrementen eingelesen werden muß, da das Berechnen der Inkremente im Programm zwar nicht zu zeitaufwendig wäre, jedoch den Programmumfang so stark aufblähen würde, daß für die eigentlichen Testroutinen, die dann ja wesentlich komplizierter sind, kaum Platz bliebe.

Bei unserem Beispiel, dem Modul 144, werden je nach Beschaffenheit des Restfaktors entweder 72 ($n \bmod 3 = 1$) oder 48 ($n \bmod 3 = 2$) x -Werte mit 4 Tests überdeckt. Das bedeutet, daß bei n in der Größenordnung 10^{10} mit den ersten 4 Tests 3720 bzw. 3050 Testfaktoren p überdeckt werden. Dieser Bereich für p nimmt mit

wachsendem x bis auf ca. 44 in beiden Fällen ab; bis dahin ist die Testmethode nach Algorithmus E schneller.

Zum Abschluß sei noch die Vorgehensweise erläutert, mit der die hoffentlich optimalen Übergangsstellen von E nach F gefunden wurden. Die "zeitfressenden" Schleifen im Programm sind die Programmsequenzen 052 bis 110 für Algorithmus E und 262 bis 324 für Algorithmus F. Die Schleife für E wird bei meinem Rechner in $\alpha = 1,160$ sec durchlaufen, während für die Schleife in F die Zeit $\beta = 1,669$ sec benötigt wird (Bei der folgenden Konstantenberechnung geht letztendlich nur das Verhältnis β/α ein, so daß unterschiedliche Rechnerlaufzeiten kompensiert wären).

Insgesamt muß vom Programm jeder in Frage kommende Faktor p im Bereich 0 bis \sqrt{n} getestet werden, wobei das in folgender Reihenfolge geschieht:

Von 0 bis $k_m \cdot \sqrt{n}$ mit Algorithmus E und anschließend von \sqrt{n} bis $k_m \cdot \sqrt{n}$ mit Algorithmus F; dabei ist $m = n \bmod 3$ ($= 1$ oder 2) und es ist $0 < k_1 < k_2 < 1$. Es sei weiter $B_m = 72$ oder 48 (für $m = 1$ oder 2). Im Algorithmus F gilt:

$$n = p \cdot q = (x-y)(x+y) = x^2 - y^2$$

$$\text{d.h.} \quad p(x) = x - \sqrt{x^2 - n} \quad \text{bzw.} \quad x(p) = (p^2 - n)/2p$$

Gesucht wird nun zunächst eine Funktion $f(p, n)$, die die Differenz des Verhältnisses der überstrichenen Testfaktorbereiche zu dem Verhältnis der Rechenzeiten für diese Bereiche angibt.

$$\begin{aligned} f(p, n) &= \frac{p(x) - p(x - B_m)}{30} - \frac{\beta}{\alpha} \\ &= \left(\sqrt{\frac{n + p^2}{2p} + B_m^2} - n + \frac{p^2 - n}{2p} - B_m \right) \cdot \frac{1}{30} - \frac{\beta}{\alpha} \end{aligned}$$

Die Faktoren k_m werden dann durch lineare Approximation der Nullstellen p dieser Funktion in Abhängigkeit von n ermittelt. Die Nullstellen dieser Funktion für verschiedene n -Werte (mit stärkerer Gewichtung großer Werte) wurden mit SOLVE (bzw. SOL) des Mathe I Moduls berechnet. Die lineare Approximation dieser Ergebnisse mit dem Kurvenanpassungsprogramm der Standardprogrammammlung ergibt für

$$n \bmod 3 = 1: R^2 = 1,00000; a = 21,79642; b = 0,48023$$

und für

$$n \bmod 3 = 2: R^2 = 1,00000; a = 21,74261; b = 0,55692$$

Die absoluten Glieder a können vernachlässigt werden, da die Schleife in E bei einem Durchlauf schon 30 p -Werte überdeckt. Es ergibt sich also für das Programm:

$$k_1 = \underline{0,480} \quad \text{und} \quad k_2 = \underline{0,557}$$

Die längste Rechenzeit ergibt sich für $p = 10^{10} - 71$ (größte Primzahl $p < 10^{10}$ mit $p \bmod 3 = 2$): 45 min 46 sec, dagegen für $10^{10} - 33$ (größte Primzahl $< 10^{10}$): 41 min 23 sec.

Ein "ungünstiger" Fall ist sicher $55697 \cdot 179533$ mit 45 min 38 sec, denn hier würde der kleinere Faktor mit Algorithmus E direkt gefunden, wenn nicht zu F übergegangen würde.

55681·179591 mit 35 min 19 sec liegt kurz vor diesem Übergang. Diese "Ungereimtheit" liegt einfach daran, daß mit Algorithmus F von der Wurzel an abwärts getestet wird; dafür wird dann ja auch beispielsweise 99991·100003 in 30 min 31 sec gefunden (alle Rechenzeiten ohne angeschlossenen Printer).

Diese Beispiele zeigen, daß mit Algorithmus F für 52 % des gesamten Testbereiches nur 26,3 % bzw. für 44,3 % nur 22,8 % der Gesamtrechenzeit benötigt werden.

Die Rechenzeit für das untenstehende Programm, das die Zahlen unterhalb 2^{19} , die die größten Primzahlen $< 2^{19}$ enthalten, faktorisiert, beträgt knapp 50 min.

01*LBL "P"	524201	524281
02 CF 26	IS PRIME	=7*74897
03 88		524280
04 STO 17	524202	=2†3*3*5*17
05 2	=2*3*7†2	*257
06 ENTER†	*1783	524281
07 19	524203	=269*1949
08 Y†X	IS PRIME	524282
09 87		=2*11*23831
10 -	524204	
	=2†2*29*4519	524283
11*LBL 01	524205	=3*174761
12 XEQ "PRM"	=3†3*5*11	524284
	*353	=2†2*131071
13 RCL 00	524206	
14 1	=2*262103	524285
15 +	524207	=5*23*47*97
16 DSE 17	=113*4639	524286
17 GTU 01	524208	=2*3†3*7*19
18 SF 26	=2†4*3*67	*73
19 BEEP	*163	524287
20 END	524209	IS PRIME
	=7*7400	524288
		=2†19

USEKS' PROGRAM LIBRARY EUROPE
 EU PÄÄSIE BENÜTZER-PROGRAMMBIBLIOTHEK
 BIBLIOTHÈQUE EUROPÉENNE DE PROGRAMMES UTILISATEURS
 LIBRERIA EUROPEA DEGLI UTILIZZATORI

Page 1 of 8

HP 41C PROGRAM SUBMITTAL FORM
 PROGRAMMFORMBLATT/DOCUMENTATION DU PROGRAMME/GENERALITÀ SUL PROGRAMMA

Program Title Programmstitel Titre du programme Título del programa		<u>PRIMFAKTORZERLEGUNG</u>	
Category No. Kategorie Nr. Category N° Categorie N°	Name Rubrik Rubrique Nome della categoria	<u>201 Zahlentheorie</u>	
No. of program lines Anzahl Programmzeilen Nombre de lignes de programme N° de lignes de programme	<u>502</u>	No. of data registers Anzahl des benutzten Datenregister Nombre de registres de données N° de registres utilisés	<u>116 (total)</u>
Recommended HP 41C System configuration Empfohlene System-Konfiguration Configuration recommandée Configurazione raccomandata			
Empfohlene System-Konfiguration Configuration recommandée Configurazione raccomandata	Part #1 <u>Memory Modul</u> <u>(Printer)</u>	Part #2 Part #4	
This program requires the following programs as subroutines: Dieses Programm benutzt folgende Programme als Unterprogramme: Ces programme utilise les programmes suivants comme sous-programmes: Questo programma usa i seguenti programmi come sottonprogrammi:			
HP Applications ROM HP Applications ROM ROM d'applications HP ROM di applicazioni HP	Program Name: Program: Nom du programme: Programma:		
Program Abstract Kurzfassung Résumé Breve descrizione del programma			
<u>Durch die Kombination von konventionellem Testverfahren und der Fermat'schen Faktorisierungsmethode werden mit dem Programm ganze Zahlen n im Bereich $2 \leq n \leq 10^{10}$ schnell faktorisiert.</u>			
<u>Die längste Rechenzeit ist ca. 46 min für die Primzahl $10^{10}-71$. $99991 \cdot 100003$ wird in ca. 31 min faktorisiert.</u>			
Name Name/Nom/Nome <u>Friedrich Hillebrandt</u>			
Address Adresse/Address/Indirizzo <u>Jülicher Straße 443</u>			
City Ort Località Città	Postal Code Postleitzahl Code postal C.A.P.	Country Land Paese	<u>Aachen</u> <u>5100</u> <u>Deutschland</u>
ACKNOWLEDGMENT AND AGREEMENT Erklärung und Ermächtigung/Dichiarazione e Autorizzazione			
To the best of my knowledge, I have the right to contribute this program material without breaching any obligation concerning confidential information of other persons or organizations. I am contributing this program material on a nonconfidential nonobligatory basis to Hewlett-Packard S.A. ("HP") for inclusion in its program library, and I agree that HP may use, duplicate, modify, publish, and distribute the program material, and authorize others to do so without obligation or liability of any kind. HP may publish my name and address as the contributor, to facilitate user inquiries pertaining to this program material.			
Ich versichere nach bestem Wissen, dass ich über mein Programmrecht frei verfügen kann, ohne dass sich dadurch bei HP andere Programmisten oder mich betreffende Verpflichtungen gegenüber Dritten oder sonstige rechtliche Nachteile ergeben. HP kann meinen Programmbeitrag ohne Lebenshaltungs- und sonstige Verpflichtungen in beliebiger Weise benutzen oder verbreiten. Gegen eine Veröffentlichung meines Namens im Zusammenhang mit meinem Programmbeitrag habe ich mich zu erklären.			
Au mieux de ma connaissance, je déclare avoir le droit de fournir le présent programme sans enfreinte des obligations de secret à l'égard d'autres personnes ou organismes. Je contribue au présent programme à la Société Hewlett-Packard S.A. (HP) sur une base non confidentielle, pour incorporation dans sa bibliothèque de programmes et j'autorise HP, qui pourra à son tour autoriser d'autres personnes, à l'utiliser, le reproduire, le modifier, le publier et le distribuer, sans obligation ni responsabilité d'aucune sorte. HP est autorisée à publier mes noms et adresse en tant qu'auteur du présent programme en vue de faciliter les échanges d'informations avec les utilisateurs de ce programme.			
Per quanto me sia a conoscenza, ho il diritto di fornire il presente programma senza violare alcun obbligo di segreto o confidenzialità verso altre persone o organismi. Fornisco il programma alla Hewlett-Packard S.A. (HP) su una base non confidenziale per includerlo nella sua biblioteca di programmi e autorizzo la società HP, la quale a sua volta potrà autorizzare in qualsiasi modo altre persone, a utilizzarlo, riprodurlo, modificarlo, pubblicarlo e distribuirlo senza obbligo né responsabilità di alcuna specie. La Hewlett HP potrà, a sua discrezione, pubblicare il mio nome e indirizzo quale autore del presente programma, onde facilitare le scambie d'informazioni con gli utilizzatori di questo.			
Date Datum Date	Signature Unterschrift Signature Firma		
<u>07.03.81</u>			

15176-81

USLA INSTRUCTIONS I
 PROGRAMMABLAUF I
 INSTRUCTIONS D'EMPLOI I
 NORME OPERATIVE I

Page 2 of 8

Step Schritt Pty Phase	Instructions Operation Instructions Tauxions	Variables Dimensione Dimens Dim	Function(s) Taux(s) Taux(s) Taux	Result Resultat Resultat Resultat
1	Programm eingeben *			
2	Faktorisierung von n	n	XEQ "PRM"	n = P ₁ ^{e₁} * P ₂ ^{e₂} * ... ! oder IS PRIME
3	Faktorisierung von n mit Einzelanzeige der Primzahlpotenzen	n	SF 00 XEQ "PRM"	n = P ₁ ^{e₁} * P ₂ ^{e₂} ! oder IS PRIME
	Nach Beendigung der Faktorisierung ertönt BEEP; die Ergebnisse können mit R/S oder XEQ 10 beliebig oft abgerufen werden (Allerdings auch bei gesetztem Flag 00 nur in der gepackten Form von 2))			
	Ist der Printer angeschlossen und eingeschaltet, so werden die Ergebnisse gemäß 2) oder 3) ausgedruckt.			
	* Wird das Programm ohne angeschlossenen Printer eingetastet, so wird Zeile 024 (siehe Listing Seite 5) zu XEQ "PRA" (d.h. das Programm wird um 3 Bytes länger, die Funktion ändert sich nicht). Es darf dann im gesamten Programmspeicher kein globales Label "PRA" existieren.			

PROGRAM DESCRIPTION 1

Page 2 of 1

PROGRAMMBSCHREIBUNG I
DESCRIPTION DU PROGRAMME I
DESCRIZIONE DEL PROGRAMMA IApplication, Equations, Variables
Anwendung, Gleichungen, Variablen
Applicazioni, Equazioni, Variabili
Applicazioni, Equazioni, Variabili

Mit dem Programm wird für eine natürliche Zahl n ($n = \pi \cdot 10^e$) die kanonische Zerlegung in ihre Primfaktoren ermittelt.

Dabei wird im ersten Programmteil (E) unter Ausnutzung des Kongruenzverhaltens modulo 30 der Testteiler bis ca. $0.5 \cdot \sqrt{n}$ die Zahl faktorisiert.

Der dann verbleibende Restfaktor, der nur noch aus höchstens zwei Faktoren bestehen kann, wird über den Ansatz $n = (x-y)(x+y)$ nach der Fermat'schen Methode (F) weiter untersucht, wobei für die Inkrementierung der x -Werte deren Kongruenzverhalten modulo 144 ausgenutzt wird.

Literatur: D.E.Knuth, The Art of Computer Programming, Vol. II, 4.5.4.

Operating limits and Warnings
Grenzen und Einschränkungen
Limites et avertissements
Limiti operativi e avvertenze

Nach Verwendung des Programms als Unterprogramm müssen die Sonderfunktionsflags 12, 21, 25 und 29 und das Anzeigeformat ggf. neu gesetzt werden.

Die eingegebene Zahl n muß im Bereich $2 \leq n \leq 10^{10}$ liegen; falls sie nicht ganzzahlig ist, wird gerundet. Zu kleine oder zu große Eingaben werden vom Programm nicht abgefragt.

This program has been verified only with respect to the numerical example given in Program Description II. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER IPI NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER IPI NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Dieses Programm wurde lediglich anhand des in der Programmbeschreibung II enthaltenen Zahlenbeispiels überprüft. Der Benutzer erhält und benutzt das Programmmaterial auf eigenes Risiko, ohne es deshalb als geeignet, ohne weitere anderweitige Zusicherungen oder Beschreibungen zu empfangen. WEDER IPI NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER IPI NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Le présent programme n'a été vérifié qu'en ce qui concerne l'exemple numérique indiqué dans la description du programme II. L'utilisateur accepte et utilise le présent programme À SON PROPRIÉTAIRE RISQUE et sans se préoccuper ni s'appuyer sur toute représentation ou description concernant le matériel du programme. NI IPI NOR THE CONTRIBUTOR MAKE ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER IPI NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Questo programma è stato verificato soltanto per quanto concerne l'esempio numerico indicato nella Descrizione II del Programma II. L'utente accetta e utilizza il presente programma A SUO INTERO RISCHIO, senza fidarsi o appoggiarsi ad alcuna garanzia, esplicita o implicita, con riferimento al presente programma, né alla sua commerciabilità o idoneità per uno scopo particolare. NI IPI NOR THE CONTRIBUTOR MAKE ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER IPI NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

PROGRAM DESCRIPTION II

PROGRAMMBESCHREIBUNG II
DESCRIPTION III: PROGRAMME II
DESCRIZIONE DEL PROGRAMMA II

Page 4 of 8

Example
Beispiel
Esempio

Die Rechenzeit ist nicht nur vom Wert des größten Primfaktors abhängig, sondern auch von verschiedenen anderen Umständen.

Generell sind die Rechenzeiten für Zahlen n mit $n \bmod 3 = 2$ etwas länger als für Zahlen n mit $n \bmod 3 = 1$.

Die längste Rechenzeit ergibt sich für $n = 10^{10} - 71$; die beiden noch darüber liegenden Primzahlen unter 10^{10} werden schneller als solche erkannt, da sie kongruent 1 modulo 3 sind.

Die schnellsten Rechenzeiten ergeben sich bei nicht angeschlossenem Printer; bei ausgeschaltetem aber angeschlossenem Printer sind sie unwesentlich länger. Wesentlich länger sind sie bei eingeschaltetem Printer.

Es ist möglich, nach beendigter Faktorisierung den Printer anzuschließen, bzw. einzuschalten und dann mit RIS (bzw. XEQ 10) einen Ausdruck zu bekommen.

220	7,000000000 ENTER†	2,00 ENTER†
=2†2*5*11	7,999999999 Y†X	27,00 Y†X
	5764800,909 ***	111,40 -
	XEQ 'PRN'	134217616,6 ***
284	5764801	134217617
=2†2*7†1	=7†8	IS PRIME
34765731	9970592519	
=3†2*7*11*13	=2143*2153	SF 00
*17*227	*2161	5907942612
		=2†2
		*3†2
		*11†2
		*13
		*17†2
		*19†2
36939357	2 ENTER†	
=3†2*7*13*23	17 ENTER†	
*37*53	1 -	
	Y†X	
	2 ENTER†	
	17 Y†X	
	1 -	
	0	
2172649216		
=2†8*257		
*33023		
		RUN
	8589869056	5907942612
	=2†16*131071	=2†2*3†2
		*11†2*13
		*17†2*19†2

PROGRAM LISTING
 PROGRAMMAUFLISTUNG
 LISTAGE DU PROGRAMME
 LISTATO DI PROGRAMMA

Page 5 of 8

Line Zeile Ligne Linea	Keystrokes Touchéage Touches Tasti	Comments Kommensur Commentaires Comentarii	Line /ze Ligne Linea	Key pressed Tastendrück Touches Tasti	Comments Kommensur Commentaires Comentarii
01	014LBL *PRM* SF 05 SF 12 CF 29 FIX 0 RND STO 00 2 STO 02 4 STO 04 + STO 06 SIGN 11 STO 08 + STO 09 CLA ASTO 11 ASTO 12 ARCL 00		51	CF 06 CF 07 RCL 07 3 MOD GTO IND X	
05		<i>Allgemeine Initialisierung</i>	55	120*LBL 02 RCL 07 4 MOD 3 X=Y? GTO 03 18 STO 01 6 SF 06 GTO 02	
10	SF 21 SF 25 PPR CF 21 RVIEH	<i>Test, ob Printer eingeschaltet</i>	60	140*LBL 03 12 STO 01 SF 07 GTO 02	
15	RCL 00 ENTER1 ENTER1 STO 07 RCL 02 MOD X=0? XEQ IND 04 CLX 3 MOD X=0? XEQ IND 04 CLX 5 MOD X=0? XEQ IND 04 CLX 7 MOD X=0? XEQ IND 04 XEQ 00	<i>Vorab-Test der Faktoren p=2, 3, 5 und 7 im Algor. E</i>	65	145*LBL 01 RCL 07 12 MOD ST+ X RCL 07 9 MOD + XEQ IND X GTO 02	
20			70	156*LBL 03 16 STO 01 2 RTN	
25	52*LBL 00 CLX RCL 04 LASTX + MOD X=0? XEQ IND 04 CLX RCL 02 LASTX + MOD X=0? XEQ IND 04 CLX RCL 04 LASTX + MOD X=0? XEQ IND 04 CLX RCL 02 LASTX + MOD X=0? XEQ IND 04 CLX RCL 02 LASTX + MOD X=0? XEQ IND 04 CLX RCL 06 LASTX + MOD X=0? XEQ IND 04 CLX RCL 02 LASTX + MOD X=0? XEQ IND 04 CLX RCL 06 LASTX + MOD X=0? XEQ IND 04 BSE 10 GTO 00	<i>Berechnung der Inkremente für Algorithm. F</i>	75	161*LBL 06 18 STO 01 14 RTN 166*LBL 09 8 STO 01 46 RTN 171*LBL 15 20 STO 01 16 RTN 176*LBL 10 4 STO 01 32 RTN 181*LBL 21 28 STO 01 8 RTN 186*LBL 02 STO 02 RCL 05 RCL 01 STO 03 ST+ X - RCL 02 - STO 04 RCL 07 16 MOD GTO IND X	<i>(Kongruenz- verh. mod 144)</i>
30		<i>(Kongruenz- verh. mod 30)</i>	80	200*LBL 03 201*LBL 11 CF 07 RCL 01 X< 02 STO 01 RCL 03 X< 04 STO 03	
35		<i>(Schleifenindex Reg. 10 wurde spätestens in Zeile 057 berechnet)</i>	85	209*LBL 01 210*LBL 13 FS?C 06 GTO 11 RCL 02 X< 04 STO 02	
40			90	216*LBL 05 217*LBL 07 218*LBL 09 219*LBL 15	
45	CLX LASTX X12 X1Y? GTO 97	<i>Test, ob Faktorisierung abgeschlossen</i>	95		
50	ST+ Y CLX LASTX ST+ X / STO 10	<i>Schleifenzähler für Algor. F (Rest Zeile 253)</i>	100		

179,180-81

Please use paper plus or similar weights. Above the type may often be pressed.
 Bitte Lesungs mit Papierton wählen. Kibetone können Druck machen!

SVP utiliser de la colle
 Per l'incollatura della colla

«do press on other impressions»
 «do stamp»

PROGRAM LISTING
PROGRAMM AUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Page 6 of 8

Line Zeil Line	Keystrokes Tastendruck Tast	Comments Anmerkungen Commentaires Commenti	Line Zeil Line	Key pressed Tastendruck Touches Tast	Comments Anmerkungen Commentaires Comments
01	RCL 07 SORT RCL X RCL 05 MOD - STO 06 XY2 RCL 07 -	<i>Initialisierung von x^2-n</i>	51	347+LBL 97 RCL 07 ENTERT SIGN STO 07 X(Y) X+Y? XEQ 01 BEEP XEQ 02 SF 21 GTO 02	<i>Faktorisierung ist abgeschlossen</i>
05	RCL 04 2 / FS? 07 CLX	<i>Anfangsinkre für Algor. F</i>	55	359+LBL 10 SF 12 SF 21 CLR ARCL 00 AVIEW RCL 09 I E3 / II +	<i>Anzeige- Routine</i>
10	XEQ 07 X=0? GTO 01 RDN RCL 01 XEQ 07 X=0? GTO 01 RDN RCL 02 XEQ 07 X=0? GTO 01 RDN RCL 03 XEQ 07 X=0? GTO 01	<i>Vorab-Test der ersten X-Werte</i>	60	370+LBL 14 CLR ARCL IND X ISG X ARCL IND X AVIEW ISG X GTO 14	<i>RTN in Zeile 384 ist RTN des Gesamtpro- gramms</i>
15	CLX RCL 10 RCL 06 - RCL 05 / INT STO 10 ISG 10	<i>Schleifenzähler für Algor. F</i>	65	370+LBL 02 RDN CLX CLD CF 12 CF 21 RTN GTO 10	
20			70		
25	262+LBL 12 CLX RCL 04 XY2 RCL 06 LASTX ST+ 06 ST+ Y + + + ENTERT SORT FRC X=0? GTO 01 CLX RCL 01 XY2 RCL 06 LASTX ST+ 06 ST+ X + + + ENTERT SORT FRC X=0? GTO 01 CLX RCL 02 XY2 RCL 06 LASTX ST+ 06 ST+ X + + + ENTERT SORT FRC X=0? GTO 01 CLX RCL 03 XY2 RCL 06 LASTX ST+ 06 ST+ X + + + ENTERT SORT FRC X=0? GTO 01 BSE 10 GTO 12 GTO 97	<i>Testschleife für Algor. F</i> <i>(Kongruenz- verh. mod 144)</i>	75	386+LBL 04 RCL 07 LASTX MOD X=0? GTO 01 CLX LASTX ST/ 07 SIGN ST+ Y PBN GTO 04	<i>Faktor gefunden (Reduktion von n, Hochzählen des Exponenten)</i>
30			80	399+LBL 01 +* FS?C 05 "+" RCL 00 LASTX ARCL X X=Y? *IS PRIME? SIGN RT X(=Y? GTO 01 *+! ARCL X +	<i>Herstellung der Zusätzlichen Anzeige</i>
35			85	414+LBL 01 FS? 25 SF 21 FS? 00 AVIEW CF 21 ASTO 2 ASHF ASTO T CLR ARCL IND 00 ARCL IND 09 ARCL 2 ARCL T ASTO 03 ASHF ASTO 01 ASHF ASTO Y CLR ASTO X X=Y? GTO 01 RCL 03 STO IND 00 RCL 01 GTO 13	<i>"Anhängen" des neuen Anzeigteils</i>
40	326+LBL 07 XY2 RCL 06 LASTX ST+ 06 ST+ X + + ENTERT X=0? RTN SORT FRC RTN	<i>Test-U-Pam für mögliche negative x^2-n Werte</i>	90	441+LBL 01 XEQ 03 RT RT 2 ST+ 00 ST+ 09 RDN STO IND 00 X(Y)	<i>neue Anzeige zu lang</i>
45			95		
50	341+LBL 01 RCL 06 LASTX - FRC XEQ 04	<i>Faktor wurde mit Algor. F gefunden</i>	100		

Not on Paper other than 40 gsm. Adhesive tape must adhere properly.
 sur Papier ou sur Papier autre que 40 gsm. Adhésif doit adhérer correctement.

Not on Paper other than 40 gsm. Adhesive tape must adhere properly.
 sur Papier ou sur Papier autre que 40 gsm. Adhésif doit adhérer correctement.

Not on Paper other than 40 gsm. Adhesive tape must adhere properly.
 sur Papier ou sur Papier autre que 40 gsm. Adhésif doit adhérer correctement.

PROGRAM LISTING
 PROGRAMMAUFLISTUNG
 LISTAGE DU PROGRAMME
 LISTATO DI PROGRAMMA

Page 7 of 8

Line Zelle Ligne Linea	Keystrokes Tastendrücke Touches Tasti	Comments Kommentare Commentaires Commenti	Line Zelle Ligne Linea	Key pressed Tastendrücke Touches Tasti	Comments Kommentare Commentaires Commenti
01	451*LBL 13 STO IND 09 ARCL IND 00 ARCL IND 09 FC? 00 AVIEW	neue Anzeige daßt ins Anzeigeregister	31		
05			55		
10	457*LBL 00 LASTX STO 01 RCL 07 SORT K(Y) - 30 / RCL 07 3 MOD XEQ IND X RT * INT STO 10 RCL 07 RCL 07 RCL 07 RCL 01 ABS RTM	Berechnung des neuen Schleifenindex für Algor. E und der Periodenlänge in Algor. F	60		
15	480*LBL 00 481*LBL 01 72 STO 05 .48 RTM		65		
20	486*LBL 02 48 STO 05 .557 RTM		70		
25	491*LBL 03 FC? 00 FC? 25 RTM CLA SF 21 ARCL IND 00 ARCL IND 09 AVIEW CLA CF 21 END	U-Pgm, das bei eingeschalteten Printer eine fertige Anzeigezeile druckt	75		
30			80		
35			85		
40			90		
45			95		
50			100		

181, 182-81

Please use paper glue or arch binding. Adhesive tape may affect print.
 Bitte Leimung mit Papieren verwenden. Klebstoffe können Druck beeinträchtigen.

SV P' utiliser de la colle à papier pour fixer les feuilles. Les collants adhésifs peuvent altérer l'impression.
 Per favore usare la colla per carta. Il collante adesivo può alterare lo stampato.

Allgemeine Instruktionen

	INPUT	TASTEN	OUTPUT
1. Eingabe		Lbl --	
Fkt		
		
		RTN	
2. Anwahl Prgm		XEQ GQ	A=?
	A	R/S	B=?
	B	R/S	N=?
	N	R/S	FUNCTION NAME?
	Name	R/S	F= ----
3. Eingabe A/B		XEQ a	siehe 2.
4. Eingabe N		XEQ A	N=?
	N	R/S	F= ----

weiter 1., 2., 3., 4.

Bemerkungen

1. siehe Programmlinie O26: Function Name? und ff
Für Besitzer des MATH1A Moduls: XEQ ROM XFN
2. Programmlinien O55, O63, O75 X<>X erfüllt die Funktion einer NOP-Funktion.
Einzugeben als: XEQ Alpha X<> Alpha · X
3. Programm eingeben; Daten eingeben in folgende Register:

Reg.-Nr.	Konstante
7	Z ₁
8	Z ₂
9	Z ₃
10	Z ₄
11	Z ₅
12	Z ₆
13	Z ₇
14	Z ₈
15	Z ₉
16	Z ₁₀

Die Daten sind mit 7,016 XEQ Alpha WDTAX Alpha auf eine Kartenhälfte zu beschreiben.

Detlev Bock

10 PUNKTE GAUß QUADRATUR

```

01+LBL "GQ"
02 1.488743
39 E-1
03 STO 07
04 2.955242
247 E-1
05 STO 08
06 4.333953
941 E-1
07 STO 09
08 2.692667
193 E-1
09 STO 10
10 6.794095
683 E-1
11 STO 11
12 2.190863
625 E-1
13 STO 12
14 8.650633
667 E-1
15 STO 13
16 1.494513
492 E-1
17 STO 14
18 9.739065
28 E-1
19 STO 15
20 6.667134
43 E-2
21 STO 16
22 CLST
23+LBL a
24 "B=?"
25 PROMPT
26 STO 01
27 "A=?"
28 PROMPT
29 STO 00
30 SF 00
31+LBL A
32 "N=?"
33 PROMPT
34 RCL 01
35 RCL 00
36 -
37 X<>Y
38 /
39 2
40 /
41 STO 02
42 SF 00
43 RCL 00
44 X<>Y
45 -
46 STO 03
47 "FKT NAM
E?-"
48 AON
49 PROMPT
50 ASTO 05
51 AOFF
52+LBL E
53 RCL 03
54 RCL 02
55 +
56 LASTX
57 +
58 RCL 01
59 X<=Y?
60 GTO e
61 X<>Y
62 STO 03
63 0
64 FS?C 00
65 STO 04
66 6
67 STO 06
68 XEQ b
69 XEQ b
70 XEQ b
71 XEQ b
72 XEQ b
73 GTO E
74+LBL b
75 ISG 06
76 X<> X
77 RCL 03
78 RCL 02
79 RCL IND
06
80 *
81 -
82 XEQ IND
05
83 ISG 06
84 X<> X
85 RCL IND
06
86 *
87 ST+ 04
88 DSE 06
89 RCL 03
90 RCL 02
91 RCL IND
06
92 *
93 +
94 XEQ IND
05
95 ISG 06
96 X<> X
97 RCL IND
06
98 *
99 ST+ 04
100 RTN
101+LBL e
102 RCL 02
103 RCL 04
104 *
105 "F="
106 ARCL X
107 PROMPT
108 END

```

Zum Problem Zufallzahlengenerator von Erwin Hartmann kann ich sagen, daß die Routine aus der HP-Standardprogrammammlung meines Erachtens schon in Ordnung ist. Die Verschiebung der Häufigkeit nach größeren Zahlen erfolgt durch den Befehl Zeile 46 SQRT in seinem Programm. Dieser Befehl gehört nicht zum Zufallzahlengenerator, sondern ist im Programm "Arithmetik-Lehrgang" offensichtlich eingefügt, um eine Bevorzugung von schwierigeren Aufgaben mit größeren Zahlenwerten zu erzielen.

Auch das Erscheinen von \emptyset ist ganz richtig. Der Generator erzeugt Zufallszahlen von 0-0,999..., wenn 0 als Zufallszahl nicht gewünscht wird, muß eben die anschließende Weiterverarbeitung richtig erfolgen.

Anschließend eine Formel, um gleichverteilte Zufallszahlen innerhalb einer maximalen und minimalen Zahl aus obigen Zufallszahlen von 0-1 zu errechnen.

```
INT(n(max-min+1)+min)
n....Zufallszahl 0-0,999...
max..maximale gewünschte Zahl
min..minimale gewünschte Zahl
```

In meinem Programm "Startroutine für Zufallzahlengenerator" muß
Zeile 03 natürlich richtig heißen: 'OFF, ON'

Pseudozufallszahlengeneratoren, wie zum Beispiel im Programm 'MEMORY' verwendet, erfordern die Eingabe einer Startzahl zwischen 0 und 1. Hier ist die Möglichkeit der Manipulation durch Eingabe einer bekannten Zahl gegeben.

Das nebenstehende Programm vermeidet diesen Umstand. Nach Programmstart wird in einer Endlosschleife die Zahl Pi aufaddiert. Die Schleife wird verlassen durch kurzes Ausschalten des Rechners und Wiedereinschalten. Das Programm wird dann automatisch fortgesetzt. Die so gewonnene Startzahl steht in R 00. Da die Schleife in der Sekunde etwa 4 mal durchlaufen wird, ist nach einigen Sekunden der Stoppzeitpunkt nicht mehr zu bestimmen und die Startzahl unabhängig vom Spieler.

In meinem Programm "Startroutine für Zufallszahlengenerator" muß Zeile 03 natürlich richtig heißen: 'OFF, ON'

Programm zur Überprüfung der durch einen Zufallszahlengenerator erzeugten Zahlen

Das Programm speichert die Anzahl der generierten Zufallszahlen im Register mit der Nummer, die der Zufallszahl entspricht. Diese Register können dann durch XEQ A angezeigt werden. Während des Programmablaufs wird ständig die Zahl der bisher generierten Zahlen angezeigt.

Das Programm kann jederzeit durch R/S gestoppt, die Register mit XEQ A angesehen und mit R/S weitergeführt werden.

Status: Size = höchste zu generierende Zahl + 5

- Register n+1 = Ausgangszahl für Zufallsgenerator
- Register n+2 = Höchstzahl + 1
- Register n+3 = Zufallszahl
- Register n+4 = Anzahl der generierten Zahlen

n = Höchstzahl (im Beispielprogramm = 49)

```

01+LBL "KDM
02 CLRG
03 FIX 0
04 .258369
05 STO 50
06 50
07 STO 51
08 CF 29
09+LBL 01
10 1
11 ST+ 53
12 RCL 53
13 VIEW X
14 XEQ "RND
M
15 STO 52
16 1
17 ST+ IND
52
    
```

Status
 Auswahl
 abspalten

```

18 GTO 01
19+LBL A
20 CLST
21 CLA
22+LBL 02
23 "Z"
24 ARCL Y
25 "F"
26 ARCL IND
Y
27 AVIEW
28 PSE
29 RDN
30 1
31 ST+ Y
32 RDN
33 RCL 51
34 X>Y?
35 GTO 02
36 STOP
37 GTO 01
38+LBL "RND
M
39 RCL 50
40 9821
41 *
42 .211327
43 +
44 FRC
45 STO 50
46 SQRT
47 RCL 51
48 *
49 INT
50 END
    
```

Register anzeigen
 Hauptauswahl
 Zufallszahlengenerator (s. 24/26)

Erwin Hartmann

Eulersche Gammafunktion

Die Eulersche Gammafunktion ist in der Zahlentheorie bis zur theoretischen Physik eine oft verwendete Funktion. Sie ist durch ein uneigentliches Integral definiert:

$$\Gamma(x) = \int_0^{\infty} e^{-t} t^{x-1} dt, \quad \Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

und von C.F. Gauß für alle $x \neq 0, -1, -2, \dots$ durch den Grenzwert

$$\Gamma(x) = \lim_{n \rightarrow \infty} \frac{n! n^x}{x(x+1)(x+2)\dots(x+n)}$$

erklärt. ($n^x = e^{x \log n}$ mit positiv-reellem Logarithmus)

Bei diesem Programm wird als Näherungsberechnung für die Funktionswerte die Stirlingsche Formel benutzt:

$$\Gamma(x+1) \approx \sqrt{2\pi x} \quad x^x \exp\left[\frac{-x+1}{12x}\right]$$

Für $x = 40$ liefert das Programm ein Ergebnis mit einem Fehler von 10^{-7} bis 10^{-8} . Für $x = 45$ wird Gamma von x mit

$$\Gamma(x) = \frac{\Gamma(z)}{(z-1)(z-2)(z-3)\dots z}$$

wobei $z=45+\text{Dezimalteil von } x$.

Instruktionen:

	INPUT	TASTEN	DISPLAY
1. nur beim Durchlauf	x	XEQ GM	I(x)=====
2.	x	A	S.O.

Anmerkung: bei Verwendung eines Druckers kann das GAMMA-Symbol mit "7 XEQ Alpha ACX / ACCHR Alpha" erzeugt werden.

Detlev Bock

PRP **

```
01*LBL *GAM      75*LBL 02
   MA            76 LASTX
02 CF 20        77 1
03 FIX 3        78 -
04 RAD          79 FS? 01
05 CF 01        80 1 E2
06 X<=0?       81 FACT
07 XEQ 00
08 STO 01      82*LBL 08
09 FRC        83 *I(X)*=-
10 X=0?       84 ARCL X
11 GTO 02     85 AVIEW
12 45         86 .END.
13 RCL 01
14 X>Y?
15 GTO 09
16 RDN
17 +
18 STO 02
19 R↑
20 -
21 STO 04
22 RCL 02
23 XEQ 09
24 RCL 02

25*LBL 01
26 1
27 -
28 /
29 LASTX
30 DSE 04
31 GTO 01
32 FC? 01
33 GTO 03
34 *
35 PI
36 /
37 RCL 01
38 LASTX
39 *
40 SIN
41 *
42 1/X
43 CHS
44 GTO 08
45*LBL 09
46 1
47 -
48 STO 03
49 LN
50 LASTX
51 *
52 LASTX
53 -
54 LASTX
55 12
56 *
57 1/X
58 +
59 E↑X
60 RCL 03
61 PI
62 *
63 2
64 *
65 SQRT
66 *
67 RTN

68*LBL 00
69 SF 01
70 CHS
71 RTN

72*LBL 03
73 RDN
74 GTO 08
```


Unterprogramm RUKUN: Differentialgleichungssystem nach Runge-Kutta

Gegeben seien N (gekoppelte) Differentialgleichungen erster Ordnung:

$$\frac{dy_i}{dx} = y_i' = f_i(x, y_1, \dots, y_N) \quad i = 1 \dots N$$

mit den N Anfangsbuchstaben

$$y_i(x(\emptyset)) = y_i^{(\emptyset)}.$$

Die Lösungen $y_i(x)$ werden - ausgehend von $x(\emptyset)$ und $y_i^{(\emptyset)}$ - punktweise durch Anwendung des Verfahrens von Runge-Kutta berechnet: zu einem kleinen Zuwachs h in x wird in vier Stufen ein kleiner Zuwachs q_i in y_i ermittelt. Der Index i läuft jeweils von 1 bis N.

1. Stufe: $x^I = x(\emptyset); \quad y_i^I = y_i^{(\emptyset)}$

$$k_i^I = \frac{h}{2} * f_i(x^I, y_1^I, \dots, y_N^I)$$

2. Stufe: $x^{II} = x(\emptyset) + \frac{h}{2}; \quad y_i^{II} = y_i^{(\emptyset)} + k_i^I$

$$k_i^{II} = \frac{h}{2} * f_i(x^{II}, y_1^{II}, \dots, y_N^{II})$$

3. Stufe: $x^{III} = x^{II}; \quad y_i^{III} = y_i^{(\emptyset)} + k_i^{II}$

$$k_i^{III} = \frac{h}{2} * f_i(x^{III}, y_1^{III}, \dots, y_N^{III})$$

4. Stufe: $x^{IV} = x(\emptyset) + h; \quad y_i^{IV} = y_i^{(\emptyset)} + 2 * k_i^{III}$

$$k_i^{IV} = \frac{h}{2} * f_i(x^{IV}, y_1^{IV}, \dots, y_N^{IV})$$

Die neuen Punkte der Lösungsfunktionen ergeben sich mit

$$q_i = \frac{1}{3} * (k_i^I + 2 * k_i^{II} + 2 * k_i^{III} + k_i^{IV}) \quad \text{zu}$$

$$x^{(1)} = x(\emptyset) + h; \quad y_i^{(1)} = y_i^{(\emptyset)} + q_i$$

Für den nächsten Schritt sind die $x^{(1)}, y_i^{(1)}$ wieder als Anfangswerte $x(\emptyset), y_i^{(\emptyset)}$ zu interpretieren, woraus sich dann wieder die nächsten Punkte $x^{(1)}, y_i^{(1)}$ ergeben usf.

RUKUN muß als Unterprogramm von einem Hauptprogramm aufgerufen werden, das folgende Aufgaben erfüllen soll:

1. Ggfs. Zuweisung von $x(\emptyset), y_i^{(\emptyset)}$ und h zu den entsprechenden Registern (siehe Speicherplan)
2. Ggfs. Zuweisung des Funktionsnamen für die f_i (s.u.) an das Register $R_{\emptyset 7}$.

3. Ggfs. Erzeugung einer Schleife, die nur nach jedem m-ten Integrationsschritt Ergebnisse ausdrucken läßt.
4. Bildung der auszudruckenden Funktionswerte aus den $x^{(j)}$, $y_i^{(j)}$.
5. Anzeige bzw. Ausdruck der gewünschten Werte.
6. Beendigung des Programmes bei Erreichen eines Endwertes x_E .

Das Funktionssystem der f_i ($i = 1 \dots N$) muß vom Benutzer programmiert werden, beginnend mit LBL "FN", endend mit END, wobei "FN" ein beliebiger Funktionsname (max. 6 Alpha-Zeichen) ist.

Speicherplan:

$R_{\emptyset\emptyset}$: Adresse für dynamische Speicherzuweisung	$R_{\emptyset 5}$: Startadresse $S \geq S$
$R_{\emptyset 1}$: Schleifenzähler für N	$R_{\emptyset 6}$: Inkrement h
$R_{\emptyset 2}$: S-1	$R_{\emptyset 7}$: "FN"
$R_{\emptyset 3}$: x^I ; x^{II} ; x^{III} ; x^{IV}	$R_{\emptyset 8}$: $x^{(\emptyset)}$; $x^{(1)}$
$R_{\emptyset 4}$: N	
$R_S \dots R_{S-1+N}$: $y_1^{(\emptyset)} \dots y_N^{(\emptyset)}$; $y_1^{(1)} \dots y_N^{(1)}$	
$R_{S+N} \dots R_{S-1+2N}$: $y_1^I \dots y_N^I$	
$R_{S+2N} \dots R_{S-1+3N}$: $y_1^I \dots y_N^I$; \dots ; $y_1^{IV} \dots y_N^{IV}$	
$R_{S+3N} \dots R_{S-1+4N}$: $k_1^I \dots k_N^I$; \dots ; $k_1^{IV} \dots k_N^{IV}$	
$R_{S+4N} \dots R_{S-1+5N}$: $l_1^I \dots l_N^I$; \dots ; $l_1^{IV} \dots l_N^{IV}$ (Zwischensummen der k)	

Die Speicher $R_{\emptyset\emptyset}$ bis $R_{\emptyset 8}$ sind fest belegt, während die übrigen Speicher, in Anpassung an N und an die Komplexität des Funktionensystems der f_i dynamisch zugewiesen werden können, beginnend mit R_S und endend mit R_{S-1+5N} ; S wird in $R_{\emptyset 5}$ festgelegt.

Die Funktionen f_i im Unterprogramm "FN" sind mit den Argumentwerten aus den Speichern $R_{\emptyset 3}$ und $R_{S+2N} \dots R_{S-1+3N}$ zu bilden und dann nach $R_{S+N} \dots R_{S-1+2N}$ abzuspeichern.

Beispiel:

Im folgenden wird die Differentialgleichung $y'' = -y$ mit den Anfangsbedingungen $y(\emptyset) = \emptyset$; $y'(\emptyset) = 1$ zugrundegelegt, deren Lösung bekanntlich $y(x) = \sin x$ (x in Radian) lautet.

Durch $y_1 = y$ und $y_2 = y'$ wird hieraus ein System von 2 Differentialgleichungen 1. Ordnung:

$$\begin{aligned} y_1' &= y_2 & y_1(\emptyset) &= \emptyset \\ y_2' &= -y_1 & y_2(\emptyset) &= 1 \end{aligned}$$

Der Einfachheit halber soll $R_{\emptyset 9}$ das erste Register des dynamischen Speicherbereiches sein, also $S = 9$. Wegen $N = 2$ lautet dann das Funktionsunterprogramm für f_1 und f_2 mit dem Namen ABLEIT:

LBL ^T ABLEIT	CHS
RCL 14	STO 12
STO 11	END
RCL 13	

Das Hauptprogramm HAUPT geht von folgender Speicherbelegung aus:

R ₀₄ : N = 2	R ₀₅ : S = 9	R ₀₆ : h = 0.01
R ₀₇ : ABLEIT	R ₀₈ : x (0) = 0	R ₀₉ : y ₁ (0) = 0
R ₁₀ : Y ₂ (0) = 1		

Weiterhin wird von HAUPT verlangt, daß nach jeweils 10 Integrationsschritten x und y₁ zur Anzeige gebracht werden. Für diesen Schrittzähler wird der hinter dem hier definierten dynamischen Speicherbereich liegende Speicher R₁₉ benutzt.

