Best of

PRISMA

CCD - Computerclub Deutschland e.V.

Best of
PRISMA

Die digitale Ausgabe des Buchs "Best of PRISMA" wurde 2016 erstellt. Dazu wurde ein Exemplar des Buchs zerlegt, eingescannt und aufbereitet. Die Aufbereitung umfasste die Erstellung von PDF-Dokumenten, die Anwendung optischer Zeichenerkennung sowie die Ergänzung von Lesezeichen zur Navigation.
Um das Ganze etwas zu beleben wurden Titelblatt und Rückseite koloriert. Die Originalversionen dieser Seiten finden sich am Ende des Dokuments.

Das Buch wurde freundlicherweise von Jürgen Keller zur Verfügung gestellt. Die Aufbereitung erfolgte durch Martin Hepperle.

Hinweis:
Die Druckvorlagen des Buchs wurde offensichtlich durch Verkleinerung von A4 auf A5 hergestellt. Da die Druckqualität des Originals nicht besonders hoch war, sind manche Ausdrucke des HP-41 Thermodruckers schwer lesbar.
Im PDF Dokument kann die Ansicht bequem vergrößert werden. Alternativ lassen sich solche Seiten auch vergrößert auf A4 Papier ausdrucken und damit die Vorlagengröße wieder herstellen.

Verbesserungen und Ergänzungen bitte an
Martin.Hepperle@MH-AeroTools.de
oder
JKeller@gmx.ch

Best of

PRISMA

1. Auflage 0 - 2600 · 1986

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

Inhalt

Inhalt

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

— 1. —
Tips & Tricks

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 mod x) zu berechnen, kann man ganz einfach auch -y 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 Zeilennumerierung 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=0?, GTO 01, ABS, LOG, INT, X<=0?, GTO 01, ST-Y 3, /, INT, FS? 29, ST- Y, LBL 01, RDN, X<0?, 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 0, LBL 00, INT, X=Y?, RTN, STO Z, RCL Y, SF 25, LBL 01, +, 2, /, ARCL IND X, FC? 25, GTO 00, RND, STO Y, R↗, GTO 01.

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 Permanentspeicher.

---

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 0 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 4,777779213 E18 lassen sich mit 1643,8766 E9 oder 4777779213 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 03; FC? 07; GTO 03. 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? 07; GTO 03. 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>0? und X≤0?, FS? und FC?. Zu X<Y? und X<0? fehlen auf dem HP41 die Umkehrungen X≥Y? und X≥0?;
Wer genau nachdenkt, wird merken, daß unser oberes Beispiel genausogut funktioniert, wenn es FC? 07; GTO 03; X=0?; GTO 03 lautet. Aber kann jetzt auch noch zusammengefaßt werden? Ja, nämlich zu: FS? 07; X=0?; GTO 03. Mit diesem Wissen können wir nun auch dem Rechner die Tests programmieren, die nicht vorhanden sind: X≥Y? und X≥0? lassen sich so also durch X≠Y?; X>Y? bzw. X≤Y?; X=Y? und X≠0?; X>0? bzw. X≤0?; X=0? 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.

Unterprogrammrücksprungadressen 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 (schnelleren) 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 Formelei findet man in vielen Standardfunktionen, z.B. statt der Schrittfolge ×, 100, /, geht es mit % viel besser. Andere Beispiele (x = Inhalt des X-Registers, y = Inhalt des Y-Registers): x·y/ 100 = %; y·sin(x) nach P-R in Y, y·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↑2, LOG = 2·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 soch 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	∝ aus	- alpha mode off
AON	∝ ein	- alpha mode on
ARCL	zurückrufen	- alpha recall
ASHF	Löscht 6 ∝ Zeichen	- alpha shift
ASIN	arc sin	- arc sine
ASN	Fkt. Zuordnung	- assign
ASTO	∝ speichern	- alpha store
ATAN	arc tg	- arc tangnent
AVIEW	∝ betrachten	- alpha view
BEEP	Akust. Signale	- beep
BST	1 Schritt zurück	- backstep
CAT	Catalog	- catalog
CF	Flag löschen	- clear flag
CHS	Vorzeicherwechsel	- change sign
CLA	Cl. Register	- clear alpha register
CLD	Cl. Anzeige	- clear display
CLP	Cl. Programm	- clear program
CLRG	Cl. Datenspeicher	- clear registers
CL Σ	Cl. der 6 Statistik R.	- clear statistical registers
CLST	Cl. Stack	- clear stack
CLX	Cl. x Register	- clear x
COPY	Kopieren oder Umspeichern 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 one
LOG	log	- common log
MEAN	arithmet. Mittel	- mean
MOD	Modulo Funktion	- modulo
OCT	Dezimal Oktal (DEC)	- decimal to octal
OFF	Ausschaltung	
ON	Einschaltung	
P-R	Polar rechtw. 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
SQRT	$\sqrt{x}$	- 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=0?	$x = 0$?	Bei Nein: Sprung
X≠0?	$x \neq 0$?	
X<0?	$x < 0$?	
X<=0?	$x \leq 0$?	
X>0?	$x > 0$?	
X=Y?	$x = y$?	
X≠Y?	$x \neq y$?	
X<Y?	$x < y$?	
X<=Y?	$x \leq y$?	
X>Y?	$x > y$?	
X<>	Austausch X mit Y.Z, T,L oder Daten Reg.	
X<>Y	$x \rightleftarrows y$	
XEQ	Aufruf einer Fkt.	- execute
X↑2	x^2	
Y↑X	y^x	
EEX	Exponent von 1o	- 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 Flg.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 LOG
08 INT
09 X<=0?
10 GTO 01
11 ST- Y
12 3
13 /
14 INT
15 FS? 29
16 ST- Y
17*LBL 01
18 RDN
19 X<0?
20 CLX
21 SKPCHR
22 RDN
23 ACX
24 END
LBL'ACXR
END      - 41 BYTES
RY=ANZ. STELLEN VOR KOMA
EINSCHL. VORZ. -2
RX=BERECHNET
```

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.

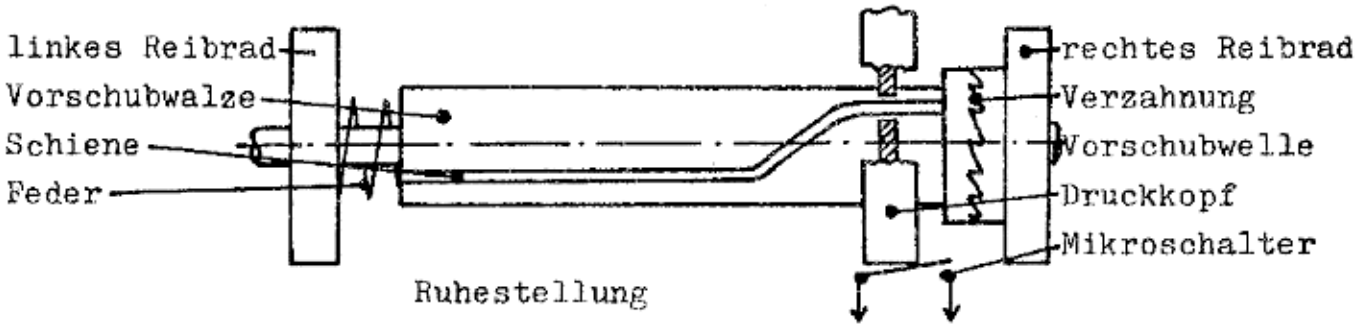
– 2. –
Barcode-Programme

Winfried Maschke
Eichhardtstraße 3
5276 Wiehl

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

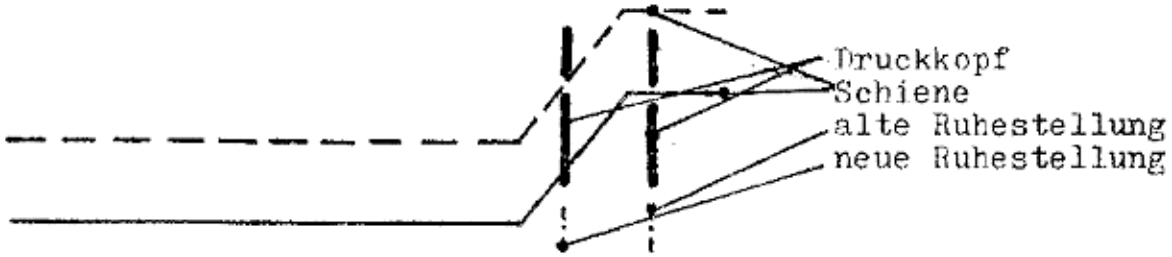
Der normale Druckervorschub

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 bewegten.



Der veränderte Druckervorschub

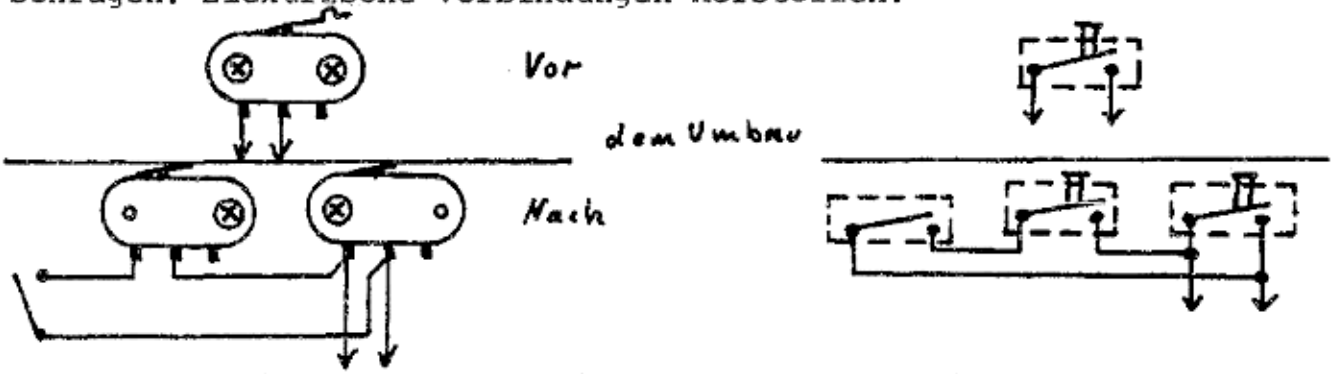
Verschiebt man den Mikroschalter nach links, so verschiebt sich auch die Ruhestellung und damit die ganze Druckzeile nach links. Wenn man die Ruhestellung weit genug nach links verschiebt, kann nur noch ein Zahn in der Verzahnung übersprungen werden, der Vorschub wird halbiert. Nun schlägt jedoch der 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 abschrägen. 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 blaudruckende 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 verblassen. 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	- 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	- 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	- Byte 1 wie Typ 1 - Byte 2 = 64 - Byte 3 bis 11 = Befehl (z.B. XEQ ABC oder STO 25)
6	Numeric Data Code	- 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	- Byte 1 wie Typ 1 - Byte 2 = 112 + Länge der Alpha-Kette - Byte 3 bis 16 = Textzeichen
8	Alpha Append Code	- 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 ØØØ).

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 Tastenzuordnungen 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 ØØØ
4. zu druckendes Programm einlesen
5. ABARP einlesen
6. Drucker: ON, MAN, halber Zeilenvorschub
7. SF 1Ø (PRIVATE) oder CF 1Ø (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 [
-2  AROT  1773  ATOX  16
/  INT  -  7  ATOX
STO [  *  -  RCL c
X<> [  -3  AROT  RDN
253  -  ATOX  16  *
X<>Y  ST+ Y  X<>Y  LASTX
*  ATOX  +  256.  /  XTOA
LASTX  *  LASTX  MOD
XTOA  SIGN  AROT  X<> [
STO c  RDN  CHS  ")"
SF 25  PURFL  CRFLD
SAVER  30  PSIZE  CLRG
R↑  ADV  FIX 0  CF 29
SF 12  ARCL X  ACA  2  -
STO 05  R↑  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  16.828
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 [

133+LBL 08
SF 02  ALENG  RCL 04
X≠Y?  0  X=Y?  ATOX
ENTER↑  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  R↑  2  /  INT
183  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  BLDSPEC
LASTX  BLDSPEC  LASTX
BLDSPEC  LASTX  BLDSPEC
LASTX  BLDSPEC  LASTX
BLDSPEC  LASTX  BLDSPEC
ACSPEC  ACSPEC  ACSPEC
ACSPEC  2  SKPCOL

335+LBL 09
CLX  FC? 09  ATOX
XEQ IND X  CLX  X<> [
ACSPEC  -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
"├0eY4ρ "  RTN

360+LBL 49
"├0ρ0ΣΓδ"  RTN

363+LBL 50
"├Q50 "  RTN

366+LBL 51
"├0νΓδ ├"  RTN

369+LBL 52
"├00Eν0"  RTN

372+LBL 53
"├0νδ‡"  RTN

375+LBL 54
"├0ν0
A<"  RTN

378+LBL 55
"├0δ δ ├"  RTN

381+LBL 56
"├0ν0" "  RTN

384+LBL 57
"├0γα‡" "  RTN

387+LBL 69
"├QδÀ
@"  RTN

390+LBL 40
"├Q××x̄α00"  RTN

393+LBL 41
"├0α00××"  RTN

396+LBL 82
"├Q0σ0" "  RTN

399+LBL 73
"├0ρ0x̄αε"  RTN

402+LBL 72
"├QΓμ#"" RTN

405+LBL 45
"├0++‡
++"  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 (dieAngaben beziehen sich auf SIZE ØØØ).

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 1Ø, 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 ØØØ
4. zu druckendes Programm einlesen
5. ABARP einlesen
6. Drucker: ON, MAN, halber Zeilenvorschub
7. SF 10 (PRIVATE) oder CF 1Ø (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 [
-2  AROT  1773  ATOX  16
/  INT  -  7  ATOX
STO [  *  -  RCL c
X<> [  -3  AROT  RDN
253  -  ATOX  16  *
X<>Y  ST+ Y  X<>Y  LASTX
*  ATOX  +  256  /  XTOA
LASTX  *  LASTX  MOD
XTOA  SIGN  AROT  X<> [
STO c  RDN  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
CLR  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
BEEP  RTN

105*LBL 02
DSE 04  GTO 08  7
STO 04  DSE 06  "
RCL 06  SEEKPT  GETX
STO [

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 Reihe-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

Unterprogramm "BP" = Byte Print Size ØØ4 161 Bytes

Dieses Programm druckt jeweils ein einzelnes Byte (8 Striche = 8 Bit), welches als dezimale Zahl im X-Register übergeben werden muß. Die einzelnen Bytes werden in umgekehrter Reihenfolge eingegeben und gedruckt. Beim ersten Aufruf von BP ist das Flag ØØ zu setzen, der Endcode der Zeile (1 Ø) wird dann automatisch gedruckt. Sind alle Bytes der Zeile gedruckt, ist das Flag Ø1 zu setzen und BP nochmals aufzurufen. Dann wird die in Speicher Ø3 aufaddierte Prüfsumme und anschließend der Anfangscode (Ø Ø) als Zeilenkopf ausgedruckt. Wird BP beim Druck von Programmen benutzt (siehe Programm BARP) und ist das Unterprogramm ROW vorhanden, wird es von BP aufgerufen. Vor dem ersten Byte ist der Speicher Ø3 zu löschen, wenn eine Prüfsumme benötigt wird.

W. Maschke

```
          PRP "BARA"

 01*LBL "BARA"
ASTO 07  ASHF  XEQ 07
XEQ 01  CLA  ARCL 07
XEQ 01  112  FS?C 02
128  GTO 05

 13*LBL "BARGTO"
29  GTO 03

 16*LBL "BARXEQ"
30

 18*LBL 03
STO 04  XEQ 07  1
GTO 04

 23*LBL "BARASN"
ENTER↑  ABS  ENTER↑
XEQ 07  10  MOD  1  X=Y?
SF 01  RDN  ST- Y  X<>Y
LASTX  /  4  FC?C 01
GTO 00  X=Y?  ISG Z

 43*LBL 00
X↑2  *  +  128  R↑  SIGN
*  X>0?  CLX  -  256
MOD  STO 04  XEQ 01  15
X<> 04  6  GTO 04

 62*LBL "BARSIZE"
6  GTO 08

 65*LBL "BARDEL"
2  GTO 08

 68*LBL "BARGTOL"
1

 70*LBL 08
STO 04  RDN  STO 05  256
MOD  ST- 05  LASTX
ST/ 05  RDN  XEQ 07
XEQ 06  RCL 05  6
GTO 04

 85*LBL "BARDE"
INT  STO 04  LASTX  FRC
 E3  *  XEQ 07  6

 94*LBL 04
RDN  XEQ IND T  RCL 04
XEQ 06  64

100*LBL 05
XEQ 06  SF 01

103*LBL 06
GTO "BP"

105*LBL 07
SF 00  CF 01  CF 03  0
STO 03  RDN  RTN

113*LBL 01
7  STO 06

116*LBL 02
ASTO 05  ASHF  "⊢++++X̄"
X<> [  X<> d  FS?C 00
SF 06  FS?C 09  SF 07
FS?C 10  SF 09  FS?C 11
SF 10  FS?C 12  SF 11
X<> d  DEC  X≠0?  SF 03
FS? 03  XEQ 06  " "
ARCL 05  DSE 06  GTO 02
.END.

          PRP "BP"

 01*LBL "BP"
.007  STO 00  RDN
FS?C 00  XEQ 05  ST+ 03
FC?C 01  GTO 00  ST- 03
RCL 03  255  MOD  XEQ 00
XEQ 04  XEQ 02  XEQ 04
XEQ 02  XEQ 03  ADV  ADV
ADV  ADV  RTN

 25*LBL 00
XEQ 04  XEQ 02  2  /
INT  LASTX  FRC  X≠0?
XEQ 02  X≠0?  XEQ 02
RDN  ISG 00  GTO 00  RTN

 41*LBL 02
2  RCL 01  PI  Y↑X
ST+ 02  RDN  ISG 01  RTN

 50*LBL 03
R↑  0  X<> 02  BLDSPEC
LASTX  BLDSPEC  LASTX
BLDSPEC  LASTX  BLDSPEC
LASTX  BLDSPEC  LASTX
BLDSPEC  LASTX  BLDSPEC
ACSPEC  ACSPEC  ACSPEC
ADV  .005  STO 01  RDN
RDN  RTN

 76*LBL 04
ISG 01  GTO 01  XEQ 03

 80*LBL 01
RCL 01  2  X>Y?  ISG 01
RDN  RDN  RTN

 88*LBL 05
ADV  SF 12  58  STO 02
RDN  XEQ 03  .END.
```

CAT 1

```
LBL"BARA
LBL"BARGTO
LBL"BARXEQ
LBL"BARASN
LBL"BARSIZE
LBL"BARDEL
LBL"BARGTOL
LBL"BARDE
END          307 BYTES
LBL"BP
END          161 BYTES
```

— 3. —
Weitere Anwendungen des Druckerumbaus

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.

Herrn
Walter Maier
Ottostr. 5
5000 Köln 1

Beispiel 2:

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

Die Datenregisterinhalte für das Querschriftprogramm sind nachstehend aufgelister. 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=	"×A@"	1 65 5 0 0 0
R02=	"ፊ X̄ፊ+)"	136 160 130 136 128 62
R03=	"θ0pA"	16 23 240 65 0 0
R04=	"qፊ$N"	113 18 36 87 0 0
R05=	"σTh0!("	9 212 164 79 161 60
R06=	"αፊθ ⊦"	4 8 16 32 96 255
R07=	" b!X̄αፊ"	32 226 161 2 4 8
R08=	"ΣáIé0@"	254 20 73 20 48 64
R09=	"qፊ$_"	113 18 36 95 0 0
R10=	" coo≮ፊ"	32 227 239 239 142 8
R11=	"BQA×××"	5 17 65 1 1 1
R12=	"σdHQ"	9 228 72 145 0 0
R13=	"⊦σ!X̄ፊ "	253 9 33 130 8 32
R14=	" @0P"	32 64 151 208 0 0
R15=	"xCi/σ>"	248 67 233 47 132 62
R16=	"zμ‰ρ')"	250 12 31 240 96 190
R17=	"\"(0')"	220 162 40 48 96 190
R18=	"zμ0BX̄ö"	250 12 23 194 2 24
R19=	"Γ‡x/+ፊ"	6 15 246 47 128 8
R20=	"γá/+E"	121 10 20 47 128 8
R21=	"Γ‡x/+á"	6 15 248 47 128 20
R22=	"zμö?+$"	250 12 24 63 128 36
R23=	"zμö0_á"	250 12 24 48 95 20
R24=	"zμö/+""	250 12 24 47 128 34
R25=	"zμö0'ö"	250 12 24 48 96 148
R26=	"zμö0@""	250 12 24 48 64 34
R27=	"t(_aGΣ"	244 40 95 225 66 254
R28=	"h/X.+"	232 47 216 174 128 0
R29=	"G‰ED"	71 241 31 196 0 0
R30=	"Σθ '!("	254 16 32 224 161 60
R31=	"⊦⊦⊦⊦⊦⊦"	255 255 255 255 255 255
R32=	""	0 0 0 0 0 0
R33=	" "	32 130 133 10 0 0
R34=	"á"	10 20 0 0 0 0
R35=	"λz‡H?"	139 250 36 72 191 162
R36=	"!≮+BFፊ"	33 252 135 194 127 8
R37=	"+0AαQC"	135 16 65 4 17 195
R38=	"γ≮QDH≮"	121 13 17 196 72 142
R39=	"μö"	12 24 0 0 0 0
R40=	"@@@×α0"	64 64 64 129 4 16
R41=	"0AX̄ααα"	16 65 2 4 4 4
R42=	"B‡α"	66 163 138 132 0 0
R43=	"@+Bα"	64 135 194 4 0 0
R44=	"0A+"	16 193 128 0 0 0
R45=	"+@"	7 192 0 0 0 0
R46=	"θ'"	48 96 0 0 0 0
R47=	"αθAαθ@"	4 16 65 4 16 64
R48=	"zeY4ρ)"	250 28 89 52 112 190
R49=	"p@×X̄Γፊ"	112 64 129 2 6 8
R50=	"≮σΓθ)"	252 24 198 16 32 190
R51=	"zμΓá ⊦"	250 12 6 8 32 127
R52=	"×+tIá0"	129 7 244 73 20 48
R53=	"zμፊ‡@⊦"	250 12 8 15 192 255
R54=	"zμ0'Aμ"	250 12 23 224 65 12
R55=	"θ Aαθ⊦"	16 32 65 4 16 255
R56=	"zμ0P')"	250 12 23 208 96 190
R57=	"aα‡P')"	97 4 15 208 96 190
R58=	"θ'"	48 96 3 6 0 0
R59=	"0A+"	16 193 128 12 24 0
R60=	"@@@Aαθ"	64 64 64 65 4 16
R61=	"+ρ‰@"	7 240 31 192 0 0
R62=	"0Aαααα"	16 65 4 4 4 4
R63=	" "	32 134 16 32 190 0
R64=	"x‡0)')"	248 15 153 62 96 190
R65=	"Γ‡x0')"	6 15 248 48 96 190
R66=	"Σμ0ρ'?"	254 12 23 240 96 191
R67=	"zፊθ ')"	250 8 16 32 96 190
R68=	"Σá(P!?"	254 20 40 80 161 63
R69=	"≮ፊ0'@⊦"	252 8 23 224 64 255
R70=	"αፊ0'@⊦"	4 8 23 224 64 255
R71=	"z≮θ ')"	250 14 16 32 96 190
R72=	"Γμ‰ρ'A"	6 12 31 240 96 193
R73=	"p@×X̄α≮"	112 64 129 2 4 28
R74=	"zμፊθ @"	250 12 8 16 32 64
R75=	"αፊθbH!"	132 136 144 226 72 161
R76=	"≮ፊθ @×"	252 8 16 32 64 129
R77=	"Γμ05αA"	6 12 25 53 113 193
R78=	"+≮01aA"	7 13 25 49 97 193
R79=	"zμö0')"	250 12 24 48 96 190
R80=	"αፊ0ρ'?"	4 8 23 240 96 191
R81=	"w≮ö0')"	251 13 24 48 96 190
R82=	"Bσ0ρ'?"	5 9 23 240 96 191
R83=	"zμ+@')"	250 12 7 192 96 190
R84=	" @×X̄α⊦"	32 64 129 2 4 127
R85=	"zμö0'A"	250 12 24 48 96 193
R86=	" !DH A"	32 161 68 72 160 193
R87=	"+≠Y0'A"	7 29 89 48 96 193
R88=	"BQABQA"	5 17 65 5 17 65
R89=	" @×BQA"	32 64 129 5 17 65
R90=	"≮0Aα0⊦"	252 16 65 4 16 127
R91=	"'@×X̄αö"	96 64 129 2 4 24
R92=	"××××××"	1 1 1 1 1 1
R93=	"θ@×X̄αμ"	48 64 129 2 4 12
R94=	" @×≮ፊ"	32 64 129 10 142 8
R95=	"≮"	252 0 0 0 0 0
R96=	"X̄α>"	2 4 62 0 0 0
R97=	"zμö?+"	250 12 24 63 128 0
R98=	"Σμö/@×"	254 12 24 47 192 129
R99=	"xፊθ?+"	248 8 16 63 128 0
R100=	"zμö? @"	250 12 24 63 160 64
R101=	"xλx/+"	248 11 248 47 128 0
R102=	" @+X̄αθ"	32 64 131 130 28 16
R103=	"r+h?+"	242 7 232 63 128 0
R104=	"Γμöo@×"	6 12 24 111 64 129
R105=	"p@×+"	112 64 129 3 8 0
R106=	"zμፊ@"	250 12 8 16 64 0
R107=	"αHs8@×"	132 200 115 56 64 129
R108=	"p@×X̄αμ"	112 64 129 2 4 12
R109=	"Γμü(+"	6 12 154 168 128 0
R110=	"Γμöo@"	6 12 24 111 64 0
R111=	"zμö/+"	250 12 24 47 128 0
R112=	"B≮ö/@"	5 252 24 47 192 0
R113=	"×rá/+"	129 242 20 47 128 0
R114=	"αፊöo@"	4 8 24 111 64 0
R115=	"Σ+'?+"	254 3 224 63 128 0
R116=	" @+X̄α"	32 160 64 131 130 4
R117=	"zμö0@"	250 12 24 48 64 0
R118=	" "(0@"	32 162 40 48 64 0
R119=	",ö0@"	138 172 152 48 64 0
R120=	"μ X̄ö@"	140 160 130 152 192 0
R121=	" @X̄ö@"	32 64 130 152 192 0
R122=	"≮QD‰@"	252 17 196 31 192 0
R123=	"P!WP"	80 161 87 208 0 0
R124=	" @×X̄αፊ"	32 64 129 2 4 8
R125=	"A+tα"	65 7 244 4 0 0
R126=	"Σ0A×!⊦"	254 16 65 1 33 127
R127=	"αፊ‰'@×"	4 8 31 224 64 129

Programmlisting:

```
01♦LBL "QSX"
SIZE? 150 X<=Y?
GTO 04 PSIZE

07♦LBL 06
"DATENKARTEN" PROMPT
RDTA

11♦LBL 04
RCL 01 "×A@++" 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 R1 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"
02 SF 12
03 "♦ÃÃø"        128 129 1 2 2 4
04 XEQ 01
05 "Ø ÃÃøå"      16 32 65 2 4 8
06 XEQ 01
07 "ÄÄL"         160 65 65 76 128 133
08 XEQ 01
09 "øÅ Å "       4 8 32 131 8 32
10 XEQ 01
11 1
12 XEQ 02
13 "♦♦♦♦ˣ"       15 1 128 0 0 0
14 XEQ 01
15 "♦♦♦♦øµ"      4 12 8 0 0 8
16 XEQ 01
17 "p♦♦"         112 224 128 129 0 0
18 XEQ 01
19 "♦ØØ♦♦♦"
20 XEQ 01        16 48 0 0 0 0
21 "ÃΓåÅ
"                128 2 6 8 7 176
22 XEQ 01
23 ADV
24 "♦♦♦♦²"       194 8 0 0 0 8
25 XEQ 01
26 "ØØµõõ"       65 66 229 12 24 16
27 XEQ 01
28 "ÃÃµøå"       65 129 2 12 4 38
29 XEQ 01
30 "p♦ø²×9"      112 4 7 1 57 0
31 XEQ 01
32 "Ñ"           241 251 247 227 208 31
33 XEQ 01
34 ADV
35 "µ:w
"                P
                 12 58 119 168 80 177
36 XEQ 01
37 "   Åå"       32 32 32 65 196 8
38 XEQ 01
39 "Ø×=="        64 128 129 1 1 1
40 XEQ 01
41 1
42 XEQ 02
43 "ØØ  Åå"      16 16 32 32 65 4
44 XEQ 01
45 " ♦åØ "       32 130 8 16 32 8
46 XEQ 01
47 11
48 SKPCOL
49 "Ø       ø"   48 146 166 192 145 4
50 XEQ 01
51 "Øå♦♦ "       145 17 20 191 0 0
52 XEQ 01
53 15
54 XEQ 02
55 "♦♦øLÜ"       4 230 76 26 0 0
56 XEQ 01
57 "ᵀδ$Gᵀ"       96 185 18 36 71 2
58 XEQ 01
59 "åØØøµ"       8 16 64 130 4 12
60 XEQ 01
61 "MØ×"         77 129 200 64 194 1
62 XEQ 01
63 "Ø1Ø"         128 240 16 7 222 48
64 XEQ 01
65 13
66 XEQ 02
67 32
68 ACCOL
69 19
70 ACCOL
71 12
72 ACCOL
73 ACCOL
74 "Ø'øz"        48 96 193 131 4 8
75 XEQ 01
76 " ØÃøå"       32 64 129 2 4 8
77 XEQ 01
78 " '           32 96 178 4 8 16
øåØ"
79 XEQ 01
80 "ØÃøåØ"       64 129 2 4 8 16
81 XEQ 01
82 "⊿ Bµ "       124 161 66 133 12 32
83 XEQ 01
84 1
85 SKPCHR
86 ADV
87 " ' ♦♦"       128 192 96 32 0 0
88 XEQ 01
89 34
90 SKPCOL
91 "αi♦♦♦"       4 49 140 0 0 0
92 XEQ 01
93 1
94 XEQ 02
95 127
96 ACCOL
97 1
98 SKPCOL
99 "SNOOPY"
100 ACA
101 1
102 SKPCOL
103 126
104 ACCOL
105 3
106 SKPCOL
107 ADV
108 7
109 ACCOL
110 8
111 ACCOL
112 "------"
113 ACA
114 6
115 ACCOL
116 7
117 ACCOL
118 3
119 SKPCOL
120 ADV
121 ADV
122 STOP
123♦LBL 01
124 ASTO X
125 ACSPEC
126 RTN
127♦LBL 02
128 SKPCOL
129 ADV
130 END
```

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 rechnereigenen 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 geshiftete 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 Programmlistings geändert werden.

```
01♦LBL "CAT"      13♦LBL 64      25 CLA         37 STO ↑
02 CF 21          14 201         26 GTO 02      38 "↳ R/S"
03 "CAT _"        15 GTO 01      27♦LBL 01      39 AVIEW
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
```

```
17 F110
28 F120 (SPACE)
32 F71FF00000237048
```

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.

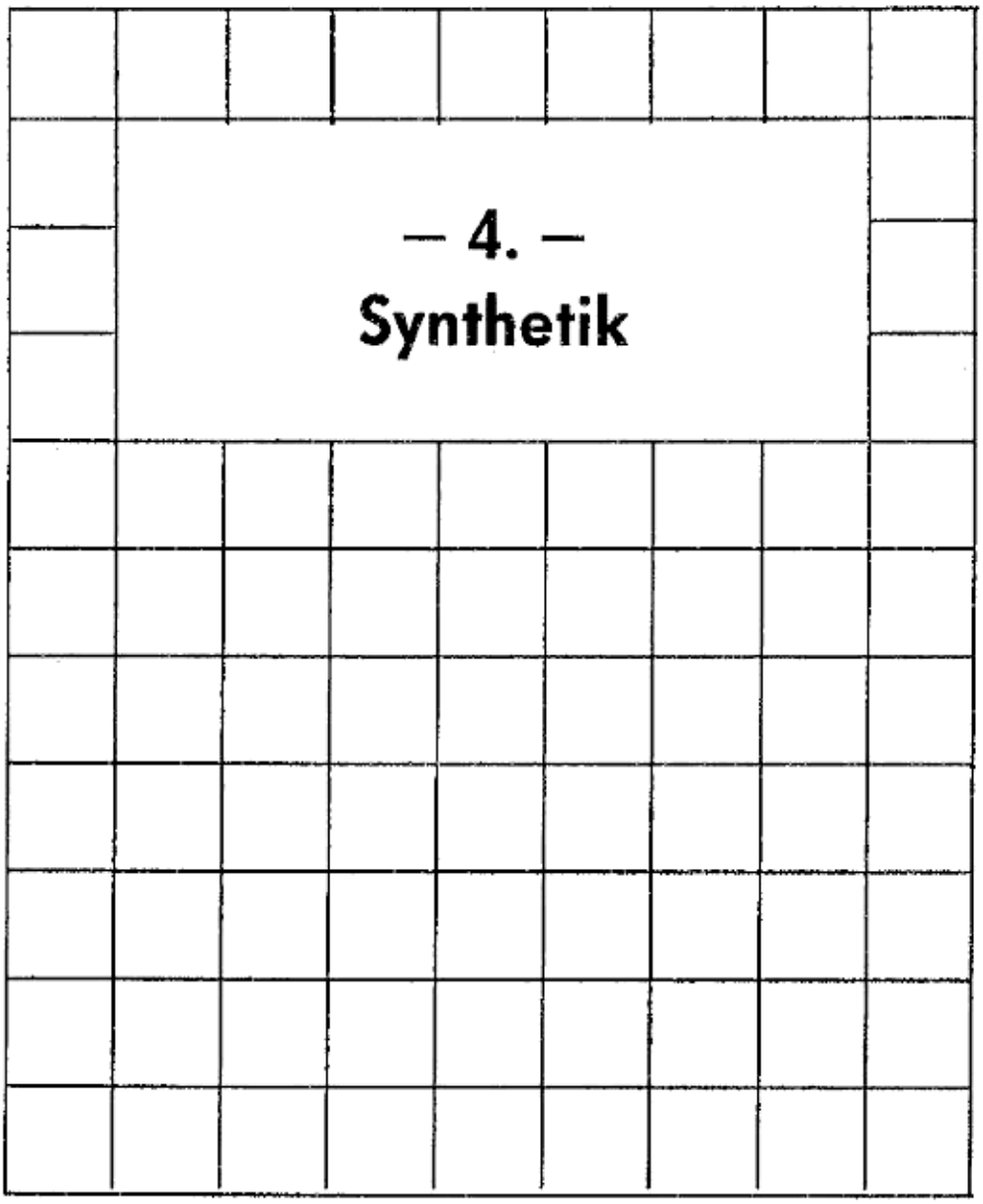
Zeilen 16-31: Die Katalogadresse wird hergestellt und ins X-Register gebracht.

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 angeschlossenem Drucker funktioniert. Allerdings sollte man bei einem CAT-Listing erst nach der R/S-Aufforderung den Drucker in den TRACE-MODE schalten.


– 4. –
Synthetik

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 "ABCDEFGHIJKLMNO" 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 .000 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 Orientierung 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 Adressen Postfix.

Zeilen A0 - A7: Dies sind die XROM-Nummern, mit denen der Rechner Module und Peripherie-Geräte ansteuert. Die Befehle haten 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, 01 - 0F) 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 B0 wird im Programmspeicher als GTO 15 angezeigt, während des Programmlaufs bewirkt er aber nichts.

Zeilen C0 - 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

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NULL 00 0	LBL 00 01 1	LBL 01 02 2	LBL 02 03 3	LBL 03 04 4	LBL 04 05 5	LBL 05 06 6	LBL 06 07 7	LBL 07 08 8	LBL 08 09 9	LBL 09 10 10	LBL 10 11 11	LBL 11 12 12	LBL 12 13 13	LBL 13 14 14	LBL 14 15 15	0
1	0 16 16	1 17 17	2 18 18	3 19 19	4 20 20	5 21 21	6 22 22	7 23 23	8 24 24	9 25 25	. 26 26	EEX 27 27	NEG 28 28	GTO 29 29	XEQ 30 30	W 31 31	1
2	RCL 00 32 32	RCL 01 33 33	RCL 02 34 34	RCL 03 35 35	RCL 04 36 36	RCL 05 37 37	RCL 06 38 38	RCL 07 39 39	RCL 08 40 40	RCL 09 41 41	RCL 10 42 42	RCL 11 43 43	RCL 12 44 44	RCL 13 45 45	RCL 14 46 46	RCL 15 47 47	2
3	STO 00 48 48	STO 01 49 49	STO 02 50 50	STO 03 51 51	STO 04 52 52	STO 05 53 53	STO 06 54 54	STO 07 55 55	STO 08 56 56	STO 09 57 57	STO 10 58 58	STO 11 59 59	STO 12 60 60	STO 13 61 61	STO 14 62 62	STO 15 63 63	3
4	+ 64 64	− 65 65	× 66 66	/ 67 67	X<Y? 68 68	X>Y? 69 69	X≤Y? 70 70	Σ+ 71 71	Σ− 72 72	HMS+ 73 73	HMS− 74 74	MOD 75 75	% 76 76	%CH 77 77	P→R 78 78	R→P 79 79	4
5	LN 80 80	X↑2 81 81	SQRT 82 82	Y↑X 83 83	CHS 84 84	E↑X 85 85	LOG 86 86	10↑X 87 87	E↑X-1 88 88	SIN 89 89	COS 90 90	TAN 91 91	ASIN 92 92	ACOS 93 93	ATAN 94 94	→DEC 95 95	5
6	1/X 96 96	ABS 97 97	FACT 98 98	X≠0? 99 99	X>0? 100 100	LN1+X 101 101	X<0? A 102	X=0? B 103	INT C 104	FRC D 105	D→R E 106	R→D F 107	→HMS G 108	→HR H 109	RND I 110	→OCT J 111	6
7	CLΣ T 112	X<>Y Z 113	PI Y 114	CLST X 115	R↑ L 116	RDN M 117	LASTX N 118	CLX O 119	X=Y? P 120	X≠Y? Q 121	SIGN R 122	X≤0? S 123	MEAN T 124	SDEV U 125	AVIEW V 126	CLD W 127	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	RAD IND 01 129	GRAD IND 02 130	ENTER↑ IND 03 131	STOP IND 04 132	RTN IND 05 133	BEEP IND 06 134	CLA IND 07 135	ASHF IND 08 136	PSE IND 09 137	CLRG IND 10 138	AOFF IND 11 139	AON IND 12 140	OFF IND 13 141	PROMPT IND 14 142	ADV IND 15 143	8
9	RCL IND 16 144	STO IND 17 145	ST+ IND 18 146	ST− IND 19 147	ST× IND 20 148	ST/ IND 21 149	ISG IND 22 150	DSE IND 23 151	VIEW IND 24 152	ΣREG IND 25 153	ASTO IND 26 154	ARCL IND 27 155	FIX IND 28 156	SCI IND 29 157	ENG IND 30 158	TONE IND 31 159	9
A	XR 0-3 IND 32 160	XR 4-7 IND 33 161	XR8-11 IND 34 162	X12-15 IND 35 163	X16-19 IND 36 164	X20-23 IND 37 165	X24-27 IND 38 166	X28-31 IND 39 167	SF IND 40 168	CF IND 41 169	FS?C IND 42 170	FC?C IND 43 171	FS? IND 44 172	FC? IND 45 173	GTO XEQ IND IND 46 174	SPARE IND 47 175	A
B	SPARE IND 48 176	GTO 00 IND 49 177	GTO 01 IND 50 178	GTO 02 IND 51 179	GTO 03 IND 52 180	GTO 04 IND 53 181	GTO 05 IND 54 182	GTO 06 IND 55 183	GTO 07 IND 56 184	GTO 08 IND 57 185	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	B
C	GLOBAL IND 64 192	GLOBAL IND 65 193	GLOBAL IND 66 194	GLOBAL IND 67 195	GLOBAL IND 68 196	GLOBAL IND 69 197	GLOBAL IND 70 198	GLOBAL IND 71 199	GLOBAL IND 72 200	GLOBAL IND 73 201	GLOBAL IND 74 202	GLOBAL IND 75 203	GLOBAL IND 76 204	GLOBAL IND 77 205	X<>-- IND 78 206	LBL -- IND 79 207	C
D	GTO -- IND 80 208	GTO -- IND 81 209	GTO -- IND 82 210	GTO -- IND 83 211	GTO -- IND 84 212	GTO -- IND 85 213	GTO -- IND 86 214	GTO -- IND 87 215	GTO -- IND 88 216	GTO -- IND 89 217	GTO -- IND 90 218	GTO -- IND 91 219	GTO -- IND 92 220	GTO -- IND 93 221	GTO -- IND 94 222	GTO -- IND 95 223	D
E	XEQ -- IND 96 224	XEQ -- IND 97 225	XEQ -- IND 98 226	XEQ -- IND 99 227	XEQ -- IND100 228	XEQ -- IND101 229	XEQ -- IND102 230	XEQ -- IND103 231	XEQ -- IND104 232	XEQ -- IND105 233	XEQ -- IND106 234	XEQ -- IND107 235	XEQ -- IND108 236	XEQ -- IND109 237	XEQ -- IND110 238	XEQ -- IND111 239	E
F	TEXT 0 IND T 240	TEXT 1 IND Z 241	TEXT 2 IND Y 242	TEXT 3 IND X 243	TEXT 4 IND L 244	TEXT 5 IND M 245	TEXT 6 IND N 246	TEXT 7 IND O 247	TEXT 8 IND P↑ 248	TEXT 9 IND Q_ 249	TEXT10 IND 250	TEXT11 IND a 251	TEXT12 IND b 252	TEXT13 IND c 253	TEXT14 IND d 254	TEXT15 IND e 255	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 Programmiertechniken, 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 aus

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 Permanentspeicher 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, 00F, 0C0 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 00F (= 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 0C0 (= 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 0C0 (= 192), die nächsten werden in Register 0C1 (= 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 00, 01 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 ⊦ und e selbst so etwas wie Flags.

Jede Taste hat nun 2 Flags: Eines in Register ⊦ 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 OO und der Statistikregister auf. Allerdings muß man, um SIZE zu finden, erst die hexadezimale Position von R OO 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 OO 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. OO1

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 O3. 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 O3, +, LBL OO, das KA von + auf Taste 11. Jetzt noch ein letztes SST, und es dauert etwas, bis CLΣ erscheint; Grund: Der Rechner überspringt die Lücke im Speicher und landet dann im Register e, welches er als Programm auffaßt, und eben CLΣ 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 .OO4 und ASN "" 12 aus, wobei unsere 1 als Code für die Taste verschwindet und statt dessen das LBL O3 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 O5 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 O5 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 ℓc (??).

Mit [←] 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 [←]), 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 [←], 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.

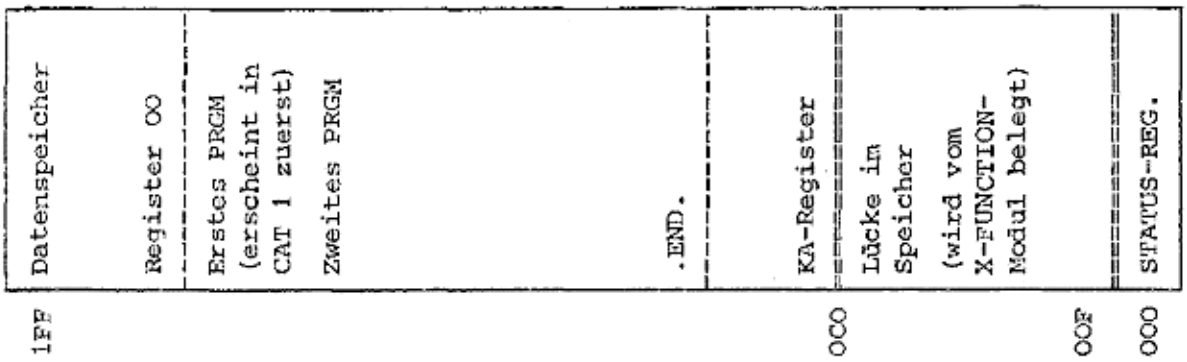
Auf diese Art und Weise kann man auch die Kurzform-STO und RCL-Befehle ausprobieren (also STO 00 - STO 15 und RCL 00 - RCL 15).

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.

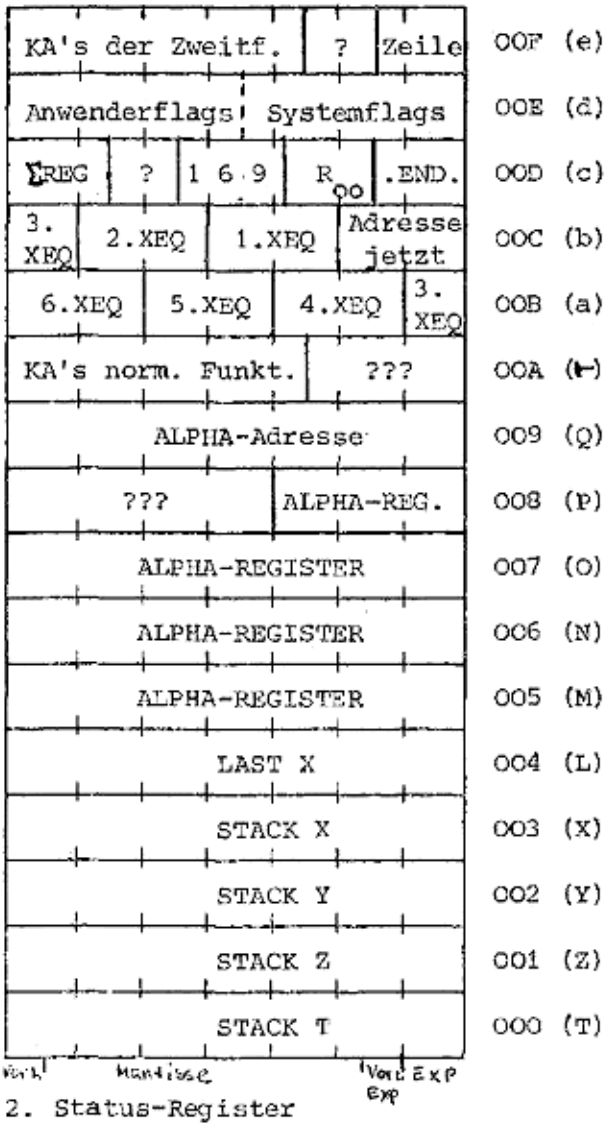
Nach diesen einfachen Experimenten müssen wir erst einmal erkunden, was der HP-41C überhaupt für Möglichkeiten bietet. Was ist z.B. ein Byte? Ein Byte besteht aus acht kleinen Schaltern, die entweder an- oder ausgeknipst werden können. Wir können nun diese 8 Bits als Binärzahl ansehen: Jedes dieser Bits 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.



1. Speicheraufbau im HP-41 CV/C+ Quad-RAM

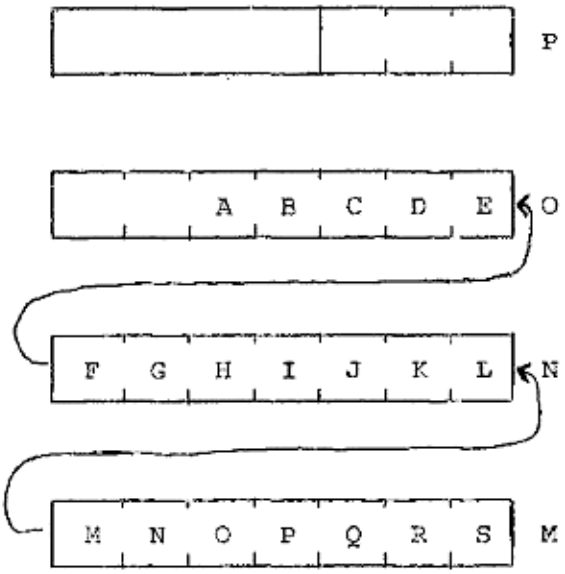


2. Status-Register

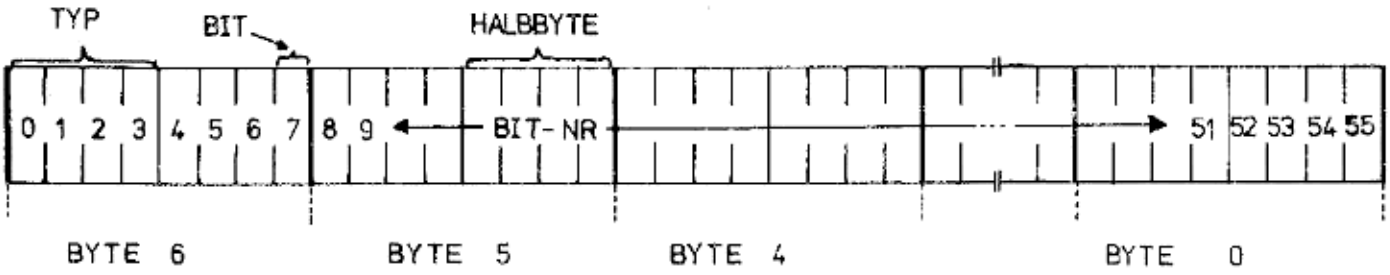
Zum .END.

240 (TEXT 0)	Byte 1
LBL 03 (nur CAT 3)	Byte 2
Funktion (z.B.SIN)	Byte 3
Taste	Byte 4
Byte 1 (Funktion)	Byte 5
Byte 2 (Adresse)	Byte 6
Taste	Byte 7

3. Aufbau eines KA-Registers



4. Das ALPHA-Register



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
⊢	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 numeriert: 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 x 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 Rechenzwerg 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 STOFLAG 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 STOFLAG 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 - Alphaketten 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.

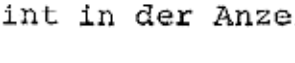
Der F7-Wolf

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 Knef sang: "... denn WOLF bleibt WOLF ...".

Erscheint in der Anzeige 

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, ␣␣DEFEKT, AVIEW, ALPHA
3. VIEW IND Z, +, BST, BST, WOLF, [←], SST, ASTO X, XEQ IND X
4. PRGM, RTN, R/S.

Nun zeigt der Rechner DEFEKT 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 herumdoktorn, 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 Vorsichtsmaßregeln!

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 einsames, 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 zwischendurch 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 Ziffereingabe 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 1,00000000 44 in der Anzeige steht -, erhält nach diesen 3 Schritten nur 1,0000000 40, da ja Flag 53 gelöscht wurde.

Soweit die Vorsichtsmaßregeln. 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 02 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	02	
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

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 02 ein Hex 02 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 E00 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 Alphazeichen, 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 umgewandelt und im Register d wieder zu einer Hexzahl zusammengeschoben. 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 - 029 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 bestimmt. Um das ganze nun nicht zu schwer zu machen, wird zur Oktalzahl 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 Rattenschwanz, 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 OO 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 ØØ ab - fertig. Dabei können diese 6 Register ganz oder teilweise außerhalb des Speichers liegen, was z.B. nach der Tastenfolge SIZE 319, 311, ΣREG IND X, SIZE O26 der Fall ist, und der Rechner reagiert auf Σ+, Σ- und CLΣ mit NONEXISTENT. Etwas komplizierter ist es da schon, die gegenwärtige Speichergröße zu bestimmen. Wählen wir auf einem einfachen C SIZE O24, dann haben wir 24 Datenspeicher (REG ØØ - 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 "Σ+S" bestimmt diese Positionen der Statistik-Register und die Speichergröße. Nach dem Programmstart mit XEQ "Σ+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 ØØ 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 Ø, 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 Ø, 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 der 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 Ziffereneingabe (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 "⊢♦♦♦♦x̄"
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

XEQ "K"
NONEXISTENT

PLNG "CD"
43 BYTES

```
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 FC?C IND T
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 ]
31 STO d
32 .END.
```

LBL'B3
END 56 BYTES

```
01♦LBL "DC"
02 OCT
03 L 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 "⊢ABC"
20 X<> \
21 CLA
22 X<> [
23 AVIEW
24 END
```

PLNG "DC"
53 BYTES

```
01♦LBL "DEMO"
02 "B♦0♦   ♦rÂ"
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 BLDSPEC
05 STO [
06 "⊢|"
07  E6
08 X<> b
09 LASTX
10 1
11 +
12 .END.
```

LBL'T2
END 29 BYTES

```
01♦LBL "Σ+$"
02 CLA
03 RCL c
04 X<> [
05 "⊢AAA"
06 X<> \
07 X<> [
08 XEQ 00
09 "AA"
10 RCL c
11 X<> [
12 X<> \
13 ASHF
14 XEQ 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 "⊢♦♦♦"
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+$
END 101 BYTES

ZEILE 30: F4 7F 00 00 03

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 00-14, 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 00, 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.