Das Hauptprogramm hat dann folgende Gestalt:

LBL ^T HAUPT	ARCL 09
10	AVIEW
STO 19	LBL 01
CLA	XEQ ^T RUKUN
FIX 1	DSE 19
ARCL 08	GTO 01
10	GTO ^T HAUPT
FIX 6	END

(0 bedeutet SPACE)

Nach der Speicherfestlegung durch XEQ SIZE 020 wird das Programm durch XEQ HAUPT gestartet. Ein Vergleich der - nach jeweils etwa 170 s Rechenzeit - erhaltenen Werte von x und y₁(x) mit den im RAD-Modus direkt errechneten Werten von y₁ = sin x zeigt, daß das Programm fehlerlos arbeitet.

Die folgenden Seiten enthalten die Struktogramme von RUKUN und seinen Unterprogrammen nach Nassi-Shneiderman sowie das vollständige Listing.

K. W. Hoenow

Unterprogramm RUKUN

Speicherbelegung bei Programmstart:

$R_{\beta 4}: N \quad R_{\beta 6}: h \quad R_{\beta 8}: x^{(\beta)}$
 $R_{\beta 5}: S \quad R_{\beta 7}: \text{"FN"}$ $R_{S-1+i}: y_i^{(\beta)} \quad i=1 \dots N$

$S-1 \rightarrow R_{\beta 2}$

$x = x^{(\beta)} \rightarrow R_{\beta 3}$

Flag $\beta \beta$ setzen

UP $\beta 2$

Flag $\beta \beta$ löschen

UP "FN"

Flags $\beta 2$ und $\beta 4$ setzen

UP $\beta 5$

Flag $\beta 2$ löschen

$x = x^{(\beta)} + \frac{h}{2} \rightarrow R_{\beta 3}$

Flag $\beta 1$ setzen

UP $\beta 2$

UP "FN"

Flag $\beta 3$ setzen

UP $\beta 5$

UP $\beta 2$

UP "FN"

UP $\beta 5$

$x = x^{(\beta)} + h \rightarrow R_{\beta 3}; \quad x^{(1)} = x \rightarrow R_{\beta 8}$

Flag $\beta 1$ löschen

UP $\beta 2$

UP "FN"

Flags $\beta 3$ und $\beta 4$ löschen

UP $\beta 5$

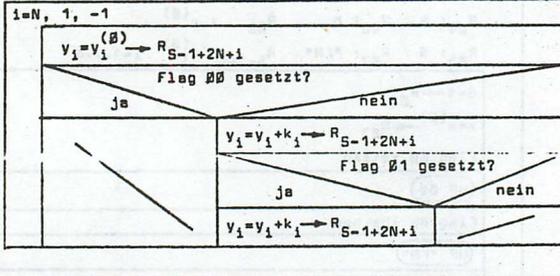
Speicherbelegung bei Programmende:

wie bei Start, jedoch $R_{\beta 8}: x^{(1)}$
 $R_{S-1+i}: y_i^{(1)} \quad i=1 \dots N$

Unterprogramm B2

Speicherbelegung bei Programmstart:

$R_{\beta 4} : N$
 $R_{S-1+i} : v_i^{(\beta)} \quad i=1 \dots N$
 $R_{S-1+3N+i} : k_i \quad i=1 \dots N$ falls Flag $\beta \beta$ gesetzt



Unterprogramm "FN"

Speicherbelegung bei Programmstart:

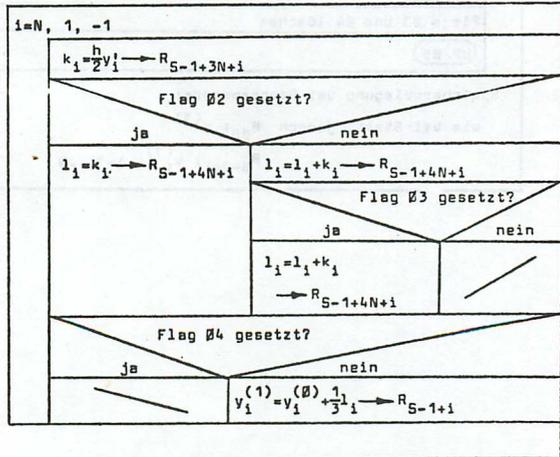
$R_{\beta 3} : x \quad R_{S-1+2N+i} : v_i \quad i=1 \dots N$

$v_i' = f_1(x, v_1, \dots, v_N) \rightarrow R_{S-1+N+i} \quad i=1 \dots N$

Unterprogramm B5

Speicherbelegung bei Programmstart:

$R_{\beta 6} : h$
 $R_{S-1+i} : v_i^{(\beta)}$ falls Flag $\beta 4$ gesetzt
 $R_{S-1+N+i} : v_i'$
 $R_{S-1+4N+i} : l_i$ falls Flag $\beta 2$ gesetzt
 $i=1 \dots N$



27 Register

```

LBLTRUKUN
RCL 05
1
-
005 STO 02
RCL 08
STO 03
SF 00
XEQ 02
010 CF 00
XEQ IND 07
SF 02
SF 04
XEQ 05
015 CF 02
RCL 08
RCL 06
2
/
020 +
STO 03
SF 01
XEQ 02
XEQ IND 07
025 SF 03
XEQ 02
XEQ IND 07
XEQ 05
030 RCL 08
RCL 06
+
STO 03
STO 08
035 CF 01
XEQ 02
XEQ IND 07
CF 03
CF 04
040 XEQ 05
RTN
-----
LBL 02      Unterprogramm
RCL 04
STO 01
045 LBL 03
RCL 01
RCL 02
+
STO 00
050 RCL IND 00
RCL 04
2
=
ST+ 00
055 RDN
STO IND 00
FS? 00
GTO 04
RCL 04
060 ST+ 00

```

```

RCL IND 00
RCL 04
ST- 00
RDN
065 ST+ IND 00
FC? 01
ST+ IND 00
LBL 04
DSE 01
070 GTO 03
-----
LBL 05      Unterprogramm
RCL 04
STO 01
075 LBL 06
RCL 01
RCL 02
+
RCL 04
080 +
STO 00
RCL IND 00
RCL 06
=
085 /
/
RCL 04
2
=
090 ST+ 00
RDN
STO IND 00
RCL 04
ST+ 00
095 RDN
FS? 02
GTO 07
ST+ IND 00
FS? 03
100 ST+ IND 00
GTO 08
LBL 07
STO IND 00
LBL 08
105 FS? 04
GTO 09
RCL IND 00
3
/
110 RCL 01
RCL 02
+
STO 00
RDN
115 ST+ IND 00
LBL 09
DSE 01
GTO 06
END

```

Auswertung aussagenlogischer Ausdrücke $A=A(p_0, \dots, p_n)$ ($n \leq 9$)

J.Schu

A ist unter einem globalen Label zu speichern. Beim Aufruf von A steht p_i in Register i. Die Programmierung erfolgt nach den Regeln der UPN. Der Stack kann wie bei arithmetischen Ausdrücken verwendet werden. Folgende Funktionen sind vorhanden:

Name	Zchn	Register	X \rightarrow LASTX	Bemerkung
AND		X,Y	ja	
EQU	\leftrightarrow	X,Y	ja	entspricht * (Mult.)
IMP	\Rightarrow	X,Y	ja	$Y \Rightarrow X$, Reihenfolge notwendig!
NOT	\neg	X	nein	entspricht CHS
OR	\vee	X,Y	ja	

Beispiel: $(p_2 \Rightarrow p_1) \vee (p_1 \wedge p_0)$ ($n=2$)

RCL O2 RCL O1 XEQ'IMP RCL O1 RCL O0 XEQ'AND XEQ'OR

Das Programm wird mit XEQ'ASL gestartet. Auf AUSSAGE? ist das Label von A mit R/S einzugeben (Rechner hält im ALPHA-Modus). Auf VAR-ANZ? ist die Anzahl der Variablen ($=n+1$) mit R/S einzugeben. Danach wird zeilenweise die Tabelle der Wahrheitswerte ausgegeben*): Die Wahrheitswerte der p_i und - durch eine Leerstelle getrennt - der Wahrheitswert von A. Die Wahrheitswerte sind W und F. In der ersten Zeile haben alle p_i den Wert W, in der letzten F. Der Wert von p_0 wechselt in jeder Zeile, der von p_1 in jeder zweiten, der von p_2 in jeder vierten etc. Die Tabelle hat 2^{n+1} Zeilen. Die Ausgabe wird mit BEEP beendet.

Zum Programm: Intern werden die Wahrheitswerte durch +1 und -1 dargestellt. Der Wahrheitswert von p_i wechselt in Zeile k (d.h. $p_i \rightarrow -p_i$), wenn gilt:

$$0 = r_{ik} := (2^{n+1} + 1 - k) \bmod 2^i$$

(dies gilt auch für die erste Zeile, da die p_i mit -1 vorbesetzt werden); dazu ist äquivalent:

$$-1 \leftarrow -r_{ik}$$

In dieser Form erfolgt die Abfrage im Programm.

Hinweis: Ist man sich nicht sicher über die Wirkung einer Funktion, so kann man sich deren Tabelle ausgeben lassen, etwa: $p_0 \Rightarrow p_1$, also: RCL O0 RCL O1 XEQ'IMP.

+) mit TONE 8

Listing

Symbole: o,l,?,j,n = Schleife, Abbruch, Abfrage, ja, nein

```

01 LBL'ASL
   'AUSGABE?
AON
04 PROMPT
AOFF
ASTO 13
   'VAR-ANZ?
08 PROMPT
   2 X<Y Y#X STO 10
LASTX 1 -
   1 E3 / STO 12
   -1 STO 00 ... STO 09
30 o LBL 10
   CLA
34 o RCL 12 STO 11
   o LBL 11
   RCL 10
   2
   RCL 11 INT
   Y#X
   MOD CHS
   -1
   ? X<Y?
   j ST# IND 11
   RCL IND 11
   ? X>0?
   j 'N
   ? X<0?
   j 'F
   i ISG 11
51 GTO 11
52 XBO IND 13
   ? X>0?
   j 'N
   ? X<0?
   j 'F
   i TONE 8 AVIEW
60 DST 10
61 GTO 10
STOP

```

```

62 LBL'AND
   #
   ? X<0?
55 j RTN
   n CLX LASTX
68 RTN
69 LBL'ECU
   #
71 RTN
72 LBL'IMP
   #
   ? X>0?
75 j RTN
   n CLX LASTX
78 RTN
79 LBL'NCT
   CHS
81 RTN
82 LBL'OR
   # CHS
   ? X>0?
86 j RTN
   n CLX LASTX
99 END

```

PRGM: 165 Byte (?)
REG: 14

$R_{00} \dots R_{09} : \pm 1 (=p_i)$
 $R_{10} : j=2^{n+1}, \dots, 1 (j:=2^{n+1}+1-k)$
 $R_{11} : i=0, \dots, n$
 $R_{12} : n/10000$ (Schleifenkontrolle für i)
 $R_{13} : \text{Aussagen-Label}$

Diskrete Fourier-Transformation

Programmzeilen: 249 444 Bytes 64 Register

Datenspeicher: (K+1).2+T+6

System Konfiguration: 1 Memorymodul, Kartenleser (Printer)

Das Programm berechnet aus einer Anzahl von Abtastwerten einer harmonischen Funktion die Frequenzkomponenten. Die Anzahl der Abtastwerte ist nur durch den Speicherraum begrenzt, die Anzahl der Frequenzkomponenten ist beliebig. Die Abtastwerte werden am Beginn fortlaufend eingegeben, die Durchführung der Transformation erfolgt dann automatisch. Das Resultat kann in rechtwinkli-ger oder polarer Form ausgegeben werden. Bei polarer Ausgabe können die Oberwellen in % der Grundwelle ausgegeben werden. Das Programm ist zur Verwendung mit oder ohne Drucker geeignet.

Wenn eine Berechnung abgeschlossen ist, können weitere Frequenzfolgen ohne neue Eingabe angehängt werden. Für spätere Verwendung können die Eingabewerte auf Magnetkarte gespeichert werden.

Das Eingabeformat kann durch Änderung des Programmschrittes 29, das Ausgabeformat durch Änderung der Schritte 158, 159, 161 bzw. 168 leicht den Erfordernissen der Genauigkeit angepaßt werden.

Die Anzahl der Abtastwerte soll mindestens doppelt so groß sein wie die Ordnungszahl der höchsten zu berechnenden Frequenz.

Grenzen und Einschränkungen:

Wenn die Abtastwerte von einer Magnetkarte eingelesen werden, muß die Anzahl der auf der Karte gespeicherten Abtastwerte und die Eingabe T=... übereinstimmen.

Der Abstand der Abtastwerte muß konstant sein.

Wenn kein Drucker angeschlossen ist, müssen aufeinanderfolgende Ausgabewerte mit R/S abgerufen werden.

Verwendete Formeln:

T = Periodendauer als Summe der Abtastwerte t

K = Anzahl der zu berechnenden Frequenzen k

k1= erste Oberwelle, welche berechnet wird

$$f(t) = \frac{a_0}{2} + \sum_1^k a_k \cdot \cos k \cdot 2 \cdot \pi \cdot t / T + b_k \cdot \sin k \cdot 2 \cdot \pi \cdot t / T =$$

$$\frac{a_0}{2} + \sum_1^k c_k \cdot e^{-i \cdot p_k}$$

$$a_k = \frac{2}{T} \cdot \sum_1^T f(t) \cdot \cos \frac{k \cdot 2 \cdot \pi \cdot t}{T} \quad b_k = \frac{2}{T} \cdot \sum_1^T f(t) \cdot \sin \frac{k \cdot 2 \cdot \pi \cdot t}{T}$$

Das vorliegende Programm ist mit keiner Verpflichtung oder Garantie irgendwelcher Art verbunden.

Der Verfasser übernimmt keine Verantwortung und keine wie immer geartete Haftung, die auf irgendeine Art aus der Benutzung dieses Programmes entsteht.

Vervielfältigung und Weiterverbreitung obigen Programmes nur mit Zustimmung des Verfassers.

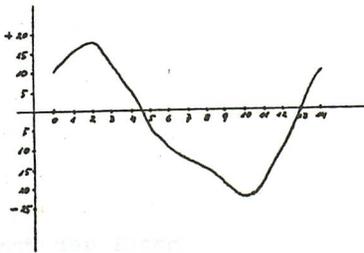
Beispiel: Eine periodische Funktion hat den unten abgebildeten Verlauf. Es sind die Harmonischen bis zur 8. Oberwelle zu bestimmen. Es stehen 30 Datenregister zur Verfügung. Bei einer Transformation können

$$K = (30 - 6 - 14) : 2 - 1 = 4 \text{ Frequenzen}$$

berechnet werden. Die Transformation muß in zwei Rechengängen durchgeführt werden.

Das Ergebnis soll rechtwinklig und polar ausgegeben werden. Bei der polaren Ausgabe sollen die Oberwellen in % der Grundschwingung berechnet werden.

$$T = 14, K = 4, K_1 = 0 (=5)$$



t	f(t)
1	16
2	18
3	12
4	5
5	-4,5
6	-10
7	-13
8	-15
9	-20
10	-22
11	-18
12	-10
13	0
14	10

Beispiel: Lösung

Eingabe	Taste		Kommentar
	A	T=14	
14	R/S	1= 16,000	Eingabe von T
16	R/S	2= 18,000	
18	R/S	3= 12,000	Eingabe der Abtastwerte
		4= 5,000	
		5= -4,500	
		6= -10,000	
		7= -13,000	
		8= -15,000	
		9= -20,000	
		10= -22,000	
		11= -18,000	
		12= -10,000	
		13= 0,000	
		14= 10,000	
4	R/S	K=4	Anzahl der Frequenzen
0	R/S	K1=0	1.Frequenz
	D	END DFT	Berechnung
		K : b : a	Ausgabe rechtwinklig
		0: 0,000: -7,357	
		1: 14,099: 11,807	
		2: 3,155: 2,255	
		3: -0,745: -0,574	
		4: 0,422: -0,067	
	E	K : PHI : c	Ausgabe polar
		0: 3,142: 7,357	
		1: 0,874: 18,390	nach K=1 mit R/S stoppen
	f e	K : PHI : c	
18,390	R/S	0: 3,142: 7,357: 20,0%	Eingabe des Wertes für
		1: 0,874: 18,390: 100,0%	100%
		2: 0,950: 3,878: 21,1%	
		3: 4,056: 0,941: 5,1%	
		4: 1,729: 0,428: 2,3%	
3	C	K=3	Start für nächste Frequenz-
5		K1=5	gruppe
		END DFT	
	D	K : b : a	
		5: -0,180: 0,017	
		6: -0,296: -0,009	
		7: 0,000: 0,500	
		8: 0,296: -0,009	
	f e	K : PHI : c	
18,390	R/S	5: 4,004: 0,181: 1,0%	
		6: 4,683: 0,297: 1,6%	
		7: 0,000: 0,500: 2,7%	
		8: 1,600: 0,297: 1,6%	

Programmablauf:

Nr.	Anweisung	Eingabe	Taste	Anzeige
1	Programmstart		A	"T="
2	Anzahl der Abtastwerte gleichen Abstands	T	R/S	"1 "
3	Amplitude des 1. Abtastwertes	f(t)	R/S	"2 "
	Schritt 3 wiederholen für alle Abtastwerte, am Schluß			"K="
4	Anzahl der zu berechnenden Frequenzen	K	R/S	"K1="
5	Ordnungszahl der 1. zu berechnenden Frequenz	K1	R/S	"1"
	während der Rechenzeit wird die fortschreitende Abarbeitung der Abtastwerte im Display angezeigt. Ende der Berechnung			BEEP "END DFT"
6	Wenn die Abtastwerte für später auf Datenkarte gespeichert werden	f b		"WDAT INP"
7	Wenn Programm mit Daten von Karte gestartet werden soll	f a		"T="
	Anzahl der Abtastwerte	T	R/S	"RDAT INP"
	Datenkarte einlesen			"K="
	weiter bei Schritt 4 Wenn T nicht mit der Anzahl der gespeicherten Abtastwerte übereinstimmt, wird Schritt 7 wiederholt.			
8	Korrektur falscher Eingaben:		B	ignor.
	richtige Werte eingeben	t	ENTER	
		f(t)	R/S	
	weiter bei Schritt 3 oder, wenn die restlichen Werte für t bereits eingegeben waren			
	weiter bei Schritt 4		C	"K="
9	Ausgabe der Koeffizienten a, b,		D	K:b:a
10	Ausgabe der Koeffizienten c, φ,		E	K:Phi:c
11	Ausgabe von c, φ, mit % Anzeige: Wert von c1 nach Schritt 10 berechnen, Prgr.stoppen		R/S f e	"c1="
	c1 für Grundwelle eingeben	c1	R/S	K:Phi:c:%
12	für Berechnung einer weiteren Folge von Frequenzen		C	"K="
	weiter bei Schritt 4 für neue Rechnung nach Schritt 1 oder 7			

Status: Size (K+1).2+T+6, Prgr.Reg. 64, User Mode ON

- Flags: 00 Pointer t setzen
 01 Eingabe von Karte, Resultat polar
 02 Ausgabe mit %
 03 negativ, ao %

- Tastenzuordnung: FOUR : 11
 Register: 00 Pointer t, k
 01 Pointer k
 02 K
 03 k1
 04 t, c1/100
 05 T
 06 t Eingabewerte
 xx "
 xx a Koeffizienten
 xx b "
 xx a "
 xx b "

PROGRAM LISTING

MA 102 Page 5 of

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line Zeile Ligne Linea	Keystrokes Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti	Line Zeile Ligne Linea	Key pressed Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti
01	*LBL "FOUR"	Programmstart	55	STO IND 01	Speicherbereich
02	CF 01		56	ISG 01	k löschen
03	*LBL 10		57	GTO 01	
04	SF 21		58	1	
05	RAD		59	STO 04	
06	CF 29		60	CF 21	
07	FIX 0		61	*LBL 02	
08	"T"	T speichern	62	CLA	Schleife DFT
09	XEQ 09		63	ARCL 04	
10	STO 05		64	AVIEW	t in Anzeige
11	FS?C 01	Rücksprung wenn	65	XEQ 00	Pointer setzen k
12	RTN	Eingabe von Karte	66	-1	
13	1		67	STO 02	Zähler für k
14	STO 04	Zähler für t1	68	*LBL 03	Schleife
15	FC? 55		69	1	
16	CF 21		70	ST+ 02	K+1
17	*LBL B	Start Dateneingabe	71	RCL 02	
18	" "		72	RCL 03	
19	FIX 0		73	+	
20	RCL 04	ti anzeigen	74	2	
21	ARCL X		75	*	k.2. .t/T
22	PROMPT	f(ti) eingeben	76	PI	
23	" "		77	*	
24	ARCL Y		78	RCL 04	
25	"t"		79	*	
26	FIX 3	Eingabe Format	80	RCL 05	
27	ARCL X		81	/	
28	AVIEW	Eingabe drucken	82	RCL IND 00	f(t)
29	X<Y		83	P-R	
30	STO 04		84	ST+ IND 01	f(t).cos
31	5		85	X<Y	
32	+		86	ISG 01	
33	X<Y		87	ST+ IND 01	f(t).sin
34	STO IND Y		88	ISG 01	nächstes k
35	1	ti+1	89	GTO 03	
36	ST+ 04		90	1	t+1
37	RCL 05		91	ST+ 04	
38	RCL 04	alle Stützwerte	92	ISG 00	nächstes t
39	X<Y?	eingegeben?	93	GTO 02	
40	GTO B		94	XEQ 00	Pointer setzen k
41	*LBL 11		95	2	
42	SF 21		96	RCL 05	2/T
43	*LBL C	Start Fouriertransf.	97	/	
44	FIX 0		98	*LBL 04	
45	"K"	K speichern	99	ST+ IND 01	Koeff. .2/T
46	XEQ 09		100	ISG 01	
47	STO 02		101	GTO 04	
48	"K1"	K1 speichern	102	SF 21	
49	XEQ 09		103	BEEP	
50	STO 03		104	"END DFT"	
51	SF 00		105	AVIEW	
52	XEQ 00	Pointer setzen k, t	106	PROMPT	
53	CLX		107	*LBL 00	
54	*LBL 01		108	RCL 05	

Please use paper glue to attach listings. Adhesive tape may affect print!
Bitte mit Papierem einkleben. Klebefilme können Druck bleichen!

S.V.P. utilisez de la colle à papier pour fixer les listings. Les rubans adhésifs peuvent altérer l'impression.
Per favore usare la colla per fissare i listati. Il nastro adesivo può alterare lo stampato!

PROGRAM LISTING

MA 102 Page 6 of

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

ke c nc na	Keystrokes Tastenseq Touches Tasti	Comments Kommentar Commentaires Commenti	Line Zeile Ligne Linea	Key pressed Tastenseq Touches Tasti	Comments Kommentar Commentaires Commenti
109	S			163 "t: "	
110	+			164 ARCL X	a(c) in Anzeige
111	1 E3			165 FC? 02	
112	/			166 GT0 07	
113	6			167 "t: "	
114	+			168 FIX 1	% Format
115	FS?C 00			169 R†	
116	ST0 00	Pointer setzen t		170 RND	% in Anzeige
117	RCL 02			171 ARCL X	
118	1			172 "+%"	
119	+			173*LBL 07	Resultat anzeigen
120	500			174 AVIEW	
121	/			175 1	ki+1
122	+			176 ST+ 00	
123	RCL 05			177 ISG 01	nächstes k
124	+			178 GT0 06	
125	ST0 01	Pointer setzen k		179 CF 01	
126	RTN			180 CF 02	
127*LBL e		Resultatausgabe		181 BEEP	
128	CLX	mit %		182 ADV	
129	"cl="	Wert für 100%		183 RTN	
130	PROMPT	speichern		184*LBL 08	Resultat polar
131	100			185 X<>Y	
132	/			186 R-P	
133	ST0 04			187 X<>Y	
134	SF 02			188 X<0?	
135*LBL E		Resultatausgabe		189 SF 03	
136	"K : PHI : c"	polar		190 2	
137	AVIEW			191 PI	
138	SF 01			192 *	
139	GT0 05			193 FC?C 03	$\varphi < 0: +2.5\%$
140*LBL D		Resultatausgabe		194 CLX	
141	"K : b : a"	rechtwinklig		195 +	
142	AVIEW			196 FC? 02	
143	CF 01			197 RTN	
144*LBL 05		Pointer setzen k		198 X<>Y	
145	XEQ 00			199 ENTER†	
146	RCL 03			200 ENTER†	
147	ST0 00	kl- ROO		201 RCL 00	
148*LBL 06		Schleife		202 X=0?	k=0: ao/2
149	FIX 0			203 SF 03	
150	CLA			204 RDN	
151	ARCL 00	ki in Anzeige		205 FS? 03	
152	"t: "			206 2	
153	RCL IND 01			207 FS?C 03	
154	ISC 01			208 /	
155	RCL IND 01			209 RCL 04	
156	FS? 01			210 /	
157	XEQ 08	Ausgabeformat		211 RDN	
158	FIX 3			212 X<>Y	
159	RND			213 RTN	
160	X<>Y			214*LBL 09	
161	RND			215 PROMPT	
162	ARCL Y	b(φ) in Anzeige		216 ARCL X	

Use paper glue to attach listings. Adhesive tape may affect print!
listings mit Papierleim einkleben. Klebefilme können Druck bleichen!

S.V.P. utiliser de la colle à papier pour fixer les listings. Les rubans adhésifs peuvent altérer l'impression!
Per favore usare la colla per fissare i listing. Il nastro adesivo può alterare lo stampato!

PROGRAM LISTING

MA 102

Page 7 of

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line Zeile Ligne Linea	Keystrokes Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti	Line Zeile Ligne Linea	Key pressed Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti
217	FS? 55		51		
218	PRA	Eingabe drucken			
219	RTH				
220	*LBL b				
221	SF 00		55		
222	XEQ 00	Pointer stzen t			
223	RCL 00				
224	1				
225	-				
226	CF 21				
227	*RDAT IMP*		60		
228	AVIEW	Eingabedaten auf			
229	RDTAX	Karte			
230	GTO 11				
231	*LBL a				
232	SF 01		65		
233	XEQ 10				
234	SF 00				
235	XEQ 00	Pointer setzen t			
236	RCL 05				
237	RCL 00				
238	1		70		
239	-				
240	CF 21				
241	*RDAT IMP*				
242	AVIEW	Eingabedaten lesen			
243	RDTAX		75		
244	RDN				
245	RCL 05				
246	X*Y?	T falsch: Eingabe			
247	GTO a	wiederholen			
248	GTO 11		80		
249	END				
	LBL*FOUR				
	END	444 BYTES			
35			85		
40			90		
45			95		
50			100		

Please use paper glue to attach listings. Adhesive tape may affect print!
Bitte Listings mit Papierkleim einkleben. Klebefolien können Druck bleichen!

S.V.P. utilisez de la colle à papier pour fixer les listings. Les rubans adhésifs peuvent altérer l'impression.
Per favore usare la colla per fissare i listing. Il nastro adesivo può alterare lo stampato!

Inverse Fourier-Transformation

Programmzeilen: 199 398 Bytes 57 Register

Datenspeicher: n.3+17

System Konfiguration: 1 Memorymodul, Kartenleser (Printer)

Das Programm berechnet aus einer Anzahl von Frequenzkomponenten die harmonische Funktion. Die Fourierkoeffizienten können in polarer oder rechtwinkliger Form vorliegen. Bei Verwendung des Printers kann die Summenfunktion auch als Diagramm dargestellt werden.

Das Programm stellt die Ergänzung zum Programm "Diskrete Fourier-Transformation" dar.

Das Programm erfordert nicht die Eingabe einer geschlossenen Folge von Frequenzen. Es brauchen nur signifikante Oberschwingungen eingegeben zu werden.

Das Ausgabeformat entspricht dem vor Abruf der Berechnung gespeicherten Format.

Bei Darstellung der Funktion durch den Printer:

Name: FT
X-Achse: Zeit
Y-Achse: $f(t)$

Grenzen und Einschränkungen:

Der Phasenwinkel ϕ muß im Winkelmaß Radiant eingegeben werden.

Verwendete Formeln:

T = Periodendauer (frei wählbar)

a_k, b_k, c_k, k = Fourierkoeffizienten der Ordnungszahl k

n = Anzahl der gegebenen Frequenzkomponenten

Das vorliegende Programm ist mit keiner Verpflichtung oder Garantie irgendwelcher Art verbunden.

Der Verfasser übernimmt keine Verantwortung und keine wie immer geartete Haftung, die auf irgendeine Art aus der Benutzung dieses Programmes entsteht.

Vervielfältigung und Weiterverbreitung obigen Programmes nur mit Zustimmung des Verfassers.

Programmablauf:

Nr.	Anweisung	Eingabe	Taste	Anzeige
1	Programmstart		F	'REC'
2	Wenn die Eingabe polar erf. soll		f c	'POL'
	Taste f c dient als Schalter		R/S	'a0='
3	Eingabe von a0	a0	R/S	'N='
4	Anzahl der gegebenen Frequ.	n	R/S	'K='
5	Ordnungszahl der 1. Frequenz	k	R/S	'b=' ('PHI=')
	Faktor bk (ϕk in RAD)	bk (ϕk)	R/S	'a=' ('c=')
	Faktor ak (ck)	ak (ck)	R/S	'K='
	Schritt 5 wiederholen für alle Frequenzkomponenten			BEEP 'END INPUT'
6	Korrektur einer falschen Eingabe		B	'K='
	falschen Wert k eingeben	k	R/S	'K='
	nach Schritt 5 und richtige Werte eingeben			
7	Wenn die Eingabewerte auf Karte gespeichert werden sollen		f b	'WDAT INP'
8	Programmstart mit Daten von Karte		f a	'REC'
	nach Schritt 2, 4			'RDAT INP'
	Datenkarte einlesen			BEEP 'END INPUT'
	Der Wert für n muß mit den Werten auf der Karte übereinstimmen, sonst wird Schritt 8 wiederholt.			
9	Berechnung von f(t) für einzelne Werte		C	'T='
10	Periodendauer eingeben	T	R/S	'T1='
11	t eingeben	t	R/S	t:f(t)
	Schritt 11 für beliebige T wiederholen			
12	Berechnung von f(t) für eine Folge t		D	'T='
13	Periodendauer eingeben	T	R/S	'T1='
14	ersten Wert für t eingeben	t1	R/S	'dT='
15	Schrittweite dt eingeben	dt	R/S	t:f(t)
	Wenn kein Drucker angeschlossen ist, müssen folgende Ergebnisse mit R/S abgerufen werden.			
16	graphische Darstellung der Funktion		E	'T='
17	Periodendauer eingeben	T	R/S	'NAME'
	Werte für Plotroutine eingeben			
	Name FT, Y=f(t), X=t			

Status: Size n.3+17, Prgr.Reg. 57, User Mode ON

Flags: 00 Koeff.polar

01 Input von Karte, Einzelschritt

02 Plotroutine

Tastenzuordnung: INFOUR : 21

Register: 00 Plotregister

08 t

11 "

10 dt

12 T

13 (n.3+16)/EEX3+17

14 Pointer

15 (R13)-2, f(t)

ab R15 werden die Register auf Datenkarte aufgezeichnet

16 a0

17 k

18 bk (k)

19 ak (ck)

20 k

Speicherlänge variabel nach Anzahl der Glieder

"

PROGRAM LISTING

MA 103/4 Page of

PROGRAMMAUFLISTUNG LISTAGE DU PROGRAMME LISTATO DI PROGRAMMA

Line Zeile Linea	Keystrokes Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti	Line Zeile Linea	Key pressed Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti
01	LBL "INFOUR"	<i>Programmstart</i>	55	XEQ 02	
02	CF 01		56	STO IND 14	
03	LBL 09		57	ISG 14	<i>Eingabe beendet?</i>
04	SF 21		58	GTO 01	
05	RAD		59	LBL 08	
06	CF 29		60	SF 21	
07	SF 00		61	BEEP	
08	LBL c		62	"END INP"	
09	FS? 00	<i>Koeffizienten polar oder</i>	63	AVIEW	
10	"REC"	<i>rechtwinklig?</i>	64	RTN	
11	FS?C 00		65	LBL 02	
12	GTO 06		66	PROMPT	<i>Eingabe anfordern</i>
13	SF 00		67	ARCL X	
14	"POL"		68	FS? 55	
15	LBL 06		69	PRA	
16	AVIEW		70	RTN	
17	FIX 3		71	LBL 8	<i>Eingabekorrektur</i>
18	"a="		72	RCL 13	<i>Pointer setzen</i>
19	FC? 01		73	STO 14	
20	XEQ 02	<i>a₀ speichern</i>	74	"K="	
21	FC? 01		75	PROMPT	
22	STO 16		76	FIX 0	
23	"N="		77	ARCL X	
24	XEQ 02	<i>n eingeben</i>	78	"I KORR."	
25	3		79	FS? 55	
26	*	<i>Pointer berechnen</i>	80	PRA	
27	16		81	LBL 05	
28	+		82	RCL IND 14	<i>K suchen</i>
29	RCL IND X	<i>Speicherbereich prüfen</i>	83	X=Y?	
30	RDN		84	GTO 01	
31	1 E3		85	RDN	
32	/		86	ISG 14	
33	17		87	ISG 14	
34	+		88	ISG 14	
35	STO 13		89	GTO 05	
36	STO 14	<i>Pointer setzen</i>	90	GTO J	
37	FS?C 01	<i>Rückprung, wenn Eingabe</i>	91	LBL b	
38	RTN	<i>von Karte</i>	92	RCL 13	<i>Eingabedaten auf</i>
39	LBL 01		93	2	<i>Karte schreiben</i>
40	FIX 0		94	-	
41	"K="	<i>K speichern</i>	95	STO 15	
42	XEQ 02		96	CF 21	
43	STO IND 14		97	"NDAT INP"	
44	ISG 14		98	AVIEW	
45	FIX 3		99	NDTAX	
46	"b="		100	GTO 08	
47	FS? 00	<i>b_n (P_n) speichern</i>	101	LBL a	
48	"PHI="		102	SF 01	<i>Eingabedaten von</i>
49	XEQ 02		103	XEQ 09	<i>Karte lesen</i>
50	STO IND 14		104	2	
51	ISG 14		105	-	
52	"a="	<i>a_n (C_n) speichern</i>	106	CF 21	
53	FS? 00		107	"NDAT INP"	
54	"c="		108	AVIEW	

Please use paper glue to attach listings. Adhesive tape may affect print!
Bitte Listings mit Papierkleb einkleben. Klebefilm: können Druck bleichen!

S.V.P. utiliser de la colle à papier pour fixer les listings. Les rubans adhésifs peuvent altérer l'impression.
Per favore usare la colla per fissare i listati. Il nastro adesivo può alterare lo stampato.

PROGRAM LISTING

MA 103/5 Page of

PROGRAMMAUFLISTUNG LISTAGE DU PROGRAMME LISTATO DI PROGRAMMA

Line Zeile -ligne linea	Keystrokes Tastensequenz Touches Tastii	Comments Kommentar Commentaires Commenti	Line Zeile -ligne linea	Key pressed Tastensequenz Touches Tastii	Comments Kommentar Commentaires Commenti
109	RDTAX			163 ISG 14	
110	RCL 15			164 RCL IND 14	b (P)
111	X=Y?	n prüfen		165 ISG 14	
112	GTO a			166 RCL IND 14	q (c)
113	GTO 00			167 FS? 00	
114	LBL E	Funktion graphisch darstellen		168 P-R	
115	SF 02			169 RCL Z	
116	XEQ 07			170 1	
117	CF 02			171 P-R	
118	BEEP			172 ST* Z	
119	RTN			173 RDN	
120	LBL C			174 ST* Z	
121	SF 01	einzelne Werte berechnen		175 RDN	
122	CF 02			176 +	
123	GTO 07			177 ST+ 15	a _n cos + b _n sin
124	LBL D	eine Folge von Werten berechnen		178 ISG 14	
125	CF 01			179 GTO 00	
126	CF 02			180 RCL 15	Ende der Berechnung
127	LBL 07			181 FS? 02	
128	SF 21			182 RTN	
129	"T="	T speichern		183 CLA	
130	XEQ 02			184 ARCL 05	
131	STO 12			185 "t="	
132	FS? 02			186 ARCL 15	
133	GTO "PRPLOT"			187 AVIEW	
134	LBL 04	t1 speichern		188 FS? 01	
135	"T1="			189 GTO 04	t + Δt
136	PROMPT			190 RCL 10	
137	STO 00			191 ST+ 00	
138	FS? 01			192 RCL 12	
139	GTO 03			193 RCL 00	
140	"ΔT="	ΔT speichern		194 X=Y?	nächstes t
141	XEQ 02			195 GTO 03	
142	STO 10			196 BEEP	
143	RCL 00			197 ADV	
144	LBL "FT"			198 RTN	
145	LBL 03	Berechnung von f(t)		199 END	
146	STO 00			LBL "INF0UR	
147	RCL 13	Pointer setzen		LBL "FT	
148	STO 14			END	398 BYTES
149	RCL 16				
150	2				
151	/		90		
152	STO 15				
153	LBL 00	Schleife			
154	RCL IND 14				
155	RCL 00				
156	*				
157	RCL 12		95		
158	/				
159	2				
160	*				
161	PI	k. z. dt. t/T			
162	*		100		

Use paper glue to attach listings. Adhesive tape may affect print!
Listings mit Papierleim einkleben. Klebefolien können Druck bleichen!

S.V.P. utiliser de la colle à papier pour fixer les listings. Les rubans adhésifs peuvent altérer l'impression!
Per favore usare la colla per fissare i listing. Il nastro adesivo può alterare lo stampato!

MSMIMA ermittelt außer Mittelwert und Standardabweichung einer Folge von Werten auch noch deren minimalen und maximalen Wert.

Start des Programmes durch XEQ MSMIMA. Es wird $\Sigma+(\Sigma-)$ zur Symbolisierung der Eingabe angezeigt. Diese erfolgt also wie gewohnt durch $\Sigma+$, Löschung eines versehentlich falsch eingegebenen Wertes durch $\Sigma-$.

Nach Eingabe aller Werte erfolgt durch Drücken von B nacheinander die Anzeige von Mittelwert, Standardabweichung, Minimum und Maximum, und zwar bei eingeschaltetem Drucker automatisch, sonst nach Drücken von R/S.

Minimum und Maximum sind in den Speichern R00 bzw. R01 verfügbar.

Klaus Werner Hoenow

LEL *MSMIMA	16 X<Y?	38 MEAN
END	17 STO 00	39 ARCL X
105 BYTES	18 RCL 01	40 AVIEW
	19 STO 03	41 FC? 55
	20 X<>Y	42 STOP
01 *LEL "MSM	21 X>Y?	43 "S="
IMA"	22 STO 01	44 SDEV
02 EPEG 04	23 0	45 ARCL X
03 CLS	24 X<>Y	46 AVIEW
04 9 E99	25 $\Sigma+$	47 FC? 55
05 STO 00	26 RTN	48 STOP
06 CHS	27 *LEL "	49 "MIN="
07 STO 01	28 RCL 02	50 ARCL 00
08 CLST	29 STO 00	51 AVIEW
09 " $\Sigma+$ < $\Sigma-$	30 RCL 03	52 FC? 55
>"	31 STO 01	53 STOP
10 SF 27	32 0	54 "MAX="
11 PROMPT	33 R↑	55 ARCL 01
12 *LBL A	34 $\Sigma-$	56 AVIEW
13 RCL 00	35 RTN	57 ΣREG 11
14 STO 02	36 *LBL B	58 END
15 X<>Y	37 "M="	

Programmpaket Matrizenrechnung

Dieses Programmpaket umfaßt mehrere Unterprogramme zur Matrizenrechnung, die so aufgebaut sind, daß eine oder mehrere Matrizen beliebiger Reihen- und Spaltenzahl (soweit es der Speicherplatz zuläßt) den Operationen des Matrizenkalküls unterzogen werden können. Zu diesem Zweck arbeiten die einzelnen Unterprogramme mit einer dynamischen Speicherbereichszuweisung, die für jede Matrix den benötigten Speicherbedarf folgendermaßen festlegt:

1. Speicherung

Gegeben sei die $m \times n$ -Matrix A mit den Elementen a_{ij} ($i=1, \dots, m$; $j=1, \dots, n$). Die Speicherung beginnt mit dem Register R_S , wobei S die Startadresse für die Matrix A ist, mit der sie von allen Unterprogrammen aufgerufen wird. Dabei muß $S \geq 11$ gewählt werden, da die Register R_{00} bis R_{10} für Zwischenspeicherungen innerhalb der einzelnen Unterprogramme benutzt werden.

Die Matrix A nimmt insgesamt $m \times n + 3$ Register in Anspruch, die folgendermaßen belegt sind:

```

RS           : "A"
RS+1         : m
RS+2         : n
RS+3...RS+2m×n : aij (i=1,...,m; j=1,...,n)

```

R_S enthält also den Namen der Matrix (max. 6 ALPHA-Zeichen), R_S+1 die Zeilenzahl und R_S+2 die Spaltenzahl. Die folgenden Register R_S+3 bis $R_S+2+m \times n$ enthalten die $m \times n$ Matrixelemente a_{ij} zeilenweise fortlaufend.

2. Adressierung

Die Registeradresse r eines Matrixelementes a_{ij} der $m \times n$ -Matrix A, deren Startadresse S ist, errechnet sich nach dem oben Gesagten zu

$$r = S + 2 + (i - 1) \times n + j$$

Diese Adressenberechnung wird von MTAIJ durchgeführt, das daher von fast allen Unterprogrammen dieses Programmpaketes benötigt wird. MTAIJ benutzt nur den Stack.

3. Zwischenspeicherung

Die Register R_{00} bis R_{10} bleiben für die verschiedenen Unterprogramme als Zwischenspeicher reserviert, wobei der Speicherplan, außer für MTINV, so aussieht:

```

RALPHA : Name der Ergebnismatrix
R00    : Konstante oder sonstige Hilfsgröße
R01    : Startadresse der 1. Matrix
R02    : Startadresse der 2. Matrix
R03    : Startadresse der Ergebnismatrix
R04    : Element der 1. Matrix
R05    : Element der Ergebnismatrix
R06    : 1. Schleifenindex
R07    : 2. Schleifenindex
R08    : 3. Schleifenindex
R09    : Re-Initialisierung des 2. Schleifenindex
R10    : Re-Initialisierung des 3. Schleifenindex

```

Für MTINV werden die Register R_{00} bis R_{08} benutzt.

4. Unterprogrammaufruf

Bei Aufruf der einzelnen Unterprogramme dieses Programmpaketes müssen gewisse Daten im Stack bzw. im ALPHA-Register vom Benutzer bereitgestellt werden, wie es jeweils im Kopf der einzelnen Listings angegeben ist. Das gilt jedoch nicht für MTAIJ, da die aufrufenden Unterprogramme dies bereits selbsttätig tun.

5. Programmbeschreibungen

5.1 MTAIJ

Dieses Unter-Unterprogramm wurde unter Punkt 2. erklärt.

5.2 MTEIN

Die Elemente a_{ij} der Matrix A werden im Dialogverkehr vom Benutzer reihenweise eingegeben.

5.3 MTAUS

Die Elemente a_{ij} der Matrix A werden mit Benennung reihenweise (CF O1) oder spaltenweise (SF O1) ausgegeben. Bei eingeschaltetem Drucker erfolgt die Ausgabe kontinuierlich; sonst wird die Ausgabe nach jedem Element gestoppt und muß mit R/S fortgesetzt werden.

5.4 MT111

Es wird eine $m \times m$ -Einheitsmatrix E erzeugt, bei der die Elemente der Hauptdiagonalen Eins sind, alle übrigen jedoch Null.

5.5 MTADD

Dieses Unterprogramm vereinigt die Matrizenaddition mit der Multiplikation mit einem konstanten Faktor:

$$C = A + k \times B$$

bzw. $c_{ij} = a_{ij} + k \times b_{ij}$ ($i=1, \dots, m; j=1, \dots, n$)
Für $k = 1$ ergibt sich die normale Matrizenaddition. Für $A = 0$ (Nullmatrix) erhält man die Multiplikation von B mit dem konstanten Faktor k. In diesem Fall muß bei Aufruf von MTADD im T-Register des Stack eine Null stehen; die Nullmatrix 0 braucht nicht gespeichert zu sein. Die Ergebnismatrix C darf sowohl einen von A und B verschiedenen Platz in den Registern einnehmen als auch eine von beiden Matrizen ersetzen.

5.6 MTSPU

Die Spur einer $m \times m$ -Matrix A ist definiert als die Summe ihrer Diagonalelemente:

$$\text{Spur}(A) = \sum_{i=1}^m a_{ii}$$

Sie wird in dem Register gespeichert, das bei Aufruf von MTSPU als Zahl im X-Register steht.

5.7 MTSHF

Dieses Unterprogramm dient zum Kopieren oder Verschieben von Matrizen innerhalb der Register. Beim Verschieben werden die Registerinhalte der "alten" Speicherplätze nicht gelöscht. Eine Verschiebung um weniger als $m \times n + 3$ Register ist erlaubt, wobei dann natürlich "alte" Elemente durch "neue" überschrieben werden.

5.8 MTTRA

Dieses Unterprogramm erzeugt die Transponierte einer $m \times n$ -Matrix A, nämlich die $n \times m$ -Matrix B, die aus A durch Ver-

tauschen von Reihen und Spalten hervorgeht:

$$b_{ji} = a_{ij} \quad (i=1, \dots, m; j=1, \dots, n)$$

5.9 MTMUL

Gegeben seien die $l \times m$ -Matrix A und die $m \times n$ -Matrix B (Spaltenzahl von A = Reihenzahl von B). MTMUL erzeugt die $l \times n$ -Produktmatrix $C = A \times B$ nach

$$c_{ik} = \sum_{j=1}^m a_{ij} b_{jk} \quad (i=1, \dots, l; j=1, \dots, m; k=1, \dots, n)$$

Im allgemeinen ist $A \times B \neq B \times A$!

5.10 MTINV

Dieses Unterprogramm erzeugt die Inverse $B = A^{-1}$ einer $m \times m$ -Matrix A. Es gilt

$$B \times A = A \times B = E$$

mit E als Einheitsmatrix (siehe Punkt 5.4). B ist dann natürlich auch eine $m \times m$ -Matrix.

Es wird m-mal das Stiefel-Verfahren angewandt mit a_{11} als Pivotelement; nach jeder Pivotisierung werden die neuen Matrixelemente in bestimmter Weise zyklisch vertauscht. Näheres zu diesem Verfahren z.B. bei G. Venz: "Lineare Algebra für programmierbare Taschenrechner", Oldenbourg-Verlag 1980, S.24 ff.

Außer der Bedingung, daß A quadratisch und nicht-singulär sein muß (siehe Lehrbücher über Matrizenrechnung), darf bei diesem Verfahren kein Element der Hauptdiagonalen Null sein. Wenn dieses Unterprogramm zusammen mit MTMUL zur Auflösung linearer Gleichungssysteme verwendet werden soll, läßt sich diese Zusatzbedingung meist durch Umstellung der Gleichungen erreichen.

Nach Abarbeitung von MTINV hat $B = A^{-1}$ die Elemente von A in den Speichern überschrieben!

6. Beispiel

Die 3×3 -Matrix

$$A = \begin{bmatrix} 4 & 7 & 2 \\ -6 & 1 & 0 \\ 2 & 7 & 3 \end{bmatrix}$$

soll folgenden Operationen unterzogen werden:

- 6.1 Einlesen von A (MTEIN)
- 6.2 Reihenweises Ausgeben von A (MTAUS)
- 6.3 Kopieren von A nach B (MTSHF)
- 6.4 Bilden von $C = A^{-1}$ (MTINV)
- 6.5 Spaltenweises Ausgeben von C (MTAUS)
- 6.6 Bilden von $E = C \times B$ (MTMUL)
- 6.7 Reihenweises Ausgeben von E (MTAUS)
- 6.8 Bilden und Ausgeben der Spur von E (MTSPU)

Außer den angegebenen Unterprogrammen wird noch MTAIJ benötigt.

Jede Matrix benötigt nach Punkt 1. $3 \times 3 + 3 = 12$ Register, wobei hier mit R_{11} begonnen werden soll. C nimmt nach Ausführung von MTINV den Platz von A ein, so daß für insgesamt 3 Matrizen, nämlich A, B und E, Speicherplatz reserviert werden muß. Es wird gewählt:

$$\begin{aligned} S(A) &= (S(C) = 11 \\ S(B) &= 23 \\ S(E) &= 35 \end{aligned}$$

Die Spur von E soll auf dem später nicht mehr benötigten Platz R_{11} gespeichert werden.

Um Programmspeicherplatz zu sparen, wird mit Hilfe der RSUB-Funktion des Kartenlesers eine Art "Overlay"-Technik durchgeführt: Es werden zunächst nur die beiden relativ kurzen und häufig gebrauchten Unterprogramme MTAIJ und MTAUS abgespeichert; anschließend folgt das aufrufende Hauptprogramm HAUPT, wie der folgende Katalog und das Listing zeigen:

```

CAT 1
LBL*MTAIJ
END
25 BYTES
LBL*MTAUS
END
100 BYTES
LBL*HAUPT
END
171 BYTES
.END.
05 BYTES

```

01*LBL "HAU	20 XEQ "MTS	39 XEQ "MTM
PT"	HF"	UL"
02 "MTEIN"	21 "MTINV"	40 CF 01
03 AVIEW	22 AVIEW	41 35
04 RSUB	23 RSUB	42 XEQ "MTA
05 "A"	24 "C"	US"
06 "3	25 11	43 "MTSPU"
07 ENTER↑	26 XEQ "MTI	44 AVIEW
08 ENTER↑	NV"	45 RSUB
09 11	27 SF 01	46 35
10 XEQ "MTE	28 11	47 ENTER↑
IN"	29 XEQ "MTA	48 11
11 CF 01	US"	49 XEQ "MTS
12 11	30 "MTMUL"	PU"
13 XEQ "MTA	31 AVIEW	50 FIX 2
US"	32 RSUB	51 ADV
14 "MTSHF"	33 "E"	52 "SPUR<E>
15 AVIEW	34 11	="
16 RSUB	35 ENTER↑	53 ARCL 11
17 11	36 23	54 AVIEW
18 ENTER↑	37 ENTER↑	55 END
19 23	38 35	

HAUPT erfüllt folgende Funktionen:

- Steuerung des Aufrufs der Unterprogramme gemäß 6.1 bis 6.8, wobei die für die Unterprogramme benötigten Parameter im Stack bzw. im ALPHA-Register bereitgestellt werden.
- Anforderung der benötigten Unterprogramme vom Benutzer mittels RSUB, wobei der Name des Unterprogrammes in der Anzeige erscheint. Der Benutzer hat dann lediglich die zugehörige Unterprogrammkarte in den Kartenleser einzuführen, und das Hauptprogramm setzt die Ausführung automatisch fort.

Auf diese Weise werden zusätzlich zu den für MTAIJ, MTAUS und HAUPT benötigten 43 Registern lediglich noch 27 Speicherregister (Unterprogramm MTINV ist das längste) benötigt, da durch den RSUB-Befehl immer das hinter HAUPT stehende Programm überschrieben

wird. Die restlichen Register stehen als Datenspeicher für die Aufnahme der Matrizen (auch größerer!) zur Verfügung. Da für dieses Beispiel 70 Programmregister und 47 Datenregister benötigt werden, ist ein Speichermodul erforderlich.

Der folgende, im NORM-Modus ausgedruckte Streifen zeigt den genauen Verlauf der Ein- und Ausgaben:

```

XEQ "HAUPT"
MTEIN
A1.1 ?      A1.1=4.0000
 4.000      A1.2=7.0000
  RU        A1.3=2.0000
  N
A1.2 ?      A2.1=-6.0000
 7.000      A2.2=1.0000
  RU        A2.3=0.0000
  N
A1.3 ?      A3.1=2.0000
 2.000      A3.2=7.0000
  RU        A3.3=3.0000
  N
A2.1 ?      MTSHF
-6.000      MTINV
  PU
  H
A2.2 ?      C1.1=0.0000
 1.000      C2.1=0.3000
  RU        C3.1=-0.0000
  N
A2.3 ?      C1.2=-0.1400
 0.000      C2.2=0.1000
  RU        C3.2=-0.2000
  N
A3.1 ?      C1.3=-0.0400
 2.000      C2.3=-0.2400
  RU        C3.3=0.9200
  N
A3.2 ?      MTMUL
 7.000      RU
  N
A3.3 ?      E1.1=1.0000
 3.000      E1.2=0.0000
  PU        E1.3=0.0000
  N
           E2.1=0.0000
           E2.2=1.0000
           E2.3=0.0000
           E3.1=0.0000
           E3.2=0.0000
           E3.3=1.0000
           MTSPU
           SPUR<E>=3.00

```

```

5.1 MTAIJ                26*LBL 02                27 STO 09                19*LBL 01
                        27 CLA                28*LBL 02                20 RCL 07
LBL*MTAIJ                28 ARCL IND                29 CLA                21 STO 09
END                        03                30 ARCL IND                22*LBL 02
25 BYTES                29 FIX 0                01                23 RCL 09
                        30 SF 29                31 RCL 09                24 RCL 06
r=S+2+(i-1)            31 RCL 06                32 INT                25 /
                        *n+j                33 RCL 06                26 INT
A:                        33 ARCL X                34 INT                27 LASTX
T:                        34 CF 29                35 FS? 01                28 /
Z: J                    35 RCL 09                36 X<>Y                29 INT
Y: 1                    36 INT                37 FIX 0                30 STO 05
X: S(A)                37 ARCL X                38 SF 29                31 RCL 09
                        38 "I?"                39 ARCL X                32 RCL 06
                        39 X<>Y                40 CF 29                33 RCL 03
01*LBL "MTA            40 RCL 03                41 X<>Y                34 XEQ "MTA
IJ"                    41 XEQ "MTA                42 ARCL X                IJ"
02 2                    IJ"                    43 "I="                35 RCL 05
03 +                    42 FIX 3                44 X<>Y                36 STO IND
04 X<>Y                43 PROMPT                45 RCL 01                Y
05 INT                    44 STO IND                46 XEQ "MTA                37 ISG 09
06 1                    Y                    IJ"                    38 GTO 02
07 -                    45 ISG 09                47 FIX 4                39 ISG 06
08 RCL IND                46 GTO 02                48 RCL IND                40 GTO 01
Y                        47 ISG 06                X                    41 PTN
09 *                    48 GTO 01                49 RND                    42 END
10 +                    49 RTN                50 ARCL X
11 +                    50 END                51 AVIEW
12 INT                    52 FC? 55                53 STOP                5.5 MTADD
13 RTN                    54 ISG 09                54 ISG 09                LBL*MTADD
14 END                    55 GTO 02                55 GTO 02                END
                        56 ADV                56 ADV                100 BYTES
                        57 ISG 06                57 ISG 06                UNTERPROGR.
                        58 GTO 01                58 GTO 01                MTAIJ ERFOR-
                        59 RTN                59 RTN                DERLICH !
                        60 END                60 END                C=A+k*B
5.2 MTEIN                LBL*MTAUS                LBL*MTADD
END                        END                        END
88 BYTES                100 BYTES                100 BYTES
UNTERPROGR.            UNTERPROGR.            UNTERPROGR.
MTAIJ ERFOR-            MTAIJ ERFOR-            MTAIJ ERFOR-
DERLICH !                DERLICH !                DERLICH !
REIHENWEISE            REIHENWEISE            REIHENWEISE
EINGABE DER            EINGABE DER            EINGABE DER
M,N-MATRIX A            M,N-MATRIX A            M,N-MATRIX A
A: NAME(A)                A:                        A: NAME(E)
T:                        T:                        T:
Z: n                    Z:                        Z:
Y: m                    Y:                        Y: m
X: S(A)                X: S(A)                X: S(E)
01*LBL "MTE            01*LBL "MTA            01*LBL "MT1
IN"                    US"                    11"
02 ASTO IND                02 STO 01                02 ASTO IND
X                        03 1                    X
03 STO 03                04 +                    03 STO 03
04 1                    05 RCL IND                04 1
05 +                    X                        05 +
06 X<>Y                06 1 E3                06 X<>Y
07 STO IND                07 /                    07 STO 03
Y                        08 1                    08 /
08 1 E3                09 +                    09 STO 03
09 /                    10 X<>Y                X
10 1                    11 1                    10 1
11 +                    12 +                    11 1
12 STO 06                13 X<>Y                12 STO 06
13 PDN                    14 RCL IND                13 PDN
14 1                    Y                        14 1
15 +                    15 1 E3                14 +
16 X<>Y                16 /                    15 +
17 STO IND                17 1                    16 X<>Y
Y                        18 +                    17 1
18 1 E3                19 FS? 01                18 +
19 /                    20 X<>Y                19 /
20 1                    21 STO 07                20 1
21 +                    22 X<>Y                21 +
22 STO 07                23 STO 06                22 STO 07
23*LBL 01                24 ADV                23 /
24 RCL 07                25*LBL 01                24 1
25 STO 09                26 RCL 07                25 +
                        26 RCL 07                25 STO 06
                        27 STO 07                26 +
                        28 STO 07                27 STO 06
                        29 STO 07                28 /
                        30 STO 07                29 /
                        31 STO 07                30 /
                        32 STO 07                31 /
                        33 STO 07                32 /
                        34 STO 07                33 /
                        35 STO 07                34 /
                        36 STO 07                35 /
                        37 STO 07                36 /
                        38 STO 07                37 /
                        39 STO 07                38 /
                        40 STO 07                39 /
                        41 STO 07                40 /
                        42 STO 07                41 /
                        43 STO 07                42 /
                        44 STO 07                43 /
                        45 STO 07                44 /
                        46 STO 07                45 /
                        47 STO 07                46 /
                        48 STO 07                47 /
                        49 STO 07                48 /
                        50 STO 07                49 /
                        51 STO 07                50 /
                        52 STO 07                51 /
                        53 STO 07                52 /
                        54 STO 07                53 /
                        55 STO 07                54 /
                        56 STO 07                55 /
                        57 STO 07                56 /
                        58 STO 07                57 /
                        59 STO 07                58 /
                        60 STO 07                59 /
                        61 STO 07                60 /
                        62 STO 07                61 /
                        63 STO 07                62 /
                        64 STO 07                63 /
                        65 STO 07                64 /
                        66 STO 07                65 /
                        67 STO 07                66 /
                        68 STO 07                67 /
                        69 STO 07                68 /
                        70 STO 07                69 /
                        71 STO 07                70 /
                        72 STO 07                71 /
                        73 STO 07                72 /
                        74 STO 07                73 /
                        75 STO 07                74 /
                        76 STO 07                75 /
                        77 STO 07                76 /
                        78 STO 07                77 /
                        79 STO 07                78 /
                        80 STO 07                79 /
                        81 STO 07                80 /
                        82 STO 07                81 /
                        83 STO 07                82 /
                        84 STO 07                83 /
                        85 STO 07                84 /
                        86 STO 07                85 /
                        87 STO 07                86 /
                        88 STO 07                87 /
                        89 STO 07                88 /
                        90 STO 07                89 /
                        91 STO 07                90 /
                        92 STO 07                91 /
                        93 STO 07                92 /
                        94 STO 07                93 /
                        95 STO 07                94 /
                        96 STO 07                95 /
                        97 STO 07                96 /
                        98 STO 07                97 /
                        99 STO 07                98 /
                        100 STO 07                99 /
                        101 STO 07                100 /
                        102 STO 07                101 /
                        103 STO 07                102 /
                        104 STO 07                103 /
                        105 STO 07                104 /
                        106 STO 07                105 /
                        107 STO 07                106 /
                        108 STO 07                107 /
                        109 STO 07                108 /
                        110 STO 07                109 /
                        111 STO 07                110 /
                        112 STO 07                111 /
                        113 STO 07                112 /
                        114 STO 07                113 /
                        115 STO 07                114 /
                        116 STO 07                115 /
                        117 STO 07                116 /
                        118 STO 07                117 /
                        119 STO 07                118 /
                        120 STO 07                119 /
                        121 STO 07                120 /
                        122 STO 07                121 /
                        123 STO 07                122 /
                        124 STO 07                123 /
                        125 STO 07                124 /
                        126 STO 07                125 /
                        127 STO 07                126 /
                        128 STO 07                127 /
                        129 STO 07                128 /
                        130 STO 07                129 /
                        131 STO 07                130 /
                        132 STO 07                131 /
                        133 STO 07                132 /
                        134 STO 07                133 /
                        135 STO 07                134 /
                        136 STO 07                135 /
                        137 STO 07                136 /
                        138 STO 07                137 /
                        139 STO 07                138 /
                        140 STO 07                139 /
                        141 STO 07                140 /
                        142 STO 07                141 /
                        143 STO 07                142 /
                        144 STO 07                143 /
                        145 STO 07                144 /
                        146 STO 07                145 /
                        147 STO 07                146 /
                        148 STO 07                147 /
                        149 STO 07                148 /
                        150 STO 07                149 /
                        151 STO 07                150 /
                        152 STO 07                151 /
                        153 STO 07                152 /
                        154 STO 07                153 /
                        155 STO 07                154 /
                        156 STO 07                155 /
                        157 STO 07                156 /
                        158 STO 07                157 /
                        159 STO 07                158 /
                        160 STO 07                159 /
                        161 STO 07                160 /
                        162 STO 07                161 /
                        163 STO 07                162 /
                        164 STO 07                163 /
                        165 STO 07                164 /
                        166 STO 07                165 /
                        167 STO 07                166 /
                        168 STO 07                167 /
                        169 STO 07                168 /
                        170 STO 07                169 /
                        171 STO 07                170 /
                        172 STO 07                171 /
                        173 STO 07                172 /
                        174 STO 07                173 /
                        175 STO 07                174 /
                        176 STO 07                175 /
                        177 STO 07                176 /
                        178 STO 07                177 /
                        179 STO 07                178 /
                        180 STO 07                179 /
                        181 STO 07                180 /
                        182 STO 07                181 /
                        183 STO 07                182 /
                        184 STO 07                183 /
                        185 STO 07                184 /
                        186 STO 07                185 /
                        187 STO 07                186 /
                        188 STO 07                187 /
                        189 STO 07                188 /
                        190 STO 07                189 /
                        191 STO 07                190 /
                        192 STO 07                191 /
                        193 STO 07                192 /
                        194 STO 07                193 /
                        195 STO 07                194 /
                        196 STO 07                195 /
                        197 STO 07                196 /
                        198 STO 07                197 /
                        199 STO 07                198 /
                        200 STO 07                199 /
                        201 STO 07                200 /
                        202 STO 07                201 /
                        203 STO 07                202 /
                        204 STO 07                203 /
                        205 STO 07                204 /
                        206 STO 07                205 /
                        207 STO 07                206 /
                        208 STO 07                207 /
                        209 STO 07                208 /
                        210 STO 07                209 /
                        211 STO 07                210 /
                        212 STO 07                211 /
                        213 STO 07                212 /
                        214 STO 07                213 /
                        215 STO 07                214 /
                        216 STO 07                215 /
                        217 STO 07                216 /
                        218 STO 07                217 /
                        219 STO 07                218 /
                        220 STO 07                219 /
                        221 STO 07                220 /
                        222 STO 07                221 /
                        223 STO 07                222 /
                        224 STO 07                223 /
                        225 STO 07                224 /
                        226 STO 07                225 /
                        227 STO 07                226 /
                        228 STO 07                227 /
                        229 STO 07                228 /
                        230 STO 07                229 /
                        231 STO 07                230 /
                        232 STO 07                231 /
                        233 STO 07                232 /
                        234 STO 07                233 /
                        235 STO 07                234 /
                        236 STO 07                235 /
                        237 STO 07                236 /
                        238 STO 07                237 /
                        239 STO 07                238 /
                        240 STO 07                239 /
                        241 STO 07                240 /
                        242 STO 07                241 /
                        243 STO 07                242 /
                        244 STO 07                243 /
                        245 STO 07                244 /
                        246 STO 07                245 /
                        247 STO 07                246 /
                        248 STO 07                247 /
                        249 STO 07                248 /
                        250 STO 07                249 /
                        251 STO 07                250 /
                        252 STO 07                251 /
                        253 STO 07                252 /
                        254 STO 07                253 /
                        255 STO 07                254 /
                        256 STO 07                255 /
                        257 STO 07                256 /
                        258 STO 07                257 /
                        259 STO 07                258 /
                        260 STO 07                259 /
                        261 STO 07                260 /
                        262 STO 07                261 /
                        263 STO 07                262 /
                        264 STO 07                263 /
                        265 STO 07                264 /
                        266 STO 07                265 /
                        267 STO 07                266 /
                        268 STO 07                267 /
                        269 STO 07                268 /
                        270 STO 07                269 /
                        271 STO 07                270 /
                        272 STO 07                271 /
                        273 STO 07                272 /
                        274 STO 07                273 /
                        275 STO 07                274 /
                        276 STO 07                275 /
                        277 STO 07                276 /
                        278 STO 07                277 /
                        279 STO 07                278 /
                        280 STO 07                279 /
                        281 STO 07                280 /
                        282 STO 07                281 /
                        283 STO 07                282 /
                        284 STO 07                283 /
                        285 STO 07                284 /
                        286 STO 07                285 /
                        287 STO 07                286 /
                        288 STO 07                287 /
                        289 STO 07                288 /
                        290 STO 07                289 /
                        291 STO 07                290 /
                        292 STO 07                291 /
                        293 STO 07                292 /
                        294 STO 07                293 /
                        295 STO 07                294 /
                        296 STO 07                295 /
                        297 STO 07                296 /
                        298 STO 07                297 /
                        299 STO 07                298 /
                        300 STO 07                299 /
                        301 STO 07                300 /
                        302 STO 07                301 /
                        303 STO 07                302 /
                        304 STO 07                303 /
                        305 STO 07                304 /
                        306 STO 07                305 /
                        307 STO 07                306 /
                        308 STO 07                307 /
                        309 STO 07                308 /
                        310 STO 07                309 /
                        311 STO 07                310 /
                        312 STO 07                311 /
                        313 STO 07                312 /
                        314 STO 07                313 /
                        315 STO 07                314 /
                        316 STO 07                315 /
                        317 STO 07                316 /
                        318 STO 07                317 /
                        319 STO 07                318 /
                        320 STO 07                319 /
                        321 STO 07                320 /
                        322 STO 07                321 /
                        323 STO 07                322 /
                        324 STO 07                323 /
                        325 STO 07                324 /
                        326 STO 07                325 /
                        327 STO 07                326 /
                        328 STO 07                327 /
                        329 STO 07                328 /
                        330 STO 07                329 /
                        331 STO 07                330 /
                        332 STO 07                331 /
                        333 STO 07                332 /
                        334 STO 07                333 /
                        335 STO 07                334 /
                        336 STO 07                335 /
                        337 STO 07                336 /
                        338 STO 07                337 /
                        339 STO 07                338 /
                        340 STO 07                339 /
                        341 STO 07                340 /
                        342 STO 07                341 /
                        343 STO 07                342 /
                        344 STO 07                343 /
                        345 STO 07                344 /
                        346 STO 07                345 /
                        347 STO 07                346 /
                        348 STO 07                347 /
                        349 STO 07                348 /
                        350 STO 07                349 /
                        351 STO 07                350 /
                        352 STO 07                351 /
                        353 STO 07                352 /
                        354 STO 07                353 /
                        355 STO 07                354 /
                        356 STO 07                355 /
                        357 STO 07                356 /
                        358 STO 07                357 /
                        359 STO 07                358 /
                        360 STO 07                359 /
                        361 STO 07                360 /
                        362 STO 07                361 /
                        363 STO 07                362 /
                        364 STO 07                363 /
                        365 STO 07                364 /
                        366 STO 07                365 /
                        367 STO 07                366 /
                        368 STO 07                367 /
                        369 STO 07                368 /
                        370 STO 07                369 /
                        371 STO 07                370 /
                        372 STO 07                371 /
                        373 STO 07                372 /
                        374 STO 07                373 /
                        375 STO 07                374 /
                        376 STO 07                375 /
                        377 STO 07                376 /
                        378 STO 07                377 /
                        379 STO 07                378 /
                        380 STO 07                379 /
                        381 STO 07                380 /
                        382 STO 07                381 /
                        383 STO 07                382 /
                        384 STO 07                383 /
                        385 STO 07                384 /
                        386 STO 07                385 /
                        387 STO 07                386 /
                        388 STO 07                387 /
                        389 STO 07                388 /
                        390 STO 07                389 /
                        391 STO 07                390 /
                        392 STO 07                391 /
                        393 STO 07                392 /
                        394 STO 07                393 /
                        395 STO 07                394 /
                        396 STO 07                395 /
                        397 STO 07                396 /
                        398 STO 07                397 /
                        399 STO 07                398 /
                        400 STO 07                399 /
                        401 STO 07                400 /
                        402 STO 07                401 /
                        403 STO 07                402 /
                        404 STO 07                403 /
                        405 STO 07                404 /
                        406 STO 07                405 /
                        407 STO 07                406 /
                        408 STO 07                407 /
                        409 STO 07                408 /
                        410 STO 07                409 /
                        411 STO 07                410 /
                        412 STO 07                411 /
                        413 STO 07                412 /
                        414 STO 07                413 /
                        415 STO 07                414 /
                        416 STO 07                415 /
                        417 STO 07                416 /
                        418 STO 07                417 /
                        419 STO 07                418 /
                        420 STO 07                419 /
                        421 STO 07                420 /
                        422 STO 07                421 /
                        423 STO 07                422 /
                        424 STO 07                423 /
                        425 STO 07                424 /
                        426 STO 07                425 /
                        427 STO 07                426 /
                        428 STO 07                427 /
                        429 STO 07                428 /
                        430 STO 07                429 /
                        431 STO 07                430 /
                        432 STO 07                431 /
                        433 STO 07                432 /
                        434 STO 07                433 /
                        435 STO 07                434 /
                        436 STO 07                435 /
                        437 STO 07                436 /
                        438 STO 07                437 /
                        439 STO 07                438 /
                        440 STO 07                439 /
                        441 STO 07                440 /
                        442 STO 07                441 /
                        443 STO 07                442 /
                        444 STO 07                443 /
                        445 STO 07                444 /
                        446 STO 07                445 /
                        447 STO 07                446 /
                        448 STO 07                447 /
                        449 STO 07                448 /
                        450 STO 07                449 /
                        451 STO 07                450 /
                        452 STO 07                451 /
                        453 STO 07                452 /
                        454 STO 07                453 /
                        455 STO 07                454 /
                        456 STO 07                455 /
                        457 STO 07                456 /
                        458 STO 07                457 /
                        459 STO 07                458 /
                        460 STO 07                459 /
                        461 STO 07                460 /
                        462 STO 07                461 /
                        463 STO 07                462 /
                        464 STO 07                463 /
                        465 STO 07                464 /
                        466 STO 07                465 /
                        467 STO 07                466 /
                        468 STO 07                467 /
                        469 STO 07                468 /
                        470 STO 07                469 /
                        471 STO 07                470 /
                        472 STO 07                471 /
                        473 STO 07                472 /
                        474 STO 07                473 /
                        475 STO 07                474 /
                        476 STO 07                475 /
                        477 STO 07                476 /
                        478 STO 07                477 /
                        479 STO 07                478 /
                        480 STO 07                479 /
                        481 STO 07                480 /
                        482 STO 07                481 /
                        483 STO 07                482 /
                        484 STO 07                483 /
                        485 STO 07                484 /
                        486 STO 07                485 /
                        487 STO 07                486 /
                        488 STO 07                487 /
                        489 STO 07                488 /
                        490 STO 07                489 /
                        491 STO 07                490 /
                        492 STO 07                491 /
                        493 STO 07                492 /
                        494 STO 07                493 /
                        495 STO 07                494 /
                        496 STO 07                495 /
                        497 STO 07                496 /
                        498 STO 07                497 /
                        499 STO 07                498 /
                        500 STO 07                499 /
                        501 STO 07                500 /
                        502 STO 07                501 /
                        503 STO 07                502 /
                        504 STO 07                503 /
                        505 STO 07                504 /
                        506 STO 07                505 /
                        507 STO 07                506 /
                        508 STO 07                507 /
                        509 STO 07                508 /
                        510 STO 07                509 /
                        511 STO 07                510 /
                        512 STO 07                511 /
                        513 STO 07                512 /
                        514 STO 07                513 /
                        515 STO 07                514 /
                        516 STO 07                515 /
                        517 STO 07                516 /
                        518 STO 07                517 /
                        519 STO 07                518 /
                        520 STO 07                519 /
                        521 STO 07                520 /
                        522 STO 07                521 /
                        523 STO 07                522 /
                        524 STO 07                523 /
                        525 STO 07                524 /
                        526 STO 07                525 /
                        527 STO 07                526 /
                        528 STO 07                527 /
                        529 STO 07                528 /
                        530 STO 07                529 /
                        531 STO 07                530 /
                        532 STO 07                531 /
                        533 STO 07                532 /
                        534 STO 07                533 /
                        535 STO 07                534 /
                        536 STO 07                535 /
                        537 STO 07                536 /
                        538 STO 07                537 /
                        539 STO 07                538 /
                        540 STO 07                539 /
                        541 STO 07                540 /
                        542 STO 07                541 /
                        543 STO 07                542 /
                        544 STO 07                543 /
                        545 STO 07                544 /
                        546 STO 07                545 /
                        547 STO 07                546 /
                        548 STO 07                547 /
                        549 STO 07                548 /
                        550 STO 07                549 /
                        551 STO 07                550 /
                        552 STO 07                551 /
                        553 STO 07                552 /
                        554 STO 07                553 /
                        555 STO 07                554 /
                        556 STO 07                555 /
                        557 STO 07                556 /
                        558 STO 07                557 /
                        559 STO 07                558 /
                        560 STO 07                559 /
                        561 STO 07                560 /
                        562 STO 07                561 /
                        563 STO 07                562 /
                        564 STO 07                563 /
                        565 STO 07                564 /
                        566 STO 07                565 /
                        567 STO 07                566 /
                        568 STO 07                567 /
                        569 STO 07                568 /
                        570 STO 07                569 /
                        571 STO 07                570 /
                        572 STO 07                571 /
                        573 STO 07                572 /
                        574 STO 07                573 /
                        575 STO 07                574 /
                        576 STO 07                575 /
                        577 STO 07                576 /
                        578 STO 07                577 /
                        579 STO 07                578 /
                        580 STO 07                579 /
                        581 STO 07                580 /
                        582 STO 07                581 /
                        583 STO 07                582 /
                        584 STO 07                583 /
                        585 STO 07                584 /
                        586 STO 07                585 /
                        587 STO 07                586 /
                        588 STO 07                587 /
                        589 STO 07                588 /
                        590 STO 07                589 /
                        591 STO 07                590 /
                        592 STO 07                591 /
                        593 STO 07                592 /
                        594 STO 07                593 /
                        595 STO 07                594 /
                        596 STO 07                595 /
                        597 STO 07                596 /
                        598 STO 07                597 /
                        599 STO 07                598 /
                        600 STO 07                599 /
                        601 STO 07                600 /
                        602 STO 07                601 /
                        603 STO 07                602 /
                        604 STO 07                603 /
                        605 STO 07                604 /
                        606 STO 07                605 /
                        607 STO 07                606 /
                        608 STO 07                607 /
                        609 STO 07                608 /
                        610 STO 07                609 /
                        611 STO 07                610 /
                        612 STO 07                611 /
                        613 STO 07                612 /
                        614 STO 07                613 /
                        615 STO 07                614 /
                        616 STO 07                615 /
                        617 STO 07                616 /
                        618 STO 07                617 /
                        619 STO 07                618 /
                        620 STO 07                619 /
                        621 STO 07                620 /
                        622 STO 07                621 /
                        623 STO 07                622 /
                        624 STO 07                623 /
                        625 STO 07                624 /
                        626 STO 07                625 /
                        627 STO 07                626 /
                        628 STO 07                627 /
                        629 STO 07                628 /
                        630 STO 07                629 /
                        631 STO 07                630 /
                        632 STO 07                631 /
                        633 STO 07                632 /
                        634 STO 07                633 /
                        635 STO 07                634 /
                        636 STO 07                635 /
                        637 STO 07                636 /
                        638 STO 07                637 /
                        639 STO 07                638 /
                        640 STO 07                639 /
                        641 STO 07                640 /
                        642 STO 07                641 /
                        643 STO 07                642 /
                        644 STO 07                643 /
                        645 STO 07                644 /
                        646 STO 07                645 /
                        647 STO 07                646 /
                        648 STO 07                647 /
                        649 STO 07                648 /
                        650 STO 07                649 /
                        651 STO 07                650 /
                        652 STO 07                651 /
                        653 STO 07                652 /
                        654 STO 07                653 /
                        655 STO 07                654 /
                        656 STO 07                655 /
                        657 STO 07                656 /
                        658 STO 07                657 /
                        659 STO 07                658 /
                        660 STO 07                659 /
                        661 STO 07                660 /
                        662 STO 07                661 /
                        663 STO 07                662 /
                        664 STO 07                663 /
                        665 STO 07                664 /
                        666 STO 07                665 /
                        667 STO 07                666 /
                        668 STO 07                667 /
                        669 STO 07                668 /
                        670 STO 07                669 /
                        671 STO 07                670 /
                        672 STO 07                671 /
                        673 STO 07                672 /
                        674 STO 07                673 /
                        675 STO 07                674 /
                        676 STO 07                675 /
                        677 STO 07                676 /
                        678 STO 07                677 /
                        679 STO 07                678 /
                        680 STO 07                679 /
                        681 STO 07                680 /
                        682 STO 07                681 /
                        683 STO 07                682 /
                        684 STO 07                683 /
                        685 STO 07                684 /
                        686 STO 07                685 /
                        687 STO 07                686 /
                        688 STO 07                687 /
                        689 STO 07                688 /
                        690 STO 07                689 /
                        691 STO 07                690 /
                        692 STO 07                691 /
                        693 STO 07                692 /
                        694 STO 07                693 /
                        695 STO 07                694 /
                        696 STO 07                695 /
                        697 STO 07                696 /
                        698 STO 07                697 /
                        699 STO 07                698 /
                        700 STO 07                699 /
                        701 STO 07                700 /
                        702 STO 07                701 /
                        703 STO 07                702 /
                        704 STO 07               
```

```

23 RDN
24 1
25 +
26 X<>Y
27 1
28 +
29 RCL IND
X
30 STO IND
Z
31 1 E3
32 /
33 1
34 +
35 STO 07
36+LEL 01
37 RCL 07
38 STO 09
39+LBL 02
40 RCL 09
41 RCL 06
42 RCL 02
43 XEO "MTA"
IJ"
44 RCL IND
X
45 RCL 00
46 *
47 STO 05
48 RDN
49 RCL 01
50 X=0?
51 GTO 03
52 RCL Y
53 +
54 RCL 02
55 -
56 RCL IND
X
57 ST+ 05
58 RDN
59+LBL 03
60 RDN
61 RCL 02
62 -
63 RCL 03
64 +
65 RCL 05
66 STO IND
Y
67 ISG 09
68 GTO 02
69 ISG 06
70 GTO 01
71 RTN
72 END

```

5.6 MTSPU

```

LBL*MTSPU
END
49 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

Σ = SPUR(A)

A:
T:
Z:
Y: S(A)
X: REG(Σ)

01+LBL "MTS
PU"
02 STO 03
03 RDN
04 STO 01
05 1

```

```

06 +
07 RCL IND
X
08 1 E3
09 /
10 1
11 +
12 STO 06
13 CLX
14 STO IND
03
15+LBL 02
16 RCL 06
17 RCL 06
18 RCL 01
19 XEO "MTA"
IJ"
20 RCL IND
X
21 ST+ IND
03
22 ISG 06
23 GTO 02
24 RTN
25 END

```

5.7 MTSHF

```

LBL*MTSHF
END
02 BYTES

VERSCHIEBEN
DER MATRIX A
VON S(A)
NACH S(B)

A:
T:
Z:
Y: S(A)
X: S(B)

01+LBL "MTS
HF"
02 CF 00
03 X<>Y
04 STO 01
05 -
06 STO 02
07 X>0?
08 SF 00
09 RCL 01
10 1
11 +
12 RCL IND
X
13 X<>Y
14 1
15 +
16 X<>Y
17 RCL IND
Y
18 *
19 +
20 FS? 00
21 GTO 01
22 1 E3
23 /
24 RCL 01
25 +
26 GTO 02
27+LBL 01
28 RCL 01
29 1
30 -
31 1 E3
32 /
33 +
34+LBL 02
35 STO 06

```

```

36+LBL 03
37 RCL 06
38 RCL IND
X
39 X<>Y
40 RCL 02
41 +
42 X<>Y
43 STO IND
Y
44 FS? 00
45 GTO 04
46 ISG 06
47 GTO 03
48 RTN
49+LBL 04
50 DSE 06
51 GTO 03
52 RTN
53 END

```

5.8 MTTPA

```

LBL*MTTPA
END
91 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

B = A *

A: NAME(B)
T:
Z:
Y: S(A)
X: S(B)

01+LBL "MTT
RA"
02 STO 03
03 RDN
04 STO 01
05 ASTO IND
03
06 1
07 +
08 RCL IND
X
09 RCL 03
10 2
11 +
12 X<>Y
13 STO IND
Y
14 1 E3
15 /
16 1
17 +
18 STO 06
19 RCL 01
20 2
21 +
22 RCL IND
X
23 RCL 03
24 1
25 +
26 X<>Y
27 STO IND
Y
28 1 E3
29 /
30 1
31 +
32 STO 07
33+LBL 01
34 RCL 07
35 STO 09
36+LBL 02

```

```

37 RCL 09
38 RCL 06
39 RCL 01
40 XEO "MTA"
IJ"
41 RCL IND
X
42 STO 04
43 RCL 06
44 RCL 09
45 RCL 03
46 XEO "MTA"
IJ"
47 RCL 04
48 STO IND
Y
49 ISG 09
50 GTO 02
51 ISG 06
52 GTO 01
53 RTN
54 END

```

5.9 MTMUL

```

LBL*MTMUL
END
131 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

C = A * B

A: NAME(C)
T:
Z: S(A)
Y: S(B)
X: S(C)

01+LBL "MTM
UL"
02 STO 03
03 RDN
04 STO 02
05 RDN
06 STO 01
07 ASTO IND
03
08 1
09 +
10 RCL IND
X
11 RCL 03
12 1
13 +
14 X<>Y
15 STO IND
Y
16 1 E3
17 /
18 1
19 +
20 STO 06
21 RDN
22 RCL 02
23 2
24 +
25 RCL IND
X
26 1
27 ST+ T
28 RDN
29 STO IND
Z
30 1 E3
31 /
32 1
33 +

```

```

34 STO 07
35 RDN
36 1
37 -
38 RCL IND
X
39 1 E3
40 /
41 1
42 +
43 STO 08
44 LBL 01
45 RCL 07
46 STO 09
47 LBL 02
48 RCL 08
49 STO 10
50 0
51 STO 05
52 LBL 03
53 RCL 10
54 RCL 06
55 RCL 01
56 XEQ "MTR"
IJ"
57 RCL IND
X
58 STO 04
59 RCL 09
60 RCL 10
61 RCL 02
62 XEQ "MTR"
IJ"
63 RCL IND
X
64 RCL 04
65 *
66 ST+ 05
67 ISG 10
68 GTO 03
69 RCL 09
70 RCL 06
71 RCL 03
72 XEQ "MTR"
IJ"
73 RCL 05
74 STO IND
Y
75 ISG 09
76 GTO 02
77 ISG 06
78 GTO 01
79 PTN
80 END

```

```

5.10 MTINV
LBL "MTINV"
END
189 BYTES

```

```

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

```

```

B = A*(-1)
S(B) = S(A)

```

```

A: NAME(B)
T:
Z:
Y:
X: S(A)

```

```

01 LBL "MTI
NV"
02 STO 00
03 ASTO IND
X
04 1
05 +
06 RCL IND

```

```

X
07 STO 01
08 1 E3
09 /
10 1
11 +
12 STO 02
13 STO 03
14 LBL 01
15 1
16 ENTER↑
17 1
18 XEQ 10
19 RCL IND
X
20 1/X
21 STO 06
22 STO IND
Y
23 RCL 02
24 1
25 +
26 STO 05
27 LBL 02
28 RCL 05
29 1
30 XEQ 10
31 RCL 06
32 CHS
33 ST* IND
Y
34 ISG 05
35 GTO 02
36 PCL 02
37 1
38 +
39 STO 04
40 LBL 03
41 RCL 02
42 1
43 +
44 STO 05
45 LBL 04
46 1
47 RCL 04
48 XEQ 10
49 RCL IND
X
50 STO 07
51 RCL 05
52 1
53 XEQ 10
54 RCL IND
X
55 ST* 07
56 RCL 05
57 PCL 04
58 XEQ 10
59 RCL 07
60 ST+ IND
Y
61 ISG 05
62 GTO 04
63 ISG 04
64 GTO 03
65 RCL 02
66 1
67 +
68 STO 04
69 LBL 05
70 1
71 RCL 04
72 XEQ 10
73 RCL 06
74 ST* IND
Y
75 ISG 04
76 GTO 05
77 RCL 02
78 STO 05
79 LBL 06
80 RCL 05

```

```

81 INT
82 STO 07
83 RCL 01
84 STO 06
85 XEQ 10
86 RCL IND
X
87 STO 03
88 RCL 02
89 STO 04
90 LBL 07
91 RCL 06
92 XEQ 02
93 STO 06
94 RCL 07
95 XEQ 08
96 STO 07
97 PCL 06
98 XEQ 10
99 RCL IND
X
100 X<> 08
101 STO IND
Y
102 ISG 04
103 GTO 07
104 ISG 05
105 GTO 06
106 ISG 03
107 GTO 01
108 PTN
109 LBL 08
110 1
111 +
112 RCL 01
113 -
114 X>0?
115 PTN
116 PCL 01
117 +
118 PTN
119 LBL 10
120 PCL 00
121 XEQ "MTR"
IJ"
122 RTN
123 END

```

Erläuterungen zu "GLS"

Das vorliegende Programm dient dazu, n lineare Gleichungen mit n Unbekannten zu lösen. Die Koeffizienten dieser sind in einer Matrix angeordnet (Abb. 1). Das Programm dient nur der Lösung eines Gleichungssystems, auf Berechnung der Determinante und Inversen wurde verzichtet, um Speicherplatz zu sparen. Hierdurch unterscheidet es sich z.B. vom ähnlichen Programm "4x4" im Solution-Book: High level Math; dieses benötigt 102 (!) Programmregister und löst damit ausschließlich 4x4-Gleichungssysteme, während "GLS" je nach Anzahl der angeschlossenen Module bis zu 15x15-Matrizen lösen kann; es benötigt 45 Programmregister und $n(n+1)+9$ Datenregister. Die Rechenzeiten für $n=2\dots 11$ sind aus Abb. 2 ersichtlich.

a ₁₁	a ₁₂	a ₁₃	b ₁	2	3	4	5	6	7
a ₂₁	a ₂₂	a ₂₃	b ₂	9"	20"	37"	1'03"	1'37"	2'21"
a ₃₁	a ₃₂	a ₃₃	b ₃	8	9	10	11		
Abb. 1				3'17"	4'24"	5'50"	7'29"		Abb. 2

Algorithmus

Zur Anwendung kommt der sogenannte Gauß-Algorithmus mit Totalpivotisierung, der wahrscheinlich nicht allen von Euch bekannt ist. Dieser sei an einem Beispiel erläutert:

Gegeben sind die drei Gleichungen $2x_1 + 3x_2 = 5$, $x_1 + x_2 + x_3 = 0$, $6x_1 + 7x_2 + 2x_3 = 7$, in Matrixform: (1). a_{11} ist das sogenannte Pivotelement, a_{21} soll durch

$$\begin{pmatrix} 2 & 3 & 0 & 5 \\ 1 & 1 & 1 & 0 \\ 6 & 7 & 2 & 7 \end{pmatrix} \quad (1)$$

$$\begin{pmatrix} 2 & 3 & 0 & 5 \\ 0 & -0,5 & 1 & -2,5 \\ 0 & -2 & 2 & -8 \end{pmatrix} \quad (2)$$

$$\begin{pmatrix} 2 & 3 & 0 & 5 \\ 0 & 1 & -2 & 5 \\ 0 & 1 & -1 & 4 \end{pmatrix} \quad (3)$$

$$\begin{pmatrix} 1 & 0 & 3 & -5 \\ 0 & 1 & -2 & 5 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad (4)$$

$$\begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad (5)$$

Multiplikation der 1. Zeile mit einer gesuchten Zahl s und Addition der 1. zur 2. Zeile zu 0 gemacht werden, ebenso a_{31} (man darf ja Gleichungen addieren). Die gesuchten Zahlen sind $-0,5$ bzw. -3 ; daraus ergibt sich folgende Matrix (2). Da Zeilen (Gleichungen) mit beliebigen Zahlen multipliziert werden dürfen, folgt Matrix (3). Jetzt wird a_{22} als Pivotelement gewählt und das Verfahren wiederholt; diesmal sollen a_{12} und a_{32} zu 0 gemacht werden (4), entsprechend folgt (5). Wenn nun die Diagonalelemente $a_{11}\dots a_{33}$ alle 1 und die übrigen Elemente der "homogenen" Matrix (des linken Teils, 3x3) 0 sind, stehen rechts von oben nach unten die Lösungen des Gleichungssystems, also ist $x_1=-2$, $x_2=3$, $x_3=-1$. Der Rechner geht im Prinzip genauso vor, er dividiert allerdings die Zeile, die das Pivotelement enthält, durch dieses, wodurch die Diagonalelemente alle 1 werden. Noch ein Hinweis, falls ein Pivotelement 0 ist: der Rechner geht dann auf Pivotsuche, d.h. er tauscht so lange Zeilen aus (Lbl Ø3), bis er ein Pivot $\neq 0$ gefunden hat. Falls das Gleichungssystem keine eindeutige Lösung hat, versagt das Programm.

Mathe-Freaks sei deshalb empfohlen, sich nach der Ergebnisanzeige durch XEQ A die Matrix anzusehen, ob sie die Gestalt von (5) hat oder nicht. Anderen empfehle ich die Prüfroutine Lbl 14. Diese wird an das Programm angehängt, zusätzlich nach Zeile 187 RCL Ø3 X=Y? GTO 16 RDN.

Bedienung des Programms:

Tastenfolge	Anzeige	Kommentar
XEQ "GLS"	RANG?	
3 R/S	a1.1	Koeffizienten
2 R/S	a1.2	
3 R/S	a1.3	
Ø R/S	b1	
5 R/S	a2.1	
...	...	
...	b3	
7 R/S	READY	
(XEQ A	a1.1=2,ØØØØ	Überprüfung der richtigen Wert- eingabe; während Anzeige (PSE) Wertänderung möglich.
	a1.2=3,ØØØØ	
	...	
	READY	
R/S	X1=-2,ØØØØ	Ergebnisse
R/S	X2= 3,ØØØØ	
R/S	X3=-1,ØØØØ	

Listing Lbl 14

```
Lbl 14 9 RCL Ø4 1E3 / + RCL Ø3 2 + 1E5 / + STO Ø1
1E-5 - RCL Ø3 + STO ØØ LBL 17 RCL IND ØØ * RCL IND Ø1
* ISG Ø1 CLD ISG ØØ GTO 17 X≠Ø? RTN LBL 16 "NO SOLVE"
PROMPT.
```

Garantien für dieses Programm kann ich natürlich nicht übernehmen, aber über regen Zuspruch würde ich mich sehr freuen, vielleicht gelingt es jemand, das Programm zu kürzen?