Der programmierbare Byte-Jumper

Rolf Mach (75)

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^{T}XYZ
1F 07 ergibt W^{T}TUVWXYZ

Dabei steht TUVWXYZ für irgendeine Alpha-Kette, da dieser Text ein Abbild der folgenden Bytes ist. Beispielsweise wird hinter GTO^{T}XYZ LN,X^{2} und X geschrieben. Das ergibt dann GTO^{T}PQR, 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 "a"
04 TONE 9
05 STOP
06 BEEP
07 .END.
```

Zeile 03 ist Hex 1D 03 und steht in der Anzeige als GTO^{T}■■■:
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 verharrt der Rechner einen kurzen Moment auf Zeile 03 (er sucht LBL^{T}■■■, wobei ^{T}■■■ = 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 *
01♦LBL "AB"
02 SF 25
03 XEQ ""
04 "              3↑↑"
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♦ãCv♦ü08ä"
20 STO 01
21♦LBL 08
22 SF 25
23 GTO ""
24 "'uv!h80          0
-
25 DEG
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 "8ur"
46 GTO 00
47 RCL I
48 840
49 /
50 RCL \
51 6
52 *
53 *
54 TONE 9
55 STOP
56 GTO 07
57 END
```

DECODIERTE VERSION, SVEN

```
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 [
19 -
20 6
21 /
22 STO \
23 .006
24 STO 01
25♦LBL 08
26 RCL [
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+ I
53 ISG 01
```

Zur Anwendung des PBJ
ein Beispielprogramm von
Michael Fehse:

Seit der Beseitigung des PRIVATE schien das "geschützte" Programm ausgestorben. 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)

```
                XEQ "AB"
F ?
X                   RUN
(a,b) ?
          3.0000 ENTER↑
          0.0000    RUN
               VIEW X
               4.5000
```

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

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

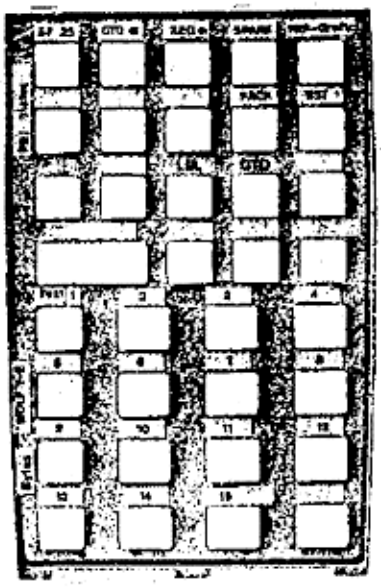
Sprung über "LBL 10" (kein ?...)

Sven

PBJ-Status

Hagen Klemp (73)

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



Erklärung des Status:

SF 25: Normale Standardfunktion

GTOα
XEQα
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:

PRGM	SF 25	PBJ Ø1	TEXT 1...15	PRGM

Der PBJ steht im Programm. Das SF 25 verhindert ein NONEXISTENT. Dann kommt der PBJ für ein Byte (PBJ Ø1) 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 0
55 STO d
56 ENTER↑
```

Eingabe:

		Ø,ØØØØ	Hex	
GTO .Ø54		54 Ø		
PRGM on		55 SF 25		Fehlerflag setzen
SF 25	(Σ+)	56 ᵀ⁻ ∠	F2 ØØ 29	3.Byte = 1.Byte PBJ
GTOα	(1/X)	57 ◆LBL ØØ	Ø1	= 2.Byte PBJ
LBL ØØ		58 RCL IND Y	9Ø F2	2.Byte = Textbyte
Bytes 2	(7)	57 LBL ØØ		
BST		56 ᵀ⁻ ∠		
BST		55 SF 25		
BST		56 ᵀ⁻2⁻⁻⁻⁻■⁻		2 Byte (F2 ØØ) auf-fressen
WOLF 2	(■ 7)	55 SF 25		
←		56 GTOᵀ■		
SST		56 GTOᵀ■		1 Byte (9Ø) auffressen
PACK	(COS)	57 ᵀ⁻W⁻⁻⁻⁻■		1 Byte (9Ø) auffressen
WOLF 1	(■ -)	56 GTOᵀ⁻		
←		56 GTOᵀ■		
PACK	(COS)			

---FERTIG---

PRGM:

	Hex
:	
54 Ø	1Ø
55 SF 25	A8 19
56 GTOᵀ■	29 Ø1
57 ᵀ■ Σ	F2 91 7E
58 ENTER↑	83
:	

```
54 0
55 SF 25
56 GTO ""
57 "Σ"
58 ENTER↑
```

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"▣	29 Ø1
57 "▣Σ	F2 91 7E
58 ENTER↑	83
⋮	

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

```
54 0
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
⋮
```

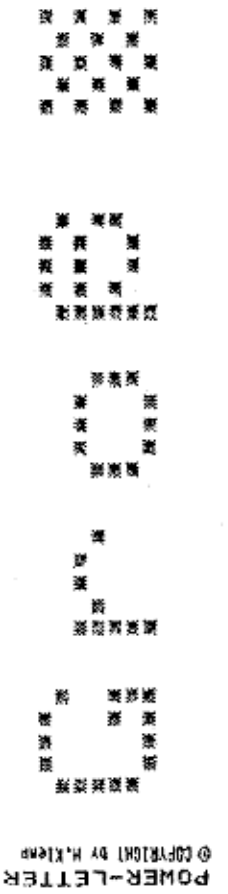
Sprung über X Bytes

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.

Eingaben ins KA-Programm: 15 PAIRS

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



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 Programmlauf 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 Programmlauf keine Änderung im Programm mehr erfolgen (vor Programmlauf 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

Funktion:	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	/
PRPLOTP	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
R02=  28,500
0
R03= "*REG3*
"
R04= -523,22
24
R05= -0,0023
R06=  4,0000
```

Normal, groß

```
R 3,1416R 3,
0000R 28,500
0R*REG3*R-52
3,2224R-0,00
  23R 4,0000
```

Gestaucht, groß

```
R00= 3,1416
R01= 3,0000
R02= 28,5000
R03= "*REG3*"
R04= -523,2224
R05= -0,0023
R06= 4,0000
```

Normal, klein

```
R 3,1416R 3,0000R 28,500
0R*REG3*R-523,2224R-0,00
          23R 4,0000
```

Gestaucht, klein

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.

```
01*LBL "83"
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 FC?C IND
                T
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
```

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.

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 Flagg 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 kreierten Zeile stehen möglicherweise noch "+'s", gefolgt von XEQ "LB". Sollen wietere 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 kreierten Zeile, "q" ist die zu löschende Zeilenzahl hinter der kreierten Zeile.

7. Wenn wir einen Fehler in der Eingabe bemerken und bereits R/S gedrückt haben, brauchen wir nur XEQ 03 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 01 daran erinnert. Ist das Byte-Loading bereits beendet, haben aber LB durch SST noch nicht verlassen, so hat XEQ 01 die gleiche Wirkung wie XEQ 03.

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 00-05 benutzen, aber nicht die Register 06-11, weil die von LB intern verwendet werden.

Beispiel zum Gebrauch:

Nehmen wir an, wir wollen eine Text-Zeile kreieren, die so aussieht: ᵀ⊢ ‾Ħ⊬⊞ (das sind 4 Zeichen, die wir anhängen (Append) wollen, mit den Hex-Codes 00, 01, 0C und 40), 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	00	01	0C	40	(hex), oder
245	127	0	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, 0, 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, 00, R/S, 01, R/S, 0C, R/S, 40, 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 006". Wir drücken SST (die Zahl 6,002 in X verdient Beachtung), gehen in den PRGM-Modus und führen DEL 006 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 generelle 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 00p".

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 00p", 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 druch 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 geshiftete 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 eingegen 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 verwenden, 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 Σ-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 mod X)/X (der Quotient) und Y 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.

Programm	Zeile	Synthetischer Code (hex)
DC	74	Text: F5 1F FO 01 69 01
LB	80	3 Byte GTO 10: DO 00 0A
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 0A
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 (Dedimal 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 "⊢*"
23 X<> ]
24 X<>Y
25 STO \
26 X<> ↑
27 "⊢*"
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 "⊢◆↓◆ᵀ◆◆
     "
41 X<> [
42 X<> \
43 X<> [
44 "⊢◆◆B"
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 "⊢◆◆◆◆◆"
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 "⊢**"
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
LBLᵀDC
LBLᵀ2D
LBLᵀOM
LBLᵀVA
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 (1981), Heft 2 (März/April), Seite 34-49. Blatt 1 von 2.

```
01◆LBL 00
02 STOP
03 GTO "++"

04◆LBL "LB"
05 FS? 50
06 GTO 00
07 "DEC/HEX
      INPT"
08 XEQ "VA"
09 CF 08
10 GTO 13

11◆LBL "L-"
12 CLA
13 XEQ "VA"
14 CF 08
15 RCL a
16 STO [
17 RCL b
18 FS? 08
19 GTO 14
20 STO \
21 SF 08

22◆LBL 13
23 RCL 11
24 CLST
25 STO 06
26 SIGN
27 ENTER↑
28 ENTER↑
29 R↑
30 GTO "++"

31◆LBL 00
32 RCL b
33 FC? 08
34 GTO 14
35 CLD
36 X<> [
37 STO a
38 X<> \
39 X<> b

40◆LBL 14
41 STO [
42 "⊢******"
43 X<> \
44 XEQ "2D"
45 2
46 /
47 INT
48 LASTX
49 FRC
50 512
51 *
52 RCL [
53 +
54 7
55 *
56 +
57 115
58 X<>Y
59 -
60 7
61 XEQ "QR"
62 ST- Z
63 X<>Y
64 CHS
65 STO 09
66 X<> Z
67 LASTX
68 XEQ "QR"
69 1,001
70 ST* 09
71 ST+ Y
72 FRC
73 ST* T
74 X<>Y
75 R↑
76 +
77 *
78 X<>Y
79 X<=0?
80 GTO 10
81 ST+ 09
82 7
83 *
84 +
85 STO 07
86 XEQ "OM"
87 X<> c
88 STO 10
89 CLST

90◆LBL 06
91 STO 11
92 CLA

93◆LBL 07
94 ASTO 08
95 X<>Y
96 ISG 06

97◆LBL 15
98 SF 09
99 FS? 08
100 RTN
101 CF 22
102 CF 23
103 FIX 0
104 CF 29
105 "#"
106 ARCL 06
107 "⊢ OF "
108 ARCL 07
109 "⊢?"
110 XEQ "VA"
111 TONE 7
112 STOP
113 FS? 48
114 GTO 14
115 FC? 22
116 GTO 19
117 GTO 08

118◆LBL 14
119 FC? 23
120 GTO 19
121 XEQ "XD"

122◆LBL 08
123 X<0?
124 GTO 03
125 ENTER↑
126 CLA
127 ARCL 08
128 XEQ "DC"
129 RCL 06
130 X<=0?
131 GTO 10
132 7
133 MOD
134 X≠0?
135 GTO 07
136 X<>Y
137 RCL 09
138 RCL 10
139 X<> c
140 RCL [
141 STO IND
          Z
142 X<>Y
143 X<> c
144 R↑
145 RCL 08
146 DSE 09
147 GTO 06
148 ISG 09

149◆LBL 20
150 CF 09
151 CLX
152 X<>Y
153 RCL 07
154 FRC
155  E3
156 *
157 AOFF
158 FIX 0
```

Programm-Gruppe LB (Load Bytes). Benötigt Unterprogramm-Gruppe DC (Decimal to Character). Erschienen im PPC Calculator Journal, Jahrgang 8 (1981), Heft 2 (März/April), Seite 34-39. Blatt 2 von 2.

```
159 "SST, DE
      L 00"
160 ARCL X
161 FIX 3
162 XEQ "VA"
163 BEEP
164 GTO 00

165◆LBL 01
166 RCL 06
167 X>0?
168 GTO 09

169◆LBL 10
170 "SST, MO
     RE +'S"
171 XEQ "VA"
172 TONE 3
173 GTO 00

174◆LBL 03
175 "*CORREC
     TION*"
176 XEQ "VA"
177 TONE 6
178 FC? 09
179 GTO 01
180 DSE 06
181 GTO 14
182 ISG 06
183 GTO 10

184◆LBL 14
185 RCL 06
186 7
187 MOD
188 X=0?
189 ISG 09
190 GTO 14
191 RCL 11
192 X=Y?
193 GTO 13
194 STO 08
195 RDN
196 STO 11
197 GTO 09

198◆LBL 13
199 TONE 4
200 6
201 ST- 06
202 R↑
203 GTO 15

204◆LBL 14
205 CLA
206 ARCL 08
207 ARCL 10
208 CLX
209 X<> \
210 STO [
211 ASTO 08

212◆LBL 09
213 RDN
214 GTO 15

215◆LBL "-B"
216 FC? 08
217 GTO 15
218 FS? 09
219 GTO 08

220◆LBL 19
221 RCL 06
222 X<=0?
223 GTO 10
224 CHS
225 ISG X
226 7
227 MOD
228 X=0?
229 GTO 14
230 CLA
231 ARCL 08

232◆LBL 11
233 "⊢◆"
234 DSE X
235 GTO 11
236 X<> [

237◆LBL 14
238 RCL 09
239 X<>Y
240 RCL 10
241 X<> c
242 X<>Y

243◆LBL 12
244 STO IND
        Z
245 CLX
246 DSE Z
247 GTO 12
248 RDN
249 X<> c
250 RDN
251 GTO 20

252◆LBL "XD"
253 "⊢◆Δ"
254 RCL [
255  E2
256 XEQ "QR"
257 29
258 ST- Z
259 -
260 ,9
261 ST* Z
262 *
263 INT
264 X<>Y
265 INT
266 16
267 *
268 +
269 RTN

270◆LBL "QR"
271 X<>Y
272 STO ]
273 X<>Y
274 MOD
275 ST- ]
276 LASTX
277 ST/ ]
278 CLX
279 X<> ]
280 X<>Y
281 END
          CAT 1
LBL"LB
LBL"L-
LBL"-B
LBL"XD
LBL"QR
END
558 BYTES
105 = <HEX>
F123
233 =<HEX
          CLA
233 = <HEX>
F27F00
253 = <HEX>
F37F0008
```

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"
02 CF 25
03 RCL 11
04 CF 07
05 SF 09

06◆LBL 01
07 XEQ "LF"
08 STO 09
09  E
10 +
11 X<>Y
12 STO 10
13 ASTO 11
14 DSE Y
15 GTO 07
16 SF 20
17 FC?C 09
18 GTO 13

19◆LBL 02
20 RCL 09
21 INT
22 LASTX
23 FRC
24  E3
25 *
26 X<>Y
27 ,5
28 FC? 10
29 SIGN
30 -
31 -
32 "REG FRE
       E: "
33 RCL d
34 FIX 1
35 ARCL Y
36 STO d
37 XEQ "VA"
38 TONE 6
39 PSE

40◆LBL 03
41 "PRE↑POS
    T↑KEY"
42 CLST
43 XEQ "VA"
44 TONE 7
45 STOP
46 GTO 14

47◆LBL "1K"
48 CF 20
49 SF 07

50◆LBL "+K"
51◆LBL 14
52 STO 08
53 RDN
54 STO 07
55 RDN
56 STO 06
57 CF 09
58 RCL 10
59 SIGN
60 FS? 20
61 X≠0?
62 GTO 01

63◆LBL 13
64 RCL 08
65 INT
66 X=0?
67 FS? 07
68 FC?C 20
69 GTO 02
70 X<0?
71 SF 09
72 ABS
73 STO Z
74 44
75 -
76 ABS
77 2
78 X<Y?
79 DSE T
80 R↑
81 STO Y
82  E1
83 ST/ Z
84 MOD
85 8
86 *
87 ENTER↑
88 CF 08
89 LASTX
90 FS? 09
91 ST+ Y
92 R↑
93 INT
94 X≠0?
95 X>Y?
96 GTO 08
97 R↑
98 +
99 ST+ Z
100 X<>Y
101 X<=Y?
102 CLX
103 X≠0?
104 SF 08

105 +
106 36
107 -
108 X>0?
109 GTO 08
110 FC? 09
111 RCL "
112 FS? 09
113 RCL e
114 FC? 08
115 GTO 14
116 STO [
117 "⊢*"
118 X<> [

119◆LBL 14
120 X<> d
121 FS? IND
          Y
122 GTO 09
123 SF IND Y
124 X<> d
125 FC? 08
126 GTO 14
127 STO [
128 ARCL 10
129 X<> \

130◆LBL 14
131 FC? 09
132 STO "
133 FS?C 09
134 STO e
 135 "**"
136 FS? 10
137 ARCL 11
138 X<> Z
139 RCL 07
140 RCL 06
141 XEQ "DC"
142 XEQ "DC"
143 XEQ "DC"
144 FS?C 10
145 GTO 14
146 "⊢◆◆◆"
147 ASTO 11
148 SF 10

149◆LBL 14
150 RCL 09
151 RCL 10
152 X<> c
153 RCL [
154 STO IND
          Z
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
159 FC? 10
160 ISG 09
161 SF 20
162 FS? 07
163 RTN
164 FS? 20
165 GTO 03
166 "DONE, N
     O MORE"
167 SF 09
168 GTO 14

169◆LBL 07
170 "NO ROOM
         "
171 CF 20
172 CLST

173◆LBL 14
174 FS?C 25
175 RTN
176 XEQ "VA"
177 TONE 7
178 TONE 3
179 STOP
180 GTO 01

181◆LBL 08
182 "NO SUCH
       KEY"
183 GTO 14

184◆LBL 09
185 X<> d
186 "KEY TAK
        EN"

187◆LBL 14
188 CF 09
189 CF 20
190 FS?C 25
191 RTN
192 XEQ "VA"
193 TONE 3
194 PSE
195 "KEYCODE
          ?"
196 CLST
197 RCL 08
198 XEQ "VA"
199 TONE 7
200 STOP
201 STO 08
202 GTO 01

203◆LBL "LF"
204 XEQ "E?"
205 17
206 -
207  E3
208 /
209 177
210 +
211 XEQ "OM"
212 "*0*◆◆**
             "
213 ,
214 ENTER↑
215 DSE T
216 GTO 14

217◆LBL 00
218 X<> IND
           T
219 X=Y?
220 GTO 14
221 X<> [
222 "⊢0"
223 STO \
224 ARCL X
225 RDN
226 RCL \
227 X<> IND
           T
228 ISG T
229 GTO 00

230◆LBL 14
231 X<> [
232 ARCL X
233 X<> \
234 SF 10
235 X=Y?
236 DSE T
237 CF 10
238 X<> Z
239 X<> c
240 R↑
241 RTN

242◆LBL "E?"
243 RCL c
244 XEQ "2D"
245 16
246 MOD
247 LASTX
248 X↑2
249 *
250 RCL [
251 +
252 CLA
253 END
```

```
          CAT 1
LBL'MK
LBL'1K
LBL'+K
LBL'LF
LBL'E?
END
535 BYTES
135 =
F32A2AF0
146 =
F47F000000
212 =
F72A102A0000
2A2A
```

– 5. –
Mathematik

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

PLNG "BU"

313 BYTES

```
R00= "used  "
R01= "B in  "
R02= "B out "
R03= "used  "
```

Gleichungen 4. Grades

Das Programm löst algebraisch die Gleichung

$ax^4+bx^3+cx^2+dx+e=0$ mit $a,b,c,d,e \leftarrow R$

$x \leftarrow R$ und $a \neq C$

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).

Schritt	Instruktionen	Eingabe	Anzeige
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

$x^4+x^4-\frac{25}{4}x^2-\frac{1}{4}x+1,5=0$ mit a=1, b=1, c=-6,25, d=-0,25, e=1,5

Lsg.: X1=-3, X2=-0,5, X3=0,5, X4=2

$x^3-8x^2+19x-12=0$ mit a=1, b=8, c=19, d=-12, e=0

Lsg.: X1=0 nicht zu berücks., X2=1, X3=3, X4=4

```
01  LBL"ROOT"
02  "a=?"
03  PROMPT
04  STO 03
05  STO 02
06  STO 01
07  STO 00
08  "b=?"
09  PROMPT
10  STx 00
11  "c=?"
12  PROMPT
13  STx 01
14  "d=?"
15  PROMPT
16  STx 02
17  "e=?"
18  PROMPT
19  STx 03
20  RCL 02
21  STO 07
22  X↑2
23  RCL 00
24  STx 07
25  X↑2
26  RCL 03
27  x
28  +
29  RCL 01
30  CHS
31  STO 06
32  RCL 03
33  4
34  x
35  ST- 07
36  x
37  +
38  CHS
39  STO 08
40  RCL 06
41  3
42  /
43  STO 06
44  RCL 07
45  x
46  RCL 08
47  -
48  2
49  /
50  RCL 06
```

< 1/X

```
51  3
52  Y↑X
53  -
54  STO 05
55  X↑2
56  RCL 07
57  3
58  /
59  RCL 06
60  X↑2
61  -
62  3
63  Y↑X
64  +
65  X>0?
66  GTO 00
67  CHS
68  SQRT
69  RCL 05
70  R-P
71  3
72  1/X
73  Y↑X
74  X<>Y
75  3
76  /
77  X<>Y
78  P-R
79  X<>Y
80  3
81  SQRT
82  x
83  2
84  /
85  STO 07
86  CHS
87  STO 08
88  X<>Y
89  ST- 08
90  ST- 07
91  2
92  x
93  RCL 08
94  X>Y?
95  X<>Y
96  RDN
97  RCL 07
98  X<=Y?
99  X<>Y
100 GTO 02
```

```
101 LBL 00
102 SQRT
103 ST+ 05
104 CHS
105 2
106 x
107 RCL 05
108 +
109 ENTER↑
110 SIGN
111 X<>Y
112 ABS
113 3
114 1/X
115 Y↑X
116 x
117 RCL 05
118 SIGN
119 RCL 05
120 ABS
121 3
122 1/X
123 Y↑X
124 x
125 +
126 LBL 02
127 RCL 06
128 -
129 STO 07
130 2
131 /
132 STO 08
133 ENTER↑
134 X↑2
135 RCL 03
136 -
137 ENTER↑
138 ABS
139 +
140 2
141 /
142 SQRT
143 ST- 08
144 +
145 STO 06
146 RCL 00
147 2
148 /
149 STO 05
150 RCL 05
```

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

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 Programmlauf 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  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  RDN
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?  "+ 1"
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  SQRT  30  /  INT
1  +  3086  /  HMS  END
```

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

ACHTUNG: Die Zeiten, 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:

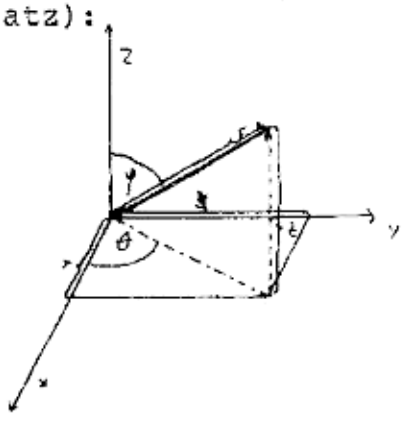
1. Bitte legt allen Programmen ein gedrucktes oder getipptes Listing bei.
2. Wer seine Programme, falls sie nicht veröffentlicht werden, wiederhaben will, der lege bitte Rückporto bei.
3. Wer mir einen ganz großen Gefallen tun will, füge beschriebene Magnetkarten bei (in diesem Fall das Rückporto nicht 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 VØ
   2. Betrag eines Vektors /V/
   3. a) Kreuzprodukt V1XV2
      b) Spatprodukt SPAT
      c) Skalarprodukt V1 ≡ V2
   4. Vektoraddition V1+V2
   5. Vektorsubtraktion V1-V2
   6. Volumen eines Vierflachs 4FLACH
   7. Winkel zwischen zwei Vektoren V1 V2
   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 1Ø 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 verwandt, deren Rechenzeiten einen mitunter nicht gerade vom Hocker hauen, wohingegen das folgende Programm die Lösung direkt berechnet.