Happy Programming!

Lorenz Tichy 418
Spardorfer Str. 51
8520 Erlangen

Listing "GLS"

01	LBL "GLS"	56	FS?	00	111	LBL	07	166	"X"
	CF 00		GTU	09		RCL	IND 01		FIX 0
	CF 29		PROMPT			STO	07		ARCL 06
	"HANG?"		STO	IND 05		LBL	05		"I="
	PROMPT	60	LBL	10		RCL	06	170	FIX 4
	STO 03		FIX	0		ST/	IND 05		ARCL IND 05
	GTU 08		1			RCL	IND 05		PROMPT
	LBL A		ST+	05		RCL	07		ISG 05
	SF 00		RTN			=			CLD
10	LBL 08		LBL	09	120	ST-	IND 01		ISG 06
	FIX 0		"I="			1			GTU 13
	RCL 03		FIX 4			ST+	05		STOP
	1 E3		RCL	IND 05		ISG	01		LBL 03
	/		RND			GTU	05		RCL 05
	1	70	ARCL	X		1		180	INT
	+		AVIEW			STO	06		STO 06
	STO 00		PSE			LBL	06		RCL 02
	STO 01		STO	IND 05		RCL	03		INT
	STO 02		GTU	10		1			RCL X
20	9		LBL	02	130	+			RCL 08
	STO 05		0			1 E3			+
	RCL 03		STO	08		/			RCL 03
	X/2		RCL	00		RCL	02		1
	LASTX		STO	05		INT			+
	+	80	RCL	02		1		190	=
	8		INT	8		-			8
	+		8			+			+
	STO 04		+			ST+	01		+
	LBL 01		RCL	03	140	RCL	00		LASTX
30	"a"		9			STO	05		RCL 03
	ARCL 00		1 E3			RCL	01		1
	"I."		/			RCL	04		+
	ARCL 01		+			X>Y?			+
	XEQ 00		STO	01		GTU	04		1 E3
	ISG 01	90	LBL	15		LBL	12	200	/
	GTU 01		RCL	15		ISG	00		+
	"b"		RCL	IND 05		CLD			STO 07
	ARCL 00		STO	06		ISG	02		LBL 11
	XEQ 00		X=0?			GTU	02		RCL IND 06
40	RCL 03		LBL	04	150	RCL	03		X<> IND 07
	ST- 01		RCL	05		9			STO IND 06
	ISG 00		INT			+			ISG 06
	GTU 01		RCL	01		RCL	03		CLD
	CF 00	100	INT			1		210	ISG 07
	2		X#Y?			+			GTU 11
	+		GTU	07		1 E5			1
	1 E5		RCL	03		/			ST+ 08
	/		2			+			GTU 15
	9		+		160	STO	05	214	END
50	+		RCL	02		RCL	02		
	STO 00		INT			FRG			
	"READY"		-			1			
	PROMPT		ST+	01		+			
	GTU 02	110	GTU	06		STO	06		
	LBL 00					LBL	13		

Lineare Regression mit Fehlerrechnung

Formeln:

```

01+LBL "LINF"
CLRG SF 21 XREG 04
"GIVE DATES" AVIEW
RCL 08 RCL 04 RCL 06
RCL 09 / * - RCL 07 /
RCL 06 XY2 RCL 09 /
- STO 08 / STO 03
MEAN XCY RCL 03 * -
STO 02 "a=" ARCL X
AVIEW "b=" ARCL X
AVIEW RCL 07 RCL 03
XY2 * RCL 05 RCL 09
RCL 02 XY2 * + +
RCL 04 RCL 06 RCL 03
* - RCL 02 * RCL 03
RCL 08 * + ST+ X
RCL T XCY - RCL 09
2 - / SRT STO 01
"S.Y=" ARCL X AVIEW
RCL 01 RCL 07 RCL 08
RCL 09 * / SRT *
"S.a=" ARCL X AVIEW
RCL 01 RCL 08 1/X
SRT * "S.b=" ARCL X
AVIEW
    
```

```

09+LBL 01
"X=?" PROMPT RCL 03 *
RCL 02 + "Y=" ARCL X
AVIEW GTO 01 .END.
    
```

$$b = \frac{\sum x_i y_i - 1/n \sum y_i \sum x_i}{\sum x_i^2 - 1/n (\sum x_i)^2}$$

Steigung

$$a = \bar{y} - b \cdot \bar{x}$$

Achsenabschnitt

Fehler (Standardabweichung)

Fehler des Fkt.wertes:

$$s_y = \sqrt{\frac{(y_i - (bx_i + a))^2}{n - 2}}$$

Fehler der Steigung:

$$s_b = \sqrt{\frac{n}{n \sum x_i^2 - (\sum x_i)^2}} \cdot s_y$$

Fehler des Achsenabschnittes:

$$s_a = \sqrt{\frac{\sum x_i^2}{n \sum x_i^2 - (\sum x_i)^2}} \cdot s_y$$

Programmumfang: 151 Bytes
SIZE 010

Zum Gebrauch des Programms diene folgendes Beispiel:

Gegeben sind die Werte:

x	1	2	3	4
y	5	8	9	10

1. XEQ' LINF
2. Nach der Aufforderung "GIVE DATES", Eingabe der Werte wie folgt:
1; ENTER; 5; $\Sigma+$, 2; ENTER; 8; $\Sigma+$, etc.
3. R/S: a=4,00; R/S: b=1,60 R/S: S.Y=0,77; R/S: S.a=0,95
R/S: S.b=0,35
4. R/S: X=? Um die Reg.gerade zu zeichnen, kann man sich einige Fkt.werte ausrechnen lassen, z.B.: 1 R/S Y=5,6; R/S: X=?
4 R/S Y=10,4

Eine Korrektur erfolgt wie unter 2., aber statt $\Sigma+$, $\Sigma-$ drücken.

Gerald Krampe

Programmpaket INTERPOLATION

- Inhalt:
1. Allgemeines
 2. Polynominterpolation
 3. Rationale Interpolation
 4. Programmbeschreibungen
 5. Algorithmen
 6. HP 41 C-Programme
 7. Beispiele

1. Allgemeines

Es seien paarweise verschiedene Punkte x_0, x_1, \dots, x_n mit dazugehörigen Funktionswerten y_0, y_1, \dots, y_n gegeben. Gesucht wird ein Polynom vom Grade n (oder eine rationale Funktion) $f(x)$ mit $f(x_k) = y_k$ für $k=0, \dots, n$.

Das Programmpaket enthält verschiedene Unterprogramme zur Berechnung des Polynoms in unterschiedlichen Formen, sowie zur Ermittlung der Funktionswerte und Ableitungswerte für beliebige x -Werte. Es werden im folgenden keine mathematischen Beweise angegeben, sondern nur die zum Verständnis unbedingt notwendigen Definitionen und Ergebnisse aufgeführt. Für weitergehende Informationen wird auf das Buch

[1] Meinardus, G.; G. Merz: Praktische Mathematik I
Mannheim, Wien, Zürich 1979 (Bibliographisches Institut)
verwiesen.

2. Polynominterpolation

Für das in Abschnitt 1 formulierte Interpolationsproblem gibt es ein eindeutig bestimmtes Polynom vom Grade n

$$(1) \quad p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n.$$

Zur Berechnung dieses Polynoms werden sogenannte dividierte Differenzen Δ_i^k verwendet, die rekursiv wie folgt definiert sind:

$$(2) \quad \begin{aligned} \Delta_i^0 &= y_i \quad (i=0, \dots, n) \\ \Delta_i^k &= \frac{\Delta_i^{k-1} - \Delta_{i+1}^{k-1}}{x_i - x_{i+k}} \end{aligned}$$

Mit $b_k = \Delta_k^k$, $k=0, \dots, n$ gilt dann für das Interpolationspolynom:

$$(3) \quad q(x) = b_0 + b_1(x-x_0) + \dots + b_n(x-x_0)(x-x_1)\dots(x-x_{n-1}).$$

Mit einem modifizierten Horner Schema (vgl. [1]) erhält man aus $q(x)$ eine weitere Darstellung des Interpolationspolynoms:

$$(4) \quad r\xi(x) = c_0 + c_1(x-\xi) + \dots + c_n(x-\xi)^n.$$

Dabei ist ξ beliebig. Für $\xi = 0$ ergibt sich $p(x)$. Man beachte, daß $c_0 = r_\xi(\xi)$ und $c_1 = r'_\xi(\xi)$. Mit dem normalen Horner Schema wird $r_\xi(x)$ aus $p(x)$ berechnet. Die Unterprogramme zur Ermittlung von $p(\xi)$, $p'(\xi)$ (bzw. $q(\xi)$, $q'(\xi)$) verwenden verkürzte Formen des vollständigen normalen (bzw. modifizierten) Horner Schemas.

3. Rationale Interpolation

Für die rationale Interpolation wird der sogenannte Thielesche Kettenbruch verwendet, d.h. die rationale Interpolationsfunktion $R(x)$ sieht wie folgt aus:

$$(5) \quad R(x) = c_0 + \frac{x-x_0}{c_1 + \frac{x-x_1}{c_2 + \dots + \frac{x-x_{n-1}}{c_n}}}$$

Zur Berechnung der c_k werden die sog. inversen Differenzen $\overset{k}{i}$ benutzt, die rekursiv wie folgt definiert sind:

$$(6) \quad \begin{aligned} \overset{0}{i} &= y_i, \quad i=0, \dots, n \\ \overset{k+1}{i} &= \frac{x_i - x_k}{\overset{k}{i} - \overset{k}{k}} \end{aligned}$$

Falls die inversen Differenzen $\overset{k}{k}$ existieren und ungleich Null sind, gilt $c_k = \overset{k}{k}$. Falls das nicht gilt, ist die rationale Interpolation in dieser Form nicht möglich.

Zur Berechnung von $R(\xi)$ wird der Kettenbruch (5) von "unten nach oben" ausgewertet.

4. Programmbeschreibungen

In den verschiedenen Unterprogrammen werden die HP 41 C-Register RO0 bis RO7 benutzt. Die Koeffizienten $\{a_k\}, \{b_k\} \dots$ können ab Register RO8 abgelegt werden. Die einzelnen Werte $\{..\}$ müssen fortlaufend gespeichert werden, z.B. x_0 -RO8, x_1 -RO9, x_2 -R10.. Für die Rechnungen werden weiterhin die Anfangsadressen der Koeffizientenvektoren benötigt, so gibt z.B. $A(x_0)$ die Adresse von x_0 an, im obigen Beispiel also $A(x_0) = 8$. Was in welchem Register abgespeichert ist, zeigt die folgende Tabelle. Ein π zeigt dabei an, daß das entsprechende Register neben den bezeichneten ebenfalls benutzt wird. Die Bedeutung von i, j, k, l ist aus den Beschreibungen der benutzten Algorithmen ersichtlich.

UPrgm	RO0	RO1	RO2	RO3	RO4	RO5	RO6	RO7
PM		n	$A(v_0)$	$A(w_0)$		i	j	
DD	π	n	$A(y_0)$	$A(b_0)$	$A(x_0)$	i	π	k
MH	ξ	n	$A(b_0)$	$A(c_0)$	$A(x_0)$	i	j	k
BM	ξ	n		$A(b_0)$	$A(x_0)$	i	j	
NH	ξ	n	$A(a_0)$	$A(c_0)$		π	j	k
BN	ξ	n	$A(a_0)$			i		
ID	l	n	$A(y_0)$	$A(c_0)$	$A(x_0)$	i	j	k
BK	ξ	n		$A(c_0)$	$A(x_0)$	i	j	

4.1 Unterprogramm PM

Verschiebt $\{v_k\}$ nach $\{w_k\}$, benötigt $A(v_0)$, $A(w_0)$, n .
Wenn $A(v_0) = A(w_0)$ ist, wird nichts verschoben.

4.2 Unterprogramm DD

Berechnet aus $\{x_k\}$, $\{y_k\}$ die dividierten Differenzen $b_k = \Delta_k^k$.
Benötigt $A(x_0)$, $A(y_0)$, $A(b_0)$, n .

4.3 Unterprogramm MH

Modifiziertes Hornerschema. Berechnet aus $\{x_k\}$, $\{b_k\}$ in (3) die Werte $\{c_k\}$ in (4). Benötigt $A(x_0)$, $A(b_0)$, $A(c_0)$, n , ξ .

4.4 Unterprogramm BM

Berechnet $q(\xi)$ und $q'(\xi)$ in (3)
Benötigt $A(x_0)$, $A(b_0)$, n , ξ .

4.5 Unterprogramm NH

Normales Hornerschema. Berechnet $\{c_k\}$ in (4) aus $\{a_k\}$ in (1).
Benötigt $A(a_0)$, $A(b_0)$, n , ξ .

4.6 Unterprogramm BN

Berechnet $p(\xi)$ und $p'(\xi)$ aus (1).
Benötigt $A(a_0)$, n , ξ .

4.7 Unterprogramm ID

Berechnet aus $\{x_k\}$, $\{y_k\}$ die inversen Differenzen $\{c_k\}$ in (5).
Benötigt $A(x_0)$, $A(y_0)$, $A(c_0)$, n .

4.8 Unterprogramm BK

Berechnet den Thieleschen Kettenbruch $R(\xi)$ in (5).
Benötigt $A(x_0)$, $A(c_0)$, n , ξ .

Es ist im allgemeinen möglich, daß die zu berechnenden Koeffizienten die alten Werte überschreiben, z.B. ist im Unterprogramm DD $A(b_0) = A(y_0)$ möglich. Dies spart zwar Speicherplatz, nur sind dann die alten Werte natürlich nicht mehr ansprechbar.

5. Algorithmen

Im folgenden werden für die im Abschnitt 4. beschriebenen Unterprogramme die benutzten Algorithmen angegeben. Die verwendete Notation dürfte allgemein verständlich sein, so bedeutet z.B. die Schreibweise "for $k = i(j)n$ ", daß entsprechend dem BASIC-Befehl "FOR $K = I$ TO N STEP J " verfahren wird. Die Algorithmen orientieren sich stark an den HP 41 C-Möglichkeiten; für eine Übertragung in höhere Programmiersprachen kann man sie noch vereinfachen. Die Schreibweise "PM(v_0, w_0, n)" bedeutet, daß die $n+1$ Werte $\{v_k\}$ mit dem Unterprogramm PM nach $\{w_k\}$ verschoben werden.

Die Unterprogramme BM und BN liefern die Werte $q(\xi)$, $q'(\xi)$ (bzw. $p(\xi)$, $p'(\xi)$) in den Stackregistern X und Y. $R(\xi)$ wird von BK in X geliefert.

BM und BN liefern neben den Funktionswerten des Interpolationspolynoms auch die Werte der ersten Ableitung. Wem die angegebenen Unterprogramme zu langsam oder zu speicherplatzaufwendig sind, kann leicht an Hand der Algorithmenlistings diese so modifizieren, daß sie nur noch die Funktionswerte berechnen.

PM

```

if  $A(v_0) = A(w_0)$  return
j=0
for i=0(1)n
   $w_i = v_i$ 
   $j = j + 1$ 
next i

```

ME

```

PM( $b_0, c_0, n$ )
for k=n(-1)1
  i=n;  $x=c_i$ ; i=i-1
  for j=k-1(-1)0
     $x = (\xi - x_j) \cdot x + c_j$ 
     $c_i = x$ ; i=i-1
  next j
next k

```

DD

```

PM( $y_0, b_0, n$ )
for k=1(1)n
   $y = b_{k-1}$ 
  for i=k(1)n
     $x = (y - b_i) / (x_{i-k} - x_i)$ 
     $y = b_i$ ;  $b_i = x$ 
  next i
next k

```

BK

```

 $x = c_n$ ;  $j = n - 1$ 
for i=n-1(-1)0
   $x = (\xi - x_i) / x + c_j$ 
   $j = j - 1$ 
next i

```

BN

```

 $y = 0$ 
 $x = a_n$ 
for i=n-1(-1)0
   $y = y \cdot \xi + x$ 
   $x = x \cdot \xi + a_i$ 
next i

```

BM

```

j=n;  $y=0$ ;  $x=b_n$ ;  $j=j-1$ 
for i=n-1(-1)0
   $y = (\xi - x_i) \cdot y + x$ 
   $x = (\xi - x_i) \cdot x + b_j$ 
   $j = j - 1$ 
next i

```

ID

```

PM( $y_0, c_0, n$ )
for i=0(1)n-1
  l=i; k=l+1
  for j=i+1(1)n
     $c_k = (x_i - x_j) / (c_1 - c_k)$ 
     $k = k + 1$ 
  next j
  l=l+1
next i

```

NH

```

PM( $a_0, c_0, n$ )
for j=0(1)n
   $x = 0$ 
  for k=n(-1)j
     $x = c_k + j \cdot x$ ;  $c_k = x$ 
  next k
next j

```

01*LBL "PM"	35 RTN	01*LBL "BN"	01*LBL "NN"	25 *
02 RCL 02	36*LBL 04	02 RCL 03	02 XEQ "PM"	26 RCL IND 06
03 RCL 03	37 STO Y	03 E	03 RCL 03	27 +
04 X=Y?	38 RCL 01	04 -	04 RCL 01	28 DSE 06
05 RTN	39 +	05 E3	05 +	29 DSE 05
06 RCL 01	40 E	06 /	06 E3	30 GTO 01
07 +	41 -	07 RCL 03	07 /	31 END
08 E3	42 E3	08 RCL 01	08 RCL 03	
09 /	43 /	09 +	09 +	01*LBL "BK"
10 RCL 03	44 +	10 +	10 STO 06	02 RCL 04
11 +	45 END	11 STO 05	11*LBL 01	03 E
12 STO 05		12 ,	12 RCL 01	04 -
13 RCL 02		13 RCL IND 05	13 RCL 03	05 ,1
14 ,9	01*LBL "DD"	14 DSE 05	14 +	06 %
15 +	02 XEQ "PM"	15*LBL 01	15 STO 07	07 +
16 STO 06	03 RCL IND 03	16 RCL 00	16 RCL 06	08 RCL 01
17*LBL 01	04 STO 00	17 ST* Z	17 STO 05	09 +
18 RCL IND 06	05 RCL 03	18 X<>Y	18 RCL 00	10 STO 05
19 STO IND 05	06 RCL 01	19 ST+ Z	19 ENTER↑	11 RCL 03
20 ISG 06	07 +	20 *	20 ENTER↑	12 RCL 01
21 ISG 05	08 E3	21 RCL IND 05	21 ENTER↑	13 +
22 GTO 01	09 /	22 +	22 CLX	14 STO 06
23 END	10 RCL 03	23 DSE 05	23*LBL 02	15 RCL IND 06
	11 +	24 GTO 01	24 *	16 DSE 06
	12 E	25 END	25 RCL IND 07	17*LBL 01
	13 +		26 +	18 1/X
01*LBL "ID"	14 STO 07	01*LBL "MH"	27 STO IND 07	19 RCL 00
02 XEQ "PM"	15*LBL 01	02 XEQ "PM"	28 DSE 07	20 RCL IND 05
03 RCL 04	16 RCL 04	03 RCL 01	29 ISG 05	21 -
04 XEQ 04	17 ,9	04 STO 07	30 GTO 02	22 *
05 STO 05	18 +	05 RCL 04	31 ISG 06	23 RCL IND 06
06 RCL 03	19 STO 06	06 E	32 GTO 01	24 +
07 XEQ 04	20 RCL 07	07 -	33 END	25 DSE 06
08 STO 00	21 STO 05	08 ,1		26 DSE 05
09*LBL 01	22 +	09 %		27 GTO 01
10 RCL 05	23 RCL 03	10 +	01*LBL "BM"	28 END
11 1,001	24 -	11 STO 06	02 RCL 03	
12 +	25 X<> 00	12*LBL 02	03 RCL 01	LBL"PM
13 STO 06	26*LBL 02	13 RCL 03	04 +	END 37 BYTES
14 RCL 00	27 RCL IND 05	14 RCL 01	05 STO 06	LBL"DD
15 LASTX	28 -	15 +	06 RCL 04	END 67 BYTES
16 +	29 RCL IND 06	16 STO 05	07 E	LBL"MH
17 STO 07	30 RCL IND 00	17 RCL 07	08 -	END 56 BYTES
18*LBL 02	31 -	18 ST+ 06	09 ,1	LBL"BM
19 RCL IND 05	32 /	19 RCL IND 05	10 %	END 48 BYTES
20 RCL IND 06	33 X<> IND 05	20 DSE 05	11 +	LBL"NH
21 -	34 ISG 00	21*LBL 03	12 RCL 01	END 51 BYTES
22 RCL IND 00	35 ISG 06	22 RCL 00	13 +	LBL"BN
23 RCL IND 07	36 ISG 05	23 RCL IND 06	14 STO 05	END 40 BYTES
24 -	37 GTO 02	24 -	15 ,	LBL"ID
25 /	38 RCL IND 07	25 *	16 RCL IND 06	END 76 BYTES
26 STO IND 07	39 STO 00	26 RCL IND 05	17 DSE 06	LBL"BK
27 ISG 07	40 ISG 07	27 +	18*LBL 01	END 43 BYTES
28*LBL 03	41 GTO 01	28 STO IND 05	19 RCL 00	
29 ISG 06	42 END	29 DSE 05	20 RCL IND 05	
30 GTO 02		30 DSE 06	21 -	
31 ISG 00		31 GTO 03	22 ST* Z	
32*LBL 03		32 DSE 07	23 X<>Y	
33 ISG 05		33 GTO 02	24 ST+ Z	
34 GTO 01		34 END		

7. Beispiele

Gegeben seien die folgenden x- und y-Werte:

x	-2	-1	0	1	2
y	81	16	1	0	1

Man löse mit Polynominterpolation folgende Aufgaben:

- Bestimme $q(x)$, d.h. $\{b_k\}$ in (3), sowie $q(.5)$ und $q'(.5)$
- Bestimme $p(x)$, d.h. $\{c_k\}$ für $\xi=0$, $p(.5)$ und $p'(.5)$
- Bestimme $p_1(x)$, d.h. $\{c_k\}$ für $\xi=1$

Lösungen:

- a) Es ist $n=4$, wähle $A(x_0)=10$, $A(y_0)=15$ und $A(b_0)=20$.

```
-2 STO 10  -1 STO 11  0 STO 12  1 STO 13  2 STO 14
81 STO 15 16 STO 16  1 STO 17  0 STO 18  1 STO 19
4 STO 01 10 STO 04 15 STO 02 20 STO 03
```

Die $\{b_k\}$ erhält man dann mit XEQ 'DD

$b_0=81$, $b_1=-65$, $b_2=25$, $b_3=-6$, $b_4=1$

.5 STO 00 YEQ 'BM ergibt: $q(.5)=0.0625=1/16$, $q'(.5)=-.5=-1/2$

- b) Die $\{c_k\}$ sollen die $\{b_k\}$ überschreiben, es ist $\xi=0$.

```
20 STO 02  0 STO 00  XEQ 'MH ergibt:
```

$c_0=1$, $c_1=-4$, $c_2=6$, $c_3=-4$, $c_4=1$, d.h. für $p(x)$ erhält man:

$p(x) = 1 - 4x + 6x^2 - 4x^3 + x^4$

.5 STO 00 XEQ 'BN ergibt wieder $p(.5)=0.0625$ und $p'(.5)=-0.5$

- c) Die neuen $\{c_k\}$ -Werte sollen wieder die alten überschreiben.

```
1 STO 00 XEQ 'NH ergibt  $p_1(x) = (x-1)^4$ .
```

Ein zweites Beispiel soll die rationale Interpolation behandeln.
Dazu seien die folgenden x- und y-Werte gegeben:

x	-2	-1	0	1	2
y	.2	0	1	2	1.8

Es soll der Thielesche Kettenbruch, d.h. die $\{c_k\}$ in (5), berechnet werden, außerdem ist nach interpolierten y-Werten für die x-Werte -1.5, -0.5, 0.5, 1.5 gefragt.

Lösung:

Es ist wieder $n=4$. Wähle $A(x_0)=8$, $A(y_0)=14$, $A(c_0)=20$.

```
-2 STO 08  -1 STO 09  0 STO 10  1 STO 11  2 STO 12
.2 STO 14  0 STO 15  1 STO 16  2 STO 17  1.8 STO 18
4 STO 01  8 STO 04  14 STO 02  20 STO 03
```

Mit XEQ 'ID erhält man jetzt die Koeffizienten $\{c_k\}$

$$\begin{aligned} c_0 &= 0.2000 = 1/5 & R(x) &= \frac{1}{5} + \frac{x+2}{-5 + \frac{x+1}{\frac{3}{15} + \frac{x}{6 + \frac{x-1}{2/3}}}} \\ c_1 &= 5.0000 = -5 \\ c_2 &= 1.3333 = 2/15 \\ c_3 &= 6.0000 = 6 \\ c_4 &= 0.6667 = 2/3 \end{aligned}$$

Berechnet man den Kettenbruch von "unten nach oben", so ergibt sich nach einigen Umformungen die interpolierende rationale Funktion $R(x)$

$$R(x) = \frac{1 + 2x + x^2}{1 + x^2}$$

Die geforderten interpolierten Werte werden wie folgt berechnet:

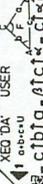
-1.5 STO 00	XEQ 'BK	R(-1.5) = 0.0769
-0.5 STO 00	XEQ 'BK	R(-0.5) = 0.2000
0.5 STO 00	XEQ 'BK	R(0.5) = 1.8000
1.5 STO 00	XEQ 'BK	R(1.5) = 1.9231

Zu den angegebenen Beispielen: Sie wurden so gewählt, daß die zu interpolierenden Funktionen von den entsprechenden Unterprogrammen reproduziert werden. Das ist didaktisch vielleicht nicht besonders geschickt, zeigt aber, daß die Unterprogramme richtig arbeiten.

Anmerkungen

1. Polynominterpolation ist immer möglich, falls die $\{x_k\}$ paarweise verschieden sind. Wenn n groß ist, kann es aber sein, daß (besonders nahe der Endpunkte) starke Oszillationen auftreten. Das kann durch Zeichnen der Funktion kontrolliert werden. Bei der rationalen Interpolation kann es vorkommen, daß Differenzen verschwinden und ID/BK mit 'DATA ERROR' (Division durch 0) abbrechen.
2. Selbstverständlich müssen nicht unbedingt alle Unterprogramme gleichzeitig im Speicher sein. Außerdem gibt es natürlich weitere Möglichkeiten, Speicherplatz zu sparen: Benötigt man z.B. bei der rationalen Interpolation die y -Werte nicht weiter, so kann man $A(y_0)$ in RO3 speichern und löscht den Schritt "XEQ 'PM" in ID. Das Unterprogramm PM wird dann nicht benötigt, und die $\{c_k\}$ überschreiben die $\{y_k\}$.

Dreiecksberechnungen
(weiterentwickelt für
HP-41C aus SD-07A für HP 67)

XEQ DA USER

 SIZE 008
 Fabrik-A-B-C-U
 c b f a g h i e r c i k f a c b f r c i b f r

Stroinski
Februar 81

01 *LBL "DR"	Haupt-Label	54 *LBL B	Routine uSw	106 *LBL E	Routine SSW	141 *LBL 04	Schluss-Routine
02 SF 27	User-Modus	55 STO 04	a speichern	107 STO 06	a speichern	142 RCL 00	Fläche
03 *STOP		56 RDN 03	c speichern	108 SIN	b speichern	143 RCL 01	berechnen
04 *LBL A		57 STO 03	b speichern	109 X<Y	a speichern	144 *	und
05 STO 01		58 RDN 03	a speichern	110 STO 02	b speichern	145 /	speichern
06 RDN 01		59 STO 05	b speichern	111 * X<Y	c speichern	146 STO 00	Ausgabe-Routine
07 STO 02		60 RCL 04	a errechnen	112 Y<Y	a errechnen	147 /	Umfang errechnen
08 STO 03		61 XEQ 00	a sin $\beta = h$	113 STO 03	b errechnen	148 *LBL 01	und
09 STO 03		62 STO 06	und	114 * X<Y	c errechnen	149 SF 21	
10 RDN		63 *LBL 06	c sin $\beta = h$	115 ASIN	a speichern	150 * 007	
11 +		64 RCL 05	c cos β bilden	116 STO 05	a errechnen	151 RCL 01	
12 +		65 RCL 03		117 RCL 06	a speichern	152 RCL 02	
13 +		66 P-R		118 XEQ 00	a errechnen	153 RCL 03	
14 2		67 X<Y		119 STO 04	a speichern	154 +	
15 /		68 STO 00	h speichern	120 XEQ 06	a errechnen	155 +	
16 STO 07		69 RCL 06	sin γ und cos γ	121 *LBL 05	a errechnen	156 STO 07	
17 X+2		70 1	bilden sin β	122 RCL 03	a errechnen	157 *Fab c < A < A	
18 LASTX		71 P-R	b = c * sin γ	123 RCL 02	1. Lsg. rechnen	B < C U-	
19 RCL 02		72 RDN		124 X<Y?	2. Lösung	158 AVIEW	
20 *		73 /	b speichern	125 STO 02	möglich?	159 *LBL 03	
21 -		74 STO 02	a = c * cos β	126 RCL 05		160 RDN	
22 RCL 03		75 R↑	+ b * cos γ	127 COS	Für 2. Lösung	161 VIEW IND	
23 RCL 01		76 *	a speichern	128 CHS	$\beta = \pi - \beta$	X	
24 *		77 +		129 ACOS	a errechnen	162 ENTER↑	
25 /		78 STO 01		130 STO 05	a errechnen	163 3	
26 SORT		79 STO 04		131 RCL 06	a errechnen	164 3	
27 ACOS		80 *LBL C		132 XEQ 00	a errechnen	165 MOD	
28 2		81 STO 06	Routine SuW	133 XEQ 00	a errechnen	166 INT	
29 *		82 X<Y	a speichern	134 STO 04	a errechnen	167 X=0?	
30 STO 05		83 STO 04	a speichern	135 *LBL 00	a errechnen	168 AD	
31 SIN		84 XEQ 00	b speichern	136 +	Unterprogramm	169 ISG Y	
32 RCL 03		85 STO 05		137 COS	zur Berechnung	170 STO 03	
33 * STO 00		86 X<Y		138 CHS	des 3. Winkels	171 *LBL 02	
34 STO 00		87 STO 03		139 ACOS		172 ADV	
35 RCL 07		88 STO 06		140 RTN		173 END	
36 X+2		89 *LBL D					
37 LASTX		90 STO 02	Routine SuS				
38 RCL 03		91 RDN	b speichern				
39 *		92 STO 04	a speichern				
40 -		93 RDN 03	a speichern				
41 RCL 02		94 STO 03	c speichern				
42 RCL 01		95 RCL 04	b sin $\alpha = h_c$ und				
43		96 RCL 02	b cos α				
44		97 P-R	bilden				
45 SORT		98 RCL 03	h _c				
46 ACOS		99	b * cos $\alpha - c$				
47 2		100 R-P	a speichern				
48		101 STO 01	a speichern				
49 XEQ 00		102 RCL 03	Stack ordnen				
50 STO 04		103 RCL 02					
51 XEQ 00		104 RCL 01					
52 STO 04		105 STO H					
53 STO 04							

141 *LBL 04
 142 RCL 00
 143 RCL 01
 144 *
 145 /
 146 STO 00
 147 /
 148 *LBL 01
 149 SF 21
 150 * 007
 151 RCL 01
 152 RCL 02
 153 RCL 03
 154 +
 155 +
 156 STO 07
 157 *Fab c < A < A
 B < C U-
 158 AVIEW
 159 *LBL 03
 160 RDN
 161 VIEW IND
 X
 162 ENTER↑
 163 3
 164 3
 165 MOD
 166 INT
 167 X=0?
 168 AD
 169 ISG Y
 170 STO 03
 171 *LBL 02
 172 ADV
 173 END

SIZE 008
 TOT. REG. 40
 USER MODUS:
 A,B,C,D,E.
 WIEDERHOLEN:
 Im Fall E werten, falls vorhanden, zwei Werte-Serien ausgegeben!
 FALL A...D:
 XEQ 01
 FALL E:
 XEQ 01
 ZULETZT GEZEIGTE LSG. ALTERNATIV-LOESUNG.

Register-Inhalte:
 00 Fläche
 01 b
 02 b
 03 c
 04 X (A)
 05 A (B)
 06 A (C)
 07 Umfang

Autor: Andreas Trögel

Plotten einer Funktion

LBL PL SIZE O17
389 Bytes

Peripherie:
Graphik-Drucker

Dieses Programm plottet den Graphen einer Funktion. Das Programm ist für den umgebauten Drucker ausgelegt. Im Programm wird eine maximale Druckbreite von 133 Spalten (= 19 Zeichen) angenommen. Für größere Druckbreiten müssen nur 2 Programmzeilen geändert werden (siehe Programmlisting).

01*LBL "PL"	52 ADV	103 GTO 03	154 GTO 09
02 CF 00	53 FIX 3	104*LBL 02	155 LASTX
03 AON	54 "STEPS OF "	105 SF 00	156 FRC
04 "FUNKTION ?"	55 ARCL 11	106 ,	157 ST+ IND 13
05 PROMPT	56 ACA	107 STO IND 13	158 CLX
06 AOFF	57 ADV	108 ISG 13	159 STO IND 14
07 ASTO 06	58 ADV	109 GTO 02	160*LBL 09
08 "Y MIN?"	59 "-----"	110*LBL 03	161 ISG 14
09 PROMPT	60 ASTO L	111 E-3	162 GTO 08
10 STO 07	61 ARCL L	112 ST+ 00	163 5 E-3
11 "Y MAX?"	62 ARCL L	113 ST+ X	164 STO 13
12 PROMPT	63 "+"	114 ST+ 01	165 RCL IND 13
13 STO 08	64 ACA	115 ST+ X	166 STO 14
14 "X MIN?"	65 ADV	116 ST+ 02	167*LBL 10
15 PROMPT	66 132 +SPALTENBREITE-1	117 ST+ X	168 RCL IND 13
16 STO 09	67 RCL 08	118 ST+ 03	169 INT
17 "X MAX?"	68 RCL 07	119 ST+ X	170 X=0?
18 PROMPT	69 -	120 ST+ 04	171 GTO 11
19 STO 10	70 /	121 ST+ X	172 RCL 14
20 "X INC?"	71 STO 12	122 ST+ 05	173 -
21 PROMPT	72*LBL 12	123 5 E-3	174 E
22 STO 11	73 5 E-3	124 STO 13	175 -
23 "PLOT OF "	74 STO 13	125*LBL 06	176 X)0?
24 ARCL 06	75*LBL 01	126 RCL 13	177 SKPCOL
25 ACA	76 RCL 09	127 STO 14	178 RCL IND 13
26 ADV	77 XEQ IND 06	128 E	179 INT
27 ADV	78 RCL 07	129 -	180 STO 14.
28 FIX 1	79 -	130 RCL IND X	181 LASTX
29 "YMIN: "	80 RCL 12	131*LBL 07	182 FRC
30 ACA	81 *	132 RCL IND 14	183 E3
31 RCL 07	82 E	133 X)Y?	184 *
32 ACX	83 +	134 X)Y	185 ACCOL
33 ADV	84 FIX 0	135 STO IND 14	186*LBL 11
34 ADV	85 RND	136 RDN	187 ISG 13
35 "YMAX: "	86 133 +SPALTENBREITE	137 ISG 14	188 GTO 10
36 ACA	87 X)Y	138 GTO 07	189 133
37 RCL 08	88 X)Y?	139 STO IND Y	190 RCL 14
38 ACX	89 ,	140 ISG 13	191 -
39 ADV	90 ENTER↑	141 GTO 06	192 SKPCOL
40 ADV	91 ,	142 4 E-3	193 ADV
41 "XMIN: "	92 X)Y?	143 STO 14	194 FC?C 00
42 ACA	93 X)Y	144 E-3	195 GTO 12
43 RCL 09	94 STO IND 13	145 +	196 BEEP
44 ACX	95 RCL 11	146 STO 13	197 ACA
45 ADV	96 ST+ 09	147*LBL 08	198 ADV
46 ADV	97 RCL 10	148 ISG 13	199 ADV
47 "XMAX: "	98 RCL 09	149 RCL IND 13	200 ADV
48 ACA	99 X)Y?	150 INT	201 ADV
49 RCL 10	100 GTO 02	151 RCL IND 14	202 ADV
50 ACX	101 ISG 13	152 INT	203 END
51 ADV	102 GTO 01	153 X)Y?	

Autor: Andreas Trögel

Donald Duck, Goofy

LBL DD, LBL GO

SIZE 000
714 Bytes

Peripherie: synthetische
Graphik-Drucker Textzeilen

Hier ein weiteres Beispiel für Graphik mit dem umgebauten HP-Drucker.
Programm einlesen, R/S...



R/S...



Programm-Listing:

Rechts neben den Textzeilen stehen die Textbytes dezimal aufgeschlüsselt.

01*LBL "DD"		40 "0+++00+##"	16 3 247 206 17 204 16
02 "G-AN"		41 XEQ 01	31 128 0 0 0 0 0
03 PROMPT		42 1	
04 ADV		43 SKPCOL	
05 SF 12		44 ADV	
06 "Q @@@0αδ @α"	17 192 64 64 64 128 129	45 "0τα 0+Δ"	16 194 116 130 4 142 32
07 XEQ 01	16 4 8 32 64 129 4	46 XEQ 01	16 129 3 227 199 136 8
08 "00AΓ>(HQ xΔ0+"	16 16 65 6 12 40 72 17	47 "0 02xα+β"	16 32 129 231 207 144 32
09 XEQ 01	32 129 2 8 48 0	48 XEQ 01	16 225 50 2 4 15 112
10 ADV		49 2	
11 "Q+ΓΓr##0Q+↓rΓ(x"	17 3 7 6 14 31 64 17 3 7	50 SKPCOL	
12 RCL \	17 30 60 120	51 ADV	
13 ACSPEC		52 "0b#0000xΔ(P"	16 226 98 36 24 16 48 17
14 3		53 XEQ 01	64 129 2 13 40 80
15 SKPCOL		54 "0 xΔ0 0xΔα+"	16 192 161 2 20 24 32 17
16 XEQ 02		55 XEQ 01	129 2 8 11 157 0
17 "0α(xQv0+*"	17 225 199 135 22 60 120	56 2	
18 XEQ 01	17 195 135 12 24 0 0	57 SKPCOL	
19 0		58 ADV	
20 ACCOL		59 "09eα(PQ.Jα&b"	16 57 130 101 20 40 80
21 2		60 XEQ 01	17 74 133 10 20 38 98
22 ACCOL		61 "0 @'+++"	16 192 64 96 0 0 0
23 124		62 XEQ 02	
24 ACCOL		63 4	
25 ADV		64 SKPCOL	
26 "0+0+ ##0Eαδ"	16 48 192 16 27 193 129	65 ADV	
27 XEQ 01	9 137 18 0 0 0 0	66 ADV	
28 "0H+++x0-Qxαδ"	16 72 146 2 16 1 17 2 4	67 ADV	
29 XEQ 01	8 34 0 0 0	68 ADV	
30 "01μrΓ"	16 136 49 140 12 14 6	69 ADV	
31 XEQ 02		70 RTN	
32 ADV		71 GTO "GO"	
33 "Qα↓+x0Δ0+*"	17 248 4 7 1 1 2 16 8 16	72*LBL 01	
34 XEQ 01	206 0 0 0	73 RCL \	
35 "Q ' @μQ+##"	17 192 96 32 64 129 12	74 ACSPEC	
36 XEQ 01	17 192 15 192 0 0 0	75*LBL 02	
37 ADV		76 RCL [
38 "00# ##0<↓"	17 224 48 15 192 16 3 7	77 ACSPEC	
39 XEQ 01	143 159 128 0 0 0	78 RTN	

79+LBL "G0"						115 XEQ 02
80 SF 12						116 ADV
81 "G-AN"						117 "0+ αΔ00+0"L" 16 32 130 4 8 16 24 35
82 PROMPT						118 XEQ 01 137 34 76 0 0 0
83 ADV						119 "0+E0xp(μΓx" 16 160 198 136 131 15 30
84 "0= θαΔ " 16 1 132 192 192 129 129						120 XEQ 01 16 120 112 40 12 6 2
85 XEQ 01 16 4 8 32 128 128 129						121 "0+αf +" 16 4 15 223 188 0 0
86 "0B0+0"						122 XEQ 02
87 XEQ 02						123 ADV
88 ADV						124 96
89 "0+0 A2e00 0xα" 16 32 65 50 28 16 16 32						125 ACCOL
90 XEQ 01 64 129 2 4 0 0						126 "0fE(x0α" 17 195 135 15 30 60 120
91 "0"p00+0"						127 XEQ 01 17 243 247 227 192 130 4
92 XEQ 02 17 34 112 130 24 0 0						128 "00 00+0 Δ" 16 16 32 64 64 17 131
93 ADV						129 XEQ 01 199 239 223 191 124 0 0
94 "0= r# 16 1 199 207 211 167 124						130 ADV
95 RCL \ 16 6 2 10 30 46 0						131 "0Δ0 Γ0 E" 16 8 16 32 193 131 6 16
96 ACSPEC						132 RCL \ 32 243 239 223 159 30
97 5						133 ACSPEC
98 SKPCOL						134 2
99 XEQ 02						135 ACCOL
100 "0z Δ00+0" 17 122 244 193 195 191						136 ACCOL
101 XEQ 01 124 17 195 135 232 64 0						137 10
102 ADV 0						138 SKPCOL
103 "0=xΔx0Δ+ 00" 17 1 2 7 128 129 2 16 8						139 XEQ 02
104 XEQ 01 32 130 24 64 0						140 2
105 "0c5+(00+0Δ>" 17 99 53 143 60 24 16 13						141 ACCOL
106 XEQ 01 220 31 62 0 0 0						142 0
107 "0 ** 16 251 247 231 192 0 0						143 ACCOL
108 XEQ 02						144 ADV
109 ADV						145 ADV
110 "0xαΔ0 HQ+0+0x" 16 123 4 8 16 32 72 17						146 ADV
111 XEQ 01 194 2 0 0 0 0						147 ADV
112 "00+ 0"0x0Δb" 16 16 32 16 130 96 17						148 ADV
113 XEQ 01 195 135 2 5 7 98 0						149 END
114 "0P0+0" 16 80 247 142 24 0 0						

Autor: Andreas Trögel

HIGH-RESOLUTION-PLOTTER FÜR 2 FUNKTIONEN

LBL PLOT SIZE 023 Drucker (MAN) Synthetisch
 426 Bytes

Das Programm PLOT dient zur graphischen Darstellung zweier Funktionen. Der Unterschied zu PRPLOT besteht in der deutlich höheren Auflösung und der Tatsache, daß man 2 Funktionen darstellen kann.

Leider benötigt dieses Programm zur Ausführung auch wesentlich mehr Zeit als PRPLOT.

Zuerst werden beide Funktionen in den Programmspeicher eingegeben. Will man nur eine Funktion drucken, kann man als zweite Funktion die X-Achse drucken lassen. Die Funktion für die X-Achse ist schon im Programm eingebaut und wird bei der Abfrage der Funktionsnamen mit X=0 eingegeben.

Das Programm wird mit XEQ PLOT gestartet, worauf der Rechner nach beiden Funktionsnamen, den Druckbereichen und der Schrittweite für x fragt. Im Anschluß daran beginnt der Ausdruck.

Einen neuen Ausdruck der Funktionen über anderen Bereichen erhält man durch XEQ NEU.

01*LBL "PLOT"	31 "X INC?"	61 9.022	91 ST+ 07
02 CF 12	32 PROMPT	62 STO 00	92 RCL 07
03 CF 13	33 STO 04	63*LBL 01	93 RCL 03
04 AON	34 "PLOT "	64 RCL 07	94 XCY?
05 "NAME 1?"	35 ARCL 05	65 XEQ IND 05	95 GTO 04
06 PROMPT	36 "+ AND "	66 CF 00	96 ISG 00
07 ASTO 05	37 ARCL 06	67*LBL 02	97 GTO 01
08 "NAME 2?"	38 PRA	68 RCL 02	98 XEQ 05
09 PROMPT	39 ENG 3	69 -	99 4
10 ASTO 06	40 "Y: "	70 RCL 00	100 RCL 04
11 AOFF	41 ARCL 02	71 *	101 *
12*LBL "NEU"	42 "+ TO "	72 1	102 ST+ 07
13 167	43 ARCL 00	73 +	103 RCL 07
14 "Y MIN?"	44 PRA	74 FIX 0	104 RCL 03
15 PROMPT	45 "X: "	75 RND	105 XCY?
16 STO 02	46 ARCL 01	76 160	106 PRA
17 "Y MAX?"	47 "+ TO "	77 XCY?	107 XCY?
18 PROMPT	48 ARCL 03	78 X<0?	108 RTN
19 STO 00	49 PRA	79 CLX	109 GTO 00
20 X<Y	50 "dx="	80 X>Y?	110*LBL 04
21 -	51 ARCL 04	81 CLX	111 SF 01
22 /	52 PRA	82 STO IND 00	112 CLX
23 STO 00	53 "-----"	83 FS? 00	113 STO IND 00
24 "X MIN?"	54 ASTO L	84 RTN	114 ISG 00
25 PROMPT	55 ARCL L	85 ISG 00	115 GTO 04
26 STO 01	56 ARCL L	86 RCL 07	116*LBL 05
27 STO 07	57 ARCL L	87 SF 00	117 9.02302
28 "X MAX?"	58 PRA	88 XEQ IND 06	118 STO 00
29 PROMPT	59 CF 01	89 XEQ 02	119 1
30 STO 03	60*LBL 00	90 RCL 04	120 +

121 STO 01	144 ISG 00	167 GTO 12	190 X<> 01
122 5 E-4	145 GTO 07	168 X<Y?	191 -
123*LBL 03	146 STO IND Y	169 GTO 13	192 1
124 ST+ X	147 ISG 01	170 X<>Y	193 -
125 ST+ IND 00	148 GTO 06	171 -	194 SKPCOL
126 ST+ IND 01	149 9.021	172*LBL 13	195 RCL IND 00
127 ISG 00	150 STO 00	173 ST+ IND 01	196 FRC
128 ISG 01	151 1 E-3	174*LBL 12	197 1 E3
129 GTO 03	152 +	175 CLX	198 *
130 10.022	153 STO 01	176 STO IND 00	199 ACCOL
131 STO 01	154*LBL 08	177*LBL 09	200*LBL 11
132*LBL 06	155 ISG 01	178 ISG 00	201 ISG 00
133 RCL 01	156 RCL IND 00	179 GTO 08	202 GTO 10
134 STO 00	157 INT	180 9.022	203 PRBUF
135 1	158 RCL IND 01	181 STO 00	204 FS?C 01
136 -	159 INT	182 CLX	205 PRA
137 RCL IND X	160 X*Y?	183 STO 01	206 RTN
138*LBL 07	161 GTO 09	184*LBL 10	207*LBL "X=0"
139 RCL IND 00	162 LASTX	185 RCL IND 00	208 .
140 X<Y?	163 FRC	186 INT	209 .END.
141 X<>Y	164 RCL IND 00	187 X=0?	
142 STO IND 00	165 FRC	188 GTO 11	
143 RDN	166 X=Y?	189 ENTER†	

Beispiel :

PLOT SIN+2 AND COS
 Y: -1.000E0 TO 1.000E0
 X: 0.000E0 TO 360.0E0
 Δx=1.000E0



Autor: Andreas Trögel

MULTILOTTER (für bis zu 6 Funktionen)

LBL PLOT SIZE O27 Drucker (MAN) Synthetik
657 Bytes

Die Plottfunktion des HP-Druckers in der ROM-Routine ist sehr be-
schränkt. Es läßt sich mit der PRPLOT-Anweisung nur eine Funktion
ausdrucken. Mit dem Programm PLOT lassen sich bis zu 6 Funktionen
gleichzeitig drucken. Außerdem hat man die Möglichkeit, beliebig
viele Streifen (SPLITS) zu drucken, die dann später nebeneinander-
geklebt werden können.

Eingabehinweise:

1. Anzahl der zu plottenden Funktionen
2. Funktionsnamen
3. Anzahl der Splits
4. x-Werte, y-Werte (wie bei PRPLOT)

XEQ NEW erzeugt einen Neustart des Programms mit gleichen Funktions-
namen.

01*LBL "PLOT"	36 ISG Y	71 X<>Y	106 " Y: "
02*LBL 00	37 GTO 01	72 X<=Y?	107 ARCL 01
03 6	38 AOFF	73 GTO 04	108 "t to "
04 "NO. OF F(X)?"	39*LBL "NEW"	74 "X INC?"	109 ARCL 02
05 PROMPT	40*LBL 02	75 PROMPT	110 "t +"
06 X)0?	41 "SPLITS"	76 X)0?	111 PRA
07 X)Y?	42 PROMPT	77 GTO 05	112 " X: "
08 GTO 00	43 STO 07	78 RCL 04	113 ARCL 04
09 .1	44 X<=0?	79 RCL 05	114 "t to "
10 %	45 GTO 02	80 -	115 ARCL 05
11 ST+ Y	46*LBL 03	81 X<>Y	116 "t ↓"
12 13	47 "Y MIN?"	82 /	117 PRA
13 LASTX	48 PROMPT	83*LBL 05	118 " Axis: "
14 %	49 STO 01	84 STO 06	119 ARCL 08
15 +	50 "Y MAX?"	85 RCL 00	120 PRA
16 +	51 PROMPT	86 RCL 11	121 FIX 3
17 ISG X	52 STO 02	87 +	122 " Steps of "
18 STO 00	53 RCL 01	88 RDN	123 ARCL 06
19 X<>Y	54 -	89 "ox**+:"	124 PRA
20 STO 11	55 X<=0?	90 ASTO [125 FIX 0
21 +	56 GTO 03	91*LBL 06	126 7
22 1.1	57 RCL 07	92 ASTO Y	127 CHS
23 AON	58 /	93 ASHF	128 STO 12
24 FIX 0	59 STO 03	94 ASTO X	129 RCL 03
25 CF 29	60 "AXIS"	95 ACK	130 161
26*LBL 01	61 PROMPT	96 " is Plot of: "	131 /
27 CF 23	62 STO 08	97 ARCL IND T	132 X<> 0?
28 "NAME F"	63*LBL 04	98 ACA	133*LBL 15
29 ARCL X	64 "X MIN?"	99 PRBUF	134 " SPLIT "
30 "t ?"	65 PROMPT	100 CLA	135 ARCL X
31 PROMPT	66 STO 04	101 X<>Y	136 AVIEW
32 CF? 23	67 "X MAX?"	102 X<> [137 SF 13
33 GTO 01	68 PROMPT	103 ISG T	138 XEQ 14
34 ASTO IND Y	69 STO 05	104 GTO 06	139 RCL 08
35 ISG X	70 RCL 04	105 FIX 2	140 RCL 01

141 -	191 +	241 GTO 10	290 RCL 06
142 RCL 03	192 X<0?	242 RDN	291 ST+ 10
143 X<Y	193 RCL 12	243 CLX	292 RCL 05
144 XYY?	194 STO IND \	244 RCL IND T	293 RCL 10
145 CHS	195 E2	245 INT	294 X<=Y?
146 RCL 07	196 ST/ [246 RCL Z	295 GTO 16
147 /	197 ISG \	247 -	296 XEQ 14
148 RND	198 GTO 07	248 4	297 RCL 03
149 .1	199 RCL 00	249 X<=Y?	298 ST+ 01
150 +	200 STO [250 GTO 10	299 RCL 02
151 X<0?	201 DSE [251*LBL 12	300 RCL 01
152 RCL 12	202*LBL 08	252 RCL [301 -
153 STO 09	203 ENTER†	253 DSE X	302 RCL 03
154 RCL 04	204 ABS	254 BEEP	303 /
155 STO 10	205 DSE X	255 RCL IND X	304 RND
156 CLA	206 BEEP	256 STO \	305 CF 13
157 SF 03	207 RCL IND X	257 STO IND [306 X>0?
158 RCL d	208*LBL 09	258 GTO 11	307 GTO 15
159 STO [209 RCL IND L	259*LBL 10	308 BEEP
160 ASTO [210 X<Y?	260 X<) T	309 "End of "
161 "f "	211 X<Y	261 ENTER†	310 "Multiplotting"
162 RCL [212 STO IND L	262 X<) \	311 PRA
163 STO]	213 RDN	263 -	312 ADV
164*LBL 16	214 ISG L	264 4	313 ADV
165 RCL 1	215 GTO 09	265 -	314 ADV
166 X<) d	216 STO IND Y	266 X<0?	315 ADV
167 *+000+	217 X<) Z	267 GTO 11	316 RTH
168 STO]	218 ISG X	268 SKPCOL	317*LBL 14
169 RCL 00	219 GTO 08	269 119	318 3
170 STO \	220 RCL 1	270 ACCOL	319 SKPCOL
171 RCL 09	221 X<) d	271 3	320 127
172 STO 13	222 STO]	272 ST- \	321 ACCOL
173*LBL 07	223 RCL 12	273 GTO 11	322 "-----"
174 RCL \	224 STO \	274*LBL 05	323 RCL [
175 RCL 11	225*LBL 10	275 X<) Z	324 STO \
176 +	226 RCL IND [276 ENTER†	325 STO]
177 RCL IND X	227 X<0?	277 X<) \	326 "f--"
178 RCL 16	228 GTO 11	278 -	327 ACA
179 XEQ IND Y	229 ENTER†	279 RCL 12	328 X<Y
180 RCL 01	230 FRC	280 +	329 ACCOL
181 -	231 ST- Y	281 X<0?	330 PRBUF
182 RCL 03	232 E2	282 GTO 12	331 .END.
183 X<Y	233 *	283 SKPCOL	
184 XYY?	234 E1	284 X<Y	ZEILE 09
185 CHS	235 X*Y?	285 ACCHR	F6:0A:6F:78:2A:2B:01
186 RCL 07	236 GTO 05	286*LBL 11	
187 /	237 RCL [287 ISG [
188 RND	238 ISG X	288 GTO 10	
189 RCL [239 CHS	289 PRBUF	ZEILE 167
190 FRC	240 X>0?		F7:07:98:84:24:30:10:07

Autor: Robert Klauc

Derby (Pferderennen)

LBL DERBY

SIZE 030
1109 Bytes

Peripherie:
Drucker (Man)

Dieses Programm simuliert ein Pferderennen über 1 Meile. Der Spieler kann bis zu zehn Wetten auf die 8 Pferde setzen. Es gibt 3 verschiedene Wettarten (Positionen):

1. Sieg auf (1)
2. auf ersten oder zweiten Platz (2)
3. auf ersten, zweiten oder dritten Platz (3)

Gestartet wird das Programm mit XEQ DERBY, worauf der Rechner nach einer Anfangszahl für den Zufallszahlengenerator fragt. Danach wird die Pferdenummer, der Pferdename sowie das Wettverhältnis ausgedruckt. Jetzt kann gewettet werden.

Zuerst wird die Pferdenummer (1..8), dann der Einsatz (max. 800) und schließlich die Position (1..3) eingegeben. Sind alle Wetten eingegeben, wird auf die Frage "PFERD NR. ?" einfach R/S gedrückt. Während des Rennens werden die Positionen der Pferde an 5 verschiedenen Stationen ausgegeben. Hinter jedem Pferdenamen steht die jeweilige Entfernung zum führenden Pferd in Pferdelängen.

Nach dem Zieleinlauf werden die Wetten ausbezahlt und der gesamte Gewinn bzw. Verlust berechnet.

01*LBL "DERBY"	57 RCL 09	113 ACA	169 GTO 19
02 CLRG	58 ACX	114 X<>Y	170 12,018
03 FIX 0	59 XEQ IND 09	115 E4	171 STO 19
04 CF 29	60 " "	116 /	172*LBL 26
05 SF 12	61 ARCL IND 09	117 ST+ IND 09	173 RCL 19
06 "0<X<1"	62 "t:1"	118 41	174 STO 10
07 PROMPT	63 ACA	119 ACCHR	175 E
08 STO 00	64 PRBUF	120 PRBUF	176 -
09 ADV	65 ISG 09	121 ISG 09	177 RCL IND X
10 "DERBY"	66 GTO 16	122 GTO 18	178*LBL 14
11 PRA	67 ADV	123*LBL 17	179 RCL IND 10
12 ,02	68 "POSITION: "	124 ADV	180 X<Y?
13*LBL 10	69 "t1,2 ODER 3"	125 SF 00	181 X<>Y
14 CF IND X	70 PRA	126 "START IN 1 "	182 STO IND 10
15 ISG X	71 "EINSATZ: "	127 "t-MINUTE"	183 RDN
16 GTO 10	72 "tBIS 800\$"	128 PRA	184 ISG 10
17 1,008	73 PRA	129 1,008	185 GTO 14
18 STO 09	74 20,029	130 STO 09	186 STO IND Y
19*LBL 09	75 STO 09	131*LBL 00	187 ISG 19
20 16	76*LBL 18	132 E1	188 GTO 26
21 XEQ 50	77 ,	133 RCL 09	189 "DAS GENNEN IST"
22 INT	78 "PFERD NR. ?"	134 +	190 "t- GESTARTET"
23 E	79 PROMPT	135 LASTX	191 FS?C 00
24 X<>Y	80 X=0?	136 INT	192 PRA
25 X<=Y?	81 GTO 17	137 E6	193 11,018
26 2	82 STO IND 09	138 /	194 STO 10
27 FS? IND X	83 "EINSATZ ?"	139 STO IND Y	195 RCL 11
28 GTO 09	84 PROMPT	140 ISG 09	196 RND
29 SF IND X	85 RCL 09	141 GTO 00	197*LBL 11
30 STO IND 09	86 19	142 21,025	198 ST- IND 10
31 ISG 09	87 -	143 STO 09	199 ISG 10
32 GTO 09	88 "WETTE "	144*LBL 27	200 GTO 11
33 2,008	89 ARCL X	145 11,018	201 ADV
34 STO 09	90 "t: "	146 STO 10	202 XEQ IND 09
35*LBL 12	91 PRA	147 FIX 1	203 "POS. PFERD "
36 RCL 09	92 RDN	148*LBL 19	204 "t- DAHINTER"
37 STO 10	93 CLA	149 4	205 PRA
38 E	94 ARCL X	150 XEQ 50	206 1,008
39 -	95 "t\$ AUF"	151 4	207 STO 10
40 RCL IND X	96 ACA	152 XEQ 50	208 FIX 0
41*LBL 13	97 RCL IND 09	153 +	209*LBL 15
42 RCL IND 10	98 XEQ IND X	154 2	210 RCL 10
43 X<Y?	99 X<>Y	155 /	211 ACX
44 X<>Y	100 E3	156 RCL IND 10	212 E1
45 STO IND 10	101 /	157 E1	213 +
46 RDN	102 FRC	158 *	214 RCL IND X
47 ISG 10	103 ST+ IND 09	159 FRC	215 ENTER↑
48 GTO 13	104 40	160 E5	216 ENTER↑
49 STO IND Y	105 ACCHR	161 *	217 E1
50 ISG 09	106 "POSITION ?"	162 X<>Y	218 *
51 GTO 12	107 PROMPT	163 RCL IND Y	219 FRC
52 1,008	108 3	164 LOG	220 E5
53 STO 09	109 X<Y?	165 +	221 *
54 CF 12	110 3	166 RND	222 XEQ IND X
55 CF 13	111 CLA	167 ST+ IND 10	223 FIX 1
56*LBL 16	112 ARCL Y	168 ISG 10	224 RDN

225 RND	281 E3	337 RTN
226 " "	282 *	338*LBL 02
227 X=0?	283 FRC	339 " MAN O WAR "
228 ARCL X	284 E1	340 ACA
229 ACA	285 *	341 RTN
230 PRBUF	286 INT	342*LBL 01
231 FIX 0	287 STO 11	343 " OROFINO "
232 ISG 10	288 RCL IND 10	344 ACA
233 GTO 15	289 FRC	345 RTN
234 ISG 09	290 E1	346*LBL 05
235 GTO 27	291 *	347 " APRIL RUN "
236 ADV	292 X<=Y?	348 ACA
237 11,018	293 GTO 31	349 RTN
238 STO 09	294 "ZERREISSE DEN "	350*LBL 08
239*LBL 29	295 "WETTSCHEIN"	351 " DERBYSTAR "
240 RCL IND 09	296 PRA	352 ACA
241 E1	297*LBL 32	353 RTN
242 *	298 ISG 09	354*LBL 07
243 FRC	299 GTO 28	355 " PICABILLY "
244 ES	300 FIX 0	356 ACA
245 *	301 ADV	357 RTN
246 RCL 09	302 "DU "	358*LBL 03
247 E1	303 RCL 12	359 " GALLANT FOX "
248 -	304 X>0?	360 ACA
249 INT	305 "FGEWANNST"	361 RTN
250 E1	306 X<=0?	362*LBL 06
251 /	307 "VERLORST"	363 " LANDSGIRL "
252 ST+ IND Y	308 "F INSG. "	364 ACA
253 ISG 09	309 ABS	365 RTN
254 GTO 29	310 ARCL X	366*LBL 04
255 ,	311 "F-\$"	367 " GOLD RIVER "
256 STO 12	312 PRA	368 ACA
257 20,029	313 STOP	369 RTN
258 STO 09	314*LBL 31	370*LBL 21
259*LBL 28	315 RCL IND 09	371 "KURZ NACH "
260 FIX 0	316 RCL 19	372 "F-DEN START"
261 RCL 09	317 RCL IND Y	373 PRA
262 19	318 INT	374 RTH
263 -	319 *	375*LBL 22
264 "WETTE "	320 RCL 11	376 "NACH EINER "
265 ARCL X	321 /	377 "F-1/4 MEILE"
266 "F-:"	322 ST+ 12	378 PRA
267 RCL IND 09	323 "DU KASSIERST "	379 RTN
268 X=0?	324 ARCL X	380*LBL 23
269 GTO 32	325 "F-\$"	381 "NACH DER "
270 PRA	326 PRA	382 "HALBEN DISTANZ"
271 INT	327 GTO 32	383 PRA
272 STO 10	328*LBL 50	384 RTN
273 RCL IND 09	329 RCL 00	385*LBL 24
274 FRC	330 9821	386 "VOR DER "
275 E3	331 *	387 "ZIELGERADE"
276 *	332 ,211327	388 PRA
277 INT	333 +	389 RTN
278 STO 19	334 FRC	390*LBL 25
279 ST- 12	335 STO 00	391 "ZIELEINLAUF"
280 RCL IND 09	336 *	392 PRA
		393 END

ELEKTROTECHNIKAufgenommene Programme

NAME	LABEL	QUELLE	AUTOR
Stern-Dreieck- Umwandlung	ST DR	Werner Stroinski	Werner Stroinski
Vierpole	BB ST KO	Prisma 1980 Weihnachtshefte	H.G. Lutke Uphues
Impedanz-Anpassung	IM	Heinrich Henze	Heinrich Henze
Hochpass-Tiefpass	HP TP	Prisma Weihnachtshefte	Heinrich Henze
Widerstandsthermo- meter	NI PT	Prisma 1981 S 366-S 369	Werner Stroinski
Addition zweier Sinusschwingungen	AD	Prisma 1981 S 92	Jörg Meyer
Rauschoptimierung	RF	nicht angegeben	nicht angegeben

Nicht aufgenommene Programme

NAME	LABEL	QUELLE	AUTOR
Ständerstrom	ASM	Prisma 1980 Weihnachtshefte	Klaus Schmitt
HF-Tapete	HF	Prisma 1980 Weihnachtshefte	Heinrich Henze
Abschwächer	ABS	Prisma 1980 Weihnachtshefte	Heinrich Henze
Widerstandsbest. aus Farbcode	OHM	Prisma 1980 Weihnachtshefte	Raimund Berg
Dreieck-Stern- Umwandlung	DS SD	Prisma 1981 S 91	Jörg Meyer
Ohmsche Dämpf.gl.	RD	Prisma 1981 S 141-S 142	Werner Stroinski
Trafo- und Netzteil- konzeption	GL TRF	Prisma 1981 S 304-S 310	Heinrich Henze
Thermodynamik	BIL	Prisma 1981 S 322-S 325	E. Barchewitz

01*LBL "RF"	58 ARCL X	115 STO 16	172 ARCL 11
02 ADV	59 "F MV ?"	116 ST+ 00	173 "F MV"
03 XEQ 11	60 PROMPT	117 XEQ 00	174 PRA
04 SF 12	61 STO 11	118 ENTER†	175*LBL 05
05 "RAUSCHFAKTOR"	62 25,9	119 X<> 15	176 RCL 12
06 AVIEW	63 RCL 01	120 X<>Y	177 "F: "
07 CF 12	64 *	121 RTN	178 ARCL X
08 XEQ 11	65 STO 05	122*LBL 01	179 "F db"
09*LBL B	66 RCL 02	123 RCL 14	180 AVIEW
10 CF 00	67 RCL 04	124 XEQ 03	181 RCL 00
11 "Ic ?"	68 +	125 X<Y?	182 RCL 16
12 PROMPT	69 STO 08	126 GTO 01	183 -
13 STO 00	70 CLX	127 E1	184 "Ic: "
14 SF 00	71 FC? 00	128 CHS	185 ARCL X
15*LBL A	72 STO 00	129 ST/ 14	186 "F MA"
16 FIX 0	73 STO 16	130*LBL 02	187 AVIEW
17 " ?"	74 E3	131 RCL 14	188 1.606 E-11
18 ASTO 13	75 STO 15	132 XEQ 03	189 RCL 04
19 RCL 01	76 ,1	133 X<Y?	190 *
20 "B: "	77 STO 14	134 GTO 02	191 RCL 10
21 ASTO 17	78 FIX 2	135*LBL 04	192 *
22 ARCL X	79 FC? 00	136 X<>Y	193 SORT
23 ARCL 13	80 GTO 01	137 LOG	194 XEQ 06
24 PROMPT	81*LBL 00	138 E1	195 RCL 12
25 STO 01	82 RCL 05	139 *	196 +
26 RCL 02	83 RCL 00	140 STO 12	197 "RP: "
27 "Rbb: "	84 CF 21	141 CF 00	198 ARCL X
28 ASTO 18	85 VIEW X	142 SF 21	199 "F db"
29 ARCL X	86 /	143 TONE 9	200 AVIEW
30 ARCL 13	87 STO 06	144 CLA	201 CHS
31 PROMPT	88 RCL 01	145 FC? 55	202 RCL 11
32 STO 02	89 /	146 GTO 05	203 XEQ 06
33 RCL 03	90 RCL 03	147 FIX 0	204 +
34 "Re: "	91 +	148 ADV	205 "RA: "
35 ASTO 19	92 STO 07	149 ARCL 17	206 ARCL X
36 ARCL X	93 RCL 08	150 ARCL 01	207 "F db"
37 ARCL 13	94 ENTER†	151 PRA	208 AVIEW
38 PROMPT	95 ENTER†	152 CLA	209 XEQ 11
39 STO 03	96 RCL 06	153 ARCL 18	210 GTO A
40 RCL 04	97 +	154 ARCL 02	211*LBL 06
41 "RG: "	98 LASTX	155 PRA	212 775
42 ASTO 20	99 /	156 CLA	213 /
43 ARCL X	100 STO 09	157 ARCL 19	214 LOG
44 ARCL 13	101 /	158 ARCL 03	215 20
45 PROMPT	102 RCL 07	159 PRA	216 *
46 STO 04	103 +	160 CLA	217 RTN
47 RCL 10	104 RCL 09	161 ARCL 20	218*LBL 11
48 "b: "	105 X+2	162 ARCL 04	219 FC? 55
49 ASTO 21	106 RCL 04	163 PRA	220 RTN
50 ARCL X	107 /	164 FIX 2	221 "-----"
51 "F KHZ? "	108 *	165 CLA	222 ACA
52 PROMPT	109 FS? 00	166 ARCL 21	223 ACA
53 STO 10	110 X<>Y	167 ARCL 10	224 ACA
54 RCL 11	111 FC?C 00	168 "F KHZ"	225 ACA
55 FIX 1	112 RTN	169 PRA	226 ADV
56 "Ue: "	113 GTO 04	170 CLA	227 END
57 ASTO 22	114*LBL 03	171 ARCL 22	

Stern-Dreieck-Umwandlung

DR SIZE 000
ST 190 Bytes

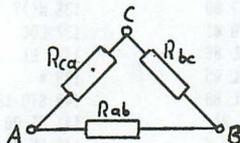
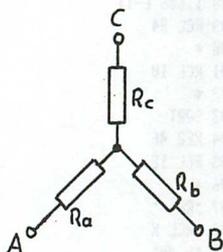
Dieses Programm wandelt die Elemente einer Sternschaltung in die Werte einer gleichwertigen Dreieckschaltung um.

Sternschaltung

Ra
ENTER
Rb
ENTER
Rc
XEQ "DR"

Dreieckschaltung

Rbc
ENTER
Rca
ENTER
Rab
XEQ "ST"



XEQ DR

Ra=100.00E0
Rb=200.00E0
Rc=300.00E0

Rbc=1.1000E3
Rca=550.00E0
Rab=366.67E0

XEQ ST

Rbc=1.1000E3
Rca=550.00E0
Rab=366.67E0

Ra=100.00E0
Rb=200.00E0
Rc=300.00E0

01*LBL "DR"	51 FS? 00	101 /
02 CF 01	52 RTN	102 FS? 00
03 FS? 00	53*LBL 02	103 RTN
04 GTO 03	54 ADV	104 GTO 03
05 XEQ 00	55 "Rbc="	105*LBL 03
06 FC? 55	56 ARCL Z	106 CF 10
07 GTO 03	57 AVIEW	107 CF 10
08*LBL 01	58 "Rca="	108 ENG 4
09 ADV	59 ARCL Y	109 END
10 "Ra="	60 AVIEW	
11 ARCL Z	61 "Rab="	
12 AVIEW	62 ARCL X	
13 "Rb="	63 AVIEW	
14 ARCL Y	64 RTN	
15 AVIEW	65*LBL "ST"	
16 "Rc="	66 SF 01	
17 ARCL X	67 FS? 00	
18 AVIEW	68 GTO 04	
19 FS? 01	69 XEQ 00	
20 RTN	70 FS? 55	
21*LBL 03	71 XEQ 02	
22 STO L	72*LBL 04	
23 RDN	73 STO L	
24 ST+ L	74 RDN	
25 RDN	75 ST+ L	
26 ST* L	76 RDN	
27 RDN	77 ST+ L	
28 X< L	78 RDN	
29 RDN	79 X< L	
30 STO L	80 RDN	
31 RDN	81 STO L	
32 ST* L	82 RDN	
33 RDN	83 ST* L	
34 RDN	84 X< L	
35 ST+ L	85 RDN	
36 X< L	86 ST* L	
37 RDN	87 X< L	
38 1/X	88 RDN	
39 RDN	89 RDN	
40 1/X	90 ST* L	
41 RDN	91 X< L	
42 1/X	92 RT	
43 X<Y	93 X<Y	
44 *	94 RDN	
45 RT	95 /	
46 LASTX	96 X< Z	
47 *	97 LASTX	
48 RT	98 /	
49 LASTX	99 X<Y	
50 *	100 LASTX	

Betriebsdämpfungsmaß und Betriebsdämpfungswinkel von Vierpolen

BB
ST
KO

SIZE O34
1002 Bytes

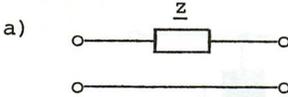
Drucker MAN
zwei Memorys

Synthetik
User

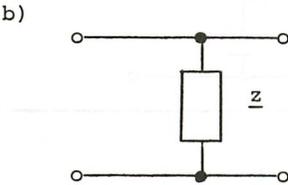
Dieses, für den HP-41 geschriebene Programm, berechnet das Betriebsdämpfungsmaß und den Betriebsdämpfungswinkel einer Schaltung, die am Eingang mit einem Generator (Innenwiderstand $\underline{z}_i = R_i + jx_i$) und am Ausgang mit einem Lastwiderstand ($\underline{z}_L = R_L + jx_L$) abgeschlossen ist.

SCHALTUNG

KETTENMATRIX



$$A = \begin{matrix} \underline{A}_{11} = 1 & \underline{A}_{12} = \underline{z} \\ \underline{A}_{21} = 0 & \underline{A}_{22} = 1 \end{matrix}$$



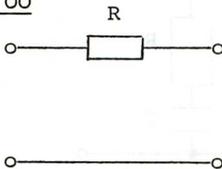
$$A = \begin{matrix} \underline{A}_{11} = 1 & \underline{A}_{12} = 0 \\ \underline{A}_{21} = 1/\underline{z} & \underline{A}_{22} = 1 \end{matrix}$$

Werden mehrere Vierpole in Reihe geschaltet, so müssen die einzelnen Kettenmatrizen multipliziert werden, was nach den Regeln der Matrixrechnung geschieht. Da es sich hier um komplexe Kettenparameter handelt, nehmen diese Berechnungen (Addition, Multiplikation, Division, Wurzelziehen) einen nicht unerheblichen Teil des Programms und der Rechenzeit ein.

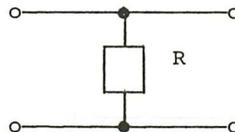
Mit dem hier vorgestellten Programm lassen sich folgende unvollkommene, aber natürlich auch vollkommene, Vierpole verarbeiten. Es können aber auch einige hinzugefügt oder ersetzt werden, dazu dienen Label 23, 24, ...

Die Zahlenangaben der folgenden Vierpole geben die Nummern des Labels an, unter denen der entsprechende Vierpol gefunden wird, um dessen Kettenmatrix aufzustellen.

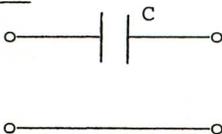
LBL 00



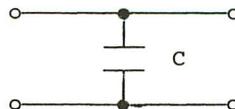
LBL 01



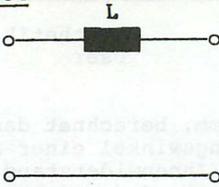
LBL 02



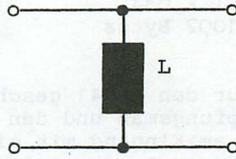
LBL 03



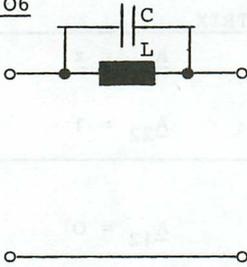
LBL 04



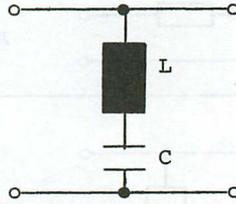
LBL 05



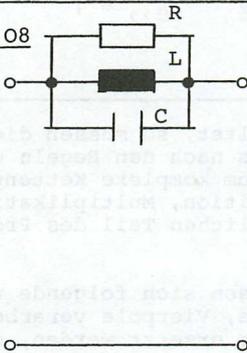
LBL 06



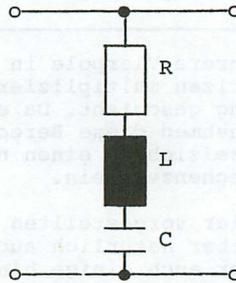
LBL 07



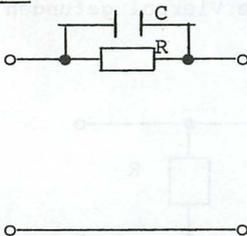
LBL 08



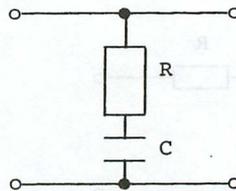
LBL 09

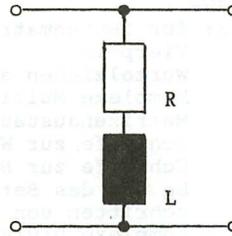
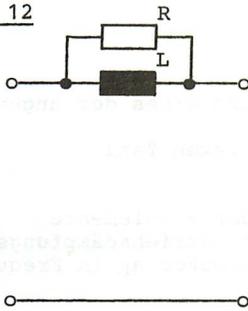


LBL 10



LBL 11





Zur Programmbedienung

Die Reihenfolge der Vierpole muß im Anschluß an LBL A zusätzlich zu dem bestehenden Programm programmiert werden, und zwar vom Eingang zum Ausgang der Vierpolkette.

Bei der Werteingabe für die Schaltelemente ist die Reihenfolge R,L,C für alle Vierpole gleich. Besteht ein Vierpol aus nur einem Schaltelement, muß natürlich nur ein Schaltelement dem Rechner eingetippt werden. Zur Eingabe dieser Werte, sowie der Werte des Generatorinnenwiderstandes und des Lastwiderstandes wird das Alpha-Label BB verwendet und für den Berechnungsstart ST. Nach dessen Aufruf müssen noch Anfangsfrequenz f , Frequenzschrittweite Δf und die Endfrequenz f_{\max} eingegeben werden. Danach startet das Programm mit der Berechnung und endet mit der Anzeige des Wertes $f=f_{\max}+\Delta f$. Es wurde diese Eingaberoutine gewählt, um Berechnungen mit anderen Frequenzen an der selben Schaltung leicht vornehmen zu können, wofür einfach nur wieder das Label ST aufgerufen werden muß.

Da bei Schaltungen mit vielen Bauelementen eine Fehleingabe vorkommen kann, ist auch eine Korrekturroutine in das Programm aufgenommen worden. Wenn z.B. nach Eingabe des 10. Schaltelements festgestellt wird, daß das 2. Bauelement falsch eingegeben wurde, kann man hier die Eingaberoutine unterbrechen, die Nummer (hier 2) in die Anzeige tippen und mit XEQ"KO" die Korrekturroutine aufrufen, wonach im Display wieder R,L,C?n (hier n=2) erscheint. Jetzt gibt man den richtigen Wert ein, drückt R/S, und man kann nach Anzeige von R,L,C?n an der Stelle der Eingaberoutine fortfahren, wo man sie unterbrochen hatte.

Registerbelegung

R00 - R19: für komplexe Matrizen
 R20: Frequenz f
 R21: Frequenzschrittweite Δf
 R22: $2 \times \text{PI} \times f = \omega$
 R23: Verschiedenes
 R24: Verschiedenes
 R25: Endfrequenz f_{\max}
 R26: Zähler
 R27: reeller Innenwiderstand des Generators R_i
 R28: imaginärer Innenwiderstand des Generators jx_i
 R29: reeller Lastwiderstand R_L
 R30: imaginärer Lastwiderstand jx_L
 R31: Inneninduktivität L_i bzw. Innenkapazität $-C_i$ des Generators
 R32: Lastinduktivität L_L bzw. Lastkapazität $-C_L$
 R33:....: Werte der Schaltelemente, beginnend mit n=1

Labelbelegung

LBL 00 - 13:	für Kettenmatrizenaufstellung eines der angegebenen Vierpole
LBL 14:	Wurzelziehen aus einer komplexen Zahl
LBL 15:	komplexe Multiplikation
LBL 16:	Matrizenaustausch
LBL 17:	Schleife zur Wert-Eingabe der Bauelemente
LBL 18/21:	Schleife zur Berechnung des Betriebsdämpfungswinkels β_B und des Betriebsdämpfungsmaßes a_B in Frequenzschritten von Δf
LBL 19:	komplexe Division
LBL 20:	Sprungadresse für komplexe Division in das Unterprogramm der komplexen Multiplikation
LBL 22:	komplexe Matrizenmultiplikation
"BB":	Eingabe des Generatorinnenwiderstandes z_L , des Lastwiderstandes Z_L , sowie der Werte der Schaltelemente
"ST":	Eingabe von f , Δf , f_{max} und Programmstart
"KO":	Korrektur einer falschen Eingabe eines Schaltelementes

Schlußbemerkung

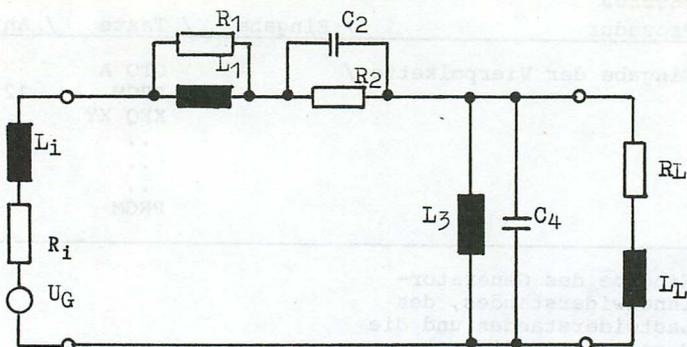
Soll das Programm in dieser Form verwendet werden, ist ein Drucker notwendig. Steht dieser aber nicht zur Verfügung, müssen die Druckbefehle gelöscht werden (gemeint ist: PRA). Programmzeile 102 kann durch einen AVIEW-Befehl, und Zeilen 193 und 197 durch je einen PROMPT-Befehl ersetzt werden. Sind diese Änderungen vorgenommen worden, muß nach jedem Ergebnis das Programm durch R/S wieder gestartet werden.

Literaturhinweis

- E. Philipnow: Grundlagen der Elektrotechnik
Brühl/Jansen/Vogt: Nachrichtenübertragungstechnik I
Stefan Hamerlie: Wobbelkurven ohne Meßgerät ermittelt
Elektronik 5/1978

Programmbedienung

Schritt	Prozedur	Eingabe	Taste	Anzeige
1	/ Eingabe der Vierpolkette /		GTO A PRGM XEQ XY PRGM	123 LBL ₁ A
2	/ Eingabe des Generator- innenwiderstandes, des Lastwiderstandes und die Werte der Bauelemente Vorbereitung		XEQ"BB"	RI?
	<u>Achtung</u>	R_i	R/S	LI, -CI?
	Falls Innenkapazität oder	L_i bzw. $-C_i$	R/S	RL?
	Lastkapazität vorhanden,	R_L	R/S	LL, -CL?
	müssen diese mit neg.	L_L bzw. $-C_L$	R/S	R, L, C?1
	Vorzeichen eingegeben	r, l oder C_1	R/S	usw.
	werden.	usw.	usw.	
	Sind alle Schaltelemente eingegeben, weiter mit 4			
3	/ Korrektur einer R, L, C- Eingabe (n-tes Bauelement)	n R, L oder C_n	XEQ"KO" R/S	R, L, C?n
4	/ <u>Frequenzeingaben</u>		XEQ"ST"	F?
	Startfrequenz	f	R/S	dF?
	Frequenzschrittweite	f	R/S	F _{max} ?
	Endfrequenz	f_{max}	R/S	
	Berechnungsstart			
	Rechnung endet mit			$f_{max} + f$



$R_i = 600 \text{ OHM}$
 $L_i = 1 \text{ E-6 H}$
 $R_L = 600 \text{ OHM}$
 $L_L = 0,5 \text{ E-3 H}$

$R_1 = 10000 \text{ OHM}$
 $L_1 = 2,387 \text{ H}$
 $R_2 = 10000 \text{ OHM}$
 $C_2 = 15,79 \text{ E-9 F}$
 $L_3 = 5,68 \text{ E-3 H}$
 $C_4 = 6,63 \text{ E-6 F}$

$R1? = 600,00000000 \text{ E0}$
 $L1, -C1? = 1,000000000 \text{ E-6}$
 $R_L? = 600,00000000 \text{ E0}$
 $L_L, -C_L? = 500,00000000 \text{ E-6}$

$F = 860,00000000 \text{ HZ}$
 $aB = 27,28356704 \text{ E0 dB}$
 $bB = 61,31158556 \text{ E0 GRAD}$

$F = 880,00000000 \text{ HZ}$
 $aB = 30,14967348 \text{ E0 dB}$
 $bB = 69,50731768 \text{ E0 GRAD}$

$R, L, C?1 = 10,00000000 \text{ E3}$
 $R, L, C?2 = 2,3870000000 \text{ E3}$
 $R, L, C?3 = 10,00000000 \text{ E3}$
 $R, L, C?4 = 15,97000000 \text{ E-9}$
 $R, L, C?5 = 5,6800000000 \text{ E-3}$
 $R, L, C?6 = 6,6300000000 \text{ E-6}$

$F = 900,00000000 \text{ HZ}$
 $aB = 32,33727274 \text{ E0 dB}$
 $bB = 73,89173730 \text{ E0 GRAD}$

BANDPASS

$F? = 760,00000000 \text{ HZ}$
 $dF? = 20,00000000 \text{ E0 HZ}$
 $F \text{ MAX?} = 900,00000000 \text{ HZ}$

123*LBL A
 124 XEQ 12
 125 XEQ 10
 126 XEQ 05
 127 XEQ 03
 128 RCL 30

$F = 760,00000000 \text{ HZ}$
 $aB = 30,77797780 \text{ E0 dB}$
 $bB = -70,76079140 \text{ E0 GRAD}$

$F = 780,00000000 \text{ HZ}$
 $aB = 27,69521014 \text{ E0 dB}$
 $bB = -62,56698329 \text{ E0 GRAD}$

$F = 800,00000000 \text{ HZ}$
 $aB = 23,72786074 \text{ E0 dB}$
 $bB = -44,05666164 \text{ E0 GRAD}$

$F = 820,00000000 \text{ HZ}$
 $aB = 20,81697244 \text{ E0 dB}$
 $bB = -551,0328094 \text{ E-3 GRAD}$

$F = 840,00000000 \text{ HZ}$
 $aB = 23,53203216 \text{ E0 dB}$
 $bB = 42,85409582 \text{ E0 GRAD}$

```

PRP "BB"

01*LBL "BB"
ENG 9 DEG "R1?"
PROMPT "I=" ARCL X
PRA STO 27 "LI,-CI?"
PROMPT "I=" ARCL X
PRA STO 31 "RL?"
PROMPT "I=" ARCL X
PRA STO 29 "LL,-CL?"
PROMPT "I=" ARCL X
PRA STO 32 ADV ADV
33 STO 26 E STO 25
CF 29

35*LBL 17
"R,L,C?" FIX 0 ARCL 25
ENG 9 PROMPT "I="
ARCL X PRA STO IND 26
E ST+ 26 ST+ 25
GTO 17

49*LBL "ST"
SF 29 ADV "F?" PROMPT
"I=" ARCL X "I HZ"
PRA "dF?" PROMPT "I="
ARCL X "I HZ" PRA
STO 21 - STO 20
"F MAX?" PROMPT "I="
ARCL X "I HZ" PRA
STO 25 ADV ADV

76*LBL 18
ADV E STO 00 STO 06
. STO 01 STO 02
STO 03 STO 04 STO 05
STO 07 RCL 21 ST+ 20
RCL 25 RCL 20 X<Y?
GTO 21 ADV ADV ADV
STOP

98*LBL 21
"F=" ARCL X "I HZ"
PRA 2 PI * * STO 22
RCL 31 X<Y * STO 28
LASTX RCL 32 * STO 30
RCL 26 INT E3 / 33
+ STO 26

123*LBL A
RCL 30 X<0? 1/X
STO 30 RCL 29 RCL 28
X<0? 1/X STO 28
RCL 27 XEQ 19 XEQ 14
STO 08 X<Y STO 09
X<Y RCL 01 RCL 00
XEQ 15 STO 00 X<Y
STO 01 RCL 28 RCL 27
RCL 30 RCL 29 XEQ 15
XEQ 14 STO 10 X<Y
STO 11 X<Y RCL 03
RCL 02 RDN RDN XEQ 19
ST+ 00 X<Y ST+ 01
RCL 05 RCL 04 RCL 11
RCL 10 XEQ 15 ST+ 00
X<Y ST+ 01 RCL 07
RCL 06 RCL 09 RCL 08
XEQ 19 ST+ 00 X<Y
ST+ 01 RCL 01 2 /
RCL 00 2 / R-P LOG
20 * "aB=" ARCL X
"I dB" PRA "bB="
ARCL Y "I GRAD" PRA
GTO 18

199*LBL 00
RCL IND 26 STO 16 .
STO 17 STO 18 STO 19
GTO 22

207*LBL 01
RCL IND 26 1/X STO 18
. STO 19 STO 16
STO 17 GTO 22

216*LBL 02
RCL IND 26 RCL 22 *
1/X CHS STO 17 .
STO 16 STO 18 STO 19
GTO 22

228*LBL 03
RCL IND 26 RCL 22 *
STO 19 . STO 18
STO 16 STO 17 GTO 22

238*LBL 04
RCL IND 26 RCL 22 *
STO 17 . STO 16
STO 18 STO 19 GTO 22

248*LBL 05
RCL IND 26 RCL 22 *
1/X CHS STO 19 .
STO 18 STO 17 STO 16
GTO 22

260*LBL 06
RCL IND 26 RCL 22 *
1/X ISG 26 RCL IND 26
RCL 22 * - 1/X
STO 17 . STO 16
STO 18 STO 19 GTO 22

277*LBL 07
RCL IND 26 ISG 26
RCL IND 26 RCL 22 *
1/X X<Y RCL 22 * -
1/X STO 19 . STO 18
STO 17 STO 16 GTO 22

295*LBL 08
RCL IND 26 1/X STO 23
X+2 ISG 26 RCL IND 26
RCL 22 * 1/X CHS
ISG 26 RCL IND 26
RCL 22 * + STO 24
X+2 + 1/X ENTER↑
ENTER↑ RCL 23 *
STO 16 X<Y RCL 24 *
CHS STO 17 . STO 18
STO 19 GTO 22

329*LBL 09
RCL IND 26 STO 23 X+2
ISG 26 RCL IND 26
RCL 22 * ISG 26
RCL IND 26 RCL 22 *
1/X - STO 24 X+2 +
1/X ENTER↑ ENTER↑
RCL 23 * STO 18 X<Y
RCL 24 * CHS STO 19
. STO 16 STO 17
GTO 22

361*LBL 10
RCL IND 26 STO 23
ISG 26 RCL IND 26
RCL 22 * * STO 24
X+2 E + 1/X ENTER↑
ENTER↑ RCL 23 *
STO 16 X<Y RCL 24 *
CHS RCL 23 * STO 17
. STO 18 STO 19
GTO 22

390*LBL 11
RCL IND 26 STO 23 X+2
ISG 26 RCL IND 26
RCL 22 * 1/X STO 24
X+2 + 1/X ENTER↑
ENTER↑ RCL 23 *
STO 18 X<Y RCL 24 *
STO 19 . STO 16
STO 17 GTO 22

```

416*LBL 12
RCL IND 26 STO 23
ISG 26 RCL IND 26
RCL 22 * STO 24 /
ENTER↑ 1/X + 1/X
ENTER↑ ENTER↑ RCL 24
* STO 16 X<Y RCL 23
* STO 17 . STO 18
STO 19 GTO 22

442*LBL 13
RCL IND 26 STO 23 X↑2
ISG 26 RCL IND 26
RCL 22 * STO 24 X↑2
+ 1/X ENTER↑ ENTER↑
RCL 23 * STO 18 X<Y
RCL 24 * CHS STO 19
. STO 16 STO 17

467*LBL 22
RCL 01 STO 09 RCL 00
STO 08 RCL 17 RCL 16
XEQ 15 STO 10 X<Y
STO 11 RCL 03 ST+ 11
RCL 02 ST+ 10 RCL 19
RCL 18 XEQ 15 ST+ 08
X<Y ST+ 09 RCL 05
STO 13 RCL 04 STO 12
RCL 17 RCL 16 XEQ 15
STO 14 X<Y STO 15
RCL 07 ST+ 15 RCL 06
ST+ 14 RCL 19 RCL 18
XEQ 15 ST+ 12 X<Y
ST+ 13 15.007 STO 23
7 STO 24

512*LBL 16
RCL IND 23 STO IND 24
E ST- 24 DSE 23
GTO 16 ISG 26 RTN

521*LBL 19
R-P 1/X X<Y CHS
GTO 20

527*LBL 15
R-P X<Y

530*LBL 20
RDN RDN R-P R↑ *
RDN + R↑ P-R RTN

541*LBL 14
R-P SQRT X<Y 2 /
ENTER↑ SIN RCL Z *
X<Y COS RCL Z * RTN

556*LBL "K0"
FIX 0 "KOR.:" ARCL X
PRA "R,L,C?" ARCL X
32 + ENG 9 PROMPT
"t=" ARCL X PRA
STO IND Y ADV GTO 17
END

Impedanz - Anpassung

IM SIZE 006
 105 Bytes

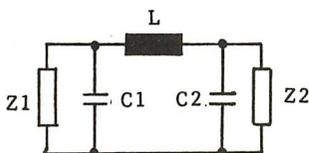
Transformierende Tiefpässe werden zur Anpassung von verschiedener Impedanzen gebraucht.