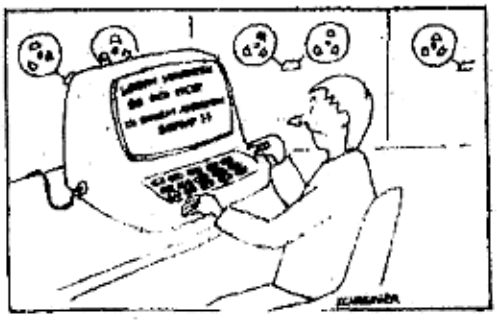
Nun noch eine Skizze zum VEKTOR-Programm (für weitere Formeln o.ä. fehlt mir der Platz):



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

```
LBL'VEKTOR
LBL'INPUT
LBL'V0
LBL'/V/
LBL'V1*V2
LBL'V1XV2
LBL'V1-V2
LBL'V1+V2
LBL'SPAT
LBL'4FLACH
LBL'∠ V1 V2
LBL'C-P
LBL'P-C
END          555 BYTES
```

```
CAT'6
   'INPUT 11
     'V0 12
    '/V/ 13
   'SPAT 14
 '4FLACH 15
'∠ V1 V2 23
    'C-P 24
    'P-C 25
  'V1-V2 51
  'V1+V2 61
  'V1*V2 71
  'V1XV2 81
```



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

$$a = \begin{pmatrix} 3 \\ 1 \\ -2 \end{pmatrix}$$

   Überprüfen Sie das Ergebnis mit der Funktion x V1 V2

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

```
                XEQ "INPUT"
N↑X↑Y↑Z
                1,00 ENTER↑
                5,00 ENTER↑
                3,00 ENTER↑
               -1,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                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  →  (13, -13, 26)
Y= -13,00
X=  13,00

                3,00  XEQ a
α               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    ***
b               XEQ "∠ V1 V2"
N1↑N2
                1,00 ENTER↑
                3,00    RUN
               90,00    ***
               XEQ "∠ V1 V2"
N1↑N2
                2,00 ENTER↑
                3,00    RUN
               90,00    ***
```

②

```
N               XEQ "/V/"
                3,00    RUN
               31,84    ***
```

③

```
                XEQ "V0"
N
                3,00    RUN
                     PRSTK

T=  0,82
Z=  0,82  →  (0,41, -0,41, 0,82)
Y= -0,41
X=  0,41
```

④

```
                XEQ "INPUT"
N↑X↑Y↑Z
                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  φ
Y=  18,43   θ
X=  3,74    r

                XEQ "INPUT"
N↑X↑Y↑Z
                2,00 ENTER↑
z-Achse         0,00 ENTER↑
                0,00 ENTER↑
                1,00    RUN
               XEQ "∠ V1 V2"
N1↑N2
                1,00 ENTER↑
                2,00    RUN
              122,31    ***

                XEQ "INPUT"
N↑X↑Y↑Z
Projektion      3,00 ENTER↑
von V auf       3,00 ENTER↑
XY-Ebene        1,00 ENTER↑
                0,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                4,00 ENTER↑
X-Achse         1,00 ENTER↑
                0,00 ENTER↑
                0,00    RUN
               XEQ "∠ V1 V2"
N1↑N2
                3,00 ENTER↑
                4,00    RUN
               18,43    ***
```

⑤

```
                XEQ "INPUT"
N↑X↑Y↑Z
                1,00 ENTER↑
                1,00 ENTER↑
                2,00 ENTER↑
                0,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                2,00 ENTER↑
                2,00 ENTER↑
                3,00 ENTER↑
                1,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                3,00 ENTER↑
                1,00 ENTER↑
                5,00 ENTER↑
                3,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                4,00 ENTER↑
                1,00 ENTER↑
Berechnung      2,00 ENTER↑
d. Seiten-      1,00    RUN
vektoren
                XEQ "V1-V2"
N1↑N2
                2,00 ENTER↑
                1,00    RUN
                2,00  XEQ a

                XEQ "V1-V2"
N1↑N2
                3,00 ENTER↑
                1,00    RUN
                3,00  XEQ a

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

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

Lösung der 6. Aufgabe:

Durch Gleichsetzen entsteht folgendes Gleichungssystem:

$$\begin{pmatrix}3\\7\\9\end{pmatrix} + \lambda_s \begin{pmatrix}-6\\1\\-5\end{pmatrix} = \begin{pmatrix}6\\3\\7\end{pmatrix} + \mu_s \begin{pmatrix}2\\-2\\-4\end{pmatrix} + \nu_s \begin{pmatrix}-1\\3\\-16\end{pmatrix}$$

Daraus folgt

$$\begin{pmatrix}3\\7\\9\end{pmatrix} - \begin{pmatrix}6\\3\\7\end{pmatrix} = \mu_s \begin{pmatrix}2\\-2\\-4\end{pmatrix} + \nu_s \begin{pmatrix}-1\\3\\-16\end{pmatrix} + \lambda_s \begin{pmatrix}6\\-1\\5\end{pmatrix}$$

$$\Rightarrow \quad \lambda_s = \frac{\left| \left[\begin{pmatrix}3\\7\\9\end{pmatrix} - \begin{pmatrix}6\\3\\7\end{pmatrix}\right] \begin{pmatrix}2\\-2\\-4\end{pmatrix} \begin{pmatrix}-1\\3\\-16\end{pmatrix} \right|}{\left| \begin{pmatrix}2\\-2\\-4\end{pmatrix} \begin{pmatrix}-1\\3\\-16\end{pmatrix} \begin{pmatrix}6\\-1\\5\end{pmatrix} \right|}$$

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

(6)

```
                XEQ "INPUT"
N↑X↑Y↑Z
                1,00 ENTER↑
                3,00 ENTER↑
                7,00 ENTER↑
                9,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                2,00 ENTER↑
                6,00 ENTER↑
                3,00 ENTER↑
                7,00    RUN

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

                XEQ "INPUT"
N↑X↑Y↑Z
                2,00 ENTER↑
                2,00 ENTER↑
               -2,00 ENTER↑
               -4,00    RUN

                XEQ "INPUT"
N↑X↑Y↑Z
                3,00 ENTER↑
               -1,00 ENTER↑
                3,00 ENTER↑
              -16,00    RUN
                XEQ "INPUT"
N↑X↑Y↑Z
                4,00 ENTER↑
                6,00 ENTER↑
               -1,00 ENTER↑
                5,00    RUN

                XEQ "SPAT"
N1↑N2↑N3
Zähler          1,00 ENTER↑
                2,00 ENTER↑
determinante    3,00    RUN
               20,00    ***
                    STO 34

                XEQ "SPAT"
N1↑N2↑N3
Nenner-         2,00 ENTER↑
                3,00 ENTER↑
determinante    4,00    RUN
              248,00    ***

                    ST/ 34
                    RCL 34
λr             0,08    ***
```

Durch Einsetzen von $\lambda_s = 0{,}08$ in die Ausgangsgleichung erhält man:

$$\vec{x}_s = \begin{pmatrix}2{,}54\\7{,}08\\8{,}53\end{pmatrix}$$

Der Schnittpunkt liegt bei

$S\,(2{,}54 \,|\, 7{,}08 \,|\, 8{,}53)$

282, 283-81

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

Programm Lottozahlengenerator:

```
01  LBL L
02  CF 27
03  FIX Ø
04  CF 29
05  1,ØØ6
06  STO Ø7
07  CLX
08  "QUELLZAHL ?"
09  PROMPT
10  SF 27
11  X=Ø?
12  GTO A
13  STO ØØ
14  LBL A
15  RCL Ø7
16  ENTER
17  CLX
18  LBL Ø2
19  STO IND Y
20  ISG Y
21  GTO Ø2
22  LBL Ø1
23  RCL ØØ
24  9821
25  x
26  Ø,211327
27  +
28  FRC
29  STO ØØ
30  5Ø
31  x
32  INT
33  X=Ø?
34  GTO Ø1
35  LBL Ø3
36  RCL Ø7
37  X<>Y
38  LBL Ø4
39  RCL IND Y
40  X=Ø?
41  GTO Ø5
42  X=Y?
43  GTO Ø1
44  RDN
45  ISG Y
46  GTO Ø4
47  LBL Ø5
48  RDN
49  STO IND Y
50  ISG Y
51  GTO Ø1
52  RCL Ø7
53  TONE Ø
54  LBL Ø6
55  6,ØØ6
56  X<>Y
57  X=Y?
58  TONE 9
59  RCL IND X
60  STOP
61  RDN
62  ISG X
63  GTO Ø6
64  LBL CLR
65  CF 27
66  SF 29
67  FIX 2
68  CLA
69  CLST
70  -
71  OFF
72  GTO CLR
73  END
```

Programmdokumentation

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'BK. 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

```
LBL'BK
FIX Ø
STO Ø7
STOP
STO Ø8
STO Ø9
LBL Ø8
RCL Ø9
-
X=Ø?
GTO 27
X>Ø?
GTO Ø8
RCL Ø9
+
x<> Ø9
GTO Ø8
RCL Ø9
+
x<> Ø9
GTO Ø8
LBL 27
RCL Ø7
XEQ A
STO Ø7
RCL Ø8
XEQ A
STO Ø8
CLA
ARCL Ø7
' ⊢ / '
APPEND
ARCL Ø8
PROMPT
GTO'BK
LBL A
RCL Ø9
+
RTN
END.
```

10 Register

Speicherbelegung:

Ø7: Z
Ø8: N
Ø9: ggT(Z,N)

Primfaktorzerlegung mit dem HP-41C Friedrich Hillebrandt (10)

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.

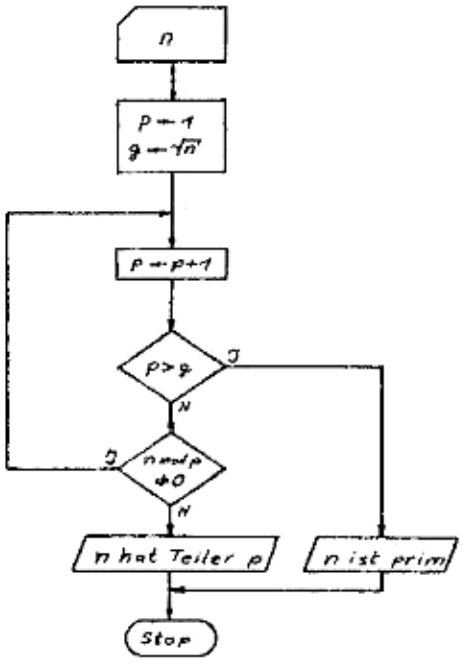
Ich habe nicht nur die Anregung, sondern auch viele Realisierungswege dem Werk von D.E. Knuth (The Art of Computer Programming, Vol. II, Ch. 4.5.4) entnommen.

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 SQRT
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 ARCL 01
23*LBL 01
24 BEEP
25 PROMPT
26 END
```



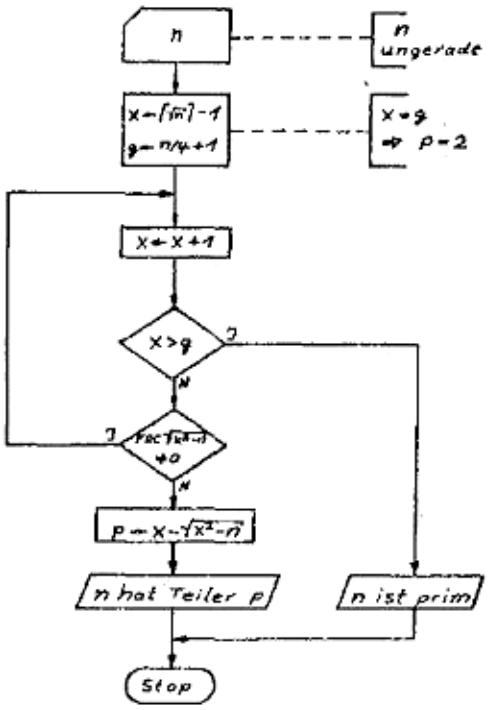
Algorithmus_F (Fermatsche Methode)

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 "SEK
02 STO 02
03 "IST PRIM"
04 VIEW 00
05 2
06 MOD
07 1/X
08 RCL 00
09 SQRT
10 RCL X
11 CHS
12 I
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 FRC
37 X=0?
38 GTO 00
39 RCL 01
40 LASTX
41 -
42 "HAT TEILER "
43 ARCL 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 $x = \lceil\sqrt{n}\rceil = \lceil\sqrt{1001423}\rceil = 1001$ 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 weiterfaktorisiert 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 Primteiler bis einschließl.		2	3	5	7	11	13	p
Bereich, den eine Schleife überdeckt	1	2	6	30	210	2310	30030	$\Pi = 2\cdot3\cdot5\cdot\ldots\cdot p$
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 x 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 mod 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 mod 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 mod 9 aufgeführt; das sind natürlich 9 Stück. Die Eintragungen in der Tabelle sind jeweils die Werte $x^2 - n$ mod 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**2-N MOD 9	X 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	8
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	1	1	1	5	2

Es ist im Algorithmus F also beispielsweise unsinnig, einen x-Wert zu testen, für den gilt: x mod 9 = 3 und gleichzeitig n mod 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:
$x \leftarrow \lceil \sqrt{n}/9 \rfloor \cdot 9$ und dann jeweils um 3 erhöhen.

n mod 9 = 1:
$x \leftarrow \lceil \sqrt{n}/9 \rfloor \cdot 9 + 1$ und dann abwechselnd um 7 und 2 erhöhen.

n mod 9 = 4:
$x \leftarrow \lceil \sqrt{n}/9 \rfloor \cdot 9 + 2$ und dann abwechselnd um 5 und 4 erhöhen.

n mod 9 = 7:
$x \leftarrow \lceil \sqrt{n}/9 \rfloor \cdot 9 + 4$ 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 = 109
N MOD 2 = 1
N MOD 72 = 37
N MOD 3 = 1
N MOD 48 = 13
N MOD 4 = 1
N MOD 36 = 1
N MOD 6 = 1
N MOD 24 = 13
N MOD 8 = 5
N MOD 18 = 1
N MOD 9 = 1
N MOD 16 = 13
N MOD 12 = 1

ANFANGSINCREMENT:

     1      ***

SCHLEIFENINCREMENTE:

    16      ***
    38      ***
    16      ***
     2      ***
    16      ***
    38      ***
    16      ***
     2      ***

------------
```

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

ANFANGSINCREMENT:

     5      ***

SCHLEIFENINCREMENTE:

     8      ***
    46      ***
     8      ***
    10      ***
     8      ***
    46      ***
     8      ***
    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 mod 3 = 1) oder 48 (n mod 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 α = 1,160 sec durchlaufen, während für die Schleife in F die Zeit β = 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 mod 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:

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

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

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.

$$f(p,n) = \frac{p(x) - p(x-B_m)}{30} - \frac{\beta}{\alpha}$$

$$= \left(\sqrt{\frac{n + p^2}{2p} + B_m{}^2 - n} + \frac{p^2 - n}{2p} - B_m\right) \cdot \frac{1}{30} - \frac{\beta}{\alpha}$$

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 Standardprogrammsammlung ergibt für
n mod 3 = 1: R^2 = 1,00000; a = 21,79642; b = 0,48023
und für
n mod 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 = <u>0,480</u> und k_2 = <u>0,557</u>

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

Ein "ungünstiger" Fall ist sicher 55697·179533 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"
02 CF 26
03 88
04 STO 17
05 2
06 ENTER↑
07 19
08 Y↑X
09 87
10 -

11◆LBL 01
12 XEQ "PRM
   "
13 RCL 00
14 1
15 +
16 DSE 17
17 GTO 01
18 SF 26
19 BEEP
20 END
```

```
524201
IS PRIME

524202
=2*3*7↑2
*1783

524203
IS PRIME

524204
=2↑2*29*4519

524205
=3↑3*5*11
*353

524206
=2*262103

524207
=113*4639

524208
=2↑4*3*67
*163

524209
=7*7490
```

```
5242↑
=7*74897

524280
=2↑3*3*5*17
*257

524281
=269*1949

524282
=2*11*23831

524283
=3*174761

524284
=2↑2*131071

524285
=5*23*47*97

524286
=2*3↑3*7*19
*73

524287
IS PRIME

524288
=2↑19
```

USERS' PROGRAM LIBRARY EUROPE
EUROPÄISCHE BENÜTZER-PROGRAMMBIBLIOTHEK
BIBLIOTHÈQUE EUROPÉENNE DE PROGRAMMES UTILISATEURS
LIBRERIA EUROPEA DEGLI UTILIZZATORI

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

Program Title (Programmtitel / Titre du programme / Titolo del programma): PRIMFAKTORZERLEGUNG

Category No. (Kategorie Nr. / Catégorie N° / Categoria N°): 201
Name (Rubrik / Rubrique / Nome della categoria): Zahlentheorie

No. of program lines (Anzahl Programmzeilen / Nombre de lignes de programme / N° de lignes de programma): 502

No. of data registers (Anzahl der benötigten Datenspeicher / Nombre de registres de données / N° di registri utilizzati): 116 (total)

Recommended HP 41C System configuration (Empfohlene System-Konfiguration / Configuration recommandée / Configurazione raccomandata):
Port #1: Memory Modul
Port #2:
Port #3: (Printer)
Port #4:

This program requires the following programs as subroutines: (Dieses Programm benutzt folgende Programme als Unterprogramme: / Ce programme utilise les programmes suivants comme sous-programmes: / Questo programma usa i seguenti programmi come subroutines:)

HP Applications ROM (HP Applikations ROM / ROM d'application HP / ROM di applicazione HP):

Program Names: (Programm / Noms de programmes / Programmi):

Program Abstract (Kurzbeschreibung / Résumé / Breve descrizione del programma):

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.

Die längste Rechenzeit ist ca. 46 min für die Primzahl $10^{10}-71$. 99991 · 100003 wird in ca. 31 min faktorisiert.

Name (Name / Nome): Friedrich Hillebrandt

Address (Strasse / Adresse / Indirizzo): Jülicher Straße 443

City (Ort / Lieu / Città): Aachen
Postal Code (Postleitzahl / Code postal / C.A.P.): 5100
Country (Land / Pays / Paese): Deutschland

ACKNOWLEDGMENT AND AGREEMENT
Erklärung und Ermächtigung / Déclaration et Autorisation / Dichiarazione e Autorizzazione

To the best of my knowledge, I have the right to contribute this program material without breaching any obligation concerning confidentiality or 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 meinen Programmbeitrag frei verfügen kann, ohne dass ich dadurch bei HP, anderen Programmbenutzern oder auch irgendwelchen Vertragspartnern gegenüber Dritten oder sonstige rechtliche Risiken eingehe. HP kann meinen Programmbeitrag ohne Gebührenhinweis und sonstige Verpflichtungen in jeder Weise benutzen oder verwerten. Gegen eine Veröffentlichung meines Namens in Zusammenhang mit meinem Programmbeitrag habe ich nichts einzuwenden.

Au mieux de ma connaissance, je déclare avoir le droit de fournir le présent programme sans enfreindre des obligations de secret à l'égard d'autres personnes ou organismes. Je le prends gracieusement à la bonne Hewlett-Packard S.A. (HP) qui me l'a demandé pour sa bibliothèque de programmes et j'autorise HP, qui pourra à son tour accepter d'autres personnes, à vendre, le reproduire, le modifier, le publier et le distribuer, sans obligations ni responsabilités d'aucune sorte. HP est autorisé à publier mes nom et adresse en tant qu'auteur du présent programme, en vue de faciliter les échanges d'informations avec les auteurs de ce programme.

Per quanto ne sia a conoscenza, ho il diritto di fornire questo programma senza violare alcun obbligo di segretezza o confidenzialità verso altre persone o organismi. Fornisco questo programma alla Hewlett-Packard S.A. (HP) senza alcun vincolo particolare per includerlo nella sua biblioteca di programmi e autorizzo la società HP, la quale a sua volta potrà usare, duplicare, modificare, pubblicare e distribuire questo programma senza alcun obbligo di responsabilità di alcuna specie. La società HP potrà, a sua discrezione, pubblicare il mio nome e indirizzo quale autore del programma onde facilitare le scambio di informazioni con gli utilizzatori del medesimo.

Date (Datum / Date / Data): 01.03.81
Signature (Unterschrift / Signature / Firma):

19-1954-1

USER INSTRUCTIONS

PROGRAMMABLAUF
INSTRUCTIONS D'EMPLOI
NORME OPERATIVE

Step Schritt Pas Passo	Instructions Operazione Instrucciones Instruzioni	Variables Datenangabe Données Dati	Function(s) Taste(n) Touche(s) Tasti	Result Resultat Resultat Risultato
1	Programm eingeben *			
2	Faktorisierung von n	n	XEQ "PRM"	n = $p_1^{e_1} * p_2^{e_2} \ldots$! oder IS PRIME
3	Faktorisierung von n mit Einzelanzeige der Primzahlpotenzen	n	SF 00 XEQ "PRM"	n = $p_1^{e_1}$ * $p_2^{e_2}$! 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 I

PROGRAMMBESCHREIBUNG I
DESCRIPTION DU PROGRAMME I
DESCRIZIONE DEL PROGRAMMA I

Application, Equations, Variables
Anwendung, Gleichungen, Veränderliche
Application, Equations, Variables
Application, Equations, Variables

Mit dem Programm wird für eine natürliche Zahl n ($n = \pi p^e$) 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 restrictions
Limitazioni 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.

PROGRAM DESCRIPTION II

PROGRAMMBESCHREIBUNG II
DESCRIPTION DU PROGRAMME II
DESCRIZIONE DEL PROGRAMMA II

Example
Beispiel
Exemple
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, noch beendigter Faktorisierung den Printer anzuschließen, bzw. einzuschalten und dann mit R/S (bzw. XEQ 10) einen Ausdruck zu bekommen.

```
220
=2↑2*5*11

284
=2↑2*71


34765731
=3↑2*7*11*13
*17*227

36939357
=3↑2*7*13*23
*37*53


2172649216
=2↑8*257
*33023

2181168896
=2↑8*8520191
```

```
7,000000000 ENTER↑
7,999999999   Y↑X
5764888,939   ***
XEQ 'PRM'
5764801
=7↑8

9970592519
=2143*2153
*2161

2 ENTER↑
17 ENTER↑
1   -
Y↑X
2 ENTER↑
17   Y↑X
1   -
*
8589869056
=2↑16*131071
```

```
2,00 ENTER↑
27,00   Y↑X
111,40   -
134217616,6   ***

134217617
IS PRIME

SF 00
5907942612
=2↑2
*3↑2
*11↑2
*13
*17↑2
*19↑2

RUN
5907942612
=2↑2*3↑2
*11↑2*13
*17↑2*19↑2
```

PROGRAM LISTING

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line	Keystrokes	Comments	Line	Key pressed	Comments
00	01*LBL "PRM"		51	CF 06 CF 07 RCL 07 3	
	SF 05 SF 12 CF 29			MOD GTO IND X	
	FIX 0 RND STO 00 2				
	STO 02 4 STO 04 +	Allgemeine		128*LBL 02	
05	STO 06 SIGN 11 STO 08	Initialisierung	55	RCL 07 4 MOD 3 X=Y?	
	+ STO 09 CLA ASTO 11			GTO 03 18 STO 01 6	
	ASTO 12 ARCL 00			SF 06 GTO 02	
				140*LBL 03	
10	SF 21 SF 25 PRA CF 21	Test, ob Printer	60	12 STO 01 SF 07	
	AVIEW	eingeschaltet		GTO 02	
				145*LBL 01	
	RCL 00 ENTER↑ ENTER↑			RCL 07 12 MOD ST+ X	
15	STO 07 RCL 02 MOD	Vorab-Test der	65	RCL 07 9 MOD +	
	X=0? XEQ IND 04 CLX 3	Faktoren p=2,		XEQ IND X GTO 02	
	MOD X=0? XEQ IND 04	3, 5 und 7			
	CLX 5 MOD X=0?	im Algor. E		156*LBL 03	
	XEQ IND 04 CLX 7 MOD			16 STO 01 2 RTN	
20	X=0? XEQ IND 04 XEQ 00		70		
				161*LBL 06	Berechnung
				10 STO 01 14 RTN	der
					Inkremente
				166*LBL 09	für
	52*LBL 00			8 STO 01 46 RTN	Algorithm. E
25	CLX RCL 04 LASTX +		75		
	MOD X=0? XEQ IND 04			171*LBL 15	
	CLX RCL 02 LASTX +	Testschleife		20 STO 01 16 RTN	(Kongruenz-
	MOD X=0? XEQ IND 04	für Algor. E			verh. mod 7!!)
	CLX RCL 04 LASTX +			176*LBL 18	
	MOD X=0? XEQ IND 04	(Kongruenz-		4 STO 01 32 RTN	
30	CLX RCL 02 LASTX +	verh. mod 30)	80		
	MOD X=0? XEQ IND 04			181*LBL 21	
	CLX RCL 04 LASTX +			28 STO 01 8 RTN	
	MOD X=0? XEQ IND 04	(Schleifenindex			
	CLX RCL 06 LASTX +	Reg 10 wurde		186*LBL 02	
35	MOD X=0? XEQ IND 04	spätestens	85	STO 02 RCL 05 RCL 01	
	CLX RCL 02 LASTX +	in Zelle 057		STO 03 ST+ X - RCL 02	
	MOD X=0? XEQ IND 04-	berechnet)		- STO 04 RCL 07 16	
	CLX RCL 06 LASTX +			MOD GTO IND X	
	MOD X=0? XEQ IND 04				
40	DSE 10 GTO 00		90	208*LBL 03	
				201*LBL 11	
				CF 07 RCL 01 X<> 02	
		Test, ob		STO 01 RCL 03 X<> 04	
	CLX LASTX X↑2 X>Y?	Faktorisierung		STO 03	
45	GTO 97	abgeschlossen	95		
				209*LBL 01	
				210*LBL 13	
				FS?C 06 GTO 11 RCL 02	
		Schleifenzähler		X<> 04 STO 02	
	ST+ Y CLX LASTX ST+ X	für Algor. F			
5	/ STO 10	(Rest Zeile 253)	101	216*LBL 05	
				217*LBL 07	
				218*LBL 09	
				219*LBL 15	

176'180—28

PROGRAM LISTING

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line	Keystrokes	Comments	Line	Key pressed	Comments
01	RCL 07 SQRT RCL X	Initialisierung	51	347*LBL 97	
	RCL 05 MOD - STO 06	von x^2-n		RCL 07 ENTER↑ SIGN	Faktorisierung
	X↑2 RCL 07 -			STO 07 X<>Y X≠Y?	ist
				XEQ 01 BEEP XEQ 02	abgeschlossen
06	RCL 04 2 / FS? 07	Anfangsinkrem	56	SF 21 GTO 02	
	CLX	für Algor. F			
				359*LBL 10	
	XEQ 07 X=0? GTO 01			SF 12 SF 21 CLR	
10	RDN RCL 01 XEQ 07	Vorab-Test	60	ARCL 00 AVIEW RCL 09	
	X=0? GTO 01 RDN	der ersten		E3 / E1 +	Anzeige-
	RCL 02 XEQ 07 X=0?	x-Werte			Routine
	GTO 01 RDN RCL 03			378*LBL 14	
	XEQ 07 X=0? GTO 01			CLA ARCL IND X ISG X	RTN in
15			65	ARCL IND X AVIEW ISG X	Zeile 384
	CLX RCL 10 RCL 06 -			GTO 14	ist RTN des
	RCL 05 / INT STO 10	Schleifenzähler			Gesamtpro-
	ISG 10	für Algor. F		378*LBL 02	gramms
20			70	RDN CLX CLD CF 12	
				CF 21 RTN GTO 10	
	262*LBL 12				
	CLX RCL 04 X↑2 RCL 06			386*LBL 04	
	LASTX ST+ 05 ST+ Y *			RCL 07 LASTX MOD X=0?	Faktor gefunden
25	* + ENTER↑ SQRT FRC		75	GTO 01 CLX LASTX	(Reduktion von
	X=0? GTO 01 CLX			ST/ 07 SIGN ST+ Y RDN	n, Hochzählen
	RCL 01 X↑2 RCL 06			GTO 04	der Exponenten)
	LASTX ST+ 06 ST+ X *	Testschleife			
	* + ENTER↑ SQRT FRC	für Algor. F		399*LBL 01	
30	X=0? GTO 01 CLX		80	"n" FS?C 05 "="	Herstellung
	RCL 02 X↑2 RCL 06			RCL 00 LASTX ARCL X	der
	LASTX ST+ 06 ST+ X *	(Kongruenz-		X=Y? "IS PRIME" SIGN	zusätzlichen
	* + ENTER↑ SQRT FRC	verh. mod 144)		R↑ X<=Y? GTO 01 "F1"	Anzeige
	X=0? GTO 01 CLX			ARCL X	
	RCL 03 X↑2 RCL 06				
35	LASTX ST+ 06 ST+ X *		85		
	* + ENTER↑ SQRT FRC			414*LBL 01	
	X=0? GTO 01 BSE 10			FS? 25 SF 21 FS? 00	
	GTO 12 GTO 97			AVIEW CF 21 ASTO Z	
				ASHF ASTO T CLA	"Anhängen"
40			90	ARCL IND 00 ARCL IND 09	des neuen
	326*LBL 07	Test-U-Pgm		ARCL Z ARCL T ASTO 03	Anzeigeteils
	X↑2 RCL 06 LASTX	für mögliche		ASHF ASTO 01 ASHF	
	ST+ 06 ST+ X * * *	negative		ASTO Y CLA ASTO X	
	ENTER↑ X<0? RTN SQRT	x^2-n Werte		X≠Y? GTO 01 RCL 03	
45	FRC RTN		95	STO IND 00 RCL 01	
				GTO 13	
	341*LBL 01	Faktor wurde			
	RCL 06 LASTX - FRC	mit Algor. F		441*LBL 01	
	XEQ 04	gefunden	100	XEQ 03 R↑ R↑ 2	neue Anzeige
				ST+ 00 ST+ 09 RDN	zu lang
				STO IND 00 X<>Y	

116

PROGRAM LISTING

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line	Keystrokes	Comments	Line	Key pressed	Comments
	451*LBL 12				
	STO IND 89 RCL IND 88	neue Anzeige			
	RCL IND 89 FC? 00	paßt ins			
	AVIEW	Anzeigeregister			
	457*LBL 00				
	LASTX STO 01 RCL 07				
	SQRT X<>Y - 38 /				
	RCL 07 3 MOD				
	XEQ IND X Rt * INT	Berechnung			
	STO 10 RCL 07 RCL 07	des neuen			
	RCL 07 RCL 01 ABS RTN	Schleifenindex			
	488*LBL 08	für Alger. E			
	481*LBL 01	und der			
	72 STO 05 ,48 RTN	Periodenlänge			
		in Alger. F			
	486*LBL 02				
	46 STO 05 ,557 RTN				
	491*LBL 03				
	FC? 00 FC? 25 RTN CLA	V-Pgm, das			
	SF 21 ARCL IND 88	bei eingeschalte-			
	ARCL IND 89 AVIEW CLA	tem Printer			
	CF 21 END	eine fertige			
		Anzeigezeile			
		druckt			

181, 182 – 81

REGISTERS, STATUS, FLAGS

REGISTERBELEGUNG, FLAGS, BETRIEBSARTEN
REGISTRES, INDICATEURS, MODES OPERATOIRES
REGISTRI, MODI OPERATIVI, FLAGS

Registers

00	Nr. Input (perm)	
01	Inkrement in F	
	Anzeigehilfsp.	
	Testfaktor p	
03	Inkrement in E	
	" " F	
04	Inkrement in E	
	" " F	
05	Periodenlänge	
	in F	
06	Inkrement in E	
	x in F	
07	n (reduziert)	
08	Index	
09	Index	
10	Zählvariable	
11	Anzeige-register	
12		
13		
14		
15		

Status

Size 017 Pgm 099 — User Mode

Eng ☐ Fix ☒ 0 Sci ☐ — Usr ☒

Deg ☒ Rad ☐ Grad ☐ — Usr ☐

	Purpose	SET	CLR	Flags
00	Einzelanzeige	X	X	gepackte Anzeige
01				
02				
03				
04				
05	kein Faktor gef.	X	X	erster Faktor gef.
06	n mod 4 = 1	X		
07	n mod 4 = 3	X		
08				
09				
10				
11	Anzeigehilfe			
12		X		
13				
14				
15				
16				
17				
18				
19				
20				
21	Printer Enable	X	X	
22	Number Input			
23	Alpha Input			
24	Range Error			
25	Trace Mode Printer "on"	X	X	Printer "off" oder
26	Audio Chk			nicht angeschlossen
27	User Mode			
28	Decimal Point			
29	Digit Grouping		X	

Assignments

Tastenbelegung / Assignations / Assegnazioni

Function	Key	Function	Key
PRM	14		

183-81

10-Punkte Gauß-Quadratur

Das folgende Programm benutzt die 10 Pkte Quadratur nach Gauß, um schnell und genau das Integral einer als Tastenfolge definierten Funktion zu berechnen.