XEQ"IM"

Eingabe: Eingangsimpedanz R/S
Eingabe: Ausgangsimpedanz R/S
Eingabe: Mittenfrequenz R/S
Eingabe: Bandbreite R/S

C2, C1 und L werden nach kurzer Rechenzeit ausgegeben

Vorausbedingung: Eingangsimpedanz > Ausgangsimpedanz



Z1?500.0E0
Z2?50.00E0
F?1.000E6
<>F?100.0E3
C2=15.29E-9
C1=4.837E-9
L=6.893E-6

```
01*LBL "IM"            33 RCL 04
02 ENG 3                34 RCL 05
03 "Z1?"                35 RCL 00
04 PROMPT              36 +
05 XEQ 01               37 *
06 STO 00               38 1/X
07 "Z2?"                39 "C1="
08 PROMPT              40 XEQ 01
09 XEQ 01               41 RCL 03
10 STO 01               42 4
11 "F?"                 43 PI
12 PROMPT              44 *
13 XEQ 01               45 RCL 02
14 STO 02               46 X^2
15 "<>F?"               47 *
16 PROMPT              48 /
17 XEQ 01               49 RCL 05
18 STO 03               50 ST+ X
19 PI                    51 RCL 00
20 *                    52 +
21 STO 04               53 RCL 01
22 RCL 01               54 +
23 RCL 00               55 *
24 *                    56 "L="
25 SQRT                 57*LBL 01
26 STO 05               58 ARCL X
27 RCL 01               59 AVIEW
28 +                    60 FC? 55
29 *                    61 STOP
30 1/X                  62 END
31 "C2="
32 XEQ 01
```

Aktives Hochpaßfilter 2. Ordnung
Tiefpaßfilter mit Einfachmitkopplung

HP SIZE 007
 TP 195 Bytes

Das Programm Hochpaßfilter errechnet ein aktives Filter 2. Ordnung.
 Für Filter höherer Ordnung muß die Berechnung für jedes Teilfilter mit den entsprechenden Koeffizienten durchgeführt werden.

Das Programm Tiefpaßfilter 2. Ordnung mit Einfachmitkopplung berechnet die Widerstände R1, R2 und den Kondensator C2.

Für Hochpaßfilter
 XEQ"HP"

Für Tiefpaßfilter
 XEQ"TP"

Programm auf Abfrage bedienen

F = Grenzfrequenz Hz
 A1 = je nach Filtertyp
 B1 =
 C1 = Kondensatorwerte F
 C2 =



Ausgaben

Hochpaßfilter R1, R2
 Tiefpaßfilter R1, R2, C2

Beispiel
 XEQ"HP"

XEQ"TP"

F?1.000E3
 A1?1.362E0
 B1?618.0E-3

F?1.000E3
 A1?1.362E0
 B1?618.0E-3

C1?10.00E-9
 C2?10.00E-9

C1?10.00E-9
 C2?10.00E-9

R1=23.38E3
 R2=17.53E3

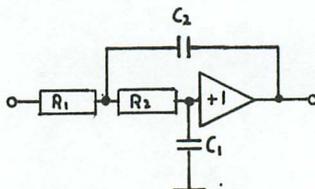
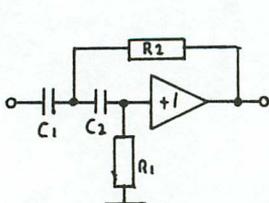
R1=7.885E3
 R2=13.79E3
 C2=14.40E-9

C1?5.700E-9
 C2?10.00E-9

C1?1.000E-9
 C2?1.500E-9

R1=32.19E3
 R2=22.34E3

R1=72.22E3
 R2=144.5E3
 C2=1.500E-9



01+LBL "HP"	61 RCL 01
02 SF 01	62 *
03 GTO 00	63 RCL 02
04+LBL "TP"	64 *
05 CF 01	65 RCL 03
06+LBL 00	66 *
07 ADV	67 STO Z
08 ENG 3	68 /
09 "F?"	69 "R1="
10 PROMPT	70 XEQ 01
11 XEQ 01	71 RCL 06
12 STO 01	72 RCL 00
13 "A1?"	73 +
14 PROMPT	74 RCL Z
15 XEQ 01	75 /
16 STO 04	76 "R2="
17 "B1?"	77 XEQ 01
18 PROMPT	78 RCL 03
19 XEQ 01	79 "C2="
20 STO 05	80 GTO 01
21+LBL A	81+LBL 02
22 ADV	82 1.2
23 "C1?"	83 ST* 03
24 PROMPT	84 GTO B
25 XEQ 01	85+LBL 03
26 STO 02	86 2
27 "C2?"	87 PI
28 PROMPT	88 *
29 XEQ 01	89 RCL 01
30 STO 03	90 *
31 ADV	91 STO 06
32+LBL B	92 RCL 04
33 FS? 01	93 *
34 GTO 03	94 1/X
35 RCL 04	95 RCL 02
36 X+2	96 1/X
37 RCL 03	97 RCL 03
38 X+2	98 1/X
39 *	99 +
40 4	100 *
41 RCL 05	101 "R1="
42 *	102 XEQ 01
43 RCL 02	103 RCL 04
44 *	104 RCL 05
45 RCL 03	105 RCL 06
46 *	106 *
47 -	107 /
48 X<0?	108 RCL 02
49 GTO 02	109 RCL 03
50 SQRT	110 +
51 STO 06	111 1/X
52 RCL 04	112 *
53 RCL 03	113 "R2="
54 *	114+LBL 01
55 STO 00	115 ARCL X
56 X<>Y	116 AVIEW
57 -	117 FC? 55
58 4	118 STOP
59 PI	119 END
60 *	

Widerstandsthermometer

PT SIZE 006
NI 443 Bytes

Dieses Programm errechnet die Temperatur nach Eingabe des spezifischen Widerstandes oder den spezifischen Widerstand nach Eingabe der Temperatur. Siehe DIN 43760

XEQ"NI"

Anzeige: T?

Eingabe der Temperatur und R/S (Anzeige kann geändert werden, durch einen Druck auf Taste R/S ohne Eingabe)

Nach kurzer Rechenzeit wird der dazugehörige Widerstand angezeigt.

Wurde nur R/S gedrückt, erscheint R? in der Anzeige (Programm fordert zur Eingabe des Widerstandes auf)

XEQ"PT"

gleiche Eingabebedingung wie bei XEQ"NI"

Das Programm überprüft, ob der eingegebene Wert innerhalb des durch DIN 43760 definierten Geltungsbereichs liegt.

NI 100
DIN 43 760 / 10.00

T=130.00 C
R=183.345 OHM

T=-60.00 C
R=69.520 OHM

T=50.26 C
R=129.265 OHM

R=70.258 OHM
T=-58.42 C

PT 100
DIN 43 760 / 10.00

T=150.00 C
R=157.315 OHM

T=850.00 C
R=390.263 OHM

T=-200.00 C
R=10.493 OHM

R=18.500 OHM
T=-199.98 C

01*LBL "PT"	53 RDN	105*LBL 00	157 FS? 03
02 XEQ 16	54 FC? 03	106 "R?"	158 GTO 14
03 ,390802	55 -200,01	107 ,	159 +
04 STO 01	56 FS? 03	108 FS? 03	160 CHS
05 580195 E-10	57 -60,01	109 STO 04	161 FC? 01
06 CHS	58 X\Y?	110 FC? 03	162 GTO 06
07 STO 02	59 GTO 01	111 STO 05	163*LBL 14
08 42735 E-12	60 RDN	112 PROMPT	164 GTO 03
09 STO 03	61 ADV	113 FC?C 22	165*LBL 04
10 E2	62 XEQ 06	114 GTO 01	166 RCL 00
11 CHS	63*LBL 03	115 FC? 03	167 -
12 /	64 FS? 03	116 390,266	168 ABS
13 STO 04	65 GTO 10	117 FS? 03	169 E-4
14 GTO 15	66 X<0?	118 223,23	170 X\Y?
15*LBL "NI"	67 SF 00	119 X<Y?	171 GTO 06
16 XEQ 16	68*LBL 10	120 GTO 00	172 LASTX
17 ,5485	69 ENTER↑	121 RDN	173 FC? 03
18 STO 01	70 ENTER↑	122 FC? 03	174 ST+ 05
19 665 E-6	71 FS? 03	123 18,491	175 FS? 03
20 STO 02	72 GTO 11	124 FS? 03	176 ST+ 04
21 2805 E-12	73 RCL 04	125 69,45	177 RCL 00
22 STO 03	74 *	126 X\Y?	178 FC? 03
23 SF 03	75 RCL 03	127 GTO 00	179 RCL 05
24*LBL 15	76 +	128 RDN	180 FS? 03
25 FC? 55	77 *	129 ADV	181 RCL 04
26 GTO 01	78 RCL 02	130 XEQ 07	182 -
27 ADV	79 +	131 STO 00	183 GTO 05
28 SF 12	80 FC?C 00	132*LBL 05	184*LBL 06
29 FC? 03	81 X<) L	133 FS? 03	185 "I="
30 " PT"	82 GTO 12	134 SF 01	186 FS?C 01
31 FS? 03	83*LBL 16	135 E2	187 X<) T
32 " NI"	84 CF 01	136 FS? 03	188 FIX 2
33 "+ 100"	85 SF 02	137 GTO 13	189 RND
34 PRA	86 CF 22	138 X\Y?	190 ARCL X
35 CF 12	87 CF 03	139 SF 01	191 "+ C"
36 " DIN 43 760 /"	88 RTN	140*LBL 13	192 ,
37 "+ 10.00"	89*LBL 11	141 -	193 GTO 08
38 PRA	90 RCL 03	142 RCL 02	194*LBL 07
39*LBL 01	91 *	143 /	195 "R="
40 "T?"	92 *	144 RCL 01	196 FIX 3
41 E	93 RCL 02	145 LASTX	197 RND
42 PROMPT	94 +	146 /	198 ARCL X
43 FC?C 22	95*LBL 12	147 2	199 "+ OHM"
44 GTO 00	96 *	148 /	200 E
45 FC? 03	97 RCL 01	149 X↑2	201*LBL 08
46 CF 00	98 +	150 LASTX	202 X<)Y
47 FC? 03	99 *	151 RDN	203 AVIEW
48 850,01	100 E2	152 +	204 FS?C 02
49 FS? 03	101 +	153 SQRT	205 RTN
50 100,01	102 FS? 01	154 R↑	206 SF 02
51 X<Y?	103 GTO 04	155 FS? 03	207 GTO IND Y
52 GTO 01	104 GTO 07	156 -	208 END

Addition von zwei Sinusschwingungen gleicher Frequenz

AD

SIZE 004
086 Bytes

User

Dieses Programm addiert zwei sinusförmige Schwingungen gleicher Frequenz zu ihrer resultierenden.

XEQ"AD"

Auf Abfrage Amplitude 1 und 2 sowie Winkel 1 und 2 eingeben. Nach jeder Eingabe R/S.

Im Display erscheint, nachdem Winkel 2 eingegeben wurde und R/S gedrückt wurde, die resultierende Amplitude, und nach einem weiteren Druck auf Taste R/S erscheint der resultierende Nullphasenwinkel.

PRP "AD"

01+LBL "AD"
"A1?" PROMPT STO 00
"A2?" PROMPT STO 01
"Z1?" PROMPT STO 02
"Z2?" PROMPT STO 03

14+LBL B
RCL 00 X↑2 RCL 01 X↑2
+ RCL 00 RCL 01 *
ST+ X RCL 02 RCL 03 -
COS * + SQRT "A="

34+LBL C
RCL 02 SIN RCL 00 *
RCL 03 SIN RCL 01 *
+ RCL 02 COS RCL 00 *
* RCL 03 COS RCL 01
* + / ATAN "Z="

ARCL X PROMPT END

	– 8. – Spiele								

Minischach

LBL MS LBL P SIZE 083 evtl. KARTENLESER NUR "5"
LBL 5 BYTES 968 DRUCKER (NORM) SYNTHETISCH

Mit diesem Programm können Sie Schach gegen Ihren HP 41C(V) spielen. Dieses Spiel wird auf 5x5 Feldern statt auf 8x8 Feldern gespielt.

Es gelten die Standardschachregeln bis auf folgende Ausnahmen:

- a) die Aufstellung der Figuren ist wie folgt:
schwarz: König, Dame, Läufer, Springer, Turm
Bauer, Bauer, Bauer, Bauer, Bauer
weiß: Bauer, Bauer, Bauer, Bauer, Bauer
König, Dame, Läufer, Springer, Turm
- b) es gibt keine Rochade.
- c) der erste Bauernzug darf nur 1 Feld betragen.
- d) es gibt kein en passant.

Das Programm, genannt MS, besteht aus 3 einzelnen Files: MS, P und 5. Das Hauptprogramm MS ist unabhängig von den beiden Unterprogrammen.

Wenn Sie keinen Drucker benutzen, brauchen Sie die beiden Programme P und 5 nicht zu laden.

Für das Hauptprogramm benötigen Sie eine Datenkarte, auf der einige wichtige Konstanten sowie die "BLDSPEC" gespeichert sind.

Anmerkungen:

SIZE muß exakt auf 083 stehen.

Sie können wählen, ob das Brett nach jedem Zug (SF 00) oder nach HP-Zug (CF 00) ausgedruckt werden soll.

Der Rechner macht nur legale Züge.

Ihre Züge werden nicht auf Legalität untersucht!

Wenn Ihrem König Schach geboten wird, können Sie dies der Anzeige entnehmen (VON xx NACH yy, SCHACH).

Es gibt aber zwei Situationen, in denen der Rechner das Schach nicht anzeigt:

1. wenn ein Bauer in eine andere Figur verwandelt wurde.
2. wenn nicht von der gezogenen Figur, sondern von einer anderen Figur die Bedrohung ausgeht (siehe Abb. 1).

Sie spielen immer die weißen Figuren, während der Rechner die schwarzen Figuren spielt.

Der HP 41 C(V) wandelt seinen Bauern immer in eine Dame um.

Wenn Sie einen Bauern auf die 1. Zeile ziehen, fragt der Rechner FIGUR?

Geben Sie bitte folgende Werte für die umgewandelte Figur ein:

Dame: 5,09
Läufer: 4,03
Springer: 3,03
Turm: 2,05

```
  1 2 3 4 5
1  ♔ ♖ ♗ ♘ ♙
2  ♚ ♜ ♝ ♞ ♟
3  ♠ ♡ ♢ ♣ ♤
4  ♥ ♦ ♧ ♨ ♩
5  ♪ ♫ ♬ ♭ ♮
```

ICH ZIEHE
VON 42 NACH 53

Abb. 1

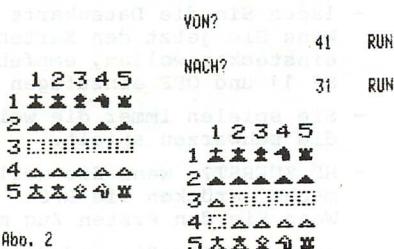
```
  1 2 3 4 5
1  ♔ ♖ ♗ ♘ ♙
2  ♚ ♜ ♝ ♞ ♟
3  ♠ ♡ ♢ ♣ ♤
4  ♥ ♦ ♧ ♨ ♩
5  ♪ ♫ ♬ ♭ ♮
```

Wenn der Rechner keinen legalen Zug findet, zeigt er SCHACHMATT
 → SIE HABEN GEWONNEN an.

Wenn Sie schachmatt sind oder eine Pattsituation entsteht, geben Sie statt Ihres Zuges -1 für schachmatt oder 0 für patt ein. Der Rechner gibt dann SCHACHMATT ICH HABE GEWONNEN bzw. PATT aus.

Ihr Zug:

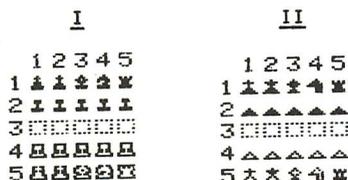
Beantworten Sie die beiden Fragen VON?, NACH? mit dem Anfangsfeld und dem Endfeld Ihres Zuges (Zeile/Spalte). Ein Beispiel sehen Sie in Abbildung 2. Schach oder schachmatt erkennt der Rechner selbständig. Die Bauernumwandlung wurde vorher schon beschrieben.



Die Datenkarte:

R18= 13,000	R28= 8,000
R19= -13,000	R29= -8,000
R20= 15,000	R30= 6,000
R21= -15,000	R31= -6,000
R22= 5,000	R32= 24,027
R23= -5,000	R33= 16,023
R24= 7,000	R34= 28,031
R25= -7,000	R35= 24,031
R26= 1,000	R36= 24,031
R27= -1,000	

Abb. 2



In R 37 bis R 49 sind die "BLDSPEC=" für die Figuren gespeichert. Für fast jede Figur sind zwei Darstellungen angegeben. Sie können natürlich auch Ihre eigenen Figuren entwerfen.

Vorschläge:

R37 = schwarzer König	= I:	0	96	122	127	122	96	0
	II:	64	98	114	127	114	98	64
R38 = schwarze Dame	= I:	0	96	114	127	114	96	0
	II:	64	98	118	127	118	98	64
R39 = schwarzer Läufer	= I:	0	100	110	123	110	100	0
	II:	0	68	110	127	110	68	0
R40 = schwarzer Springer	= I:	0	108	102	119	126	108	0
	II:	8	12	14	127	126	124	0
R41 = schwarzer Turm	= I:	0	102	124	126	124	102	0
	II:	0	102	124	126	124	102	0
R42 = schwarzer Bauer	= I:	0	96	102	126	102	96	0
	II:	64	96	112	120	112	96	64
R43 = freies Feld	=I+II:	85	0	65	0	65	0	85
R44 = weißer Bauer	= I:	96	95	89	65	89	95	96
	II:	64	96	80	72	80	96	64
R45 = weißer Turm	= I:	103	89	67	65	67	89	103
	II:	0	102	124	102	124	102	0
R46 = weißer Springer	= I:	110	83	89	72	65	83	126
	II:	8	12	10	121	66	124	0
R47 = weißer Läufer	= I:	110	91	81	68	81	91	110
	II:	0	68	106	113	106	68	0
R48 = weiße Dame	= I:	112	95	77	64	77	95	112
	II:	64	98	86	79	86	98	64
R49 = weißer König	= I:	112	95	69	64	69	95	112
	II:	64	98	82	79	82	98	64

01*LBL MS*
CLAG FIX 0 CF 04
CF 29 18.049 RDTAX 9
STO 16 ST- 17 6.5
STO 78 ST- 50 5.09
STO 79 ST- 51 4.03
STO 80 ST- 52 1.01
STO 71 STO 72 STO 73
STO 74 STO 75 ST- 57
ST- 58 ST- 59 ST- 60
ST- 61 INT - STO 81
ST- 53 2.05 STO 82
ST- 54 "A" 55.078
SIGN

41*LBL 01
RCL IND L X=0?
ASTO IND L ISG L
GTO 01 SREG 64 CLX
ASTO 69 FS? 55 XEQ "P"
CF 23 AON "HP ZUERST?"
PROMPT ROFF FC? 23
GTO 00

59*LBL 99
"VON?" PROMPT "ICH"
SF 04 X<0? GTO 04
X=0? GTO 05 XEQ 06
STO 00 "NACH?" PROMPT
XEQ 06 STO 01 CLX
X<> IND 00 STO IND 01
XEQ 07 FC? 55 GTO 00
FS? 00 XEQ "P"

82*LBL 00
"ICH ZIEHE" AVIEW
CF 17 FS? 55 SF 17
FS? 55 XEQ "S" PI
STO 09 82.049 STO 02
CHS STO 00

96*LBL 11
RCL IND 02 SIGN X=0?
GTO 00 LASTX X<0?
XEQ 12

104*LBL 00
DSE 02 GTO 11 FS? 17
XEQ "S" RCL 09 PI
X=Y? GTO 05 "SIE"
CF 04 -25 RCL 00 X<Y?
GTO 04 CLX X<> IND 12
STO IND 13 XEQ 08
"VON" RCL 12 XEQ 09
"I NACH" RCL 13
XEQ 09 RCL 00 FRC 10
* FRC X=0?
"I. SCHACH" BEEP AVIEW

FS? 55 XEQ "P" FC? 55
STOP GTO 99

143*LBL 07
54 RCL 01 X<Y? RTN 2
RCL IND 01 X<Y? RTN
"FIGUR?" PROMPT
STO IND 01 RTN
!
156*LBL 08
ABS 2 X<Y? RTN 78
RCL 13 X<Y? RTN -5.09
STO IND 13 RTN

168*LBL 04
ASTO X "SCHACHMATT"
AVIEW BEEP CLA ARCL X
"I HABE" FC? 04 "I-N"
"I GEMONNEN" PROMPT

180*LBL 05
"PATT" BEEP PROMPT

184*LBL 09
INT 42 - ENTER↑
ENTER↑ 7 / INT 7
X<Y * LASTX RDN -
R↑ 10 * + ARCL X
RTN

205*LBL 06
10 / INT 7 LASTX
RDN * R↑ FRC 10 *
+ 42 + RTN

221*LBL 12
STO 03 ABS CF 05
CF 06 CF 07 2 X<Y?
GTO 13 X<Y 30 +
XEQ IND X RCL IND X
STO 04

236*LBL 14
RCL 02 STO 05

239*LBL 15
RCL IND 04 ST+ 05
RCL 05 50 X<Y? GTO 00
X<Y XEQ 08 FS? 18
GTO 00 X<0? GTO 00
CF 09 X=0? SF 09
XEQ 12 FS? 05 GTO 00
FS? 09 GTO 15

260*LBL 00
ISG 04 GTO 14 RTN

264*LBL 36
SF 07

266*LBL 33
SF 05

268*LBL 32
269*LBL 34
270*LBL 35
RTN

272*LBL 13
SF 06 RCL 02 7 XEQ 09
FS? 18 1 X=0? XEQ 12
RCL 02 8 XEQ 09
FS? 18 CLX X<0?
XEQ 12 RCL 02 6
XEQ 09 FS? 18 RTN
X<=0? RTN

295*LBL 12
CF 08 STO 07 FRC 1 E2
* STO 06 RCL 2 STO 08
.4 FS? 07 ST- 06
FS? 06 XEQ 12 FC? 07
XEQ 13 RCL 00 RCL 06
X<=Y? RTN RCL 03
FS? 08 -5.09
STO IND 08 CLX
STO IND 02 50.082
STO 11 STO 09 CF 19

325*LBL 21
RCL IND 11 SIGN X=0?
GTO 00 LASTX X<=0?
GTO 00 XEQ 07 FS? 19
GTO 04

336*LBL 00
ISG 11 GTO 21 RCL 09
STO 00 RCL 02 STO 12
RCL 08 STO 13

345*LBL 04
RCL 03 STO IND 02
RCL 07 STO IND 08 RTN

351*LBL 12
.5 ST+ 06 84 RCL 08
X<Y? RTN SF 08 9
ST+ 06 RTN

362*LBL 13
FS? 06 GTO 13 RCL 03
30 - RCL IND X STO 01

370*LBL 03
RCL 08 STO 10

373*LBL 10
RCL IND 01 ST+ 10
RCL 10 50 X>Y? GTO 00
X<>Y XEQ 08 FS? 18
GTO 00 XEQ 12 X=Y?
RTN FS? 05 GTO 00
LASTX X=0? GTO 10

392*LBL 00
ISG 01 GTO 03 RTN

396*LBL 13
RCL 08 8 XEQ 00
RCL 08 6

402*LBL 00
XEQ 09 FS? 18 RTN

406*LBL 12
INT 6 X=Y? RTN .41
ST+ 06 RDN RTN

415*LBL 07
CF 01 CF 02 CF 03 2
X>Y? GTO 12 X<>Y 30
+ XEQ IND X RCL IND X
STO 14

428*LBL 28
RCL 11 STO 15

431*LBL 29
RCL IND 14 ST+ 15
RCL 15 XEQ 08 FS? 18
GTO 00 X>0? GTO 00
CF 10 X=0? SF 10
XEQ 13 FS? 19 RTN
FS? 01 GTO 00 FS? 10
GTO 29

450*LBL 00
ISG 14 GTO 28 RTN

454*LBL 36
SF 03

456*LBL 33
SF 01

458*LBL 32
459*LBL 34
460*LBL 35
RTN

462*LBL 12
SF 02 RCL 11 RCL 17
XEQ 09 FS? 18 1 X=0?
XEQ 13 FS? 19 RTN
RCL 11 RCL 29 XEQ 00
FS? 19 RTN RCL 11
RCL 31

480*LBL 00
XEQ 09 FS? 18 RTN
X=0? X>0? RTN

487*LBL 13
FRC ABS 1 E2 *
FS? 03 .4 FS? 03 -
FS? 02 XEQ 13 RCL 06
X<>Y - RCL 00 X<>Y
X<=Y? SF 19 X<=Y? RTN
RCL 09 X<>Y X<>Y
STO 09 RTN

512*LBL 13
.5 + RCL Z 54 X<>Y
CF 04 X<=Y? SF 04
RCL Z 9 FC? 04 CLX +
RTN

527*LBL 09
+

529*LBL 08
CF 10 SF 25 RCL IND X
SIGN FS?C 25 X=0?
SF 18 LASTX .END.

Beispiel:

01*LBL "P"
ADV SF 12 9 SKPCOL
49,053 STO 13

00*LBL 00
ACCHR 2 SKPCOL X<>Y
ISG X GTO 00 PRBUF
50,054 STO 15

10*LBL 01
RCL 13 ACCHR

21*LBL 02
2 SKPCOL RCL IND 15
INT 43 + RCL IND X
ACSPEC ISG 15 GTO 02
2,007 ST+ 15 ISG 13
GTO 01 ADV ADV CF 12
RTN

40*LBL "5"
RCL d STO [ARCL 17
ARCL 19 RCL [STO d
FC?C 15 SF 15 RCL d
STO ["I-12" RCL \
STO d END

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

HP ZUERST?
N
VON?
41 RUN
NACH?
31 RUN
ICH ZIEHE
VON 22 NACH 31

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

VON?
42 RUN
NACH?
32 RUN
ICH ZIEHE
VON 23 NACH 32

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

VON?
43 RUN
NACH?
32 RUN
ICH ZIEHE
VON 21 NACH 32

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

VON?
54 RUN
NACH?
33 RUN
ICH ZIEHE
VON 24 NACH 33

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

VON?
55 RUN
NACH?
54 RUN
ICH ZIEHE
VON 33 NACH 44

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

VON?
54 RUN
NACH?
44 RUN
ICH ZIEHE
VON 14 NACH 33

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

VON?
52 RUN
NACH?
32 RUN
ICH ZIEHE
VON 12 NACH 32

1 2 3 4 5
1 1 1 1 1
2 1 1 1 1
3 1 1 1 1
4 1 1 1 1
5 1 1 1 1

Logic

LBL LOGIC SIZE 019
 BYTES 670

Das Programm dient zur Prüfung logischer Gesetze (wahr/falsch) und zur Lösung von LOGICALS.

I. Prüfen logischer Gesetze

1. Eingabe des logischen Gesetzes als Programm unter LBL P.
2. Initialisierung des Hauptprogramms durch XEQ LOGIC.
3. Auf die Frage "ELEMENTE ?" wird die Anzahl der Variablen des Gesetzes eingegeben (max. 12).
4. Darauf permutiert der Rechner die möglichen Wahrheitsverteilungen auf die n Elemente durch und zeigt an:
 - a) "WAHR" wenn das Gesetz gültig ist
 - b) "FALSCH" wenn die Wahrheitswertbelegung zu einem falschen Resultat führt; diese Belegung wird anschließend angezeigt, z.B.: ---OIOOIO
5. Im Fall 4.a) führt R/S zur Anzeige "END", da keine weiteren Permutationen der Wahrheitswerte mehr möglich sind.
6. Im Fall 4.b) permutiert der Rechner die restlichen Wahrheitswertbelegungen durch (falls dies nicht bereits die letzte Permutation war).
7. Will man die Befragung einer ganz bestimmten Wahrheitswertbelegung durchführen, so wird die Abfrage durch XEQ PS initialisiert, der Rechner fragt dann nach der Anzahl der Elemente und darauf nach dem Wahrheitswert (0,1) der einzelnen Elemente: "x. WERT". Nach dem letzten Wahrheitswert untersucht der Rechner die Belegung und weist das Ergebnis aus: "WAHR" oder "FALSCH". Für die Fortführung der Abfrage mit permutierter Wahrheitswertverteilung gehe zu 5.

II. Lösen eines LOGICALS

1. 'INTELLIGENTE' Eingabe der Aussagen des LOGICALS als Programm unter dem LBL P (siehe Beispiel).
2. Initialisierung des Hauptprogramms durch XEQ LOGIC.
3. Auf die Frage "ELEMENTE ?" wird die Anzahl der Variablen des LogicalS eingegeben.
4. Darauf permutiert der Rechner die möglichen Wahrheitswertverteilungen durch und zeigt an:
 - a) "UNLOESBAR" wenn das Logical keine Lösung hat
 - b) "LOESUNG" und anschließend die Wahrheitswertbelegung, z.B. "001001010", die zur Lösung führte (nicht unbedingt die einzige Lösung).
5. Eventuelle Befragung der restlichen Permutationen mittels R/S. Dies führt allerdings zur Anzeige "END", falls keine weiteren Permutationen existieren.
6. Will man die Befragung einer ganz bestimmten Wahrheitswertbelegung durchführen, so wird die Abfrage mit XEQ PS gestartet; der Rechner fragt dann nach der Anzahl der Elemente und darauf nach dem Wahrheitswert der einzelnen Elemente: "x. WERT". Nach dem letzten Wahrheitswert untersucht der Rechner die Belegung und weist das Ergebnis aus: "LOESUNG" oder "UNLOESBAR". Für die Fortführung der Abfrage mit permutierter Wahrheitswertverteilung gehe zu 5.

Beispiele

I. Prüfen eines logischen Gesetzes

$\vdash (A \wedge B \rightarrow C) \leftrightarrow (B \wedge A \rightarrow C)$ mit $A=1, B=2, C=3$

Eingabe des Problems als Programm:

LBL P

1	Die Formel enthält 3 Elemente (A, B, C).
ENTER	Also nach XEQ LOGIC wird die Frage
2	"ELEMENTE ?" mit 3 beantwortet.
XEQ AND	Der Rechner permutiert nun sämtliche
3	Wahrheitswertbelegungen durch.
XEQ IF	Findet er keine Belegung, bei der das
2	zu prüfende Gesetz falsch wäre, kündigt
ENTER	er dies durch BEEP und "WAHR" an.
1	
XEQ AND	
3	
XEQ IF	
XEQ EQUI	

II. Lösen eines Logicals

(Quelle: 99 Logeleien, Zweistein)

Einige Familienmitglieder der Meiers werden zu Besuch erwartet. Folgendes ist gewiß:

1. Wenn Vater Meier kommt, dann auch Frau Meier.
 2. Mindestens einer der beiden Söhne Uwe und Kay kommt.
 3. Entweder kommt Frau Meier oder Timm (der dritte Sohn).
 4. Entweder kommen Timm und Kay oder beide nicht.
 5. Wenn Uwe kommt, dann auch Kay und Vater Meier.
- Wer kommt?

Sie Vater=1, Mutter=2, Kay=3, Uwe=4, Timm=5; also 5 Elemente. Dann lassen sich die 'Gleichungen' wie folgt schreiben:

1. $1 \rightarrow 2$
2. $4 \vee 3$
3. $2 \vee 5$
4. $(5 \wedge 3) \rightarrow (\neg 5 \wedge \neg 3)$
5. $4 \rightarrow (3 \wedge 1)$

Verständlicher Weise müssen die fünf Aussagen noch miteinander verknüpft werden, da sie ja als Gesamtheit Gültigkeit haben. Dies geschieht mit der \rightarrow -Funktion (XEQ).

LBL P	2	3	XEQ IF
1	ENTER	XEQ NON	XEQ Σ
ENTER	5	XEQ AND	
2	XEQ XOR	XEQ XOR	
XEQ IF	XEQ Σ	XEQ Σ	
XEQ Σ	5	4	
4	ENTER	ENTER	
ENTER	3	3	
3	XEQ AND	ENTER	
XEQ OR	5	1	
XEQ Σ	XEQ NON	XEQ AND	

Nun folgt der Programmstart mit XEQ LOGIC. Auf die Frage "ELEMENTE" wird die Anzahl der Elemente eingegeben: 5. Dann permutiert der Rechner die verschiedenen Personengruppierungen durch, bis er eine Kombination findet, die zur Gesamtheit der Aussagen 1. bis 5. keine Widerspruch enthält und weist diese Kombination als "LOESUNG" aus, hier 00101, was in diesem Beispiel folgendes bedeutet: Nur Person 3 und Person 5 kommen zu Besuch, mithin Kay und Timm. Läßt man den Rechner weiter permutieren mittels R/S, so findet er keine weitere "LOESUNG" mehr (die Lösung ist also eindeutig!) und weist für die restlichen Permutationen "UNLOESBAR" aus.

01*LBL "LOGIC"	50 SF 06	99 PROMPT	148 RCL 18	197 STO IND 13
02 ZREG 01	51*LBL 00	100 GTO 04	149 -	198 DSE 15
03 CF 04	52 ISG 15	101*LBL 05	150 FACT	199 GTO 11
04 CF 05	53 GTO 02	102 FS?C 01	151 /	200*LBL 12
05*LBL 15	54 FC?C 03	103 GTO 06	152 STO 17	201 CF 03
06 FIX 0	55 SF 00	104 "W"	153 RTN	202 DSE 17
07 CF 00	56 E	105 BEEP	154*LBL 09	203 RTN
08 CF 01	57 FC? 04	106 GTO 00	155 RCL IND 13	204 CF 02
09 CF 02	58 RCL 17	107*LBL 06	156 ISG 13	205 DSE 18
10 CF 03	59 FC?C 06	108 "UNLOESBAR"	157 RCL IND 13	206 GTO 17
11 CF 06	60 STO 17	109 TONE 2	158 X?Y?	207 CLX
12 CF 22	61 AVIEW	110*LBL 00	159 GTO 00	208 STO IND 00
13 CF 29	62 TONE 0	111 PROMPT	160 X*Y?	209 RTN
14 CLRG	63 XEQ "P"	112 FS?C 00	161 SF 03	210*LBL "PS"
15 12	64 ASTO 13	113 GTO 04	162 RCL 00	211 SF 04
16 "ELEMENTE?"	65 ASHF	114 SF 02	163 RCL 13	212 SF 05
17 TONE 9	66 ASTO 14	115*LBL 17	164 X?Y?	213 XEQ 15
18*LBL 01	67 FS?C 01	116 FC?C 05	165 GTO 09	214 E
19 AVIEW	68 GTO 03	117 FS? 00	166 GTO 12	215 RCL 01
20 PSE	69 RCL IND X	118 GTO 05	167*LBL 00	216 STO 18
21 FC? 22	70 X=0?	119 1,9	168 E	217*LBL 13
22 GTO 01	71 GTO 17	120 STO 13	169 STO IND 13	218 RDN
23 X=0?	72 SF 03	121 FS? 02	170 ST- 13	219 CLA
24 X?Y?	73 "F:"	122 GTO 00	171 CLX	220 ARCL Y
25 GTO 15	74 TONE 2	123 CLS	172 STO IND 13	221 "I.WERT"
26 STO 00	75 GTO 00	124 ZREG 07	173 STO 15	222 CF 22
27 STO 18	76*LBL 03	125 CLS	174 FC?C 03	223 TONE 9
28 E3	77 RCL 15	126 ZREG 01	175 GTO 12	224*LBL 14
29 /	78 X=0?	127 RCL 18	176 RCL 13	225 AVIEW
30 E	79 GTO 17	128 STO 15	177 INT	226 PSE
31 +	80 "LOESUNG"	129 E	178 E	227 FC? 22
32 STO 16	81 BEEP	130*LBL 07	179 -	228 GTO 14
33 FS? 05	82*LBL 00	131 STO IND 15	180 E3	229 X?Y?
34 RTN	83 CF 05	132 DSE 15	181 /	230 GTO 13
35 GTO 17	84 AVIEW	133 GTO 07	182 STO 14	231 STO IND Z
36*LBL 16	85 PSE	134*LBL 00	183 ISG 14	232 ST+ 18
37 19,9	86 CLA	135 XEQ "PM"	184*LBL 10	233 ISG Z
38 STO 14	87 FS?C 03	136 SF 02	185 RCL IND 14	234 GTO 13
39 RCL 16	88 "----"	137 GTO 16	186 X=0?	235 XEQ 08
40 STO 15	89 ARCL 13	138*LBL "PM"	187 GTO 11	236 GTO 16
41 CLA	90 ARCL 14	139 FS? 02	188 E	237*LBL "XOR"
42*LBL 02	91 PROMPT	140 GTO 09	189 STO IND 14	238 SF 03
43 ARCL IND 15	92 SF 02	141*LBL 08	190 ST+ 15	239*LBL "EQUI"
44 RCL IND 15	93 FC?C 00	142 RCL 00	191 ISG 14	240 STO 13
45 X=0?	94 GTO 17	143 FACT	192 GTO 10	241 X<> IND 13
46 SF 03	95*LBL 04	144 RCL 18	193*LBL 11	242 STO IND 13
47 X=0?	96 CF 02	145 FACT	194 E	243 X<>Y
48 GTO 00	97 CF 04	146 /	195 ST- 13	244 STO 13
49 FS? 03	98 "END"	147 RCL 00	196 CLX	245 X<> IND 13

246 STO IND 13	305 X<> IND 13
247 X>Y?	306 ST* IND 14
248 X<>Y	307 X<> IND 13
249 X#0?	308 X<>Y
250 GTO 00	309 GTO 19
251 X<>Y	310*LBL "AND"
252 X=0?	311 XEQ "AND"
253 GTO 00	312*LBL "NON"
254 +	313 STO 13
255 FC?C 03	314 X<> IND 13
256 CLX	315 X#0?
257 GTO 18	316 GTO 00
258*LBL 00	317 X<> IND 13
259 X<> 13	318 GTO 18
260 +	319*LBL 00
261 FS?C 03	320 X<> IND 13
262 CLX	321 CLX
263 GTO 18	322*LBL 18
264*LBL "IF"	323 STO IND 14
265 STO 13	324 RDN
266 X<> IND 13	325*LBL 19
267 X#0?	326 RCL 14
268 GTO 00	327 ISG 14
269 X<> IND 13	328 RTN
270 X<>Y	329*LBL "Σ"
271 STO 13	330 SF 01
272 X<> IND 13	331 19,9
273 X=0?	332 STO 14
274 GTO 00	333 RCL IND Y
275 X<> IND 13	334 ST* 15
276 RDN	335 X=0?
277 CLX	336 GTO 17
278 GTO 18	337 END
279*LBL 00	
280 X<> IND 13	
281 +	
282 GTO 18	
283*LBL "OR"	
284 STO 13	
285 X<> IND 13	
286 STO IND 14	
287 X<> IND 13	
288 RDN	
289 STO 13	
290 X<> IND 13	
291 ST+ IND 14	
292 X<> IND 13	
293 RDN	
294 GTO 19	
295*LBL "NOR"	
296 XEQ "OR"	
297 GTO "NON"	
298*LBL "AND"	
299 STO 13	
300 X<> IND 13	
301 STO IND 14	
302 X<> IND 13	
303 RDN	
304 STO 13	

Liebe Clubmitglieder!

Die Reaktion auf das Bubble-Sort Programm war zwar nicht ganz so groß wie bei SORT3, aber einige abdruckreife Programme habe ich doch erhalten. Herzlichen Dank!

Ein paar Seiten weiter werdet Ihr eine Optimierung des DEZ-HEX Programmes von prisma 218/81 finden. Dort fehlt nach Zeile 11 ein CLA. Zum Schluß noch eine Frage: WER kann mir das Barcode-Heft für die Programmsammlung "Höhere Mathematik" leihen? Ansonsten viel Spaß mit den Programmen, und schreibt mal wieder.

Euer Andreas

BUBBLE SORT

SIZE 000 !!!

EINGABE: AAA.EEE R/S.
HOBEI AAA=ANFANGSPEC.
EEE=ENDREGISTER

DAS PROGRAMM VERWENDET
NUR STACK UND LAST X-REG!
NACH DEM SORTIEREN STEHT
DIE HOCHSTE ZAHL IN
KLEINSTEM REGISTER.
BEI ALPHA-DATEN ER-
SCHEINT ALPHA DATA

© COPYRIGHT by
H.Klemp 8/1981
6152

Happy
Programming
Ins.(Grad.) Hasen Klemp (73)
Dianastr. 25
1000 Berlin 28

```
01*LBL "BSORT"  
02 STO L  
03*LBL 01  
04 ENTER↑  
05 ISG X  
06 GTO 05  
07 GTO 04  
08*LBL 05  
09 RCL IND Y  
10 RCL IND Y  
11 X<=Y?  
12 GTO 03  
13 X< IND T  
14 X< IND Z  
15 SF 00  
16*LBL 03  
17 RT  
18 ISG X  
19 GTO 01  
20*LBL 04  
21 RCL L  
22 FS?C 00  
23 GTO 01  
24 END  
END 49 BYTES
```

*Das Programm wird etwas schneller, wenn man
nach Zeile 24
0001 -
einfügt.
:indicates*

P.S.: Nähere Informationen zum Sortieren finden sich in CHIP 7/81

Das Programm noch einmal in groß

```
01*LBL "BSORT"      13 X< IND T  
02 STO L            14 X< IND Z  
03*LBL 01           15 SF 00  
04 ENTER↑          16*LBL 03  
05 ISG X            17 RT  
06 GTO 05           18 ISG X  
07 GTO 04           19 GTO 01  
08*LBL 05           20*LBL 04  
09 RCL IND Y        21 RCL L  
10 RCL IND Y        22 FS?C 00  
11 X<=Y?           23 GTO 01  
12 GTO 03           24 END
```

Berechnung der Monatslohnsteuer mit dem HP-41C

Raimund Berg, Hagener Straße 200, 5910 Kreuztal-Eichen

Das Programm dient zur monatlichen Lohn- und Gehaltsabrechnung. Es errechnet den Bruttolohn, die steuer- und sozialversicherungspflichtigen Beträge, die Abzüge einschließlich VWL und Sozialversicherung und bestimmt den Auszahlungsbetrag. Mit dem Programm kann in einfacher Weise die monatliche Lohn-/Gehaltsabrechnung erstellt oder überprüft werden.

Ausführung: Nach jeder Dateneingabe und jedem PROMPT Rechner mit R/S starten. Bei der Dateneingabe dürfen 2 Stackregister benutzt werden. Ist eine Eingabe nicht erforderlich, nur R/S drücken.

Schritt:	Tasten:	Anzeige:	Eingaben:
1	XEQ'LST	ST.KL?	Steuerklasse 1 bis 6 eingeben
2	R/S	KINDER?	Kinderanzahl laut Lohnsteuerkarte eingeben
3	R/S	RELIGION?	Religionsgemeinschaft laut Lohnsteuerkarte eingeben: E = ev, K = rk, Space = ohne 1. Buchstabe = Steuerpflichtiger 2. Buchstabe = Ehegatte Beispiel: KK = beide Ehegatten rk E_ = Steuerpflichtiger. ev, Ehegatte ohne Religionsgemeinschaft Rechner hält im ALPHA-Modus an.
4	R/S	ALTER?	Lebensalter eingeben. Ab 63 Jahre keine Arbeitslosenversicherungsbeiträge, ab 62 Jahre Altersentlastungsbeitrag
5	R/S	GEHALT?	Bei Angestellten monatliches Grundgehalt eingeben. Weiter bei Schritt 15. Wenn Arbeiter, weiter bei Schritt 6.
6	R/S	ECKLOHN?	Basisstundenlohn eingeben
7	R/S	GSTD.=?	Gesamte Arbeitsstunden einschl. Überstunden eingeben
8	R/S	GSTD.=...	Betrag der Gesamtstunden
9	R/S	Z-STD=?	Zusätzliche Stunden mit Zuschlag eingeben (z.B. Überstunden, Nachtstunden, Sonntagsstunden, Stunden mit Schmutzzulage usw.). Hier wird nur die zusätzliche Vergütung berechnet. Der Grundlohn ist bei Schritt 7 mit berücksichtigt worden. Ohne Eingabe, weiter bei 15.
10	R/S	% ?	Zulage in % des Ecklohns. Wenn keine Eingabe, weiter bei Schritt 15.

Schritt:	Tasten:	Anzeige:	Eingaben:
11	R/S	%=...	Betrag der Zulage pro Stunde
12	R/S	Z-STD=...	Betrag der Zulage gesamt
13	R/S	STPFL=?	Zulage steuerpflichtig? (Nacht-, Sonntags-, Feiertags- usw. -zu- lagen sind steuer- und sozial- versicherungsfrei) Wenn ja, 'J eingeben, Rechner hält im ALPHA-Modus an.
14	R/S	SOZV.=?	Zulage sozialversicherungspflich- tig? Wenn ja, 'J eingeben, Rech- ner hält im ALPHA-Modus an. Rechner geht nach Schritt 9.
15	R/S	VWL AN/AG?	Vermögenswirksame Leistungen: Anteil Arbeitnehmer, ENTER , An- teil Arbeitgeber. Ist ein Anteil gleich Null, so muß diese mit eingegeben werden.
16	R/S	GRATI.?	Ggfs. Weihnachtsgratifikation eingeben. Bei Dateneingabe wird DM 600,- Steuerfreibetrag und bis zu DM 100,- Sozialversiche- rungsfreibetrag berechnet.
17	R/S	SONSTIGES?	Sonstige Bezüge, die zum Brutto- lohn gehören, eingeben (z.B. zu- sätzliches Urlaubsgeld, Lohnfort- zahlung usw.). Rechner fragt an- schließend, ob der Betrag sozial- versicherungs- bzw. steuerpflichtig ist. Ist der Betrag nur teilweise sozialversicherungs- oder steuer- pflichtig, müssen die Teilbeträge getrennt eingegeben werden. Wenn keine Eingabe, weiter bei Schritt 20.
18	R/S	STPFL=?	Betrag steuerpflichtig? Wenn ja, 'J eingeben, Rechner hält im ALPHA-Modus an.
19	R/S	SOZV.=?	Betrag sozialversicherungspflich- tig? Wenn ja, 'J eingeben, Rech- ner hält im ALPHA-Modus an. Weiter bei Schritt 17.
20	R/S	FREIBETRAG?	Steuerfreibetrag laut Lohnsteuer- karte eingeben (nicht den Weih- nachtsfreibetrag, Altersentla- stungsbetrag, steuerfreie Bezüge).
21	R/S	STPFL=...	steuerpflichtiger Betrag
22	R/S	SOZV.=...	sozialversicherungspflichtiger Betrag
23	R/S	BRUTTO=...	Bruttolohn bzw. -gehalt
24	R/S	LST=...	Lohnsteuerbetrag

Schritt:	Tasten:	Anzeige:	Eingaben:
25	R/S	E KST=...	Wenn ev. Kirchensteuerpflicht, erfolgt Anzeige.
26	R/S	K KST=...	Wenn rk. Kirchensteuerpflicht, erfolgt Anzeige.
27	R/S	G=...	Krankenversicherungsarbeitnehmeranteil. Das Programm berücksichtigt bei den Sozialversicherungen die Beitragsbemessungsgrenze sowie die Geringverdienergrenze. Ab der Beitragsbemessungsgrenze sind Angestellte beitragsfrei.
28	R/S	K=... L=... oder	Arbeitnehmeranteil Rentenversicherung Arbeiter Arbeitnehmeranteil Rentenversicherung Angestellte
29	R/S	M=...	Arbeitnehmeranteil Arbeitslosenversicherung. Ab 63 Jahre beitragsfrei.
30	R/S	VWL=...	VWL-Gesamtabzug
31	R/S	ABZUEGE=?	Abzüge für Abschlagszahlungen, Lohnaufrechnung, usw.
32	R/S	ABZUEGE=...	Gesamtabzüge
33	R/S	NETTO=...	Nettolohn
34	R/S	ERS.K=181,50	Arbeitgeberanteil der Krankenversicherung für freiwillig versicherte Angestellte mit einem Gehalt über der Beitragsbemessungsgrenze.
35	R/S	SPARZUL=...	Sparzulage für VWL-Beträge
36	R/S	ERSTATTUNG ?	Erstattungen für Fahrgelder, Auslagen usw. eingeben
37	R/S	AUSZAHLEN=...	Auszahlungsbetrag

Abzüge vom Bruttolohn: Schritte 24, 25, 26, 27, 28, 29, 30, 31
Hinzurechnungen: Schritte 34, 35, 36

Das Programm überprüft nicht alle Daten vollständig auf Richtigkeit. Somit kann bei falscher Eingabe u.U. eine fehlerhafte Berechnung erfolgen. Zeigt das Programm DATA ERROR an, empfiehlt es sich, mit XEQ'LST neu zu beginnen.

Das Programm berücksichtigt VWL bis DM 52,-/Monat. Der darüber gehende Betrag erhält keine Sparzulage und muß im Rahmen der Abzüge (Schritt 31) eingegeben werden.

Das Programm behandelt jeden Angestellten mit einem Bruttolohn über der Beitragsbemessungsgrenze als freiwillig krankenversichert. Ist dies nicht der Fall, so müssen die DM 181,50 Arbeitgeberanteil an der Krankenversicherung nicht hinzugerechnet werden und DM 181,50 dem Nettolohn abgezogen werden. Der Beitrag von DM 181,50 gilt natürlich nur bei einem Beitragssatz von 11,0 % bzw. 5,5 % für jeweils Arbeitgeber und Arbeitnehmer.

Die Erstattungs- und Abzugsbeträge müssen jeweils in einer Summe eingegeben werden.

Das Programm muß ggfs. den jeweiligen Erfordernissen angepaßt werden, da die Lohnabrechnung ziemlich umfangreich sein kann und die Konstanten sich recht schnell ändern können. Das abgedruckte Programm benutzt folgende einprogrammierte Werte:

Zeile:	Bemerkung:
146	Höchstbetrag der zulagefähigen VWL
172	Altersentlastungsbetrag DM 250,-/Monat höchstens
193	Vorsorgepauschale 9 % vom Jahresgehalt
195	Vorsorgepauschale höchstens jedoch 9 % der Jahresrentenbemessungsgrenze
202	Weihnachtsfreibetrag DM 600,- pauschal
204, 205	Weihnachtsfreibetrag der Sozialversicherung bis zu DM 100,-
240-249	Rundung auf die jeweiligen Eingangsstufen des Monats- tarifs, ab 1981 DM 4,50-Stufen.
254	Summe der Jahresfreibeträge Steuerklasse I und IV ohne Vorsorgepauschale
260	zusätzlicher Jahresfreibetrag bei der Steuerklasse II, wenn Kinder zu berücksichtigen
262	Grundjahresfreibeträge Steuerklasse II
267	Summe der Jahresfreibeträge Steuerklasse III ohne Vor- sorgepauschale
293	Summe der Jahresfreibeträge Steuerklasse V
295	Rundungsbetrag Steuerklasse VI
329-338	Berechnung des Kinderfreibetrags bei der Kirchensteuer
344	Kirchensteuersatz 8 %, in Berlin, Hessen, Niedersachsen, Nordrhein-Westfalen, Rheinland-Pfalz, Saarland und Schleswig-Holstein 9 %.
349-351	Mindestbetrag der Kirchensteuer, nur in Baden-Württem- berg, Hamburg, Niedersachsen, Schleswig-Holstein DM 0,60, in Hessen DM 0,30, in den anderen Ländern kein Mindest- betrag
371	die Hälfte der Monatslohnsteuer-Eingangsstufen, ab 1981 DM 2,25
374	10 % der Beitragsbemessungsgrenze, 1981 DM 440,-. Achtung: dieser Betrag ändert sich jedes Jahr.
389	Arbeitnehmeranteil an der Krankenversicherung, bei jeder Krankenkasse unterschiedlich, \emptyset ca. 5,5 %
401	Arbeitnehmeranteil an der Rentenversicherung, ab 1981 9,25 %
413	Arbeitnehmeranteil an der Arbeitslosenversicherung, 1,5 %
445	Höchstbetrag des Krankenversicherungsbeitrags: 1981: 5,5 % von (75 % der Beitragsbemessungsgrenze (DM 4.400,-))

Zeile:	Bemerkung:
461	,3 = 30 % Sparzulage bei VWL bis einschl. 2 Kinder
463	,4 = 40 % Sparzulage bei VWL ab einschl. 3 Kinder
477	Betrag zur Berechnung der Vorsorgepauschale, ab 1982 DM 4.680,-
482	Betrag zur Berechnung der Vorsorgepauschale
506, 609	Rundungsbetrag der Jahreslohnsteuer, ergibt die Eingangsstufen der Jahressteuer
523-610	Berechnung der Jahressteuer nach § 32 a EStG

Abänderung des Programms für 1980, 1981, 1982 (abgedruckt 1981)

Zeile:	1980:	1981:	1982:
195	4536	4752	9 % der Jahresbeitragsbemessungsgrenze
240	,5	3	3
242	,4	4,5	4,5
243	X	/	/
245	,4	4,5	4,5
246	/	X	X
247	1,99	1,49	1,49
254	1794	1314	1314
260	2160	2136	3348
262	2634	2178	2178
267	2544	1584	1584
295	24	18	18
371	1,25	2,25	2,25
374	420	440	10 % der Monatsbeitragsbemessungsgrenze
445	173,25	181,5	Höchstbeitrag zur Krankenversicherung bei 5,5 %
477	4200	4200	4680
505-510	LBL 18 48E3 X<>Y X≠Y? SF 07 60 FS?C 07 XEQ 17 / LASTX X<>Y INT X RTN	LBL 18 54 / INT 54 X RTN	LBL 18 54 / INT 54 X RTN
529	4,8	6	6
533	1,6	1,8	1,8

Zeile:	1980:	1981:	1982:
536	3,2	4,2	4,2
542	16019	18E3	18E3
547	10,86	3,05	3,05
549	154,42	73,76	73,76
553	925	695	695
561	2708	3034	3034
566	130019	13E4	13E4
571	,1	,09	,09
573	6,07	5,45	5,45
577	109,95	88,13	88,13
581	4800	5040	5040
585	15298	20018	20018
593	13664	14837	14837
598	3719	4213	4213
607	812	926	926

Bedeutung der Flags, wenn sie gesetzt sind:

FO0: Angestellter	F15: rk. Kirchensteuerpflicht
FO1: Steuerklasse I	F16: ev. Kirchensteuerpflicht
FO2: Steuerklasse II	F17: rk. und ev. Kirchensteuerpflicht
FO3: Steuerklasse III	F18: keine Kirchensteuerpflicht
FO4: Steuerklasse IV	F19: Altersentlastungsbetrag berücksichtigen
FO5: Steuerklasse V	F20: keine Arbeitslosenversicherung
FO6: Steuerklasse VI	F22: benutzt
FO7: benutzt	F27: gesetzt
FO8: benutzt	F29: gesetzt

Datenspeicher:

RO0: Bruttolohn	RO6: benutzt
RO1: Ecklohn	RO7: benutzt
RO2: Steuerpflichtiger Betrag	RO8: benutzt
	RO9: Anzahl Kinder
RO3: Sozialversicherungs- pflichtiger Betrag	R10: Steuerklasse, benutzt
RO4: VWL	

Status: SIZE 011, FIX 2, SF 27, CF 28, SF 29, Register: 179 = 3 Memory Module.

Das Programm wurde anhand der "Lohnsteuertabelle 1980", "Sozialversicherungs-Tabellen" vom Fachverlag Schäffer GmbH & Co., Stuttgart, und dem ESTG i.d.F.v. 21.06.1979 und i.d.F.v. 25.06.1980 erstellt. Der Verfasser übernimmt keine Garantie für die Richtigkeit des Programms und lehnt jede Haftung, die bei Benutzung des Programms entsteht, ab. Das Programm darf außerhalb des Bereichs des HP-Anwender-Club Oliver Rietschel nur mit Zustimmung des Verfassers verbreitet werden.

Raimund Berg, Hagener Straße 200, 5910 Kreuztal-Eichen

01*LBL "LST	80 CF 22	156*LBL 12	233 FC?C 07
02 0	81 "GEHALT?	157 "SONSTIG	234 ST- 02
03 STO d	-	ES? "	235 FC?C 08
04 CLRG	82 PROMPT	158 PROMPT	236 ST- 03
05 SF 29	83 FS? 22	159 FS? 22	237 AOFF
06 SF 27	84 STO 00	160 XEQ 13	238 RTN
07 FIX 2	85 FS? 22	161 FS?C 22	239*LBL 14
08 "STPFL="	86 SF 00	162 GTO 12	240 3
09 ASTO 06	87 FS?C 22	163 "FREIBET	241 +
10 "SOZV.="	88 GTO 08	RAG? "	242 4,5
11 ASTO 07	89 "ECKLOHN	164 PROMPT	243 /
12 6	? "	165 FS?C 22	244 INT
13 "ST.KL? "	90 PROMPT	166 ST- 02	245 4,5
14 PROMPT	91 X<0?	167 RCL 00	246 *
15 INT	92 LOG	168 ST+ 02	247 1,49
16 X<=0?	93 STO 01	169 ST+ 03	248 +
17 LOG	94 CF 22	170 ,4	249 RTN
18 X>Y?	95 "GSTD.=?	171 *	250*LBL 01
19 ASIN	-	172 250	251 0
20 SF IND X	96 ASTO 08	173 X>Y?	252 STO 09
21 STO 10	97 PROMPT	174 X<>Y	253*LBL 04
22 CF 22	98 FC?C 22	175 FS?C 19	254 1314
23 "KINDER?	99 0	176 ST- 02	255 GTO 15
-	100 X<0?	177 CLA	256*LBL 02
24 PROMPT	101 LOG	178 ARCL 06	257 RCL 09
25 FC?C 22	102 *	179 ARCL 02	258 X*0?
26 0	103 STO 00	180 PROMPT	259 SF 07
27 INT	104 CLA	181 CLA	260 2136
28 X<0?	105 ARCL 08	182 ARCL 07	261 ENTER↑
29 LOG	106 ARCL X	183 ARCL 03	262 2178
30 STO 09	107 PROMPT	184 PROMPT	263 FS?C 07
31 SF 18	108*LBL 09	185 "BRUTTO="	264 +
32 "RELIGIO	109 "Z-STD=?	-	265 GTO 15
N? "	-	186 ARCL 00	266*LBL 03
33 AON	110 ASTO 08	187 PROMPT	267 1584
34 PROMPT	111 PROMPT	188 RCL 02	268*LBL 15
35 AOFF	112 FC?C 22	189 XEQ 14	269 STO 10
36 ASTO 00	113 GTO 08	190 12	270 XEQ 16
37 " "	114 RCL 01	191 *	271 SF 07
38 ARCL 00	115 "? ? "	192 STO 07	272 XEQ 16
39 ASTO 00	116 PROMPT	193 9	273 RCL Z
40 ASHF	117 FC?C 22	194 %	274 +
41 ASTO Z	118 GTO 08	195 4752	275 XEQ 18
42 " "	119 %	196 X>Y?	276 RCL 10
43 ARCL 00	120 RND	197 X<>Y	277 +
44 ASHF	121 "%="	198 STO 08	278 CHS
45 ASTO X	122 ARCL X	199 GTO IND	279 RCL 07
46 "K "	123 PROMPT	10	280 +
47 ASTO Y	124 RCL Z	200*LBL 11	281 FS? 03
48 X=Y?	125 *	201 ST+ 00	282 XEQ 19
49 SF 15	126 RND	202 600	283 FC? 03
50 X=Y?	127 CLA	203 ST- 02	284 XEQ 20
51 CF 18	128 ARCL 08	204 6	285 GTO 21
52 "E "	129 ARCL X	205 /	286*LBL 05
53 ASTO Y	130 PROMPT	206 X>Y?	287*LBL 06
54 X=Y?	131 XEQ 13	207 X<>Y	288 SF 09
55 SF 16	132 GTO 09	208 ST- 03	289 0
56 X=Y?	133*LBL 08	209 RTN	290 STO 09
57 CF 18	134 "VML AM↑	210*LBL 13	291 RCL 07
58 FS? 18	AG? "	211 RDN	292 FS? 05
59 GTO 07	135 PROMPT	212 RDN	293 1044
60 "K "	136 FC?C 22	213 AON	294 FS? 06
61 FS? 15	137 GTO 10	214 CLA	295 18
62 "E "	138 X<0?	215 ARCL 06	296 -
63 ASTO Y	139 LOG	216 "+? "	297 XEQ 18
64 RDN	140 ST+ 00	217 PROMPT	298 STO 07
65 X=Y?	141 +	218 ASTO X	299 STO 10
66 GTO 07	142 LASTX	219 "J "	300 ,22
67 SF 15	143 X<>Y	220 ASTO Y	301 ST* 10
68 SF 16	144 X<Y?	221 X=Y?	302 RDN
69 SF 17	145 ASIN	222 SF 07	303 1,5
70*LBL 07	146 52	223 CLA	304 *
71 CLX	147 X<>Y	224 ARCL 07	305 XEQ 19
72 "ALTER? "	148 X>Y?	225 "I? "	306 X<> 07
73 PROMPT	149 ASIN	226 PROMPT	307 2,5
74 INT	150 STO 04	227 ASTO X	308 *
75 63	151*LBL 10	228 X=Y?	309 XEQ 19
76 X<Y?	152 "GRATI.?"	229 SF 08	310 RCL 07
77 SF 19	-	230 RDN	311 -
78 X<=Y?	153 PROMPT	231 RDN	312 RCL 10
79 SF 20	154 FS?C 22	232 ST+ 00	313 INT
	155 XEQ 11		314 X<=Y?

315 X<>Y	397 "G="	475 PROMPT	557 2200
316*LBL 21	398 ARCL X	476*LBL 16	558 +
317 1,2	399 PROMPT	477 4200	559 RCL 06
318 /	400 RCL 03	478 FS? 07	560 *
319 INT	401 9,25	479 XEQ 17	561 3034
320 10	402 FS? 07	480 FC? 03	562 +
321 /	403 CLX	481 XEQ 17	563 INT
322 "LST="	404 %	482 600	564 RTN
323 ARCL X	405 RND	483 FS?C 07	565*LBL 24
324 PROMPT	406 ST+ 01	484 XEQ 17	566 13 E4
325 STO 01	407 "K="	485 FS? 04	567 RCL 05
326 RCL 09	408 FS? 00	486 XEQ 17	568 X>Y?
327 X=0?	409 "L="	487 RCL 09	569 GT0 25
328 SF 07	410 ARCL X	488 *	570 RCL 06
329 1	411 PROMPT	489 +	571 ,09
330 -	412 CLX	490 RCL 08	572 *
331 150	413 1,5	491 X>Y?	573 5,45
332 *	414 FS?C 07	492 X<>Y	574 -
333 20	415 CLX	493 RTN	575 RCL 06
334 -	416 FS?C 20	494*LBL 17	576 *
335 50	417 CLX	495 2	577 88,13
336 X>Y?	418 %	496 /	578 +
337 X<>Y	419 RND	497 RTN	579 RCL 06
338 RDN	420 ST+ 01	498*LBL 19	580 *
339 FS? 04	421 "M="	499 2	581 5040
340 XEQ 17	422 ARCL X	500 /	582 +
341 FS?C 07	423 PROMPT	501 XEQ 20	583 RCL 06
342 CLX	424 RCL 04	502 2	584 *
343 -	425 ST+ 01	503 *	585 20018
344 8	426 "VWL="	504 RTN	586 +
345 %	427 ARCL X	505*LBL 18	587 INT
346 XEQ 22	428 PROMPT	506 54	588 RTN
347 X<=0?	429 "ABZUEGE	507 /	589*LBL 25
348 SF 18	=?"	508 INT	590 RCL 05
349 ,6	430 ASTO 10	509 54	591 ,56
350 X<=Y?	431 PROMPT	510 *	592 *
351 RDN	432 FS?C 22	511 RTN	593 14837
352 FS?C 18	433 ST+ 01	512*LBL 22	594 -
353 CLX	434 CLA	513 1 E2	595 INT
354 FS? 17	435 ARCL 10	514 *	596 RTN
355 XEQ 17	436 "+E="	515 INT	597*LBL 26
356 XEQ 22	437 ARCL 01	516 1 E2	598 4213
357 "E KST="	438 PROMPT	517 /	599 X<>Y
358 ARCL X	439 RCL 00	518 RTN	600 X>Y?
359 FS? 16	440 RCL 01	519*LBL 23	601 GT0 27
360 PROMPT	441 -	520 X<=Y?	602 0
361 "K KST="	442 "NETTO="	521 SF 08	603 RTN
362 ARCL X	443 ARCL X	522 RTN	604*LBL 27
363 FS? 15	444 PROMPT	523*LBL 20	605 ,22
364 PROMPT	445 181,5	524 FC? 09	606 *
365 ENTER↑	446 X<>Y	525 XEQ 18	607 926
366 FS? 17	447 "ERS.K="	526 STO 05	608 -
367 +	448 ARCL Y	527 1 E4	609 INT
368 ST+ 01	449 FS? 08	528 /	610 END
369 RCL 03	450 PROMPT	529 6	
370 XEQ 14	451 FS?C 08	530 X<>Y	
371 2,25	452 +	531 X>Y?	
372 -	453 RCL 09	532 SF 07	
373 STO 03	454 2	533 1,8	
374 40	455 X<Y?	534 -	
375 X>Y?	456 SF 07	535 STO 06	
376 SF 07	457 RDN	536 4,2	
377 10	458 RDN	537 -	
378 *	459 RCL 04	538 FS? 07	
379 X<=Y?	460 FC? 07	539 STO 06	
380 STO 03	461 ,3	540 FS?C 07	
381 4	462 FS?C 07	541 GT0 24	
382 /	463 ,4	542 18 E3	
383 3	464 *	543 RCL 05	
384 *	465 "SPARZUL	544 X<=Y?	
385 FS? 00	.-"	545 GT0 26	
386 XEQ 23	466 ARCL X	546 RCL 06	
387 X>Y?	467 PROMPT	547 3,05	
388 X<>Y	468 +	548 *	
389 5,5	469 "ERSTATT	549 73,76	
390 FS? 07	UNG ?"	550 -	
391 CLX	470 PROMPT	551 RCL 06	
392 FS? 08	471 FS?C 22	552 *	
393 CLX	472 +	553 695	
394 %	473 "AUSZAHL	554 +	
395 RND	EN="	555 RCL 06	
396 ST+ 01	474 ARCL X	556 *	

Name: DIN-ASA-Umrechnung

Beschreibung: Das Programm rechnet DIN-Werte in ASA-Werte um. Es ist für alle Fotofreunde interessant, die keine Umrechnungstabelle besitzen oder auf deren Apparaten keine 2 Skalen aufgedruckt sind. Die DIN- und ASA-Werte zeigen die Empfindlichkeit eines Filmes an. Je höher der Wert, um so lichtempfindlicher ist der Film.

Entwicklung: Raimund Berg, Hagener Str. 200, 5910 Kreuztal 6

Register: 22

Speicher: STO 00 Eingabewert

Status: beliebig, \uparrow AD und \uparrow DA können irgendwelchen Tasten zugeordnet werden

Ausführung: 1. ASA-Wert eingeben ($3 \leq x \leq 12800$), XEQ \uparrow AD
2. DIN-Wert eingeben ($6 \leq x \leq 42$), XEQ \uparrow DA

Listing:

001	LBL \uparrow DA	DIN \rightarrow ASA	036	LBL 03	ASA-Wert be-
	FIX 0	Status festlegen		2	rechnen
	CF 29			RCL 00	
	INT	etwaiger Dezimalteil		3	Anfangswert
	STO 00	unterdrücken, ab-		/	INT $\left \frac{\text{DIN}}{3} - 14 \right.$
		speichern		14	2
	3			-	
	MOD			ABS	
	GTO IND X	die passende Reihe		INT	
		aussuchen		y ^x	
009	LBL 00	1. Reihe		/	
	12800	Anfangswert		RTN	
	XEQ 03		048	LBL \uparrow AD	ASA \rightarrow DIN
	INT			FIX 0	Status fest-
013	LBL 04	Anzeige-Routine			legen
	CLA			CF 29	
	ARCL 00			STO 00	
	\uparrow 'DIN='			1 E 2	Umrechnung
	ARCL X			/	
	\uparrow 'ASA'			LN	$\ln \frac{\text{ASA}}{100}$
	GTO 06			2	$3 \frac{\ln \frac{\text{ASA}}{100}}{\ln 2} + 21$
019	LBL 02	2. Reihe		LN	
	1 E 4	Anfangswert		/	
	XEQ 03			3	
	700			*	
023	LBL 05	Routine, die die		21	
	x \leftrightarrow y	Unregelmäßigkeit der		+	
	x $>$ y?	Reihen ausgleicht		RND	
	GTO 04			CLA	
	1,024			ARCL 00	
	*			\uparrow 'ASA='	
	GTO 04			ARCL X	
031	LBL 01	3. Reihe		\uparrow DIN'	
	8 E 3	Anfangswert	068	LBL 06	Standardmodus
	XEQ 03			AVIEW	herstellen
	70			SF 29	
	GTO 05			FIX 2	
			072	END	

Liebe Mitglieder!

Ich habe das vorliegende Programmpaket, von dem ich überzeugt bin, daß es noch optimiert werden kann, in den letzten beiden Jahren im Urlaub benutzt. Es hat mir die 'Arbeit', d.h. die Verwaltung der Urlaubskasse wesentlich erleichtert. Es werden also mit meinem Programm all diejenigen angesprochen, die auch während des Urlaubs auf ihre Kasse achten wollen oder müssen. In der vorliegenden Form benötigt das Programmpaket zwei Memory Moduls (190 Register). Es läßt sich aber auch in einer abgemagerten Version mit nur einem Memory verwirklichen.

Da ich in beiden Jahren des Urlaubs mit dem Auto durch mehrere Länder unterwegs war, ist ein wichtiger Punkt für mich gewesen, die Umrechnung mit den Währungen meinem HP zu übertragen. Es entstand ein zweigeteiltes Programmpaket:

1. URLAUB mit den Unterprogrammen: CHANGE, UMR, RMU, WECHS, EINH.
 2. TANKEN mit den Unterprogrammen: KM, DIST, KMKO.
- Als Ergänzung dazu gibt es das Programm SUM, und mit den HP-KEY NOTES Vol. 5 No. 1 flatterte das Programm TIMER auf meinen Tisch, so daß ich dieses auch noch mit aufnahm.

Im folgenden Abschnitt gebe ich eine Kurzbeschreibung der einzelnen Programme. Die Numerierung findet sich im Programm-Listing im Anhang wieder.

- 1.1 URLAUB: Die Ausgaben werden mit Hilfe des Programmes in 5 Kategorien eingeteilt: Unterkunft, Ernährung, Andere Kosten, Fahrtkosten, Kleinkosten. Die Abspeicherung erfolgt in die Speicher 1 bis 5. Speicher 0 wird als Zwischenspeicher benutzt. Für die Benutzung mit anderen Währungen (Ausland) ist das Programm UMR als Unterprogramm erforderlich.
- 1.2 CHANGE): Das Programm CHANGE berechnet den Wechselkurs und speichert die Einheit in Speicher 7 und den Kurs in Speicher 8. Als Unterprogramm wird WECHS aufgerufen. Dies führt aber nur zu einer vernünftigen Programmausführung, falls vorher in den Speichern 18 bis 21 maximal 4 Währungseinheiten gespeichert wurden. Bei Ausführung von CHANGE wird dann der aktuelle Wechselkurs für die entsprechende Währungseinheit in den Speichern 22 bis 25 festgehalten.
- 1.3 WECHS): Das Programm WECHS berechnet den Wechselkurs und speichert die Einheit in Speicher 7 und den Kurs in Speicher 8. Als Unterprogramm wird CHANGE aufgerufen. Dies führt aber nur zu einer vernünftigen Programmausführung, falls vorher in den Speichern 18 bis 21 maximal 4 Währungseinheiten gespeichert wurden. Bei Ausführung von WECHS wird dann der aktuelle Wechselkurs für die entsprechende Währungseinheit in den Speichern 22 bis 25 festgehalten.
- 1.4 EINH: Mit EINH wird der Austausch der Währungseinheiten bzw. Kurse zwischen den von UMR benutzten Rechen Speichern 7, 8 und den Speichern 18-21, 22-25 vorgenommen, wenn man in ein neues Währungsgebiet einreist.
- 1.5 UMR: UMR rechnet ausländische Währungen in DM um. Hierzu muß die aktuelle Einheit der Währung in Speicher 7 stehen, der Kurs in Speicher 8. Befindet man sich längere Zeit in einem Währungsgebiet, kann man FLAG 00 setzen. Dies verkürzt die Ausführung. Ist FLAG 00 gelöscht, wird jedesmal nach einer Änderung der Einheit gefragt.

- 1.6 RMU: RMU leistet das Inverse zu UMR. Es rechnet DM in die in Speicher 7 vorhandene Einheit um.
- 1.7 SUM: Dieses Zusatzprogramm berechnet die Summe aller Ausgaben, die mit URLAUB gespeichert werden. Am Anfang kann man einen Höchstbetrag der Ausgaben festsetzen, der dann als Vergleichswert dient und für jede Abfrage mit SUM den Restbetrag ausrechnet. Bei Überschreiten des Höchstbetrages erfolgt auch noch eine akustische Warnung mit BEEP.
- 2.1 TANKEN: Mit dem Programm TANKEN läßt sich die Kontrolle der Treibstoffkosten sowie des Verbrauchs durchführen. Es erfordert die Eingabe des Literpreises, sowie der Treibstoffmenge. In ausländischem Währungsgebiet ruft es ebenfalls UMR als Unterprogramm auf, um die Berechnungen in DM durchzuführen. Die Treibstoffkosten werden automatisch zu den Fahrkosten addiert (Speicher Ø4) und getrennt in Speicher 17 festgehalten. Die Programme KM und DIST werden aufgerufen, um den durchschnittlichen Verbrauch mit der letzten Tankfüllung und während der gesamten Fahrt zu berechnen.
- 2.2 NACH: Dies ist ein Ergänzungsprogramm zu TANKEN. Sollte aus bestimmten Gründen, z.B. sehr teure Tankstelle, einmal der Tank nicht voll gefüllt werden, würde die Durchschnittsberechnung zu fehlerhaften Ergebnissen führen. Deshalb wird in diesem Fall mit NACH anstelle von TANKEN gearbeitet.
- 2.3 KM: KM hält insbesondere den Kilometerstand zu Beginn der Reise fest und besorgt die Speicherverwaltung der einzelnen Kilometerstände bei der Durchschnittsberechnung von TANKEN.
- 2.4 DIST: Hiermit lassen sich jederzeit die bisher zurückgelegten Wegstrecken ermitteln.
- 2.5 KMKO: Dieses Programm berechnet die Kosten pro zurückgelegtem Kilometer während des Urlaubs. In Programmzeile Ø findet sich ein additiver Faktor. Dieser muß individuell bestimmt werden. Er berücksichtigt die jährlichen Fixkosten (STEUER, VERSICHERUNG, WARTUNG, RÜCKLAGE NEUWAGENKAUF, WERTMINDERUNG). Unter Umständen kann man hier die ADAC-TABELLE heranziehen (ADAC-MOTORWELT 12/80). Man muß dann aber die Treibstoffkosten herausrechnen. Für einen OPEL-ASCONA 1,6 l ergibt sich dann monatl. Aufwand pro Kilometer ohne Treibstoffkosten bei 15000 km im Jahr: 0,2592 DM/km.
- 3.1 TIMER: Mit diesem Programm kann die Genauigkeit des Tachos überprüft werden. Das Programm wird im USER-Modus mit LBL"A" (Taste 11) gestartet. Nach einem Kilometer (gleichmäßig schneller) Fahrt z.B. von Kilometerschild zu Kilometerschild, ist Taste "B" (12) zu drücken. Hierauf wird die Durchschnittsgeschwindigkeit angezeigt. Dieser Vorgang läßt sich durch nochmaliges drücken auf Taste "B" wiederholen. Die 1,5 in Zeile 18 dient zur Geschwindigkeitskorrektur und muß eventuell dem Rechner angepaßt werden.

Speicherbelegung beim Programmpaket:

Speicher:	Inhalt:
ØØ	Zwischenspeicher
Ø1	Unterkunft
Ø2	Ernährung
Ø3	Andere Kosten
Ø4	Fahrtkosten
Ø5	Kleinkosten
Ø6	Summe der Ausgaben nach Aufruf von Summe.
Ø7	Währungseinheit
Ø8	Kurs
Ø9	Inverser Kurs
1Ø	Summe der getankten Liter Treibstoff
11	Zuletzt getankte Treibstoffmenge
12	Anfangs-Kilometerstand
13	Summe der gefahrenen Kilometer
14	Kilometer, die mit der letzten Tankfüllung erreicht wurden
15	Zwischenspeicher
16	Kilometerstand beim letzten Tanken
17	Summe der Treibstoffkosten
18	1. Währungseinheit
19	2. Währungseinheit
2Ø	3. Währungseinheit
21	4. Währungseinheit
22	Kurs 1. Währungseinheit
23	Kurs 2. Währungseinheit
24	Kurs 3. Währungseinheit
25	Kurs 4. Währungseinheit
26	Höchstbetrag der Urlaubsausgaben

Für den Benutzer unbedingt erforderliche Kenntnis der FLAGS:

Flag ØØ: Muß gesetzt werden, um bei längerem Aufenthalt in einem Währungsgebiet die Programme zu beschleunigen.

Flag Ø1: Wird automatisch gesetzt, wenn das Programm NACH benutzt wird.

Da ich keinen Drucker besitze, ist das Listen mit der Schreibmaschine erfolgt. Alphanumerische Eingaben sind mit ' ' eingeschlossen.

Im Anhang folgen jetzt die Listings der Programme und dann ein Beispiel. Für das Beispiel gebe ich hier die Werte an: In Deutschland getauscht: Für DM 363,23 erhielt ich ÖS 2500, und für DM 103,75 erhielt ich 50000 Lire. Der Start erfolgte beim Km-Stand 82691. Getankt habe ich dann wie folgt:

Liter-Preis	Liter	Km-Stand
1,32	31,83	83Ø65
1,489	1Ø,Ø7	Nachgetankt
1Ø,84 ÖS	28,04	836Ø8
98Ø Lire	1Ø,6	8372Ø

Ausgaben, die das Programm URLAUB betreffen:

Ernährung DM 15,-, Unterkunft ÖS 175,-, Ernährung ÖS 1ØØ,-, Fahrtkosten (Mautgebühr) ÖS 17Ø,-, Kleinkosten ÖS 19,5Ø.

Viel Spaß beim Benutzen des Programms und während des nächsten Urlaubs!

Olaf Gursch

01*LBL "URLAUB"
02 CLST
03*LBL 14
04 "PROGRAMM?"
05 PROMPT
06 STO 00
07 X=0?
08 GTO 14
09 5
10 X<Y?
11 GTO 14
12 FIX 2
13 1,005
14 STO 15
15*LBL 13
16 RCL 00
17 RCL 15
18 INT
19 X=Y?
20 GTO 12
21 ISG 15
22 GTO 13
23*LBL 12
24 GTO IND X
25*LBL 01
26 "UNTERKUNFT"
27 GTO 11
28*LBL 02
29 "ERNAEHRUNG"
30 GTO 11
31*LBL 03
32 "ANDERE KOSTEN"
33 GTO 11
34*LBL 04
35 "FAHRTKOSTEN"
36 GTO 11
37*LBL 05
38 "KLEINKOSTEN"
39*LBL 11
40 AVIEW
41 PSE
42 "DM? =1"
43 E
44 PROMPT
45 X=Y?
46 GTO 00
47 SF 07
48 XEQ "UMR"
49*LBL 00
50 "DM-BETRAG?"
51 FC?C 07
52 PROMPT
53 RCL 00
54 X<Y
55 ST+ IND Y
56 END

01*LBL "CHANGE"
02 "EINHEIT?"
03 AON
04 PROMPT
05 ASTO 07
06 AOFF
07 "NIEVIEL DM?"
08 PROMPT
09 ENTER↑
10 "NIEVIEL "
11 ARCL 07
12 PROMPT
13 /
14 STO 08
15 "1 "
16 ARCL 07
17 "t=" "
18 ARCL 08
19 "fDM"
20 AVIEW
21 XEQ "WECHS"
22 END

01*LBL "WECHS"
02 18
03 STO 15
04 RCL 07
05*LBL 01
06 RCL IND 15
07 X=Y?
08 GTO 02
09 E
10 ST+ 15
11 R↑
12 R↑
13 GTO 01
14*LBL 02
15 4
16 ST+ 15
17 FS? 10
18 RCL IND 15
19 FC? 10
20 RCL 08
21 FS? 10
22 STO 08
23 FC? 10
24 STO IND 15
25 END

01*LBL "EINH"
02 SF 10
03 "WELCHE WAERHUNG"
04 "t?"
05 AON
06 PROMPT
07 AOFF
08 ASTO 07
09 XEQ "WECHS"
10 CF 10
11 END

01*LBL "UMR"
02 FS? 00
03 GTO 01
04 "ZUR ZEIT: "
05 ARCL 07
06 AVIEW
07 PSE
08 "AENDERUNG? #0"
09 PROMPT
10 X=0?
11 XEQ "EINH"
12*LBL 01
13 CLX
14 "NIEVIEL "
15 ARCL 07
16 PROMPT
17 STO 15
18*LBL "UM"
19 RCL 08
20 *
21 FS? 07
22 RTN
23 CLA
24 ARCL 07
25 "t:" "
26 ARCL 15
27 "t=" "
28 ARCL X
29 PROMPT
30 END

01*LBL "RMU"
02 RCL 08
03 1/X
04 STO 09
05 "NIEVIEL DM?"
06 PROMPT
07 STO 00
08 RCL 09
09 *
10 ARCL 00
11 "fDM=" "
12 ARCL X
13 ARCL 07
14 PROMPT
15 END

01+LBL "SUM"
 02 FS? 06
 03 GTO 01
 04 "HOECHSTBETRAG?"
 05 PROMPT
 06 STO 26
 07 SF 06
 08+LBL 01
 09 1.005
 10 STO 15
 11 ,
 12 STO 06
 13+LBL 02
 14 RCL IND 15
 15 ST+ 06
 16 ISG 15
 17 GTO 02
 18 RCL 06
 19 "BISHER AUSGEGEB"
 20 "FEN: "
 21 ARCL X
 22 "FDM"
 23 AVIEW
 24 PSE
 25 RCL 26
 26 X<>
 27 -
 28 "RESIDBETAG: "
 29 ARCL X
 30 AVIEW
 31 X<=0?
 32 BEEP
 33 END

01+LBL "TIMER"
 02+LBL A
 03 "FAHRE SICHER"
 04 AVIEW
 05+LBL 01
 06 ,
 07 STO 00
 08+LBL 00
 09 PSE
 10 E
 11 ST+ 00
 12 GTO 00
 13+LBL B
 14 RCL 00
 15 E4
 16 /
 17 "HR"
 18 1,5
 19 *
 20 1/X
 21 "AV="
 22 ARCL X
 23 "FKM/H"
 24 AVIEW
 25 GTO 01
 26 END

01+LBL "TANKEN"
 02 "DM? =1"
 03 PROMPT
 04 E
 05 X=Y?
 06 SF 09
 07 "1 L KOSTET?"
 08 PROMPT
 09 FS?C 09
 10 GTO 03
 11 SF 07
 12 XEQ "UM"
 13+LBL 03
 14 "WIEVIEL LITER?"
 15 PROMPT
 16 ST+ 11
 17 ST+ 10
 18 *
 19 VIEW X
 20 ST+ 04
 21 ST+ 17
 22 FS? 01
 23 RTN
 24 XEQ "KM"
 25 RCL 14
 26 E2
 27 /
 28 RCL 11
 29 X<>Y
 30 /
 31 "AUF 100KM:"
 32 ARCL X
 33 "FL"
 34 AVIEW
 35 PSE
 36 SF 07
 37 XEQ "DIST"
 38 E2
 39 /
 40 RCL 10
 41 X<>Y
 42 /
 43 "DURCHSCHNITTLIC"
 44 "HER"
 45 AVIEW
 46 "VERBRAUCH:"
 47 ARCL X
 48 "FL"
 49 AVIEW
 50 PSE
 51 ,
 52 STO 11
 53 END
 01+LBL "NACH"
 02 SF 01
 03 "NICHT VOLL"
 04 AVIEW
 05 PSE
 06 "GETANKT? =0"
 07 X#0?
 08 PROMPT
 09 XEQ "TANKEN"
 10 CF 01
 11 END

01+LBL "KM"
 02 FS? 08
 03 GTO 01
 04 "START BEI KM?"
 05 PROMPT
 06 STO 12
 07 STO 16
 08 SF 08
 09+LBL 01
 10 "KM-STAND?"
 11 PROMPT
 12 STO 15
 13 RCL 16
 14 -
 15 STO 14
 16 ST+ 13
 17 RCL 15
 18 STO 16
 19 END

01+LBL "DIST"
 02 FS?C 07
 03 GTO 01
 04 "KM-STAND?"
 05 PROMPT
 06 GTO 02
 07+LBL 01
 08 RCL 16
 09+LBL 02
 10 RCL 12
 11 -
 12 "GEFAHRENE KM:"
 13 ARCL X
 14 AVIEW
 15 PSE
 16 END

01+LBL "KMKO"
 02 XEQ "DIST"
 03 STO 00
 04 RCL 17
 05 X<>Y
 06 /
 07 ,2592
 08 +
 09 FIX 4
 10 "1 KM KOSTETE "
 11 ARCL X
 12 "FDM"
 13 AVIEW
 14 PSE
 15 FIX 2
 16 RCL 00
 17 *
 18 "AUTOKOSTEN:"
 19 ARCL X
 20 "FDM"
 21 PROMPT
 22 END

3. TIMER: s. auch HP KEY-NOTES Vol.1.5 No.1

01 LBL'TIMER'	09 1	18 *
02 LBL A	10 ST+ 00	19 1/X
03 'SICHER FAHREN'	11 GTO 00	20 'AV='
04 AVIEW	12 LBL B	21 ARCL X
05 LBL 01	13 RCL 00	22 - KM/H
06 STO 00	14 10000	23 AVIEW
07 LBL 00	15 /	24 GTO 01
03 PSE	16 HR	25 RTN
	17 1,5	26 END

LBL'URLAUB	
END	170 BYTES
LBL'CHANGE	
END	77 BYTES
LBL'WECHS	
END	47 BYTES
LBL'EINH	
END	46 BYTES
LBL'UMR	
...UM	
END	91 BYTES
LBL'RMU	
END	41 BYTES
LBL'SUM	
END	104 BYTES
LBL'TANKEN	
END	164 BYTES
LBL'NACH	
END	50 BYTES
LBL'KM	
END	55 BYTES
LBL'DIST	
END	52 BYTES
LBL'KNKO	
END	75 BYTES
LBL'TIMER	
END	67 BYTES

PROGRAMMBEISPIEL:

EINGABE	FUNKTION	ANZEIGE
	SIZE 027	
EINGABE DES PROGRAMMPAKETES	CLRG	
	ASN'TIMER'14	
	ASN'URLAUB'13	
	ASN'SUM'14	
	ASN'CHANGE'15	
	ASN'UMR'21	
	ASN'RMU'22	
	ASN'TANKEN'23	
	ASN'NACH'24	
	ASN'KM' 25	
	ASN'DIST'32	
	ASN'KNKO'33	
	ASN'EINH'34	
OES	ASTO 18	
LIRE	ASTO 19	
	XEQ'KM'(25)	START BEI KM?
82691	R/S	
	XEQ'CHANGE'(15)	EINHEIT?
OES	R/S	WIEVIEL DM?
362,23	R/S	WIEVIEL OES
2500	R/S	1 OES=0,14.DM
	XEQ'CHANGE'(15)	EINHEIT?
LIRE	R/S	WIEVIEL DM?
103,75	R/S	WIEVIEL LIRE
50000	R/S	1 LIRE=2,08E-3DM
	XEQ'TANKEN'(23)	DM?=1
1	R/S	1 L KOSTET?
1,32	R/S	WIEVIEL LITER?
31,83	R/S	42,02
		KM-STAND?
83065	R/S	AUF 100 KM:8,51L
		GEFAHRENE KM 374,00
		DURCHSCHNITTLIC
		HER VERBRAUCH:8,51
	XEQ'URLAUB'(13)	PROGRAMM?
2	R/S	ERNAEHRUNG
		DM? =1
1	R/S	DM-BETRAG?
15	R/S	15,00
	XEQ'NACH'(24)	NICHT VOLL
		GETANKT? =0
0	R/S	DM? =1
1	R/S	1 L KOSTET?
1,489	R/S	WIEVIEL LITER?
10,07	R/S	14,99
	XEQ'URLAUB'(13)	PROGRAMM?
1	R/S	UNTERKUNFT
		DM? =1
0	R/S	ZUR ZEIT: LIRE
		AENDERUNG? # 0
2	R/S	WELCHE WAERUNG
OES	R/S	WIEVIEL OES
175	R/S	25,36
	R/S	PROGRAMM?
2	R/S	ERNAEHRUNG
		DM? =1
0	R/S	ZUR ZEIT: OES
		AENDERUNG? #0
0	R/S	WIEVIEL OES

INGABE	FUNKTION	ANZEIGE
100	R/S	14,49
	R/S	PROGRAMM?
4	R/S	FAHRTKOSTEN
		DM? = 1
5	R/S	ZUR ZEIT: OES
		AENDERUNG? ≠ 0
0	R/S	WIEVIEL OES
170	R/S	24,63
	SF 00	
	XEQ'TANKEN'(23)	DM? = 1
2	R/S	1 L KOSTET?
10,84	R/S	WIEVIEL LITER?
28,04	R/S	44,04
		KM-STAND?
83608	R/S	AUF 100 KM: 7,02 L
		GEFAHRENE KM 917,00
		DURCHSCHNITTLIC
		HER VERBRAUCH: 7,63L
	XEQ'URLAUB'(13)	PROGRAMM?
5	R/S	KLEINKOSTEN
		DM? = 1
4	R/S	WIEVIEL OES
19,5	R/S	2,83
	CF 00	
	XEQ'EINH'(34)	WELCHE WAERHUNG?
	R/S	2,08 -03
	XEQ'TANKEN'(23)	DM? = 1
	R/S	1 L KOSTET?
0	R/S	WIEVIEL LITER?
980	R/S	21,56
10,6	R/S	
		KM-STAND?
83720	R/S	AUF 100 KM: 9,46 L
		GEFAHRENE KM 1029,00
		DURCHSCHNITTLIC
		HER VERBRAUCH: 7,83 L
	XEQ'UMR'(21)	ZUR ZEIT: LIRE
		AENDERUNG? ≠ 0
	R/S	WIEVIEL LIRE
0	R/S	LIRE: 4500,00 = 9,34
4500	R/S	WIEVIEL DM?
112	R/S	112,00 DM = 53975,00 LIRE
	XEQ'SUM'(14)	HOECHSTBETRAG?
500	R/S	BISHER AUSGEGEB
		EN: 204,90 DM
		RESTBETRAG: 295,10
	XEQ'KMKO'(33)	KM-STAND?
83720	R/S	GEFAHRENE KM: 1029,00
		1 KM KOSTETE: 0,3783 DM
		AUTOKOSTEN: 389,32

BLEIBT NACHZUTRAGEN, WAS IHR EUCH LÄNGST GEDACHT HABT, IN
 KLAMMERN SIND DIE TASTEN-NUMMERN ANGEGBEN LAUT ZUORDNUNG.