Die Funktion muß als globales Label definiert werden. Bei Aufruf der Fkt ist x im X-Register vorhanden. Zur Programmierung steht der Restspeicher, der Stack und die Datenregister ab R 17 zur Verfügung.

Singularitäten oder Diskontinuietäten der Funktion verursachen einen Abbruch der Integration. Diese Schwierigkeiten können mit stückweiser Integration behoben werden.

Gleichungen

$$\int_A^B f(x)dx = \frac{b-a}{2} \sum_{1=1}^{10} w_i \; f\left(\frac{z_i(b-a)+b+a}{2}\right)$$

wobei:

$z_1 = .1488743390$
$z_2 = .2955242247$
$z_3 = .4333953941$
$z_4 = .2692667193$
$z_5 = .6794095683$
$z_6 = .2190863625$
$z_7 = .8650633667$
$z_8 = .1494513492$
$z_9 = .9739065285$
$z_{10} = .6667134430 \times 10^{-1}$

Beispiel

$$\int_1^{10} \frac{1}{x}\,dx = \ln 10 = 2.302585093$$

Die Integration mit diesem Programm liefert: 2.302578678 (N=1)

Lbl AA 1/x RTN	INPUT	TASTEN	DISPLAY
		XEQ GQ	A=?
	10	R/S	B=?
	1	R/S	N=?
	1	R/S	FUNCTION NAME?
	AA	R/S	F=2.3025

Allgemeine Instruktionen

	INPUT	TASTEN	OUTPUT
1. Eingabe Fkt		Lbl -- .... 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 026: Function Name? und ff
   Für Besitzer des MATH1A Moduls: XEQ ROM XFN

2. Programmlinien 055, 063, 075 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-Standardprogrammsammlung 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 Ø ist ganz richtig. Der Generator erzeugt Zufallszahlen von O-O,999..., wenn O als Zufallszahl nicht bewünscht wird, muß eben die anschließende Weiterverarbeitung richtig erfolgen.

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

INT(n(max-min+1)+min)

n....Zufallszahl O-O,999...
max..maximale gewünschte Zahl
min..minimale gewünschte Zahl

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Startroutine für Zufallszahlengenerator

Walter Kropf

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 weitergefahren 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 "RDM
-
 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
 18 GTO 01
 19◆LBL A
 20 CLST
 21 CLA
 22◆LBL 02
 23 "Z"
 24 ARCL Y
 25 "+ "
 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
```

Erwin Hartmann

Raimund Berg
Heusteigstraße 66

7000 Stuttgart

Stuttgart, 23.10.1980

Herrn
Oliver Rietschel
Postfach 373

2420 Eutin

Lieber Oliver!

Vielen Dank für die Übersendung von "prisma 10/80". Ich möchte dazu ein paar Sätze schreiben.

Die ersten betreffen das "Überprüfungsprogramm für Zufallszahlengeneratoren". Ich habe zwar nicht das Arithmetik-Lehrprogramm aus der Standard-Programmsammlung zur Hand, aber meines Wissens enthält der Zufallsgenerator nicht die Funktion SQRT. Mir ist er in der folgenden Form schon in mehreren Programmen begegnet und ich verwende ihn auch immer so:

```
LBL RNDM        STO n     Ausgangszahl für Zufallsgenerator
RCL n           STO m     Höchstzahl + 1
9821
x
,211327
+
FRC
STO n
RCL m
x
INT
END
```

Wenn man den Zufallsgenerator in dieser Form in das Testprogramm einsetzt, erhält man das gewünschte Ergebnis, wie die beigefügte Aufstellung beweist. (Anlage 1)

Das Programm "BK" (Kürzen von Brüchen) leistet leider trotz mehrerer Versuche nicht das, was es soll. Oder ich habe einen Fehler gemacht. Zusätzlich habe ich eine Version von mir beigelegt. (Anlage 2). Mein Programm verwendet keine Datenregister und benötigt 6 Programmspeicherregister.

Diesem Brief habe ich einen Euroscheck beigefügt für die folgenden Beträge:

Beitrag 1981	60,-- DM
Weihnachtssonderhefte	15,-- DM
2 Kartentaschen je 7,-- DM	14,-- DM
Porto, Kopien	1,-- DM
	70,-- DM

In meinem Prisma 10/80 sind leider die Seiten 10 und 35 nicht vorhanden. Bitte sende sie mir nach.

Ich habe noch eine Bitte an Dich: Kannst Du in einer der nächsten Ausgaben von Prisma folgende Notiz veröffentlichen?

> Wer hat Erfahrung mit der Monatslohnsteuertabelle? Ich möchte ein Programm zur Berechnung der monatlichen Abzüge vom Bruttolohn erstellen. Allerdings habe ich Schwierigkeiten mit der tabellenunabhängigen Berechnung der Lohnsteuer. Für jeden Tip wäre ich dankbar. Raimund Berg, Hagener Str. 200, 5910 Kreuztal.

Wie Du aus der Adresse ersiehst, wohne ich, voraussichtlich bis Sommer'82, z. Z. in Stuttgart. Meine Post soll aber weiter an meine alte Adresse gesendet werden.

Zum Schluß habe ich noch eine Frage zur Mitgliederliste. Was bedeuten die Zahlen in Klammern hinter dem Namen?

↖ Geburtsjahr im 20. Jh.

Mit freundlichem Gruß

Raimund Berg

Raimund Berg

Anlage 1:

erzeugte Zufallszahlen: 6900

Zufallszahl	Anzahl		
0	130		
1	142		
2	139		
3	134		
4	144		
5	148		
6	129		
7	127		
8	135		
9	140		
10	147		
11	141		
12	149	31	126
13	131	32	128
14	147	33	135
15	164	34	127
16	138	35	135
17	142	36	140
18	122	37	131
19	131	38	142
20	111	39	137
21	151	40	143
22	156	41	161
23	121	42	148
24	136	43	136
25	135	44	148
26	133	45	139
27	137	46	127
28	116	47	131
29	149	48	142
30	145	49	152

Ausführung:

1. Zähler eingeben
2. XEQ "BK"
3. Nenner eingeben
4. R/S
   gekürzter Bruch
   wird angezeigt

Anlage 2:

```
001 LBL "BK"
      FIX 0
      CP 29
      STO Z
      SYP
      STO Y
007 LBL 00
      MOD
      LAST X
      X<>Y
      X#0?
      GTO 00
      RDN
      STO/ Z
      STO/ Y
      CLA
      ARCL Z
      "/"
      ARCL Y
      AVIEW
021 END
```

Eulersche Gammafunktion

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

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

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

$$\Gamma(x) = \lim_{n\to\infty} \frac{n!\, n^z}{x(x+1)(x+2)\ldots(x+n)}$$

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

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

wobei z=45+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
         MA"
02 CF 29
03 FIX 3
04 RAD
05 CF 01
06 X<=0?
07 XEQ 00
08 STO 01
09 FRC
10 X=0?
11 GTO 02
12 45
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

75◆LBL 02
76 LASTX
77 1
78 -
79 FS? 01
80 1 E2
81 FACT

82◆LBL 08
83 "I<X>="
84 ARCL X
85 AVIEW
86 .END.
```

```
DIFFERENTIALGLEICHUNGEN
-----------------------
AUFLOESUNG VON **
1. ORDNUNG:
-----------
Y'=F(X,Y)
2. ORDNUNG:
-----------
Y"=F(X,Y,Y') ODER
Y'=F(X,Y,Z)
Z'=G(X,Y,Z)
4. ORDNUNG:
-----------
Y"=F(X,Y,Y',Y") ODER
Y'=F(X,Y,Z,Z')
Z'=G(X,Y,Z,Z')
ABLAUF:
FUER PLOTTEN/PRINTEN:
PRINTERSCHALTER: MAN NO
PLOTTEN: XEQ PPLOT
  NAME?: DIFFGL ...
MIT PLOT2: USER-MODE
PRINTEN U/O ANZEIGEN:
  XEQ DIFFGL
F() RESP. G() ALS EIGENE
PROGRAMME SCHREIBEN.
X STEHT IN X-REG. F()
RESP. G() MUESSEN AUCH
IM X-REG STEHEN.
STATUS:
R00-R10,R12-R19
ZUSAETZLICH BEI TYP 2:
R20-R24
ZUSAETZLICH BEI TYP 4:
R25-R28
F01,F02,F04,F10,F19-F21
INHALT:
R00:X0,R09:XMAX,R10:XINC
R12:Y0,R13:"F()",R14:Y
R15:h,R16:KC,R17:1,R18:e
R19:n,R20:20,R21:"G()"
R22:2,R23:1,R24:11
R25:Y'0,R26:Z'0,R27:Y'
R28:Z',R29:h',R30:1'
F01-F02-F04 TYP,F10:BEL.
F19:MAX,F20:PLOT,F21:BEL

VERFAHREN: RUNGE-KUTTA
AUSGABE: 1. ANZEIGEN
         2. PRINTEN
         3. PLOTTEN
VOM PROGRAMM HER ODER
EXTERN: AUXILIARY
LITERATUR: LOESUNG VON
DIFFERENTIALGLEICHUNGEN
MIT PROGRAMMIERBAREN
TASCHENRECHNERN
AUTOR: G. VENZ
VERLAG: W.SCHMIDBURG
BEARBEITET VON:
MAX HUBER (196)
```

```
01*LBL "DIF
       FGL"
02 FC? 19
03 FS? 20
04 GTO 15
05 XEQ 17
06 "PLOT:SF
       20"
07 PROMPT
08 "AUX:SF
       9"
09 PROMPT
10 "TYP=1,2
      ,4?"
11 PROMPT
12 SF IND X
13 FS? 20
14 GTO 00
15 "XMAX?"
16 PROMPT
17 STO 09
18 "XINC?"
19 PROMPT
20 STO 10
21 "e?"
22 PROMPT
23 STO 18
24 CLX
25 STO 19

26*LBL 00
27 RDN
28 "NAME:FC
      )?"
29 PROMPT
30 ASTO 13
31 FS? 01
32 GTO 05
33 "NAME:GC
      )?"
34 PROMPT
35 ASTO 21

36*LBL 05
37 AOFF
38 "X0↑Y0"
39 FC? 01
40 "↳↑Z0"
41 "↳?"
42 PROMPT
43 FC? 01
44 STO 20
45 FC? 01
46 RDN
47 STO 12
48 X<>Y
49 STO 08
50 FC? 04
51 GTO 19
52 "Y'0↑Z'0
      ?"
```

EINGABE DER WERTE.

EINGABE SCHON GESCHEHEN? WEITERBERECHNEN, F19:F20 müssen neu belegt sein.

BEI PLOT: UEBERSPRINGEN, DA SCHON ABGEFRAGT-ODER NICHT BENOETIGT.

SONDERBEDINGUNGEN.

```
53 PROMPT
54 STO 26
55 X<>Y
56 STO 25

57*LBL 19
58 FS? 19
59 RTN
60 FC? 20
61 GTO 06
62 RCL 12
63 FS? 01
64 RTN
65 RCL 20
66 XEQ "PLO
       T2"
67 RTN

68*LBL 06
69 "X"
70 8
71 XEQ 16
72 "Y"
73 12
74 XEQ 16
75 FS? 01
76 GTO 15
77 "Z"
78 20
79 XEQ 16
80 FS? 02
81 GTO 15
82 "Y'"
83 25
84 XEQ 16
85 "Z'"
86 26
87 XEQ 16

88*LBL 15
89 RCL 08
90 RCL 09
91 X<=Y?
92 GTO 17
93 CLX
94 STO 15
95 FC? 01
96 STO 23
97 FS? 04
98 STO 29
99 FS? 04
100 STO 30
101 2
102 ST/ 10
103 X↑2
104 STO 17

105*LBL 18
106 XEQ IND
       17
107 RCL 08
```

AUSGABE: WIN? PLOT MIT PLOT2? ODER PRINTEN?

BERECHNUNG VON k, 1, k', 1'.


```
108 XEQ IND
       13
109 STO 16
110 ST+ 15
111 FS? 10
112 ST+ 15
113 FS? 01
114 GTO 07
115 FS? 04
116 ST+ 29
117 RCL 08
118 XEQ IND
       21
119 STO 24
120 ST+ 23
121 FS? 10
122 ST+ 23
123 FS? 04
124 ST+ 30

125*LBL 07
126 DSE 17
127 GTO 18
128 FC? 04
129 GTO 08
130 ST- 30
131 RCL 16
132 ST- 29
133 RCL 10
134 6
135 /
136 ST* 29
137 ST* 30

138*LBL 08
139 RCL 10
140 6
141 /
142 ST* 15
143 FC? 01
144 ST* 23
145 ISG 19
146 CLX
147 FS? 04
148 GTO 11
149 RCL 15
150 ST+ 12
151 FS? 01
152 GTO 19
153 RCL 23
154 ST+ 28
155 GTO 19

156*LBL 11
157 RCL 29
158 RCL 25
159 +
160 RCL 10
161 *
162 ST+ 12
163 RCL 15
```

NAECHSTE PUNKTE.

```
164 ST+ 25
165 RCL 30
166 RCL 26
167 +
168 RCL 10
169 *
170 ST+ 20
171 RCL 23
172 ST+ 26
173 GTO 19

174*LBL 16
175 FIX 0
176 ARCL 19
177 "↳= "
178 RCL 18
179 LOG
180 1
181 +
182 FIX IND
          X
183 ARCL IND
          Y
184 AVIEW
185 RTN

186*LBL 01
187 CF 10
188 RCL 10
189 2
190 ST* 10
191 X<>Y
192 GTO 09

193*LBL 03
194 SF 10
195 RCL 10

196*LBL 09
197 ST+ 08

198*LBL 02
199 XEQ 04
200 RCL 16
201 FS? 04
202 GTO 10
203 ST+ 14
204 FS? 01
205 RTN
206 RCL 24
207 ST+ 22
208 RTN

209*LBL 10
210 RCL 25
211 X<>Y
212 +
213 STO 27
214 LASTX
215 2
216 /
```

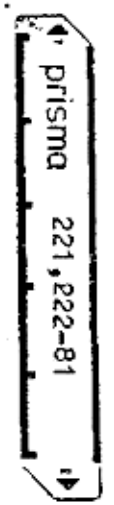
U'PROGRAMM FUER AUSGABE.

UNTERPROGRAMM FUER s.U.

```
217 -
218 RCL 10
219 *
220 ST+ 14
221 RCL 26
222 RCL 24
223 +
224 STO 28
225 LASTX
226 2
227 /
228 -
229 RCL 10
230 *
231 ST+ 22
232 RTN

233*LBL 04
234 RCL 12
235 STO 14
236 RCL 10
237 ST* 16
238 FS? 01
239 RTN
240 ST* 24
241 RCL 20
242 STO 22
243 FS? 02
244 RTN
245 RCL 25
246 STO 27
247 RCL 26
248 STO 28
249 RTN

250*LBL 17
251 CF 01
252 CF 02
253 CF 04
254 CF 10
255 CF 19
256 CF 20
257 BEEP
258 "FIN"
259 CF 21
260 AVIEW
261 SF 21
262 END
```



W'PROGRAMM FUER AUSGABE. PRINTEN U/O ANZEIGEN.

```
LBL"DIFFGL
END
52$ BYTES
```

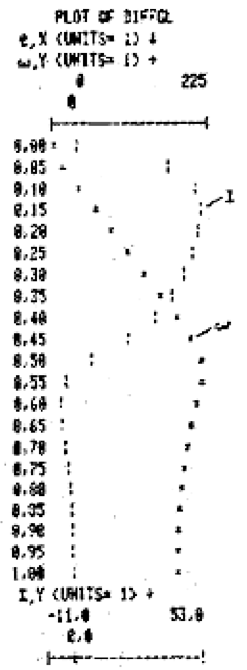
BEISPIEL:
EINSCHALTEN EINES
GLEICHSTROM-MOTORS MIT
ANKERRUECKWIRKUNG

$$\frac{dw}{dt} = \frac{c*i}{I}$$

$$\frac{di}{dt} = \frac{U-Ra\cdot i-c*w}{L}$$

```
 01*LBL "dO"      F()
1.2 RCL 22 40 / X↑2
1 + / STO 25 RCL 22
* .05 / RTN

 16*LBL "dI"      G()
225 RCL 22 3.6 * -
RCL 25 RCL 14 * -
.18 / END
```



PRINTEN DURCH DIFFGL:

```
X0= 0.000
Y0= 0.000
Z0= 0.000
X1= 0.050
Y1= 16.944
Z1= 37.123
X2= 0.100
Y2= 40.690
Z2= 48.619
X3= 0.150
Y3= 64.865
Z3= 51.179
X4= 0.200
Y4= 87.380
Z4= 50.280
X5= 0.250
Y5= 110.880
Z5= 47.730
 U S W .
```

PRINTEN EXTERN:

```
 01*LBL "AUX"
XEQ "DIFFGL" "t" 0
XEQ 16 "⊢ sec" AVIEW
"w" 12 XEQ 16
"⊢ rad/s" AVIEW "I"
20 XEQ 16 "⊢ A" AVIEW
GTO "AUX"

 19*LBL 16
FIX 0 ARCL 19 "⊢= "
RCL 18 LOG 1 +
FIX IND X ARCL IND Y
END
```

```
t0= 0.000 sec
w0= 0.000 rad/s
I0= 0.000 A
t1= 0.050 sec
w1= 16.944 rad/s
I1= 37.123 A
t2= 0.100 sec
w2= 40.690 rad/s
I2= 48.619 A
t3= 0.150 sec
w3= 64.865 rad/s
I3= 51.179 A
t4= 0.200 sec
w4= 87.380 rad/s
I4= 50.280 A
t5= 0.250 sec
w5= 110.880 rad/s
I5= 47.730 A
```

DIESES BEISPIEL BESITZT
ORDNUNG 2. FÜR BEISPIELE
MIT ORDNUNG 1 ODER 4
EMPFEHLE ICH SIN UND COS
FUNKTIONEN.
ANSONSTEN VERWEISE ICH
AUF DIE LITERATUR.

HAPPY PROGRAMMING
MAX HUBER (196)

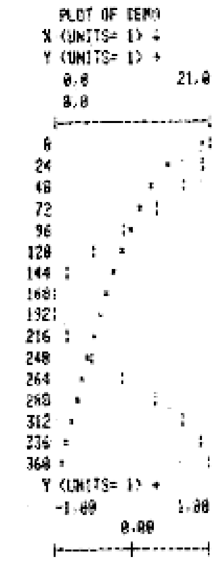
FLOTTEN VON 2 FUNKTIONEN
... PUNKTEN

PLOT2 STELLT DIE DATEN
ZUM PLOTTEN VON 2 FUNK-
TEN BEREIT. AM ENDE DES
PLOTS WIRD DIE ACHSE VON
Y2 BESCHRIFTET.
PRINTERSCHALTER: MAN ■
PROGRAMMSEQUENZ:

```
  LBL ... (UNTERPROGRAMM
  ....     (FUER "PRPLOT"
  RCL Y1
  RCL Y2
  XEQ PLOT2
  END       ODER REGPLOT
                ....
                ....
```

VORBEDINGUNG
PLOT2 MUSS IN USER-MODE
GESTARTET WERDEN. WIRD
GELOESCHT VOR EINGABE.

STATUS:
GLEICHE REG WIE PRPLOT:
R02:nom.aY2,R03:YMIN2
R04:YMAX2,F27:EINGABE?




```
01*LBL "PLO
        T2"
02 FC?C 27
03 GTO 00
04 "YMIN2?"
05 PROMPT
06 STO 03
07 RDN
08 "YMAX2?"
09 PROMPT
10 STO 04
11 RDN
```

EINGABE, MUSS DAS ERSTE MAL DURCHLAUFEN WERDEN.

```
12*LBL 00
13 RCL 03
14 -
15 RCL 04
16 RCL 03
17 -
18 /
19 RCL 02
20 INT
21 STO 02
22 *
23 FIX 0
24 RND
25 1 E3
26 /
27 ST+ 02
28 X<>Y
29 RCL 09
30 RCL 10
31 RCL 06
32 +
33 X>Y?
34 GTO 01
35 R↑
36 RTN
```

BEREITSTELLEN DER DATEN FUER REGPLOT.

```
37*LBL 01
38 R↑
39 REGPLOT
40 RCL 03
41 STO 00
42 X<=0?
43 CLX
44 X<> 04
45 STO 01
46 X<0?
47 STO 04
48 XROM "PR
      AXIS"
49 BEEP
50 "FIN"
51 PROMPT
52 END
```

LETZTE SPALTE UND ZWEITE Y-ACHSE UND SCHLUSSHALT.

**ERSTES PLOT-
PROGRAMM FOR
2 FUNKTIONEN
MIT VERSCH.
Y-ACHSEN!**
PLOT2 VERWENDET DIE X-
ACHSE ZUM PLOTTEN DER 2.
FUNKTION. IN DER ERSTEN
Y-ACHSE KANN EIN MARKIE-
RUNGSSTRICH FREI GEWÄHLT
WERDEN (z.B. NULLAGE).
IN DER ZWEITEN Y-ACHSE
WIRD DIE NULLAGE ODER
DER WERT NAHER BEI NULL
AUTOMATISCH MARKIERT.
PLOT2 BENOTIGT KEINE ZU-
SÄTZLICHEN SPEICHERREG.
IN R02 IST DIE LAGE DES
PLOTPUNKTES UND DER
X-ACHSE FESTGELEGT.
R03 BESTIMMT DEN PLOT-
CHARACTER WENN NICHT NUM
UND R04 ZEIGT, DASS DIE
ACHSE GEPLOTTET WERDEN
SOLL, WENN NUMERISCH.
DAS PLOT2-PROGRAMM ENT-
STAND ALS NEBENPRODUKT
VON DIFFGL, DA DORT ZWEI
FUNKTIONEN MIT VERSCH.
Y-ACHSEN GEPLOTTET WER-
DEN SOLLTEN.
Happy Programming
Max Huber (196)

```
LBL "PLOT2
END
87 BYTES
```

```
 01*LBL "DEMO"
COS LASTX SQRT CHS
20 + X<>Y XEQ "PLOT2"
END
```

Unterprogramm RUKUN: Differentialgleichungssystem nach Runge-Kutta

Gegeben seien N (gekoppelte) Differentialgleichungen erster Ordnung:

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

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

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

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

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

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

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

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

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

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 ... N) muß vom Benutzer programmiert werden, beginnend mit LBL "FN", endend mit END, wobei "FN" ein beliebiger Funktionsname (max. 6 Alpha-Zeichen) ist.

Speicherplan:

$R_{ØØ}$: Adresse für dynamische Speicherzuweisung
$R_{Ø1}$: Schleifenzähler für N
$R_{Ø2}$: S-1
$R_{Ø3}$: x^I; x^{II}; x^{III}; x^{IV}
$R_{Ø4}$: N
$R_{Ø5}$: Startadresse S ⩾ S
$R_{Ø6}$: Inkrement h
$R_{Ø7}$: "FN"
$R_{Ø8}$: $x^{(Ø)}$; $x^{(1)}$

$R_S \ldots R_{S-1+N}$: $y_1^{(Ø)} \ldots y_N^{(Ø)}$; $y_1^{(1)} \ldots y_N^{(1)}$

$R_{S+N} \ldots R_{S-1+2N}$: $y_1' \ldots y_N'$

$R_{S+2N} \ldots R_{S-1+3N}$: $y_1^I \ldots y_N^I$; ... ; $y_1^{IV} \ldots y_N^{IV}$

$R_{S+3N} \ldots R_{S-1+4N}$: $k_1^I \ldots k_N^I$; ... ; $k_1^{IV} \ldots k_N^{IV}$

$R_{S+4N} \ldots R_{S-1+5N}$: $l_1^I \ldots l_N^I$; ... ; $l_1^{IV} \ldots l_N^{IV}$ (Zwischensummen der k)

Die Speicher $R_{ØØ}$ bis $R_{Ø8}$ 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 $R_{Ø5}$ festgelegt.

Die Funktionen f_i im Unterprogramm "FN" sind mit den Argumentwerten aus den Speichern $R_{Ø3}$ und $R_{S+2N} \ldots R_{S-1+3N}$ zu bilden und dann nach $R_{S+N} \ldots R_{S-1+2N}$ abzuspeichern.

Beispiel:

Im folgenden wird die Differentialgleichung y" = -y mit den Anfangsbedingungen y(Ø) = Ø; y'(Ø) = 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:

$y_1' = y_2$ $\qquad$ $y_1(Ø) = Ø$

$y_2' = y_1$ $\qquad$ $y_2(Ø) = 1$

Der Einfachheit halber soll $R_{Ø9}$ 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"ABLEIT          CHS
RCL 14              STO 12
STO 11              END
RCL 13
```

Das Hauptprogramm HAUPT geht von folgender Speicherbelegung aus:

$R_{Ø4}$: N = 2 $R_{Ø5}$: S = 9 $R_{Ø6}$: h = Ø.Ø1
$R_{Ø7}$: ABLEIT $R_{Ø8}$: $x^{(Ø)}$ = Ø $R_{Ø9}$: $y_1(Ø)$ = Ø
$R_{1Ø}$: $y_2(Ø)$ = 1

Weiterhin wird von HAUPT verlangt, daß nach jeweils 1Ø Integrationsschritten x und y_1 zur Anzeige gebracht werden. Für diesen Schrittzähler wird der hinter dem hier definierten dynamischen Speicherbereich liegende Speicher R_{19} benutzt.

Das Hauptprogramm hat dann folgende Gestalt:

```
LBL"HAUPT           ARCL Ø9
1Ø                  AVIEW
STO 19              LBL Ø1
CLA                 XEQ"RUKUN
FIX 1               DSE 19
ARCL Ø8             GTO Ø1
←Ø                  GTO"HAUPT
FIX 6               END
```

(Ø bedeutet SPACE)

Nach der Speicherfestlegung durch XEQ SIZE Ø2Ø wird das Programm durch XEQ HAUPT gestartet. Ein Vergleich der - nach jeweils etwa 17Ø s Rechenzeit - erhaltenen Werte von x und $y_1(x)$ mit den im RAD-Modus direkt errechneten Werten von y_1 = 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:

$R_{Ø4}$: N $R_{Ø6}$: h $R_{Ø8}$: $x^{(Ø)}$

$R_{Ø5}$: S $R_{Ø7}$: "FN" R_{S-1+i}: $y_i^{(Ø)}$ i=1...N

$S-1 \rightarrow R_{Ø2}$

$x=x^{(Ø)} \rightarrow R_{Ø3}$

Flag Ø0 setzen

UP Ø2

Flag Ø0 löschen

UP "FN"

Flags Ø2 und Ø4 setzen

UP Ø5

Flag Ø2 löschen

$x=x^{(Ø)}+\frac{h}{2} \rightarrow R_{Ø3}$

Flag Ø1 setzen

UP Ø2

UP "FN"

Flag Ø3 setzen

UP Ø5

UP Ø2

UP "FN"

UP Ø5

$x=x^{(Ø)}+h \rightarrow R_{Ø3}$; $x^{(1)}=x \rightarrow R_{Ø8}$

Flag Ø1 löschen

UP Ø2

UP "FN"

Flags Ø3 und Ø4 löschen

UP Ø5

Speicherbelegung bei Programmende:

wie bei Start, jedoch $R_{Ø8}$: $x^{(1)}$

R_{S-1+i}: $y_i^{(1)}$ i=1...N

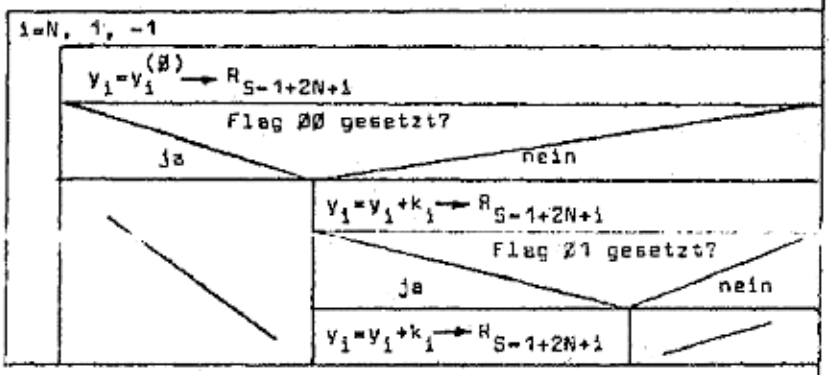
Unterprogramm Ø2

Speicherbelegung bei Programmstart:

$R_{Ø4}$: N

R_{S-1+i} : $y_i^{(Ø)}$ i=1...N

$R_{S-1+3N+i}$: k_i i=1...N falls Flag ØØ gelöscht



Unterprogramm "FN"

Speicherbelegung bei Programmstart:

$R_{Ø3}$: x $R_{S-1+2N+i}$: y_i i=1...N

$y_i'=f_i(x, y_1, \ldots y_N) \rightarrow R_{S-1+N+i}$ i=1...N

Unterprogramm Ø5

Speicherbelegung bei Programmstart:

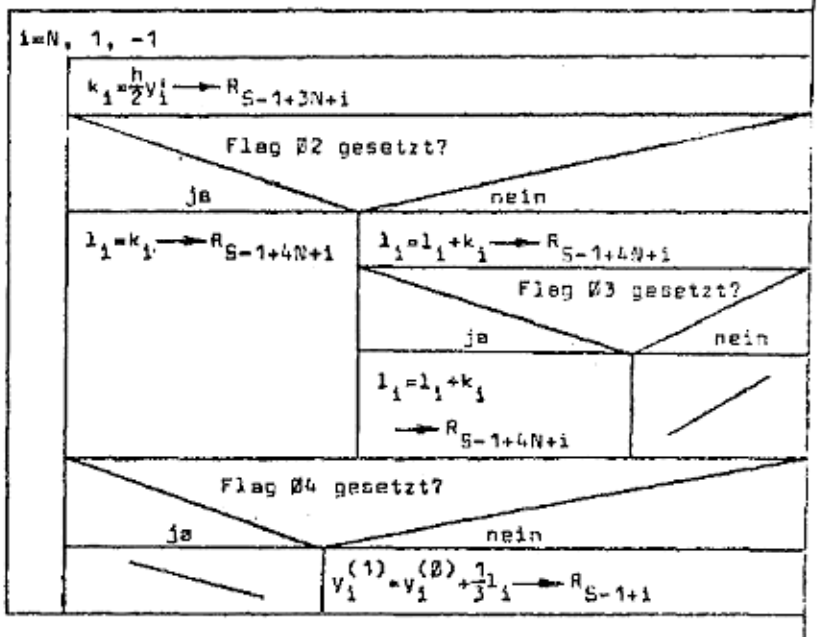
$R_{Ø6}$: h

R_{S-1+i} : $y_i^{(Ø)}$ falls Flag Ø4 gelöscht

$R_{S-1+N+i}$: y_i'

$R_{S-1+4N+i}$: l_i falls Flag Ø2 gelöscht

i=1...N



```
    LBL"RUKUN
    RCL Ø5
    1
    -
ØØ5 STO Ø2
    RCL Ø8
    STO Ø3
    SF ØØ
    XEQ Ø2
Ø1Ø CF ØØ
    XEQ IND Ø7
    SF Ø2
    SF Ø4
    XEQ Ø5
Ø15 CF Ø2
    RCL Ø8
    RCL Ø6
    2
    /
Ø2Ø +
    STO Ø3
    SF Ø1
    XEQ Ø2
    XEQ IND Ø7
Ø25 SF Ø3
    XEQ Ø5
    XEQ Ø2
    XEQ IND Ø7
    XEQ Ø5
Ø3Ø RCL Ø8
    RCL Ø6
    +
    STO Ø3
    STO Ø8
Ø35 CF Ø1
    XEQ Ø2
    XEQ IND Ø7
    CF Ø3
    CF Ø4
Ø4Ø XEQ Ø5
    RTN
    LBL Ø2   Unterprogramm
    RCL Ø4
    STO Ø1
Ø45 LBL Ø3
    RCL Ø1
    RCL Ø2
    +
    STO Ø2
Ø52 RCL IND ØØ
    RCL Ø4
    2
    *
    ST+ ØØ
Ø55 RDN
    STO IND ØØ
    FS? ØØ
    GTO Ø4
    RCL Ø4
Ø6Ø ST+ ØØ
    RCL IND ØØ
    RCL Ø4
    ST- ØØ
    RDN
Ø65 ST+ IND ØØ
    FC? Ø1
    ST+ IND ØØ
    LBL Ø4
    DSE Ø1
Ø7Ø GTO Ø3
    RTN
    LBL Ø5   Unterprogramm
    RCL Ø4
    STO Ø1
Ø75 LBL Ø6
    RCL Ø1
    RCL Ø2
    +
    RCL Ø4
Ø8Ø +
    STO ØØ
    RCL IND ØØ
    RCL Ø6
    *
Ø85 2
    /
    RCL Ø4
    2
    *
Ø9Ø ST+ ØØ
    RDN
    STO IND ØØ
    RCL Ø4
    ST+ ØØ
Ø95 RDN
    FS? Ø2
    GTO Ø7
    ST+ IND ØØ
    FS? Ø3
1ØØ ST+ IND ØØ
    GTO Ø8
    LBL Ø7
    STO IND ØØ
    LBL Ø8
1Ø5 FS? Ø4
    GTO Ø9
    RCL IND ØØ
    3
    /
11Ø RCL Ø1
    RCL Ø2
    +
    STO ØØ
    RDN
115 ST+ IND ØØ
    LBL Ø9
    DSE Ø1
    GTO Ø6
    END
```

Auswertung aussagenlogischer Ausdrücke $A=A(p_o,\ldots,p_n)$ $(n\leq 9)$

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 → LASTX	Bemerkung
AND		X,Y	ja	
EQU	⇔	X,Y	ja	entspricht × (Mult.)
IMP	⇒	X,Y	ja	Y ⇒ X, Reihenfolge notwendig!
NOT	¬	X	nein	entspricht CHS
OR	∨	X,Y	ja	

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

```
RCL 02 RCL 01 XEQ'IMP RCL 01 RCL 00 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_o wechselt in jeder Zeile, der von p_1 in jeder zweiten, der von p_2 in jeder vierten etc. Die Tabelle hat $2\uparrow(n+1)$ 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:

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

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

$$-1 < -r_{ik};$$

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: $p_o \Rightarrow p_1$, also: RCL 00 RCL 01 XEQ'IMP.

*) mit TONE 8

Listing

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

```
Ø1 LBL'ASL
   'AUSSAGE?
   AON
Ø4 PROMPT
   AOFF
   ASTO 13
   'VAR-ANZ?
Ø8 PROMPT
   2  X<>Y  Y↑X  STO 1Ø
   LASTX  1  -
   1 E3  /  STO 12
   -1  STO ØØ  ...  STO Ø9
3Ø o  LBL 1Ø
      CLA
      RCL 12  STO 11
34    o  LBL 11
         RCL 1Ø
         2
         RCL 11  INT
         Y↑X
         MOD  CHS
         -1
         ?  X<Y?
         j  ST+ IND 11
         RCL IND 11
         ?  X>Ø?
         j  'T-J
         ?  X<Ø?
         j  'T-F
      I  ISG 11
51       GTO 11
52    XE? IND 13
      ?  X>Ø?
      j  'T-J
      ?  X<Ø?
      j  'T-F
      TONE 8  AVIEW
   I  DSE 1Ø
6Ø    GTO 1Ø
61 STOP

62 LBL'AND
      *
      ?  X<Ø?
65    j  RTN
      n  CLX  LASTX
68       RTN

69 LBL'ECU
      *
71 RTN

72 LBL'IMP
      *
      ?  X>Ø?
75    j  RTN
      n  CLX  LASTX
78       RTN

79 LBL'NOT
      CHS
81 RTN

82 LBL'OR
      *  CHS
      ?  X>Ø?
85    j  RTN
      n  CLX LASTX
89       END
```

PRGM: 165 Byte (?)
REG: 14

$R_{00}..R_{09}$: ± 1 $(=p_i)$

R_{10} : $j=2^{n+1},\ldots,1$ $(j:=2^{n+1}+1-k)$

R_{11} : i=Ø,...,n

R_{12} : n/1ØØ (Schleifenkontrolle für i)

R_{13} : 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 rechtwinkliger 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

$$f(t) = \frac{ao}{2} + \sum_{1}^{k} a_k \cdot \cos\, k.2.\pi.t/T + b_k \cdot \sin\, k.2.\pi.t/T =$$

$$\frac{ao}{2} + \sum_{1}^{k} c_k \cdot e^{-i \cdot p_k}$$

$$a_k = \frac{2}{T} \cdot \sum_{1}^{T} f(t) \cdot \cos \frac{k.2.\pi.t}{T} \qquad b_k = \frac{2}{T} \cdot \sum_{1}^{T} f(t) \cdot \sin \frac{k.2.\pi.t}{T}$$

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

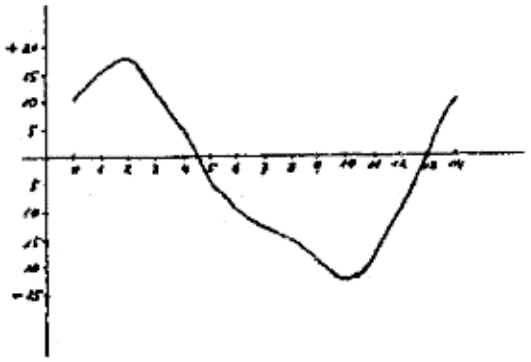
Der Verfasser übernimmt keine Verantwortung und keine wie immer geartete Haftung, die auf irgendeine Art aus der Benutzung dieses Programmes entsteht.

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 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, K1=0 (=5)

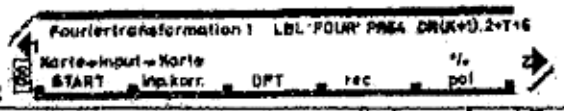


t	f(t)
1	16
2	18
3	12
4	5
5	- 4,5
6	-10
7	-13
8	-15
9	-20
10	-22
11	-18
12	-10
13	0
14	10

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
		END DFT	Berechnung
	D	K : b : a	Ausgabe rechtwinklig
		0: 0,000: -7,357	
		1: 14,899: 11,887	
		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 100%
		1: 0,874: 18,390: 100,0%	
		2: 0,950: 3,878: 21,1%	
		3: 4,056: 0,941: 5,1%	
		4: 1,729: 0,428: 2,3%	
	C		Start für nächste Frequenz-gruppe
3		K=3	
5		K1=5	
		END DFT	
	D	K : b : a	
		5: -0,188: 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: 4,804: 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		C	"K="
	weiter bei Schritt 4			
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

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	*	
22	PROMPT	f(ti) eingeben	76	PI	k.2. .t/T
23	" "		77	*	
24	ARCL Y		78	RCL 04	
25	"+= "		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 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 listati. Il nastro adesivo può alterare la stampato!

PROGRAM LISTING
PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line / Zeile / Ligne / Linea	Keystrokes / Tastenfolge / Touches / Tasti	Comments / Kommentar / Commentaires / Commenti
109	5	
110	+	
111	1 E3	
112	/	
113	6	
114	+	
115	FS?C 00	
116	STO 00	Pointer setzen t
117	RCL 02	
118	1	
119	+	
120	500	
121	/	
122	+	
123	RCL 05	
124	+	
125	STO 01	Pointer setzen k
126	RTN	
127	*LBL e	Resultatausgabe
128	CLX	mit %
129	"c1="	Wert für 100%
130	PROMPT	speichern
131	100	
132	/	
133	STO 04	
134	SF 02	
135	*LBL E	Resultatausgabe
136	"K : PHI : c"	polar
137	AVIEW	
138	SF 01	
139	GTO 05	
140	*LBL D	Resultatausgabe
141	"K : b : a"	rechtwinklig
142	AVIEW	
143	CF 01	
144	*LBL 05	
145	XEQ 00	Pointer setzen k
146	RCL 03	
147	STO 00	k1- ROO
148	*LBL 06	Schleife
149	FIX 0	
150	CLA	
151	ARCL 00	ki in Anzeige
152	"⊢: "	
153	RCL IND 01	
154	ISG 01	
155	RCL IND 01	
156	FS? 01	
157	XEQ 08	
158	FIX 3	Ausgabeformat
159	RND	
160	X<>Y	
161	RND	
162	ARCL Y	b(φ) in Anzeige

Line / Zeile / Ligne / Linea	Key pressed / Tastenfolge / Touches / Tasti	Comments / Kommentar / Commentaires / Commenti
163	"⊢: "	
164	ARCL X	a(c) in Anzeige
165	FC? 02	
166	GTO 07	
167	"⊢: "	
168	FIX 1	% Format
169	R↑	
170	RND	
171	ARCL X	% in Anzeige
172	"⊢%"	
173	*LBL 07	
174	AVIEW	Resultat anzeigen
175	1	
176	ST+ 00	ki+1
177	ISG 01	
178	GTO 06	nächstes k
179	CF 01	
180	CF 02	
181	BEEP	
182	ADV	
183	RTN	
184	*LBL 08	
185	X<>Y	Resultat polar
186	R-P	
187	X<>Y	
188	X<0?	
189	SF 03	
190	2	
191	PI	
192	*	
193	FC?C 03	φ < 0: +2π
194	CLX	
195	+	
196	FC? 02	
197	RTN	
198	X<>Y	
199	ENTER↑	
200	ENTER↑	
201	RCL 00	
202	X=0?	k=0: ao/2
203	SF 03	
204	RDN	
205	FS? 03	
206	2	
207	FS?C 03	
208	/	
209	RCL 04	
210	/	
211	RDN	
212	X<>Y	
213	RTN	
214	*LBL 09	
215	PROMPT	
216	ARCL X	

Please use paper glue to attach listings. Adhesive tape may affect print!
Bitte Listings mit Papierleim einkleben. Klebefilme können Druck bleichen!
S.V.P. utilisez de la colle à papier pour coller les listings. Les rubans adhésifs peuvent altérer l'impression!
Per favore usare la colla per fissare i listati. Il nastro adesivo può alterare la stampa!

PROGRAM LISTING
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 RTN				
	220*LBL b				
	221 SF 00		55		
	222 XEQ 00	Pointer stzen t			
	223 RCL 00				
	224 1				
	225 -				
	226 CF 21				
	227 "WDAT INP"		60		
	228 AVIEW	Eingabedaten auf			
	229 WDTAX	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		70		
	238 1				
	239 -				
	240 CF 21				
	241 "RDAT INP"				
	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 Papierleim 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 la stampa!

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 phi muß im Winkelmaß Radiant eingegeben werden.

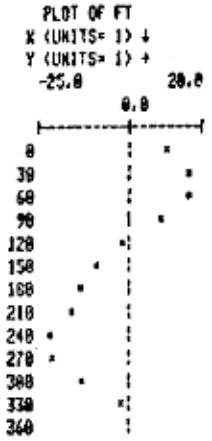
Verwendete Formeln:
T = Periodendauer (frei wählbar)
ak, bk, ck, k = Fourierkoeffizienten der Ordnungszahl k
n = Anzahl der gegebenen Frequenzkomponenten

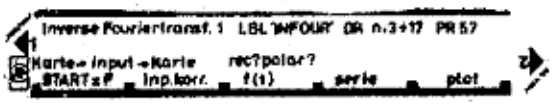
Beispiel: Eine harmonische Schwingung besteht aus folgenden Komponenten:

k	b	a	$a_o = 7,357$
1	14,099	11,807	
2	3,155	2,255	
3	-0,745	-0,574	
4	0,422	-0,067	
7	0,0	0,5	

Es soll die Funktion für die Werte t=2 und t=6 bei einer Periodenlänge von T=14 berechnet werden. Der Verlauf der Funktion ist graphisch darzustellen, wobei die Periodenlänge mit 360 bezeichnet wird.

Eingabe	Taste		Kommentar
	F	REG	Eingabe rechtwinklig
7,357 CHS	R/S	a0=-7,357	
5	R/S	N=5,000	
1	R/S	K=1	1. Komponente
14,099	R/S	b=14,099	
11,807	R/S	a=11,807	
2	R/S	K=2	2. Komponente
		b=3,155	
		a=2,255	
		K=3	
		b=-0,745	
		a=-0,574	
		K=4	
		b=0,422	
		a=-0,067	
		K=7	
		b=0,000	
		a=0,500	
	C	END INP	Berechnung einzelner Werte
14	R/S	T=14,000	
2	R/S	2,000: 17,351	
6	R/S	6,000: -9,755	
	E	T=360,000	Aufruf der Plotroutine
360	R/S		
FT	R/S		
-25	R/S		
20	R/S		
0	R/S		
0	R/S		
360	R/S		
30	R/S		





MA 103/3

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
16 a0 Datenkarte aufgezeichnet
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 Ligne Linea	Keystrokes Tastenfolge Touches Tasti	Comments Kommentar Commentaires Comments	Line Zeile Ligne Linea	Key pressed Tastenfolge Touches Tasti	Comments Kommentar Commentaires Comments
01	LBL "INFOUR"	Programmstart	55	XEQ 02	
02	CF 01		56	STO IND 14	
03	LBL 09		57	ISG 14	Eingabe beendet?
04	SF 21		58	GTO 01	
05	RAB		59	LBL 08	
06	CF 29		60	SF 21	
07	SF 00		61	BEEP	
08	LBL c		62	"END INP"	
09	FS? 00	Koeffizienten polar oder	63	AVIEW	
10	"REC"	rechtwinklig ?	64	RTN	
11	FS?C 00		65	LBL 02	
12	GTO 06		66	PROMPT	Eingabe anfordern
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 B	Eingabekorrektur
18	"a0="		72	RCL 13	Pointer setzen
19	FC? 01		73	STO 14	
20	XEQ 02	a_0 speichern	74	"K="	
21	FC? 01		75	PROMPT	
22	STO 16		76	FIX 0	
23	"N="		77	ARCL X	
24	XEQ 02	n eingeben	78	"+ KORR."	
25	3		79	FS? 55	
26	*	Pointer berechnen	80	PRA	
27	16		81	LBL 05	
28	+		82	RCL IND 14	K suchen
29	RCL IND X	Speicherbereich prüfen	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	Pointer setzen	89	GTO 05	
36	STO 14		90	GTO J	
37	FS?C 01	Rücksprung, wenn Eingabe	91	LBL b	
38	RTN	von Karte	92	RCL 13	Eingabedaten auf
39	LBL 01		93	2	Karte schreiben
40	FIX 0		94	-	
41	"K="	K speichern	95	STO 15	
42	XEQ 02		96	CF 21	
43	STO IND 14		97	"WDAT INP"	
44	ISG 14		98	AVIEW	
45	FIX 3		99	WDTAX	
46	"b="		100	GTO 00	
47	FS? 00	b_n (φ_n) speichern	101	LBL a	Eingabedaten von
48	"PHI="		102	SF 01	Karte lesen
49	XEQ 02		103	XEQ 09	
50	STO IND 14		104	2	
51	ISG 14		105	-	
52	"a="	a_n (c_n) speichern	106	CF 21	
53	FS? 00		107	"RDAT INP"	
54	"c="		108	AVIEW	

Please use paper glue to attach listings. Adhesive tape may affect print!
Bitte Listings mit Papierleim einkleben. KlebeBänder 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. Nastro adesivo può alterare la stampa!

PROGRAM LISTING

MA 103/5 Page of

PROGRAMMAUFLISTUNG
LISTAGE DU PROGRAMME
LISTATO DI PROGRAMMA

Line Zeile Ligne Linea	Keystrokes Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti
	109 RDTAX	
	110 RCL 15	
	111 X≠Y?	n prüfen
	112 GTO a	
	113 GTO 00	
	114*LBL E	
	115 SF 02	Funktion graphisch
	116 XEQ 07	darstellen
	117 CF 02	
	118 BEEP	
	119 RTN	
	120*LBL C	
	121 SF 01	einzelne Werte berechnen
	122 CF 02	
	123 GTO 07	
	124*LBL D	
	125 CF 01	eine Folge von Werten
	126 CF 02	berechnen
	127*LBL 07	
	128 SF 21	
	129 "T="	
	130 XEQ 02	T speichern
	131 STO 12	
	132 FS? 02	
	133 GTO "PRPLOT"	
	134*LBL 04	
	135 "T1="	
	136 PROMPT	t1 speichern
	137 STO 08	
	138 FS? 01	
	139 GTO 03	
	140 "dT="	
	141 XEQ 02	ΔT speichern
	142 STO 10	
	143 RCL 08	
	144*LBL "FT"	
	145*LBL 03	
	146 STO 08	Berechnung von f(t)
	147 RCL 13	
	148 STO 14	Pointer setzen
	149 RCL 16	
	150 2	
	151 /	
	152 STO 15	
	153*LBL 00	
	154 RCL IND 14	Schleife
	155 RCL 08	
	156 *	
	157 RCL 12	
	158 /	
	159 2	
	160 *	
	161 PI	k.π.x.t/T
	162 *	

Line Zeile Ligne Linea	Key pressed Tastenfolge Touches Tasti	Comments Kommentar Commentaires Commenti
	163 ISG 14	
	164 RCL IND 14	b (f)
	165 ISG 14	
	166 RCL IND 14	a (c)
	167 FS? 00	
	168 P-R	
	169 RCL Z	
	170 1	
	171 P-R	
	172 ST* Z	
	173 RDN	
	174 ST* Z	
	175 RDN	
	176 +	
	177 ST+ 15	$a_k \cdot \cos + b_k \cdot \sin$
	178 ISG 14	
	179 GTO 00	
	180 RCL 15	Ende der Berechnung
	181 FS? 02	
	182 RTN	
	183 CLA	
	184 ARCL 08	
	185 "↦ "	
	186 ARCL 15	
	187 AVIEW	
	188 FS? 01	
	189 GTO 04	
	190 RCL 10	t + Δt
	191 ST+ 08	
	192 RCL 12	
	193 RCL 08	
	194 X<=Y?	
	195 GTO 03	nächstes t
	196 BEEP	
	197 ADV	
	198 RTN	
	199 END	
	LBL"INFOUR	
	LBL"FT	
	END 398 BYTES	

Please use paper glue to attach listings. Adhesive tape may affect print!
Bitte Listings mit Papierleim einkleben. Klebeband 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 la stampa!

Programm MSMIMA SIZE 010

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 Σ+(Σ-) zur Symbolisierung der Eingabe angezeigt. Diese erfolgt also wie gewohnt durch Σ+, Löschung eines versehentlich falsch eingegebenen Wertes durch Σ-.

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

```
LBL'MSMIMA
END
105 BYTES


 01*LBL "MSM
IMA"
 02 ΣREG 04
 03 CLΣ
 04 9 E99
 05 STO 00
 06 CHS
 07 STO 01
 08 CLST
 09 " Σ+ <Σ-
>"
 10 SF 27
 11 PROMPT
 12*LBL A
 13 RCL 00
 14 STO 02
 15 X<>Y
 16 X<Y?
 17 STO 00
 18 RCL 01
 19 STO 03
 20 X<>Y
 21 X>Y?
 22 STO 01
 23 0
 24 X<>Y
 25 Σ+
 26 RTN
 27*LBL a
 28 RCL 02
 29 STO 00
 30 RCL 03
 31 STO 01
 32 0
 33 R↑
 34 Σ-
 35 RTN
 36*LBL B
 37 "M="
 38 MEAN
 39 ARCL X
 40 AVIEW
 41 FC? 55
 42 STOP
 43 "S="
 44 SDEV
 45 ARCL X
 46 AVIEW
 47 FC? 55
 48 STOP
 49 "MIN="
 50 ARCL 00
 51 AVIEW
 52 FC? 55
 53 STOP
 54 "MAX="
 55 ARCL 01
 56 AVIEW
 57 ΣREG 11
 58 END
```

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×n-Matrix A mit den Elementen a_{ij} (i=1,...,m; j=1,...,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 ≥ 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×n+3 Register in Anspruch, die folgendermaßen belegt sind:

R_S	:	"A"
R_{S+1}	:	m
R_{S+2}	:	n
R_{S+3}...$R_{S+2+m×n}$	:	a_{ij} (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 $R_{S+2+m×n}$ enthalten die m×n Matrixelemente a_{ij} zeilenweise fortlaufend.

2. Adressierung

Die Registeradresse r eines Matrixelementes a_{ij} der m×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:

R_{ALPHA}	:	Name der Ergebnismatrix
R_{00}	:	Konstante oder sonstige Hilfsgröße
R_{01}	:	Startadresse der 1. Matrix
R_{02}	:	Startadresse der 2. Matrix
R_{03}	:	Startadresse der Ergebnismatrix
R_{04}	:	Element der 1. Matrix
R_{05}	:	Element der Ergebnismatrix
R_{06}	:	1. Schleifenindex
R_{07}	:	2. Schleifenindex
R_{08}	:	3. Schleifenindex
R_{09}	:	Re-Initialisierung des 2. Schleifenindex
R_{10}	:	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 01) oder spaltenweise (SF 01) 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 mxm-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:

$$C = A + k * B$$

bzw. $c_{ij} = a_{ij} + k * b_{ij}$ $(i=1,\ldots,m;\ j=1,\ldots,n)$

Für k = 1 ergibt sich die normale Matrizenaddition. Für $A = \underline{O}$ (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 $\underline{O}$ braucht nicht gespeichert zu sein. Die Ergebnismatrix $\overline{C}$ 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 mxm-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 mxn+3 Register ist erlaubt, wobei dann natürlich "alte" Elemente durch "neue" überschrieben werden.

5.8 MTTRA
Dieses Unterprogramm erzeugt die Transponierte einer mxn-Matrix A, nämlich die nxm-Matrix B, die aus A durch Ver-

tauschen von Reihen und Spalten hervorgeht:

$$b_{ji} = a_{ij} \quad (i=1,\ldots,m;\ j=1,\ldots,n)$$

5.9 MTMUL

Gegeben seien die lxm-Matrix A und die mxn-Matrix B (Spaltenzahl von A = Reihenzahl von B). MTMUL erzeugt die lxn-Produktmatrix C = A * B nach

$$c_{ik} = \sum_{j=1}^{m} a_{ij}b_{jk} \quad (i=1,\ldots,l;\ j=1,\ldots,m;\ k=1,\ldots,n)$$

Im allgemeinen ist A * B ≠ B * A !

5.10 MTINV

Dieses Unterprogramm erzeugt die Inverse $B = A^{-1}$ einer mxm-Matrix A. Es gilt

$$B * A = A * B = E$$

mit E als Einheitsmatrix (siehe Punkt 5.4). B ist dann natürlich auch eine mxm-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 * 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*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:

S(A) = (S(C) = 11
S(B) = 23
S(E) = 35

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
PT"
 02 "MTEIN"
 03 AVIEW
 04 RSUB
 05 "A"
 06 3
 07 ENTER↑
 08 ENTER↑
 09 11
 10 XEQ "MTE
IN"
 11 CF 01
 12 11
 13 XEQ "MTA
US"
 14 "MTSHF"
 15 AVIEW
 16 RSUB
 17 11
 18 ENTER↑
 19 23
 20 XEQ "MTS
HF"
 21 "MTINV"
 22 AVIEW
 23 RSUB
 24 "C"
 25 11
 26 XEQ "MTI
NV"
 27 SF 01
 28 11
 29 XEQ "MTA
US"
 30 "MTMUL"
 31 AVIEW
 32 RSUB
 33 "E"
 34 11
 35 ENTER↑
 36 23
 37 ENTER↑
 38 35
 39 XEQ "MTM
UL"
 40 CF 01
 41 35
 42 XEQ "MTA
US"
 43 "MTSPU"
 44 AVIEW
 45 RSUB
 46 35
 47 ENTER↑
 48 11
 49 XEQ "MTS
PU"
 50 FIX 2
 51 ADV
 52 "SPUR<E>
="
 53 ARCL 11
 54 AVIEW
 55 END
```

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 ?
 4.000      RU
             N
A1.2 ?
 7.000      RU
             N
A1.3 ?
 2.000      RU
             N
A2.1 ?
-6.000      RU
             N
A2.2 ?
 1.000      RU
             N
A2.3 ?
 0.000      RU
             N
A3.1 ?
 2.000      RU
             N
A3.2 ?
 7.000      RU
             N
A3.3 ?
 3.000      RU
             N
```

```
A1.1=4.0000
A1.2=7.0000
A1.3=2.0000

A2.1=-6.0000
A2.2=1.0000
A2.3=0.0000

A3.1=2.0000
A3.2=7.0000
A3.3=3.0000

MTSHF
MTINV

C1.1=0.0600
C2.1=0.3500
C3.1=-0.0800

C1.2=-0.1400
C2.2=0.1600
C3.2=-0.2800

C1.3=-0.0400
C2.3=-0.2400
C3.3=0.9200

MTMUL

E1.1=1.0000
E1.2=0.0000
E1.3=0.0000

E2.1=0.0000
E2.2=1.0000
E2.3=0.0000

E3.1=0.0000
E3.3=0.0000
E3.3=1.0000

MTSPU

SPUR(E)=3.00
```

5.1 MTAIJ

```
LBL"MTAIJ
END
25 BYTES

n=S+2+(i-1)
        *m+J

A:
T:
Z: j
Y: i
X: S(A)

 01*LBL "MTA
IJ"
 02 2
 03 +
 04 X<>Y
 05 INT
 06 1
 07 -
 08 RCL IND
Y
 09 *
 10 +
 11 +
 12 INT
 13 RTN
 14 END
```

5.2 MTEIN

```
LBL"MTEIN
END
88 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

REIHENWEISE
EINGABE DER
m,n-MATRIX A

A: NAME(A)
T:
Z: n
Y: m
X: S(A)

 01*LBL "MTE
IN"
 02 ASTO IND
 X
 03 STO 03
 04 1
 05 +
 06 X<>Y
 07 STO IND
Y
 08 1 E3
 09 /
 10 1
 11 +
 12 STO 06
 13 RDN
 14 1
 15 +
 16 X<>Y
 17 STO IND
Y
 18 1 E3
 19 /
 20 1
 21 +
 22 STO 07
 23*LBL 01
 24 RCL 07
 25 STO 09
 26*LBL 02
 27 CLA
 28 ARCL IND
 03
 29 FIX 0
 30 SF 29
 31 RCL 06
 32 INT
 33 ARCL X
 34 CF 29
 35 RCL 09
 36 INT
 37 ARCL X
 38 "+ ?"
 39 X<>Y
 40 RCL 03
 41 XEQ "MTA
IJ"
 42 FIX 3
 43 PROMPT
 44 STO IND
Y
 45 ISG 09
 46 GTO 02
 47 ISG 06
 48 GTO 01
 49 RTN
 50 END
```

5.3 MTAUS

```
LBL"MTAUS
END
100 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

REIHENWEISE
(CF 01) ODER
SPALTENWEI-
SE (SF 01)
AUSGABE DER
m,n-MATRIX A

A:
T:
Z:
Y:
X: S(A)

 01*LBL "MTA
US"
 02 STO 01
 03 1
 04 +
 05 RCL IND
X
 06 1 E3
 07 /
 08 1
 09 +
 10 X<>Y
 11 1
 12 +
 13 X<>Y
 14 RCL IND
Y
 15 1 E3
 16 /
 17 1
 18 +
 19 FS? 01
 20 X<>Y
 21 STO 07
 22 X<>Y
 23 STO 06
 24 ADV
 25*LBL 01
 26 RCL 07
 27 STO 09
 28*LBL 02
 29 CLA
 30 ARCL IND
 01
 31 RCL 09
 32 INT
 33 RCL 06
 34 INT
 35 FS? 01
 36 X<>Y
 37 FIX 0
 38 SF 29
 39 ARCL X
 40 CF 29
 41 X<>Y
 42 ARCL X
 43 "+*"
 44 X<>Y
 45 RCL 01
 46 XEQ "MTA
IJ"
 47 FIX 4
 48 RCL IND
X
 49 RND
 50 ARCL X
 51 AVIEW
 52 FC? 55
 53 STOP
 54 ISG 09
 55 GTO 02
 56 ADV
 57 ISG 06
 58 GTO 01
 59 RTN
 60 END
```

5.4 MT111

```
LBL"MT111
END
68 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

ERZEUGUNG
DER m,m-EIN-
HEITSMATRIX
E

A: NAME(E)
T:
Z:
Y: m
X: S(E)

 01*LBL "MT1
11"
 02 ASTO IND
 X
 03 STO 03
 04 1
 05 +
 06 X<>Y
 07 STO IND
Y
 08 X<>Y
 09 1
 10 +
 11 X<>Y
 12 STO IND
Y
 13 1 E3
 14 /
 15 1
 16 +
 17 STO 06
 18 STO 07
 19*LBL 01
 20 RCL 07
 21 STO 09
 22*LBL 02
 23 RCL 09
 24 RCL 06
 25 /
 26 INT
 27 LASTX
 28 /
 29 INT
 30 STO 05
 31 RCL 09
 32 RCL 06
 33 RCL 03
 34 XEQ "MTA
IJ"
 35 RCL 05
 36 STO IND
Y
 37 ISG 09
 38 GTO 02
 39 ISG 06
 40 GTO 01
 41 RTN
 42 END
```

5.5 MTADD

```
LBL"MTADD
END
108 BYTES

UNTERPROGR.
MTAIJ ERFOR-
DERLICH !

C=A+k*B

S(C)=S(A)
ODER
S(C)=S(B)
ERLAUBT.

WENN A=0,
MUSS S(A)=0
GESETZT WER-
DEN.

A: NAME(C)
T: S(A)
Z: S(B)
Y: k
X: S(C)

 01*LBL "MTA
DD"
 02 ASTO IND
 X
 03 STO 03
 04 RDN
 05 STO 00
 06 RDN
 07 STO 02
 08 RDN
 09 STO 01
 10 X<> T
 11 1
 12 +
 13 X<>Y
 14 1
 15 +
 16 RCL IND
Y
 17 STO IND
Y
 18 1 E3
 19 /
 20 1
 21 +
 22 STO 06
```

```
 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*LBL 01
 37 RCL 07
 38 STO 09
 39*LBL 02
 40 RCL 09
 41 RCL 06
 42 RCL 02
 43 XEQ "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 02
 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 XEQ "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
82 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 MTTRA

```
LBL'MTTRA
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 XEQ "MTA
IJ"
 41 RCL IND
X
 42 STO 04
 43 RCL 06
 44 RCL 09
 45 RCL 03
 46 XEQ "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 "MTA
IJ"
 57 RCL IND
X
 58 STO 04
 59 RCL 09
 60 RCL 10
 61 RCL 02
 62 XEQ "MTA
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 "MTA
IJ"
 73 RCL 05
 74 STO IND
Y
 75 ISG 09
 76 GTO 02
 77 ISG 06
 78 GTO 01
 79 RTN
 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 RCL 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 RCL 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 08
 88 RCL 02
 89 STO 04
 90*LBL 07
 91 RCL 06
 92 XEQ 08
 93 STO 06
 94 RCL 07
 95 XEQ 08
 96 STO 07
 97 RCL 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 RTN
109*LBL 08
110 1
111 +
112 RCL 01
113 -
114 X>0?
115 RTN
116 RCL 01
117 +
118 RTN
119*LBL 10
120 RCL 00
121 XEQ "MTA
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...11 sind aus Abb. 2 ersichtlich.

a_{11}	a_{12}	a_{13}	b_1
a_{21}	a_{22}	a_{23}	b_2
a_{31}	a_{32}	a_{33}	b_3

Abb. 1

2	3	4	5	6	7
9"	20"	37"	1'03"	1'37"	2'21"

8	9	10	11
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 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 $a_{11}...a_{33}$ 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 ≠Ø gefunden hat. Falls das Gleichungssystem keine eindeutige Lösung hat, versagt das Programm.

$$\begin{pmatrix} 2 & 3 & 0 & 5 \\ 1 & 1 & 1 & 0 \\ 6 & 7 & 2 & 7 \end{pmatrix} (1) \qquad \begin{pmatrix} 2 & 3 & 0 & 5 \\ 0 & -0{,}5 & 1 & -2{,}5 \\ 0 & -2 & 2 & -8 \end{pmatrix} (2)$$

$$\begin{pmatrix} 2 & 3 & 0 & 5 \\ 0 & 1 & -2 & 5 \\ 0 & 1 & -1 & 4 \end{pmatrix} (3) \qquad \begin{pmatrix} 1 & 0 & 3 & -5 \\ 0 & 1 & -2 & 5 \\ 0 & 0 & 1 & -1 \end{pmatrix} (4)$$

$$\begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix} (5)$$

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-
	a1.2=3,ØØØØ	eingabe; während Anzeige (PSE)
	...	Wertänderung möglich.
	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"
     CF 00
     CF 29
     "RANG?"
     PROMPT
     STO 03
     GTO 08
     LBL A
     SF 00
10   LBL 08
     FIX 0
     RCL 03
     1 E3
     /
     1
     +
     STO 00
     STO 01
     STO 02
20   9
     STO 05
     RCL 03
     X/2
     LASTX
     +
     8
     +
     STO 04
     LBL 01
30   "a"
     ARCL 00
     "├."
     ARCL 01
     XEQ 00
     ISG 01
     GTO 01
     "b"
     ARCL 00
     XEQ 00
40   RCL 03
     ST- 01
     ISG 00
     GTO 01
     CF 00
     2
     +
     1 E5
     /
     9
50   +
     STO 00
     "READY"
     PROMPT
     GTO 02
     LBL 00
56   FS? 00
     GTO 09
     PROMPT
     STO IND 05
60   LBL 10
     FIX 0
     1
     ST+ 05
     RTN
     LBL 09
     "├="
     FIX 4
     RCL IND 05
     RND
70   ARCL X
     AVIEW
     PSE
     STO IND 05
     GTO 10
     LBL 02
     0
     STO 08
     RCL 00
     STO 05
80   RCL 02
     INT
     8
     +
     RCL 03
     9
     +
     1 E3
     /
     +
90   STO 01
     LBL 15
     RCL IND 05
     STO 06
     X=0?
     XEQ 03
     LBL 04
     RCL 05
     INT
     RCL 01
100  INT
     X≠Y?
     GTO 07
     RCL 07
     2
     +
     RCL 02
     INT
     -
     ST+ 01
110  GTO 06
111  LBL 07
     RCL IND 01
     STO 07
     LBL 05
     RCL 06
     ST/ IND 05
     RCL IND 05
     RCL 07
     *
120  ST- IND 01
     1
     ST+ 05
     ISG 01
     GTO 05
     1
     STO 06
     LBL 06
     RCL 03
     1
130  +
     1 E3
     /
     RCL 02
     INT
     1
     -
     +
     ST+ 01
     RCL 00
140  STO 05
     RCL 01
     RCL 04
     X>Y?
     GTO 04
     LBL 12
     ISG 00
     CLD
     ISG 02
     GTO 02
150  RCL 03
     9
     +
     RCL 03
     1
     +
     1 E5
     /
     +
     STO 05
160  RCL 02
     FRC
     1
     +
     STO 06
     LBL 13
166  "X"
     FIX 0
     ARCL 06
     "├="
170  FIX 4
     ARCL IND 05
     PROMPT
     ISG 05
     CLD
     ISG 06
     GTO 13
     STOP
     LBL 03
     RCL 05
180  INT
     STO 06
     RCL 02
     INT
     RCL X
     RCL 08
     +
     RCL 03
     1
     +
190  *
     8
     +
     +
     LASTX
     RCL 03
     1
     +
     1 E3
200  /
     +
     STO 07
     LBL 11
     RCL IND 06
     X<> IND 07
     STO IND 06
     ISG 06
     CLD
     ISG 07
210  GTO 11
     1
     ST+ 08
     GTO 15
214  END
```

Lineare Regression mit Fehlerrechnung

```
01*LBL "LINF"
CLRG  SF 21  EREG 04
"GIVE DATES"  AVIEW
RCL 08  RCL 04  RCL 06
RCL 09  /  *  -  RCL 07
RCL 06  X↑2  RCL 09  /
-  STO 08  /  STO 03
MEAN  X<>Y  RCL 03  *  -
STO 02  "a="  ARCL X
AVIEW  "b="  ARCL 03
AVIEW  RCL 07  RCL 03
X↑2  *  RCL 05  RCL 09
RCL 02  X↑2  *  +  +
RCL 04  RCL 06  RCL 03
*  -  RCL 02  *  RCL 03
RCL 08  *  +  ST+ X
RCL T  X<>Y  -  RCL 09
2  -  /  SQRT  STO 01
"S.Y="  ARCL X  AVIEW
RCL 01  RCL 07  RCL 08
RCL 09  *  /  SQRT  *
"S.a="  ARCL X  AVIEW
RCL 01  RCL 08  1/X
SQRT  *  "S.b="  ARCL X
AVIEW

89*LBL 01
"X=?"  PROMPT  RCL 03  *
RCL 02  +  "Y="  ARCL X
AVIEW  GTO 01  .END.
```

Formeln:

$$b = \frac{\Sigma x_i y_i - 1/n \; \Sigma y_i \; \Sigma x_i}{\Sigma x_i^2 - 1/n \; (\Sigma x_i)^2}$$ Steigung

$$a = \bar{y} - b.\bar{x}$$ Achsenabschnitt

Fehler (Standardabweichung)

Fehler des Fkt.wertes:

$$s_y = \sqrt{\frac{(y_i - (bx_i + a))^2}{n - 2}}$$

Fehler der Steigung:

$$s_b = \sqrt{\frac{n}{n \; \Sigma x_i^2 - (\Sigma x_i)^2}} \cdot s_y$$

Fehler des Achsenabschnittes:

$$s_a = \sqrt{\frac{\Sigma x_i^2}{n \; \Sigma x_i^2 - (\Sigma x_i)^2}} \cdot s_y$$

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; Σ+ , 2; ENTER; 8; Σ+ , 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 Σ+, Σ- drücken.

Gerald Krampe

HP-41 C

Programmpaket INTERPOLATION

Inhalt:

1. Allgemeines

Es seien paarweise verschiedene Punkte $x_0, x_1, \ldots, x_n$ mit dazugehörigen Funktionswerten $y_0, y_1, \ldots, y_n$ gegeben. Gesucht wird ein Polynom vom Grade n (oder eine rationale Funktion) f(x) mit $f(x_k) = y_k$ für k=0,..,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

$$(1) \qquad p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n.$$

Zur Berechnung dieses Polynoms werden sogenannte dividierte Differenzen Δ_i^k verwendet, die rekursiv wie folgt definiert sind:

$$(2) \qquad \Delta_i^0 = y_i \quad (i=0,..,n)$$
$$\Delta_i^k = \frac{\Delta_i^{k-1} - \Delta_{i+1}^{k-1}}{x_i - x_{i+k}}$$

Mit $b_k = \Delta_k^k$, k=0,..,n gilt dann für das Interpolationspolynom:

$$(3) \qquad q(x) = b_0 + b_1(x-x_0) + \ldots + b_n(x-x_0)(x-x_1)\ldots(x-x_{n-1}).$$

Mit einem modifizierten Hornerschema (vgl. [1]) erhält man aus q(x) eine weitere Darstellung des Interpolationspolynoms:

$$(4) \qquad r\xi(x) = c_0 + c_1(x-\xi) + \ldots + c_n(x-\xi)^n.$$

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 Hornerschema 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) Hornerschemas.

3. Rationale Interpolation

Für die rationale Interpolation wird der sogenannte Thielesche Kettenbruch verwendet, d.h. die rationale Interpolationsfunktion R(x) sieht wie folgt aus:

$$R(x) = c_o + \cfrac{x-x_o}{c_1 + \cfrac{x-x_1}{c_2 + \ddots \cfrac{x-x_{n-1}}{c_n}}} \tag{5}$$

Zur Berechnung der c_k werden die sog. inversen Differenzen $\overset{k}{i}$ benutzt, die rekursiv wie folgt definiert sind:

$$\overset{o}{i} = y_i, \quad i=o,..,n$$

$$\overset{k+1}{i} = \frac{\overset{k}{i} - \overset{k}{k}}{\overset{k}{i} - \overset{k}{k}} \tag{6}$$

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(ξ) wird der Kettenbruch (5) von "unten nach oben" ausgewertet.

4. Programmbeschreibungen

In den verschiedenen Unterprogrammen werden die HP 41 C-Register R00 bis R07 benutzt. Die Koeffizienten $\{a_k\},\{b_k\}$ ... können ab Register R08 abgelegt werden. Die einzelnen Werte {..} müssen fortlaufend gespeichert werden, z.B. x_o-R08, x_1-R09, x_2-R10.. Für die Rechnungen werden weiterhin die Anfangsadressen der Koeffizientenvektoren benötigt, so gibt z.B. $A(x_o)$ die Adresse von x_o an, im obigen Beispiel also $A(x_o) = 8$. Was in welchem Register abgespeichert ist, zeigt die folgende Tabelle. Ein * 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.

UPrgrm	R00	R01	R02	R03	R04	R05	R06	R07
PM		n	$A(v_o)$	$A(w_o)$		i	j	
DD	*	n	$A(y_o)$	$A(b_o)$	$A(x_o)$	i	*	k
MH	∫	n	$A(b_o)$	$A(c_o)$	$A(x_o)$	i	j	k
BM	∫	n		$A(b_o)$	$A(x_o)$	i	j	
NH	∫	n	$A(a_o)$	$A(c_o)$		*	j	k
BN	∫	n	$A(a_o)$			i		
ID	1	n	$A(y_o)$	$A(c_o)$	$A(x_o)$	i	j	k
BK	∫	n		$A(c_o)$	$A(x_o)$	i	j	

4.1 Unterprogramm PM

Verschiebt $\{v_k\}$ nach $\{w_k\}$, benötigt $A(v_o)$, $A(w_o)$, n.
Wenn $A(v_o) = A(w_o)$ ist, wird nichts verschoben.

4.2 Unterprogramm DD

Berechnet aus $\{x_k\}$, $\{y_k\}$ die dividierten Differenzen $b_k=\Delta_k^k$.
Benötigt $A(x_o)$, $A(y_o)$, $A(b_o)$, n.

4.3 Unterprogramm MH

Modifiziertes Hornerschema. Berechnet aus $\{x_k\}$, $\{b_k\}$ in (3) die Werte $\{c_k\}$ in (4). Benötigt $A(x_o)$, $A(b_o)$, $A(c_o)$, n, ξ.

4.4 Unterprogramm BM

Berechnet $q(\xi)$ und $q'(\xi)$ in (3)
Benötigt $A(x_o)$, $A(b_o)$, n, ξ.

4.5 Unterprogramm NH

Normales Hornerschema. Berechnet $\{c_k\}$ in (4) aus $\{a_k\}$ in (1).
Benötigt $A(a_o)$, $A(b_o)$, n, ξ.

4.6 Unterprogramm BN

Berechnet $p(\xi)$ und $p'(\xi)$ aus (1).
Benötigt $A(a_o)$, n, ξ.

4.7 Unterprogramm ID

Berechnet aus $\{x_k\}$, $\{y_k\}$ die inversen Differenzen $\{c_k\}$ in (5).
Benötigt $A(x_o)$, $A(y_o)$, $A(c_o)$, n.

4.8 Unterprogramm BK

Berechnet den Thieleschen Kettenbruch $R(\xi)$ in (5).
Benötigt $A(x_o)$, $A(c_o)$, n, ξ.

Es ist im allgemeinen möglich, daß die zu berechnenden Koeffizienten die alten Werte überschreiben, z.B. ist im Unterprogramm DD $A(b_o) = A(y_o)$ 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_o,w_o,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_j
  j=j+1
next i
```

MH

```
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=(ξ-x_j)*x + c_i
    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=(ξ-x_i)/x + c_j
  j=j-1
next i
```

BN

```
y=0
x=a_n
for i=n-1(-1)0
  y=y*ξ + x
  x=x*ξ + a_i
next i
```

BM

```
j=n; y=0; x=b_n; j=j-1
for i=n-1(-1)0
  y=(ξ-x_i)*y + x
  x=(ξ-x_i)*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_l-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+ξ*x; c_k=x
  next k
next j
```

```
01♦LBL "PM"
02 RCL 02
03 RCL 03
04 X=Y?
05 RTN
06 RCL 01
07 +
08  E3
09 /
10 RCL 03
11 +
12 STO 05
13 RCL 02
14 ,9
15 +
16 STO 06
17♦LBL 01
18 RCL IND 06
19 STO IND 05
20 ISG 06
21 ISG 05
22 GTO 01
23 END


01♦LBL "ID"
02 XEQ "PM"
03 RCL 04
04 XEQ 04
05 STO 05
06 RCL 03
07 XEQ 04
08 STO 00
09♦LBL 01
10 RCL 05
11 1,001
12 +
13 STO 06
14 RCL 00
15 LASTX
16 +
17 STO 07
18♦LBL 02
19 RCL IND 05
20 RCL IND 06
21 -
22 RCL IND 00
23 RCL IND 07
24 -
25 /
26 STO IND 07
27 ISG 07
28♦LBL 03
29 ISG 06
30 GTO 02
31 ISG 00
32♦LBL 03
33 ISG 05
34 GTO 01
35 RTN
36♦LBL 04
37 STO Y
38 RCL 01
39 +
40  E
41 -
42  E3
43 /
44 +
45 END


01♦LBL "DD"
02 XEQ "PM"
03 RCL IND 03
04 STO 00
05 RCL 03
06 RCL 01
07 +
08  E3
09 /
10 RCL 03
11 +
12  E
13 +
14 STO 07
15♦LBL 01
16 RCL 04
17 ,9
18 +
19 STO 06
20 RCL 07
21 STO 05
22 +
23 RCL 03
24 -
25 X<> 00
26♦LBL 02
27 RCL IND 05
28 -
29 RCL IND 06
30 RCL IND 00
31 -
32 /
33 X<> IND 05
34 ISG 00
35 ISG 06
36 ISG 05
37 GTO 02
38 RCL IND 07
39 STO 00
40 ISG 07
41 GTO 01
42 END


01♦LBL "BH"
02 RCL 03
03  E
04 -
05  E3
06 /
07 RCL 03
08 RCL 01
09 +
10 +
11 STO 05
12 ,
13 RCL IND 05
14 DSE 05
15♦LBL 01
16 RCL 00
17 ST* Z
18 X<>Y
19 ST+ Z
20 *
21 RCL IND 05
22 +
23 DSE 05
24 GTO 01
25 END


01♦LBL "MH"
02 XEQ "PM"
03 RCL 01
04 STO 07
05 RCL 04
06  E
07 -
08 ,1
09 %
10 +
11 STO 06
12♦LBL 02
13 RCL 03
14 RCL 01
15 +
16 STO 05
17 RCL 07
18 ST+ 06
19 RCL IND 05
20 DSE 05
21♦LBL 03
22 RCL 00
23 RCL IND 06
24 -
25 *
26 RCL IND 05
27 +
28 STO IND 05
29 DSE 05
30 DSE 06
31 GTO 03
32 DSE 07
33 GTO 02
34 END


01♦LBL "NH"
02 XEQ "PM"
03 RCL 03
04 RCL 01
05 +
06  E3
07 /
08 RCL 03
09 +
10 STO 06
11♦LBL 01
12 RCL 03
13 RCL 01
14 +
15 STO 07
16 RCL 06
17 STO 05
18 RCL 00
19 ENTER↑
20 ENTER↑
21 ENTER↑
22 CLX
23♦LBL 02
24 *
25 RCL IND 07
26 +
27 STO IND 07
28 DSE 07
29 ISG 05
30 GTO 02
31 ISG 06
32 GTO 01
33 END


01♦LBL "BM"
02 RCL 03
03 RCL 01
04 +
05 STO 06
06 RCL 04
07  E
08 -
09 ,1
10 %
11 +
12 RCL 01
13 +
14 STO 05
15 ,
16 RCL IND 06
17 DSE 06
18♦LBL 01
19 RCL 00
20 RCL IND 05
21 -
22 ST* Z
23 X<>Y
24 ST+ Z
25 *
26 RCL IND 06
27 +
28 DSE 06
29 DSE 05
30 GTO 01
31 END


01♦LBL "BK"
02 RCL 04
03  E
04 -
05 ,1
06 %
07 +
08 RCL 01
09 +
10 STO 05
11 RCL 03
12 RCL 01
13 +
14 STO 06
15 RCL IND 06
16 DSE 06
17♦LBL 01
18 1/X
19 RCL 00
20 RCL IND 05
21 -
22 *
23 RCL IND 06
24 +
25 DSE 06
26 DSE 05
27 GTO 01
28 END


LBL'PM
END          37 BYTES
LBL'DD
END          67 BYTES
LBL'MH
END          56 BYTES
LBL'BM
END          48 BYTES
LBL'NH
END          51 BYTES
LBL'BH
END          40 BYTES
LBL'ID
END          76 BYTES
LBL'BK
END          43 BYTES
```

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:

a) Bestimme q(x), d.h. $\{b_k\}$ in (3), sowie q(.5) und q'(.5)
b) Bestimme p(x), d.h. $\{c_k\}$ für ξ=0, p(.5) und p'(.5)
c) Bestimme $p_1(x)$, d.h. $\{c_k\}$ für ξ=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 ξ=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 $\{c_k\}$

c_0 = 0.2000 = 1/5
c_1 = 5.0000 = -5
c_2 = 1.3333 = 2/15
c_3 = 6.0000 = 6
c_4 = 0.6667 = 2/3

$$R(x) = \frac{1}{5} + \cfrac{x+2}{-5 + \cfrac{x+1}{\frac{3}{15} + \cfrac{x}{6 + \cfrac{x-1}{2/3}}}}$$

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)

Stroinski
Februar 81

```
01*LBL "DA"
02 SF 27
03 STOP
04*LBL A
05 STO 01
06 RDN
07 STO 02
08 RDN
09 STO 03
10 RDN
11 RDN
12 +
13 +
14 2
15 /
16 STO 07
17 X↑2
18 LASTX
19 RCL 02
20 *
21 -
22 RCL 03
23 RCL 01
24 *
25 /
26 SQRT
27 ACOS
28 2
29 *
30 STO 05
31 SIN
32 RCL 03
33 *
34 STO 00
35 RCL 07
36 X↑2
37 LASTX
38 RCL 03
39 *
40 -
41 RCL 02
42 /
43 RCL 01
44 /
45 SQRT
46 ACOS
47 2
48 *
49 STO 06
50 RCL 05
51 XEQ 00
52 STO 04
53 GTO 04
```

Haupt-Label
User-Modus

Routine SSS
a speichern
b speichern
c speichern

Stack ordnen
(a+b+c):2 = s
s speichern

$\sqrt{\frac{s(s-b)}{ac}}$
$\cos\frac{\beta}{2}$
β speichern

h_a errechnen
h_a speichern

$\sqrt{\frac{s(s-c)}{ab}}$
$\cos\frac{\gamma}{2}$
γ speichern

α errechnen
α speichern

```
54*LBL B
55 STO 04
56 RDN
57 STO 03
58 RDN
59 STO 05
60 RCL 04
61 XEQ 00
62 STO 06
63*LBL 06
64 RCL 05
65 RCL 03
66 P-R
67 X<>Y
68 STO 00
69 RCL 06
70 1
71 P-R
72 RDN
73 /
74 STO 02
75 R↑
76 *
77 +
78 STO 01
79 GTO 04
80*LBL C
81 STO 06
82 X<>Y
83 STO 04
84 XEQ 00
85 STO 05
86 X<>Y
87 STO 03
88 GTO 06
89*LBL D
90 STO 02
91 RDN
92 STO 04
93 RDN
94 STO 03
95 RCL 04
96 RCL 02
97 P-R
98 RCL 03
99 -
100 R-P
101 STO 01
102 RCL 03
103 RCL 02
104 RCL 01
105 GTO A
```

Routine USW
α speichern
c speichern
β speichern
γ errechnen
γ speichern

$c\cdot\sin\beta = h_a$ und $c\cdot\cos\beta$ bilden

h_a speichern

$\sin\gamma$ und $\cos\gamma$ bilden
$b = c\cdot\frac{\sin\beta}{\sin\gamma}$

b speichern

$a = c\cdot\cos\beta + b\cdot\cos\gamma$

a speichern

Routine SWW
γ speichern
α speichern
β speichern.
c speichern

Routine SWS
b speichern
α speichern
c speichern
$b\cdot\sin\alpha = h_c$ und $b\cos\alpha$ bilden

$h_c \rightarrow a$

$b\cdot\cos\alpha - c$
a speichern

Stack ordnen

```
106*LBL E
107 STO 06
108 SIN
109 X<>Y
110 STO 02
111 *
112 X<>Y
113 STO 03
114 /
115 ASIN
116 STO 05
117 RCL 06
118 XEQ 00
119 STO 04
120 XEQ 06
121*LBL 05
122 RCL 03
123 RCL 02
124 X<=Y?
125 GTO 02
126 RCL 05
127 COS
128 CHS
129 ACOS
130 STO 05
131 RCL 06
132 XEQ 00
133 STO 04
134 GTO 06
135*LBL 00
136 +
137 COS
138 CHS
139 ACOS
140 RTN
```

Routine SSW
γ speichern
b speichern
c speichern
β speichern
α errechnen
α speichern

1. Lsg. rechnen

2. Lösung möglich?

Für 2. Lösung
$\beta' = \pi - \beta$
errechnen
β' speichern
α' errechnen
α' speichern

2. Lösung rechn.

Unterprogramm zur Berechnung des 3. Winkels

Register-inhalte:

00	Fläche
01	a
02	b
03	c
04	α (*A)
05	β (*B)
06	γ (*C)
07	Umfang

```
141*LBL 04
142 RCL 00
143 RCL 01
144 *
145 2
146 /
147 STO 00
148*LBL 01
149 SF 21
150 ,007
151 RCL 01
152 RCL 02
153 RCL 03
154 +
155 +
156 STO 07
157 "Fabc<A<
B<C U"
158 AVIEW
159*LBL 03
160 RDN
161 VIEW IND
 X
162 ENTER↑
163 ENTER↑
164 3
165 MOD
166 INT
167 X=0?
168 ADV
169 ISG Y
170 GTO 03
171*LBL 02
172 ADV
173 END


SIZE 008
TOT. REG. 40

USER MODUS:
A,B,C,D,E.

WIEDERHOLEN:

FALL A...D:
XEQ 01

FALL E:
XEQ 01:
ZULETZT GE-
ZEIGTE LSG.
XEQ 05:
ALTERNATIV-
LOESUNG.
```

Schluß-Routine
Fläche berechnen und speichern

Ausgabe-Routine
Umfang errechnen und speichern

Text der Merkzeile

Druck/Anzeige

Stack ordnen

Leerzeilen nach 1. 4. und 7. Wert

Die Werte erscheinen in der Reihenfolge:

Fläche
a
b
c
α (*A)
β (*B)
γ (*C)
Umfang.

Im Fall E werden, falls vorhanden, zwei Werte-Serien ausgegeben!

Ohne Drucker hält das Programm nach jedem Wert an. Abruf des nächsten Wertes mit R/S-Taste!

— 6. —
Plotprogramme

Autor: Andreas Trögel

Plotten einer Funktion

LBL PL SIZE 017
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"
02 CF 00
03 AON
04 "FUNKTION ?"
05 PROMPT
06 AOFF
07 ASTO 06
08 "Y MIN?"
09 PROMPT
10 STO 07
11 "Y MAX?"
12 PROMPT
13 STO 08
14 "X MIN?"
15 PROMPT
16 STO 09
17 "X MAX?"
18 PROMPT
19 STO 10
20 "X INC?"
21 PROMPT
22 STO 11
23 "PLOT OF "
24 ARCL 06
25 ACA
26 ADV
27 ADV
28 FIX 1
29 "YMIN:"
30 ACA
31 RCL 07
32 ACX
33 ADV
34 ADV
35 "YMAX:"
36 ACA
37 RCL 08
38 ACX
39 ADV
40 ADV
41 "XMIN:"
42 ACA
43 RCL 09
44 ACX
45 ADV
46 ADV
47 "XMAX:"
48 ACA
49 RCL 10
50 ACX
51 ADV
52 ADV
53 FIX 3
54 "STEPS OF "
55 ARCL 11
56 ACA
57 ADV
58 ADV
59 "------"
60 ASTO L
61 ARCL L
62 ARCL L
63 "+-"
64 ACA
65 ADV
66 132  *SPALTENBREITE-1
67 RCL 08
68 RCL 07
69 -
70 /
71 STO 12
72*LBL 12
73 5 E-3
74 STO 13
75*LBL 01
76 RCL 09
77 XE0 IND 06
78 RCL 07
79 -
80 RCL 12
81 *
82  E
83 +
84 FIX 0
85 RND
86 133    *SPALTENBREITE
87 X<>Y
88 X>Y?
89 ,
90 ENTER↑
91 ,
92 X<=Y?
93 X<>Y
94 STO IND 13
95 RCL 11
96 ST+ 09
97 RCL 10
98 RCL 09
99 X>Y?
100 GTO 02
101 ISG 13
102 GTO 01
103 GTO 03
104*LBL 02
105 SF 00
106 ,
107 STO IND 13
108 ISG 13
109 GTO 02
110*LBL 03
111  E-3
112 ST+ 00
113 ST+ X
114 ST+ 01
115 ST+ X
116 ST+ 02
117 ST+ X
118 ST+ 03
119 ST+ X
120 ST+ 04
121 ST+ X
122 ST+ 05
123 5 E-3
124 STO 13
125*LBL 06
126 RCL 13
127 STO 14
128  E
129 -
130 RCL IND X
131*LBL 07
132 RCL IND 14
133 X<Y?
134 X<>Y
135 STO IND 14
136 RDN
137 ISG 14
138 GTO 07
139 STO IND Y
140 ISG 13
141 GTO 06
142 4 E-3
143 STO 14
144  E-3
145 +
146 STO 13
147*LBL 08
148 ISG 13
149 RCL IND 13
150 INT
151 RCL IND 14
152 INT
153 X≠Y?
154 GTO 09
155 LASTX
156 FRC
157 ST+ IND 13
158 CLX
159 STO IND 14
160*LBL 09
161 ISG 14
162 GTO 08
163 5 E-3
164 STO 13
165 RCL IND 13
166 STO 14
167*LBL 10
168 RCL IND 13
169 INT
170 X=0?
171 GTO 11
172 RCL 14
173 -
174  E
175 -
176 X>0?
177 SKPCOL
178 RCL IND 13
179 INT
180 STO 14
181 LASTX
182 FRC
183  E3
184 *
185 ACCOL
186*LBL 11
187 ISG 13
188 GTO 10
189 133
190 RCL 14
191 -
192 SKPCOL
193 ADV
194 FC?C 00
195 GTO 12
196 BEEP
197 ACA
198 ADV
199 ADV
200 ADV
201 ADV
202 ADV
203 END
```

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"
02 "G-AN"
03 PROMPT
04 ADV
05 SF 12
06 "Q        @@@@σά @α"  17 192 64 64 64 128 129
07 XEQ 01                    16 4 8 32 64 129 4
08 "@@AΓ»(HQ x̄Δ@♦"       16 16 65 6 12 40 72 17
09 XEQ 01                      32 129 2 8 48 0
10 ADV
11 "Q♦↓Γ┌Ē@Q←↓┌£(x"    17 3 7 6 14 31 64 17 3 7
12 RCL \                       17 30 60 120
13 ACSPEC
14 3
15 SKPCOL
16 XEQ 02
17 "Qā(xΩ»ô♦♦"          17 225 199 135 22 60 120
18 XEQ 01                  17 195 135 12 24 0 0
19 0
20 ACCOL
21 2
22 ACCOL
23 124
24 ACCOL
25 ADV
26 "@♦@♦       ♦♦@Eσδ"  16 48 192 16 27 193 129
27 XEQ 01                   9 137 18 0 0 0 0
28 "@H♦♦♦x̄@×Qx̄αΔ"      16 72 146 2 16 1 17 2 4
29 XEQ 01                      8 34 0 0 0
30 "@1»┌Γ"                16 136 49 140 12 14 6
31 XEQ 02
32 ADV
33 "Qα↓=♦x̄@Δ@♦♦♦"      17 248 4 7 1 1 2 16 8 16
34 XEQ 01                        206 0 0 0
35 "Q       ' @»Q♦↓♦♦♦"  17 192 96 32 64 129 12
36 XEQ 01                  17 192 15 192 0 0 0
37 ADV
38 "Q@↕        ♦♦@←↓♦"  17 224 48 15 192 16 3 7
39 XEQ 01                  143 159 128 0 0 0
40 "@←♦♦♦Q@♦Ξ♦"          16 3 247 206 17 204 16
41 XEQ 01                      31 128 0 0 0 0 0
42 1
43 SKPCOL
44 ADV
45 "@tα @←Δ"             16 194 116 130 4 142 32
46 XEQ 01                 16 129 3 227 199 136 8
47 "@  @2x̄α¢p"           16 32 129 231 207 144 32
48 XEQ 01                   16 225 50 2 4 15 112
49 2
50 SKPCOL
51 ADV
52 "@b$ö@@@@x̄∠(P"       16 226 98 36 24 16 48 17
53 XEQ 01                     64 129 2 13 40 80
54 "@       x̄αö Q x̄Δλ♦"16 192 161 2 20 24 32 17
55 XEQ 01                   129 2 8 11 157 0
56 2
57 SKPCOL
58 ADV
59 "@9eä(PQJä&b"         16 57 130 101 20 40 80
60 XEQ 01                 17 74 133 10 20 38 90
61 "@          @'♦♦♦"      16 192 64 96 0 0 0
62 XEQ 02
63 4
64 SKPCOL
65 ADV
66 ADV
67 ADV
68 ADV
69 ADV
70 RTN
71 GTO "GO"
72♦LBL 01
73 RCL \
74 ACSPEC
75♦LBL 02
76 RCL [
77 ACSPEC
78 RTN
```

```
79♦LBL "GO"
80 SF 12
81 "G-AH"
82 PROMPT
83 ADV
84 "θ×          θαδ "16 1 132 192 192 129 129
85 XEQ 01                16 4 8 32 128 128 129
86 "QΘ♦♦"                 17 5 136 160 128 0 0
87 XEQ 02
88 ADV
89 "θ♦♦ Α2e88 θx̄α"      16 32 65 50 28 16 16 32
90 XEQ 01                        64 129 2 4 0 0
91 "Q"pō♦♦"               17 34 112 130 24 0 0
92 XEQ 02
93 ADV
94 "θ×      ⌐≝         16 1 199 207 211 167 124
95 RCL \                   16 6 2 10 38 46 0
96 ACSPEC
97 5
98 SKPCOL
99 XEQ 02
       100 "Qz ∠QΘ♦♦"  17 122 244 193 195 191
101 XEQ 01             124 17 195 135 232 64 0
102 ADV                                      0
103 "Q×x̄↓x̄θδ♦ ōΘ"      17 1 2 7 128 129 2 16 8
104 XEQ 01                     32 130 24 64 0
105 "Qc5♦(ōθ♦♦∠▒ >"   17 99 53 143 60 24 16 13
106 XEQ 01                     226 31 62 0 0 0
107 "θ              ♦♦"  16 251 247 231 192 0 0
108 XEQ 02
109 ADV
110 "θwαδθ HQ♦♦♦♦x̄"      16 123 4 8 16 32 72 17
111 XEQ 01                      194 2 0 0 0 0
112 "ΘΘ♦ θ"Qx̄Θ↓b"        16 16 32 16 130 96 17
113 XEQ 01                 195 135 2 5 7 98 0
114 "ΘPō♦♦"                16 80 247 142 24 0 0
115 XEQ 02
116 ADV
117 "θ♦♦ αδθō♦⌘"L" 16 32 130 4 8 16 24 35
118 XEQ 01                137 34 76 0 0 0
119 "θ⌘£θxp(ρΓx̄" 16 160 198 136 131 15 30
120 XEQ 01          16 128 112 48 12 6 2
121 "θ♦αf ♦"         16 4 15 223 188 0 0
122 XEQ 02
123 ADV
124 96
125 ACCOL
126 "Q⌘£(xQα"     17 195 135 15 30 60 120
127 XEQ 01       17 243 247 227 192 130 4
128 "ΘΘ ΘΘ♦♦Q ∠"   16 16 32 64 64 17 131
129 XEQ 01        199 239 223 191 124 0 0
130 ADV
131 "θδΘ fΘ £"    16 8 16 32 193 131 6 16
132 RCL \           32 243 239 223 159 30
133 ACSPEC
134 2
135 ACCOL
136 ACCOL
137 10
138 SKPCOL
139 XEQ 02
140 2
141 ACCOL
142 0
143 ACCOL
144 ADV
145 ADV
146 ADV
147 ADV
148 ADV
149 END
```

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"
02 CF 12
03 CF 13
04 AON
05 "NAME 1?"
06 PROMPT
07 ASTO 05
08 "NAME 2?"
09 PROMPT
10 ASTO 06
11 AOFF
12*LBL "NEU"
13 167
14 "Y MIN?"
15 PROMPT
16 STO 02
17 "Y MAX?"
18 PROMPT
19 STO 00
20 X<>Y
21 -
22 /
23 STO 08
24 "X MIN?"
25 PROMPT
26 STO 01
27 STO 07
28 "X MAX?"
29 PROMPT
30 STO 03
31 "X INC?"
32 PROMPT
33 STO 04
34 "PLOT "
35 ARCL 05
36 "+ AND "
37 ARCL 06
38 PRA
39 ENG 3
40 "Y: "
41 ARCL 02
42 "+ TO "
43 ARCL 00
44 PRA
45 "X: "
46 ARCL 01
47 "+ TO "
48 ARCL 03
49 PRA
50 "Δx="
51 ARCL 04
52 PRA
53 "-------"
54 ASTO L
55 ARCL I
56 ARCL L
57 ARCL L
58 PRA
59 CF 01
60*LBL 00
61 9.022
62 STO 00
63*LBL 01
64 RCL 07
65 XEQ IND 05
66 CF 00
67*LBL 02
68 RCL 02
69 -
70 RCL 08
71 *
72 I
73 +
74 FIX 0
75 RND
76 160
77 X<>Y
78 X<0?
79 CLX
80 X>Y?
81 CLX
82 STO IND 00
83 FS? 00
84 RTN
85 ISG 00
86 RCL 07
87 SF 00
88 XEQ IND 00
89 XEQ 02
90 RCL 04
91 ST+ 07
92 RCL 07
93 RCL 03
94 X<Y?
95 GTO 04
96 ISG 00
97 GTO 01
98 XEQ 05
99 4
100 RCL 04
101 *
102 ST+ 07
103 RCL 07
104 RCL 03
105 X<Y?
106 PRA
107 X<Y?
108 RTN
109 GTO 00
110*LBL 04
111 SF 01
112 CLX
113 STO IND 00
114 ISG 00
115 GTO 04
116*LBL 05
117 9.02302
118 STO 00
119 I
120 +
```

```
121 STO 01
122 5 E-4
123*LBL 03
124 ST+ X
125 ST+ IND 00
126 ST+ IND 01
127 ISG 00
128 ISG 01
129 GTO 03
130 10.022
131 STO 01
132*LBL 06
133 RCL 01
134 STO 00
135 1
136 -
137 RCL IND X
138*LBL 07
139 RCL IND 00
140 X<Y?
141 X<>Y
142 STO IND 00
143 RDN
144 ISG 00
145 GTO 07
146 STO IND Y
147 ISG 01
148 GTO 06
149 9.021
150 STO 00
151 1 E-3
152 +
153 STO 01
154*LBL 08
155 ISG 01
156 RCL IND 00
157 INT
158 RCL IND 01
159 INT
160 X≠Y?
161 GTO 09
162 LASTX
163 FRC
164 RCL IND 00
165 FRC
166 X=Y?
167 GTO 12
168 X<Y?
169 GTO 13
170 X<>Y
171 -
172*LBL 13
173 ST+ IND 01
174*LBL 12
175 CLX
176 STO IND 00
177*LBL 09
178 ISG 00
179 GTO 08
180 9.022
181 STO 00
182 CLX
183 STO 01
184*LBL 10
185 RCL IND 00
186 INT
187 X=0?
188 GTO 11
189 ENTER↑
190 X<> 01
191 -
192 1
193 -
194 SKPCOL
195 RCL IND 00
196 FRC
197 1 E3
198 *
199 ACCOL
200*LBL 11
201 ISG 00
202 GTO 10
203 PRBUF
204 FS?C 01
205 PRA
206 RTN
207*LBL "X=0"
208 .
209 .END.
```

Beispiel :

```
PLOT SIN↑2 AND COS
Y: -1.000E0 TO 1.000E0
X: 0.000E0 TO 360.0E0
Δx=1.800E0
```

Autor: Andreas Trögel

MULTIPLOTTER (für bis zu 6 Funktionen)

LBL PLOT SIZE 027 Drucker (MAN) Synthetik
657 Bytes

Die Plottfunktion des HP-Druckers in der ROM-Routine ist sehr beschränkt. 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 nebeneinandergeklebt 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 Funktionsnamen.

```
01*LBL "PLOT"
02*LBL 00
03 6
04 "NO. OF F(X)?"
05 PROMPT
06 X>0?
07 X>Y?
08 GTO 00
09 .1
10 %
11 ST+ Y
12 13
13 LASTX
14 %
15 +
16 +
17 ISG X
18 STO 00
19 X<>Y
20 STO 11
21 +
22 1.1
23 AON
24 FIX 0
25 CF 29
26*LBL 01
27 CF 23
28 "NAME F"
29 ARCL X
30 "= ?"
31 PROMPT
32 FC? 23
33 GTO 01
34 ASTO IND Y
35 ISG X
36 ISG Y
37 GTO 01
38 AOFF
39*LBL "NEW"
40*LBL 02
41 "SPLITS"
42 PROMPT
43 STO 07
44 X<=0?
45 GTO 02
46*LBL 03
47 "Y MIN?"
48 PROMPT
49 STO 01
50 "Y MAX?"
51 PROMPT
52 STO 02
53 RCL 01
54 -
55 X<=0?
56 GTO 03
57 RCL 07
58 /
59 STO 03
60 