Programm: Entfernungsberechnung mit Koordinaten

a) Beschreibung

Dieses Entfernungsberechnungsprogramm ist ein speziell auf dem HP-41C entwickeltes Anwenderprogramm, das nach der Formel

$$E_{\text{(miles)}} = \cos^{-1} [\sin(NL_1) \sin(NL_2) + \cos(NL_1) \cos(NL_2) \cos(OB_2 - OB_1)] \times 60$$
abläuft.

Die Entfernung ("E") ist eine Angabe in Miles. Für die Umrechnung in km muß mit 1.8544 multipliziert werden.

Ich habe dieses Programm erstmals am 30. Juni 1980 geschrieben. Die jetzt vorliegende Fassung geht auf den 30. Juli 1980 zurück.

Noch einige Bemerkungen vorweg: Bei einem Input von negativen Werten wird richtig über den O-Meridian hinweg gerechnet. Vorsicht: Bei Input von zweimal dem gleichen Punkt wird 90 m Entfernung berechnet. Dies ist jedoch falsch.

Das Programm verlangt nun bei Beginn die Eingabe von 4 Positionen (genauer 2 mal x und y). Während der Verarbeitung gemäß der oben erwähnten Formel zeigt der Rechner im Anzeigefeld die Bemerkung "Berechnung". Programmschritt 36 und 37 können bei einer Ausgabe in Miles weggelassen werden, doch empfehle ich dann, den Programmschritt 40 ("KM") durch ("MILES") zu ersetzen.

Wenn nun die Anzeige "999.99 KM" im Display steht, kann das Programm mit (R/S) weitergeführt werden. Falls man jetzt aber mit diesem Wert rechnen will, genügt es, die Clear-Taste (←) zu bedienen. Die nächste Anzeige ist nun "NOCH EINMAL?". Wenn dort "Y" ("Yes") eingegeben wird, verzweigt das Programm wieder zum Start; wenn nicht, dann kommt die Frage "INPUT-VIEW?" (Anzeigen der Eingaben). Auch hier kann wieder mit "Y" geantwortet werden, dann zeigt die Anzeige für eine sec. "KONTROLLE:" und bringt dann für jeweils eine sec. die Werte (1-4) in dieser Form in den Display: "1. 99.99" / "2. 99.99" / "3. 99.99" / "4. 99.99". Anschließend kommt die Frage "ZUM ANFANG?" in die Anzeige. Wenn nicht "Y" eingegeben wird, zeigt der Rechner "* ENDE *".

Das im LBL 07 verwendete Zeichen "Y" kann natürlich auch mit der deutschen Eingabe "J" ausgetauscht werden. Ich persönlich ziehe jedoch "Y" vor, weil es unter dem "*" -Zeichen sehr günstig liegt und einfach zu merken ist.

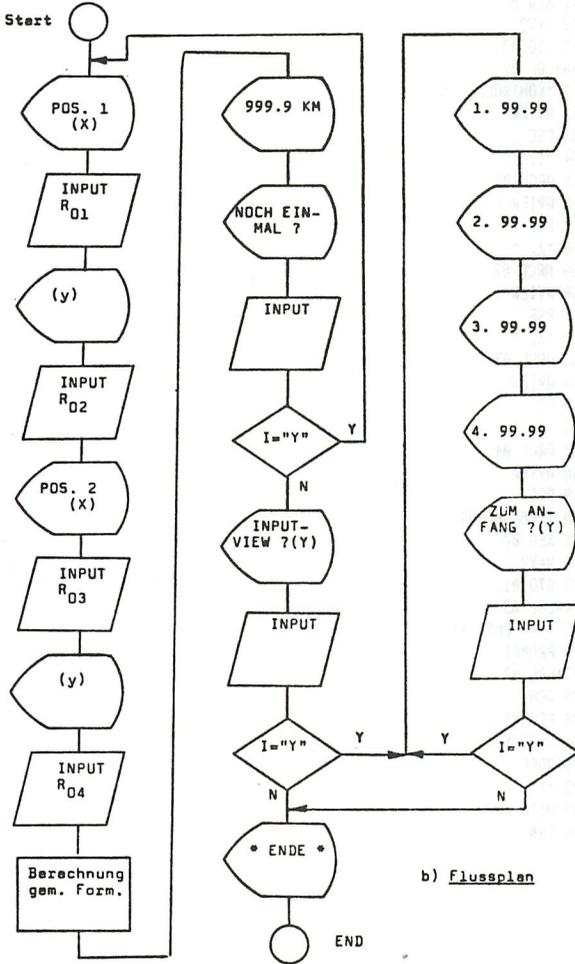
Auch der Name für das LBL 07 ist sehr willkürlich gewählt. Ich verwende in allen meinen Programmen XEQ 07 für die Verarbeitung von "Yes/NO" Eingaben.

240 Bytes

Das ganze Programm kommt auf ~~33 belegte Programmzeilen~~, was 84 95 Programmschritten entspricht. Es werden die Speicher 00-05 und das Alpha-Reg. benutzt.

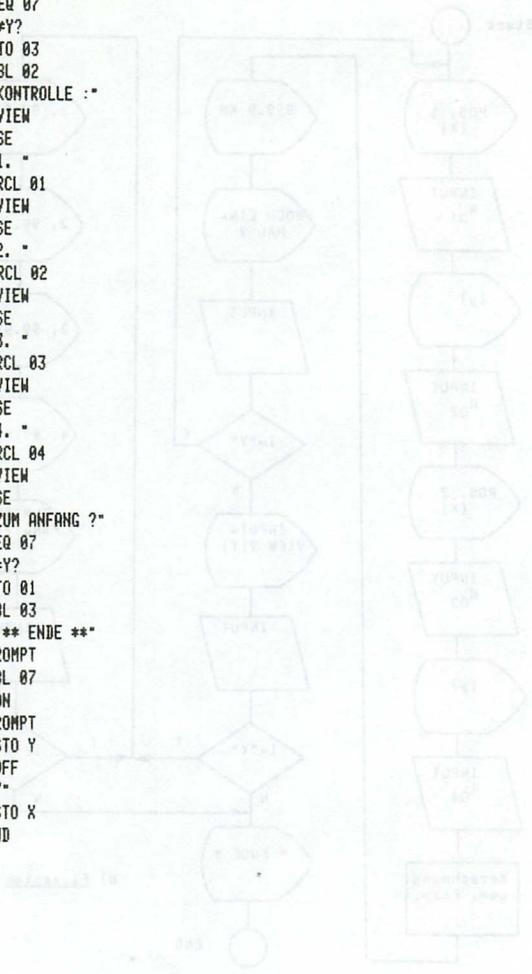
Über Ideen, wie sich mein Programm verbessern oder verkürzen ließe, wäre ich sehr dankbar.

Programm: Entfernungsberechnung mit Koordinaten



b) Flussplan

6. *LBL "ENF"	43 XEQ 07
02 *LBL 01	44 X=Y?
03 " POS. 1 (X) ?"	45 GTO 01
04 PROMPT	46 "INPUT-VIEW ?"
05 STO 01	47 XEQ 07
06 " (Y) ?"	48 X=Y?
07 PROMPT	49 GTO 03
08 STO 02	50 *LBL 02
09 " POS. 2 (X) ?"	51 "KONTROLLE :"
10 PROMPT	52 AVIEW
11 STO 03	53 PSE
12 " (Y) ?"	54 "1. "
13 PROMPT	55 ARCL 01
14 STO 04	56 AVIEW
15 " BERECHNUNG"	57 PSE
16 AVIEW	58 "2. "
17 RCL 02	59 ARCL 02
18 -	60 AVIEW
19 COS	61 PSE
20 RCL 03	62 "3. "
21 COS	63 ARCL 03
22 *	64 AVIEW
23 RCL 01	65 PSE
24 COS	66 "4. "
25 *	67 ARCL 04
26 RCL 01	68 AVIEW
27 SIN	69 PSE
28 RCL 03	70 "ZUM ANFANG ?"
29 SIN	71 XEQ 07
30 *	72 X=Y?
31 +	73 GTO 01
32 ACOS	74 *LBL 03
33 60	75 " ** ENDE **"
34 *	76 PROMPT
35 1.8544	77 *LBL 07
36 *	78 AON
37 STO 00	79 PROMPT
38 " "	80 ASTO Y
39 ARCL 00	81 AOFF
40 "+ KM"	82 "Y"
41 PROMPT	83 ASTO X
42 "NOCH EINMAL?"	84 END



Programm: Entfernungsberechnung mit Koordinaten

d) Beispiel



X = Nördliche Länge
 -X = Südliche Länge

Y = Östliche Breite
 -Y = Westliche Breite

Paris : 48.833 / 2.333

Basel : 47.566 / 7.583

Eingabe:

XEQ "ENTF"
 48.833 (R/S)
 2.333 (R/S)
 47.566 (R/S)
 7.583 (R/S)

(R/S)
 N (R/S)
 Y (R/S)

N (R/S)

Anzeige:

POS. 1 (X)
 (Y)
 POS. 2 (X)
 (Y)
 BERECHNUNG
 413.98 KM
 NOCH EINMAL?
 INPUT-VIEW?
 KONTROLLE :
 1. 48.83
 2. 2.33
 3. 47.57
 4. 7.58
 ZUM ANFANG ?
 ** ENDE **

(ca. 2 sec.)

} jeweils ca. 1 sec.

Programm: FN Flugnavigation

FN dient der vorgeschriebenen, navigatorischen Flugvorbereitung für Überlandflüge, sowie der Berechnung der Winkelkomponenten auf dem Start- und Landeplatz.

1. Spuren 1 ... 4 vom FN einlesen.
2. Programm mit XEQ FN starten.
3. Eingabe der Kennziffer n der ersten Flugteilstrecke (32) (Kennzifferbezeichnung der folgenden Flugteilstrecken erhöht sich automatisch um +1)
Kartenkurs ($^{\circ}$) und Entfernung (nm) der Flugteilstrecken im Dialog mit R/S eingeben, z.B. 150, 100 (Kartenkurs 150° , Entfernung 100 nm).
4. Eingabe der Flugdaten durch XEQ E
Magnetische Variation ($^{\circ}$), Geschwindigkeit - (nm) und Richtung ($^{\circ}$) des Höhenwindes sowie der Geschwindigkeit über Grund (V_E in kn) des Flugzeugtyps im Dialog mit R/S eingeben.
(Eingabe: Mag. Variation 3° West = -3
 3° Ost = +3)
5. Eingabe Kennziffer n der ersten gewünschten Flugteilstrecke mit R/S. Automatische Ausgabe der berechneten Navigationsdaten für die Flugteilstrecken beginnend mit der eingegebenen Kennziffer n.

	Kennziffer der Flugteilstrecke	
Automatische Ausgabe:	Kartenkurs	.
	Entfernung	nm
	Windabdriftwinkel	.
	Rechtweisender Windkurs	.
	Mißweisender Windkurs	.
	Geschwindigkeit über Grund	kn
	Flugzeit	Std,Min
	Gesamtflugzeit	Std,Min
	(nur bei letzter Flugteilstrecke)	

6. Berechnung der Windkomponenten durch XEQ F
Eingabe der Start-/Landebahnrichtung ($^{\circ}$), der Richtung ($^{\circ}$) und Geschwindigkeit (kn) des Bodenwindes im Dialog mit R/S.
Automatische Ausgabe der Gegenwind- und Rückenwind-Komponente in Knoten (Gegenwind positiv, Rückenwind negativ) und der linken oder rechten Seitenwindkomponente.
(Seitenwindkomponente von rechts positiv, Seitenwindkomponente von links negativ)
7. Datenkorrektur der vorigen Flugteilstrecken
XEQ C (ERROR-ANZEIGE, wenn Kennziffer < 1)
8. Datenkorrektur von Flugteilstrecken bei beendeter DATA-Eingabe
XEQ B. Eingabe der Kennziffer mit R/S. Anzeige der betreffenden Flugteilstrecken - DATA Neueingabe mit R/S.
9. Abspeicherung der Flugteilstrecken - DATA auf KARTE XEQ D
Erfolgt automatisch, wenn Kennziffer größer 32, d.h. in der Regel, wenn mehr als 32 Flugteilstrecken vorliegen. Bei Ausführung von XEQ E (4) wird diese Karte zuerst im Programm für den ersten Rechendurchgang verlangt.

Vom Programm werden die Register R 00 ... R 58 benutzt. Die eingelesenen DATA der Flugteilstrecken werden in den Speichern 1-12 gespeichert. Speicher 33-44 sind Rechenspeicher. In den Speichern 44-57 werden "strings" gespeichert.

Benötigt werden 2 Speichermodule.
Das Programm arbeitet mit und ohne Drucker.

% = Grad
nm = nautische Meilen
kn = Knoten

Liebe Clubfreunde!

Dieses recht umfangreiche Flugnavigationsprogramm wird hoffentlich bei allen Hobby-Flugbegeisterten auf Anklang stoßen, da es in der Flugvorbereitung die sonst verwendete Rechenscheibe weitgehend ablöst.

Mein besonderer Dank gilt Herrn Hoenow (11), der mir freundlicherweise bei der Optimierung des Programms behilflich war.

Happy Programming

Harald Wienbeck (141)
Rethwiese 13
2080 Pinneberg
(Adressenänderung!!!)

01*LBL "FH"
CF 01 SF 27 CLRG CLX
XEQ 12 BEEP

08*LBL 01
CLA ARCL 45 ARCL 44
"-?" PROMPT STO 00 32
XOY X?Y? GTO 01

19*LBL 02
RCL 00 33 X=? GTO 0
RBN FIX 0 CLA ARCL 46
"- ARCL 44 ARCL X
AVIEW STOP FIX 3
VIEW X STO INB 00 1
ST+ 00 GTO 02

39*LBL 05
TONE 0 TONE 5 "WRONG"
ARCL 47 AVIEW STOP

46*LBL 0
"WRITE LEGS" AVIEW
1.032 HD TAX SF 01 CLX
GTO "FH"

54*LBL 0
CLA ARCL 47 ARCL 48
AVIEW PSE CLA ARCL 45
ARCL 44 "-?" PROMPT
STO 00 RCL INB 00
FIX 4 STOP STO INB 00
STOP

71*LBL C
1 ST- 00 RCL 00 X=0?
GTO 05 GTO 02

78*LBL 03
RCL 00 32 XOY X?Y?
GTO 04 RCL INB 00 0
X=? GTO 04 RCL 00
FIX 0 CLA ARCL 44
ARCL 46 "- ARCL X
AVIEW PSE FIX 3
RCL INB 00 VIEW X
TONE 5 PSE PSE 1
ST+ 00 GTO 03

106*LBL 04
"END" ARCL 44 ARCL 46
AVIEW TONE 5 TONE 6
STOP

114*LBL E
ADV ADV 0 STO 40
SF? 01 XEQ 10 SF 12
CLA ARCL 48 "-"
ARCL 46 "- XEQ 11
CLA ARCL 49 "VARIA"
ARCL 54 "- ?" AVIEW
STOP FIX 0 VIEW X
STO 33 CLA ARCL 56
ARCL 57 "- ?" AVIEW
STOP VIEW X STO 34
CLA ARCL 56 ARCL 55
ARCL 54 "- ?" AVIEW
STOP VIEW X STO 35
CLA ARCL 51 "AIR"
ARCL 57 "- ?" AVIEW
STOP VIEW X STO 36
RCL 33 X?Y? GTO 06
CLA ARCL 45 ARCL 44
"-?" PROMPT STO 00

173*LBL 07
RCL INB 00 X=0? GTO 00
ADV ADV SF 12 CLA
ARCL 44 ARCL 00 XEQ 11
RCL INB 00 INT STO 41
CLA ARCL 51 "TRACK"
ARCL 53 ARCL X AVIEW
RCL INB 00 FRC 1 E3 0
STO 42 "DISTANCE"

ARCL 52 ARCL 53 ARCL X
AVIEW RCL 41 CHS
RCL 35 + STO 37 SIN
RCL 34 * RCL 36 /
ASIN FIX 1
"DRIFT ANGLE" ARCL 53
ARCL X AVIEW RCL 41 +
STO 38 XEQ 09 FIX 0
CLA ARCL 51 "HEADING"
ARCL 52 ARCL X AVIEW
RCL 33 - XEQ 09 CLA
ARCL 49 "HEADING"
ARCL 52 ARCL X AVIEW
RCL 38 RCL 36 P-R
XOY RCL 35 RCL 34
P-R RBN - RBN - R?
XOY R-P "GROUND"
ARCL 57 ARCL 53 ARCL X
AVIEW STO 39 RCL 42
XOY / ST+ 40 HNS
FIX 2 CLA ARCL 48
ARCL 50 ARCL 52 ARCL X
AVIEW 1 ST+ 00 FIX 0
GTO 07

275*LBL 06
TONE 0 TONE 5 TONE 9
CLA ARCL 46 "F ERROR"
AVIEW GTO E

284*LBL 08
ADV "END OF LEGS"
AVIEW ADV RCL 40 HNS
FIX 2 "TOTAL FL."
ARCL 50 ARCL X AVIEW
ADV ADV STOP

299*LBL 09
360 MOD RTN

303*LBL 10
CF 01 1.032 RTAX "RTN"

308*LBL F
SF 12 "TAKE OFF"
AVIEW PSE "LANDING"
AVIEW PSE ADV XEQ 13
FIX 0 CF 12 CLA
ARCL 55 ARCL 54 "- ?"

AVIEW STOP VIEW X
STO 43 CLA ARCL 56
ARCL 55 ARCL 54 "- ?"
AVIEW STOP VIEW X
STO 41 CLA ARCL 56
ARCL 57 "- ?" AVIEW
STOP VIEW X STO 42
ADV RCL 43 RCL 41
RCL 33 - - CHS
RCL 42 P-R
"HEAD"/"TAIL" ARCL 56
AVIEW PSE FIX 1
VIEW X PSE XOY
"RIGHT"/"LEFT" AVIEW
PSE "CROSS" ARCL 56
AVIEW PSE VIEW X ADV
ADV ADV ADV RTN

375*LBL 11
AVIEW "-----" ASTO X
ARCL X AVIEW CF 12
ADV RTN

384*LBL 12
"LEG" ASTO 44
"NO OF" ASTO 45
"DATA" ASTO 46
"CORREC" ASTO 47
"FLIGHT" ASTO 48
"MAG." ASTO 49
"-TIME" ASTO 50
"TRUE" ASTO 51
ASTO 52 " " ASTO

405*LBL 13
"TION" ASTO 54 "DI"
ASTO 55 "WIND" ASTO
"SPEED" ASTO 57 "E

RECHEN-
BEISPIEL :

DATA LEG 1.	20.100
DATA LEG 2.	110.200
DATA LEG 3.	200.100
DATA LEG 4.	290.200
DATA LEG 5.	
FLIGHT DATA :	

MAG. VARIATION ?	-3.
WINDSPEED ?	15.
WIND DIRECTION ?	60.
TRUE AIRSPEED ?	95.
TAKE OFF / LANDING	
DIRECTION ?	110.
WIND DIRECTION ?	60.
WINDSPEED ?	10.
+HEAD/-TAIL WIND	6.8
+RIGHT/-LEFT CROSSWIND	-7.3

LEG 1.

TRUE TRACK 20.
DISTANCE 100.
DRIFT ANGLE 5.8
TRUE HEADING 26.
MAG. HEADING 29.
GROUNDSPEED 83.
FLIGHT-TIME 1.12

LEG 2.

TRUE TRACK 110.
DISTANCE 200.
DRIFT ANGLE -6.9
TRUE HEADING 103.
MAG. HEADING 106.
GROUNDSPEED 85.
FLIGHT-TIME 2.21

LEG 3.

TRUE TRACK 200.
DISTANCE 100.
DRIFT ANGLE -5.8
TRUE HEADING 194.
MAG. HEADING 197.
GROUNDSPEED 106.
FLIGHT-TIME 0.56

LEG 4.

TRUE TRACK 290.
DISTANCE 200.
DRIFT ANGLE 6.9
TRUE HEADING 297.
MAG. HEADING 300.
GROUNDSPEED 104.
FLIGHT-TIME 1.55

END OF LEGS

TOTAL FL.-TIME 6.26

ASTRONOMIE

Aufgenommene Programme

<u>NAME</u>	<u>LABEL</u>	<u>QUELLE</u>	<u>AUTOR</u>
Mondphasen/ Finsternisse	PH FI	Prisma 1980	Günter Lelarge
Ephemeride	EP	Uwe Hommelsheim	Uwe Hommelsheim

Nicht aufgenommenes Programm

<u>NAME</u>	<u>LABEL</u>	<u>QUELLE</u>	<u>AUTOR</u>
Diverse astronom. Programme	DIV	Prisma 1981 S 406-S 414	Werner Loibl

Mondphasen / Sonnen- und Mondfinsternisse

PH	SIZE 027	Magnetkartenleser	Synthetik
FI	444 Bytes	Drucker	MAN
		ein Memory	

Dieses Programm errechnet die Mondphasen und die Sonnen- und Mondfinsternisse ab einem vorher eingegebenen Datum. Das Programm stoppt automatisch, nachdem das letzte Ereignis eines Jahres ausgedruckt wurde.

Entsprechendes Jahr in dezimaler Form (JJJJ,jj) in das X-Register, dann entweder

XEQ"PH" für Mondphasen oder
XEQ"FI" für Finsternisse

Wurde das Programm erstmals in den Speicher geladen, erscheint nach Programmstart "CARD" im Display. Datenkarte einlesen; Programm rechnet automatisch weiter.

Datenkarte muß mit folgenden Informationen beschrieben sein:

R 01:	14.765	R 13:	9.6
R 06:	0.07	R 14:	13.176398
R 07:	1.23	R 15:	153.17
R 08:	0.27	R 16:	-0.111407
R 09:	5.14528	R 17:	32.83
R 21:	693906.43	R 18:	-0.052952
R 22:	122.0019	R 19:	262.49
R 23:	365.25		
R 24:	30.6001		
R 10:	-0.985647		
R 11:	142.9		
R 12:	282.2		

Beispiel: Errechnung der Finsternisse für die zweite Hälfte des Jahres 1982

Eingabe: 1982.5
XEQ"FI"

FINST: 1982,50

MONDF: AM 6. 7.1982
UM 8.29 UHR
SONNF: AM 20. 7.1982
UM 19.54 UHR
SONNF: AM 15. 12.1982
UM 10.14 UHR
MONDF: AM 30. 12.1982
UM 12.38 UHR

01*LBL "PH"	56 FS? 00	111 X<>Y	166 RTN	221 9
02 FC? 04	57 GTO 01	112 -	167 "UM "	222 +
03 XEQ 05	58 RCL 08	113 ISG 25	168 ARCL X	223 *
04 SF 00	59 *	114*LBL 02	169 "+ UHR"	224 ST+ 03
05 CF 03	60 -	115 E	170 PRA	225 RCL 03
06 "PHAS: "	61 RCL 02	116 %	171 CF 01	226 RCL 05
07 GTO 04	62 COS	117 Rt	172*LBL 14	227 +
08*LBL "FI"	63 RCL 06	118 +	173 RCL 01	228 3
09 FC? 04	64 *	119 RCL 25	174 RCL 02	229 LN
10 XEQ 05	65 RCL 07	120 E6	175 SIN	230 P-R
11 "FINST: "	66 +	121 /	176 +	231 2
12 SF 03	67 -	122 +	177 ST+ 00	232 D-R
13 CF 00	68 X)?	123 RCL 22	178*LBL 03	233 -
14*LBL 04	69 GTO 14	124 FRC	179 9,1	234 *
15 FIX 2	70 LASTX	125 +	180 STO 25	235 ST+ 03
16 CF 29	71 RCL 21	126 STO Y	181 XEQ 00	236 RCL 03
17 SF 28	72 FRC	127 E2	182 ENTER↑	237 RCL 04
18 ADV	73 *	128 *	183 ISG 25	238 XEQ 00
19 SF 12	74 /	129 FRC	184 RCL IND 25	239 -
20 ACA	75 FS? 02	130 E4	185 +	240 +
21 CF 12	76 "MONDF: "	131 *	186 ,2	241 STO 04
22 ACX	77 FC?C 02	132 RCL 26	187 P-R	242 ENTER↑
23 PRBUF	78 "SONNF: "	133 INT	188 ISG 25	243 +
24 ADV	79*LBL 01	134 XCY?	189 RCL IND 25	244 SIN
25 STO 26	80 FIX 2	135 STOP	190 +	245 9
26 RCL 23	81 RCL 00	136 RCL Z	191 X<>Y	246 /
27 *	82 INT	137 FS? 02	192 STO 04	247 -
28 RCL 21	83 ENTER↑	138 "VOLLM: "	193 *	248 ST+ 05
29 -	84 ENTER↑	139 FS? 03	194 +	249 RCL 06
30 STO 00	85 RCL 22	140 GTO 11	195 STO 05	250 RCL 05
31 GTO 03	86 -	141 FC?C 02	196 XEQ 00	251 TAN
32*LBL 05	87 RCL 23	142 "NEUM: "	197 STO 03	252 ATAN
33 RDTA	88 /	143*LBL 11	198 XEQ 00	253 ABS
34 SF 04	89 INT	144 "+ AN "	199 STO 02	254 X<=Y?
35 RTN	90 STO 25	145 STO Y	200 -	255 GTO 12
36*LBL 12	91 RCL 23	146 INT	201 +	256 LASTX
37 CF 02	92 *	147 SF 29	202 +	257 RCL 02
38 RCL 09	93 INT	148 FIX 0	203 SIN	258 COS
39 RCL 04	94 -	149 ARCL X	204 32	259 8
40 SIN	95 ENTER↑	150 Rt	205 %	260 -
41 *	96 ENTER↑	151 FRC	206 +	261 *
42 RCL 05	97 RCL 24	152 E2	207 RCL 04	262 E
43 ENTER↑	98 /	153 *	208 +	263 %
44 +	99 INT	154 FIX 4	209 ST+ 03	264 ST+ 00
45 RCL 04	100 STO 20	155 ARCL X	210 RCL 03	265 GTO 03
46 -	101 RCL 24	156 PRA	211 RCL 04	266*LBL 00
47 SIN	102 *	157 RCL Z	212 RCL 24	267 ISG 25
48 7	103 INT	158*LBL 12	213 D-R	268 RCL IND 25
49 /	104 -	159 RCL 00	214 /	269 RCL 00
50 +	105 12	160 FRC	215 +	270 *
51 ABS	106 RCL 20	161 24	216 ST+ 02	271 ISG 25
52 RCL 05	107 E	162 *	217 RCL 02	272 RCL IND 25
53 COS	108 -	163 HMS	218 ,7	273 +
54 X)?	109 X<=Y?	164 FIX 2	219 ST* 04	274 END
55 SF 02	110 GTO 02	165 FS? 01	220 P-R	

GTO 03 in Zeile 31, GTO 14 in Zeile 63, GTO 12 in Zeile 255 und GTO 03 in Zeile 265 sollten synthetisch erzeugte "long-form"-GTO's sein, da sonst die Sprungweite nicht gespeichert werden kann, was eine Verzögerung des Programmablaufes zur Folge hätte.

Rechner-Bugs

Bug bedeutet eigentlich Insekt, Wanze oder Käfer (der Käfer heißt dagegen Beetle), ist aber auch der englische Slang-Ausdruck für Fehler. Und darum geht es hier auch: Um Fehler des HP-41C und der Peripherie-Geräte. Viele Fehler wurden in den neuen C's und CV's bereits beseitigt. Dennoch können solche Fehler manchmal zu merkwürdigen Erscheinungen führen, weshalb sie hier aufgelistet sind.

1. Σ^+ und Σ^- bewirken keinen Eintrag von X in LASTX.
2. Bei SIZE 000 lassen sich die Register 768 bis 999 mit indirekten Speicheroperationen ansprechen. Ein Teil dieser Register gehört dabei dem Programmspeicher (also Belegung mit KA's oder Programmen), ein weiterer gehört den 127 Registern des erweiterten Speichers im X-FUNCTION-Modul, und schließlich lassen sich auch die Statusregister so indirekt aufrufen. Wird SIZE geändert, verändert sich auch die Lage dieser Register. Bei SIZE 000 ist 768 das T-Register.
3. Alle Flags lassen sich indirekt adressieren. Um festzustellen, ob ein vorliegender Rechner diesen Fehler hat, muß 49, SF IND X ausgeführt werden; erscheint BAT, hat der Rechner diesen Fehler.
4. Für sehr kleine Zahlen ($<5,729577951 \text{ E-}99$) liegt der SIN außerhalb des Rechenbereiches, so daß der Rechner ihn zu 0 runden müßte. Stattdessen werden Werte ausgegeben, die um den Faktor 10^{100} zu groß sind (nur DEG/GRAD).
5. CLP löscht nicht das ganze Programm, wenn der Drucker geschlossen ist. Im NORM oder TRACE-Modus genügen dieser Funktion 233 Zeilen, im MAN-Modus werden dagegen schon 1089 Zeilen gelöscht.
6. Steht in einem HP-67/97 Programm: EEX, CHS, 7, CHS, 5, CHS, dann übersetzt der Kartenleser in den HP-41: E-7-5-.
7. Werden mit ASTO 6 Zeichen aus dem ALPHA-Register (mit mindestens 7 Zeichen gefüllt) abgespeichert, dann wird in die Bits 4-7 des angesprochenen Registers noch ein Teil des 7 Alpha-Zeichens eingeschrieben. Beispiel: ALPHA, ABCDEF, ASTO X, APPEND, G, ASTO Y. Nun versagen die beiden Tests $X=Y?$ und $X\neq Y?$, sie behaupten nämlich einfach das Gegenteil von dem, was man mit $X<>Y$ feststellen kann. Als einzige Abhilfe ist: CLA, ARCL X, ASTO X, CLA, ARCL Y, ASTO Y möglich.
8. Die Sprungadressen der lokalen GTO's und XEQ's werden nach einer Programmänderung nicht gelöscht, wenn man statt mit PRGM einfach mit ON, ON aus dem PRGM-Modus aussteigt. Beispiel: LBL A GTO OO, ENTER, LBL OO, TONE 9. Nach XEQ A ertönt TONE 9; nun PRGM, GTO .004, BEEP, ON, ON. Nun führt XEQ A dazu, daß zuerst BEEP und dann TONE 9 ertönt und nicht NONEXISTENT erscheint, weil das LBL OO gelöscht wurde. Das Programm kann auch mit SST durchgetastet werden. Abhilfe: PRGM, PRGM oder eine Operation, die PACKING auslöst (GTO .., PACK).
9. Dieser Fehler erlaubt es, in die KA-Register zu springen (also über das .END. hinweg), von wo man mit SST durch den erweiterten Speicher des X-FUNCTION-Modules (falls vorhanden) und die

Statusregister springen kann. Dazu muß man folgende Vorbereitungen treffen: MEMORY LOST, sonst funktioniert es nicht immer. Dann 1 oder 2 KA's, und in den PRGM-Modus, evtl. SIZE so groß wie möglich machen.

Für diesen Fehler nun sind folgende weitere Schritte nötig, um ihn auszunutzen: CAT 1 und sofort R/S. ALPHA, Σ , ALPHA und GTO .OOL; nun ist der Rechner auf das erste Byte des KA-Registers positioniert (TEXT 0, FO). Benutzt wird er bei der Zuweisung des WOLF.

10. Ein anderer Fehler, der erlaubt, über das .END. zu springen, ist folgender: Zunächst die gleichen Vorbereitungen wie oben treffen (s. Bug 9). SST niederdrücken und bei gedrückter SST-Taste ON betätigen. Nun SST und anschließend ON loslassen (das ganze muß schnell gehen, da der Rechner nach SST nicht zur Anzeige NULL kommen darf). Rechner wieder einschalten, PRGM, Σ , GTO .OOL; wieder befindet sich der Rechner im ersten KA-Register.
11. Mit folgenden Schritten verändert man das Flag-Register: GTO .., Σ REG OO (SIZE OO5), CL Σ , EEX 67, STO O2, EEX -97, STO O5, SF 25, SDEV oder MEAN speichert in Register d die Zahl 99 99 9? ?? 99 99 (Hex), wobei ? bedeutet, daß diese Halbbytes nicht verändert werden. Die 9en setzen also die Flags OO, O3, O4, O7, O8, 11, 12, 15, 16, 19, 36, 39, 40, 43, 44, 47, 48, 51, 52 und 55.
12. Fehlt dem Rechner Speicherraum, und ist Flag 25 gesetzt, so kann das PACKING, welches durch Einfügen eines Schrittes ausgelöst wird, zu einer falschen Zeilenummerierung führen. Beispiel: GTO .., +,+,+,+,+, STOP, LN, nun SIZE maximal wählen und außerhalb von PRGM SF 25, GTO .OOL. Nun in das Programm irgendeinen Befehl einfügen (z.B. ENTER), nach dem PACKING erscheint in der Anzeige: O2 LN
13. Wenn X oder Y negativ sind, weichen die Ergebnisse von $X \text{ MOD } Y$ und von $Y - X \cdot \text{INT}(Y/X)$ ab, da bei der internen Berechnung ein anderes INT verwendet wird. Das interne Integer ist die Gaußsche Klammerfunktion, die immer zur nächstkleineren Ganzzahl rundet, z.B. -2,3 auf -3, während das verdrahtete INT immer auf die betragsmäßig kleinere Zahl rundet, also -2,3 auf -2.
14. Für große Zahlen ergibt R-P nicht die Fehlermeldung OUT OF RANGE, sondern in X steht ein um 10^{100} zu kleiner Wert, z.B. nach 8 E99, ENTER \uparrow 6 E99, R-P.
15. Wird CAT IND auf die Stackregister angewendet, z.B. CAT IND L, führt der Rechner immer CAT 3 aus, unabhängig vom Inhalt des Registers.
16. FIX, SCI, ENG und TONE reagieren bei indirekter Adressierung auf Werte zwischen 10 - 999 mit DATA ERROR, bei Werten von 1000 oder mehr mit NONEXISTENT.
17. 64, BLDSPEC, XEQ IND X führt zum Crash. Ebenso GTO IND X oder wenn synthetisch GTO oder XEQ mit diesem Zeichen programmiert werden. Einzige Ausnahme: Das LBL Affenschwanz ist vorhanden.

18. Entgegen dem Hinweis im Handbuch wird das Dateneingabeflag 22 beim Lesen von Datenkarten nicht gesetzt.
19. Enthält die Anzeige eine Meldung, welche durch einen VIEW oder AVIEW dorthin gebracht wurde, so ist das Message-Flag (50) gesetzt. CLD löscht Flag 50 und die Anzeige. Mit SF 25 und irgend-einer Operation, die zu einer Fehlermeldung führt, läßt sich Flag 50 löschen, ohne den Inhalt zu verändern, der angezeigt wird. Dabei läuft dann die Anzeige statt des normalen Seglers durch. Beispiel: Mit folgendem Programm läßt sich der Effekt demonstrieren. "CCD", AVIEW, SF 25, SF 30 (verursacht normalerweise NONEXISTENT), LBL 00, PI, SIN, GTO 00; evtl. angeschlossene Drucker einschalten oder CF 21.
20. Fehler in der Anzeige des ALPHA-Registers treten im Zusammenhang mit SPACE und ., oder : auf; tastet man A, SPACE, :, SPACE, :, SPACE, : ein und betätigt 3 mal , dann scheint nur noch A im ALPHA-Register zu stehen. Nach AVIEW erscheint dagegen der korrekte Inhalt wieder.
21. Steht in X ein Punkt, Komma oder Doppelpunkt, und stehen im ALPHA-Register mehr als 11 Zeichen, erscheint nach dem Aussteigen aus dem ALPHA Modus das letzte ALPHA-Zeichen ganz links in der Anzeige. Jeder Tastendruck läßt es allerdings wieder verschwinden.
22. Steckt man ein ROM (z.B. MATH) in Slot 4, positioniert den Rechner in ein solches ROM-Programm, und vertauscht es dann nach dem Ausschalten mit dem Kartenleser, steht man in dessen ROM.

Das Querschriftalphabet

Mit den folgenden Tabellen können Sonderzeichen und das Alphabet in Querschrift über die Funktion BLDSPEC erzeugt werden. Das System zur Erzeugung ist dabei sehr einfach. Nehmen wir z.B. die Ziffer 9: Zunächst den Stack löschen und BLDSPEC einer Taste zuweisen (CLST, ASN "BLDSPEC" 15). Unter der Spalte COL stehen nun die Zahlen 1 - 7 und rechts daneben eine Menge Zahlen. Uns interessieren jedoch nur die, welche unter der 9 stehen, also 24, 32, 64, 126, 65, 65, 62. Wir tasten jede dieser Zahlen ein und führen danach BLDSPEC aus, also:

24, BLDSPEC, 32 BLDSPEC usw. und schon ist das Zeichen fertig !

Zweites Beispiel: Das + in Querschrift.

In der Reihe neben der 1 steht ein Strich, was bedeutet, daß für dieses Zeichen nur 6 mal BLDSPEC ausgeführt werden muß und auch nur 6 Zahlen eingegeben werden müssen. Es sind dies also: 8, 8, 62, 8, 8, 0.

Diese werden also so eingegeben:

8 BLDSPEC, 8 BLDSPEC, 62 BLDSPEC, 8 BLDSPEC, 8 BLDSPEC, 0 BLDSPEC.
Vorsicht: Die letzte 0 darf nicht weggelassen werden !

COL	X	Y	N	A	COL	O	O	I	I	COL	dr	dr	D	±	COL	□	□	□	◆
1→	99:	8:	127:	28:	1→	-:	-:	-:	-:	1→	128:	128:	-:	-:	1→	127:	127:	127:	8:
2→	20:	8:	2:	28:	2→	-:	56:	-:	-:	2→	72:	72:	49:	68:	2→	65:	65:	65:	28:
3→	8:	8:	28:	21:	3→	-:	68:	-:	68:	3→	127:	127:	73:	68:	3→	85:	85:	93:	62:
4→	20:	28:	32:	62:	4→	14:	68:	-:	124:	4→	0:	0:	73:	95:	4→	65:	73:	65:	127:
5→	99:	99:	127:	64:	5→	17:	56:	17:	64:	5→	88:	8:	74:	68:	5→	85:	85:	93:	62:
6→	0:	0:	0:	0:	6→	17:	0:	31:	0:	6→	32:	124:	68:	68:	6→	65:	65:	65:	28:
7→	0:	0:	0:	0:	7→	14:	0:	16:	0:	7→	24:	72:	0:	0:	7→	127:	127:	127:	8:

COL	↓	M	T	COL	□	□	□	□	COL	→	~	≈	≈	COL	♥	♣	♠	♣	
1→	8:	16:	127:	1:	1→	-:	-:	-:	-:	1→	8:	8:	36:	34:	1→	14:	56:	28:	-:
2→	8:	32:	66:	1:	2→	-:	-:	-:	-:	2→	24:	4:	34:	17:	2→	31:	68:	28:	-:
3→	8:	127:	4:	1:	3→	-:	116:	-:	84:	3→	40:	4:	34:	17:	3→	62:	38:	95:	28:
4→	8:	32:	8:	127:	4→	-:	84:	-:	84:	4→	8:	8:	36:	34:	4→	124:	127:	127:	62:
5→	8:	16:	4:	1:	5→	29:	92:	21:	124:	5→	10:	16:	40:	68:	5→	62:	38:	95:	62:
6→	8:	0:	66:	1:	6→	21:	0:	21:	0:	6→	12:	16:	40:	68:	6→	31:	68:	28:	127:
7→	8:	0:	127:	1:	7→	23:	0:	31:	0:	7→	8:	8:	36:	34:	7→	14:	56:	28:	127:

COL	Z	Z	+	+	COL	Y	Y	5	5	COL	≡	≤	≥	∞	COL	<	▷	>	
1→	-:	-:	-:	-:	1→	-:	-:	-:	-:	1→	-:	-:	-:	28:	1→	-:	127:	65:	-:
2→	-:	-:	-:	-:	2→	-:	-:	-:	-:	2→	42:	80:	81:	34:	2→	-:	127:	65:	-:
3→	-:	100:	-:	16:	3→	-:	28:	-:	92:	3→	42:	88:	82:	28:	3→	28:	62:	34:	-:
4→	-:	84:	-:	56:	4→	-:	16:	-:	84:	4→	42:	84:	84:	8:	4→	34:	62:	34:	-:
5→	25:	76:	4:	16:	5→	7:	124:	23:	116:	5→	42:	82:	88:	28:	5→	34:	28:	28:	-:
6→	21:	0:	14:	0:	6→	4:	0:	21:	0:	6→	42:	81:	80:	34:	6→	65:	0:	0:	-:
7→	19:	0:	4:	0:	7→	31:	0:	29:	0:	7→	0:	0:	0:	28:	7→	65:	0:	0:	-:

COL	-	-	✓	✓	COL	5	5	7	7	COL	▽	II	€	¥
1→	-:	-:	-:	-:	1→	-:	-:	-:	-:	1→	3:	-:	-:	8:
2→	-:	-:	-:	-:	2→	-:	-:	-:	-:	2→	13:	65:	-:	4:
3→	-:	16:	-:	32:	3→	-:	124:	-:	4:	3→	49:	127:	28:	68:
4→	-:	16:	-:	16:	4→	-:	84:	-:	4:	4→	65:	65:	42:	68:
5→	4:	16:	8:	8:	5→	31:	116:	1:	124:	5→	49:	127:	42:	58:
6→	4:	0:	4:	0:	6→	21:	0:	1:	0:	6→	13:	65:	0:	1:
7→	4:	0:	2:	0:	7→	29:	0:	31:	0:	7→	3:	0:	0:	0:

COL	<	<	>	>	COL	■	■	Y	Y	COL	ψ	ψ	ω	⊙	
1→	-:	-:	-:	-:	1→	-:	-:	-:	-:	1→	-:	-:	3:	56:	28:
2→	-:	-:	-:	-:	2→	-:	-:	-:	-:	2→	4:	12:	68:	34:	
3→	-:	56:	-:	-:	3→	-:	124:	-:	28:	3→	120:	16:	64:	93:	
4→	-:	68:	-:	68:	4→	-:	84:	-:	26:	4→	32:	127:	56:	85:	
5→	14:	0:	-:	56:	5→	31:	124:	7:	124:	5→	16:	16:	64:	85:	
6→	17:	0:	17:	0:	6→	21:	0:	5:	0:	6→	12:	12:	68:	34:	
7→	0:	0:	14:	0:	7→	31:	0:	31:	0:	7→	0:	3:	56:	28:	

COL	■	Γ	Γ	ds	COL	π	π	Y	Y	COL	™	□	□	□
1→	-:	16:	-:	128:	1→	-:	-:	-:	-:	1→	1:	127:	127:	127:
2→	-:	16:	64:	72:	2→	-:	-:	-:	-:	2→	15:	65:	65:	65:
3→	7:	32:	64:	127:	3→	-:	108:	-:	12:	3→	121:	65:	69:	69:
4→	5:	127:	62:	0:	4→	-:	16:	-:	112:	4→	8:	73:	65:	73:
5→	7:	1:	1:	72:	5→	27:	108:	3:	12:	5→	48:	65:	81:	81:
6→	0:	1:	1:	48:	6→	4:	0:	28:	0:	6→	8:	65:	65:	65:
7→	0:	1:	0:	72:	7→	27:	0:	3:	0:	7→	128:	127:	127:	127:

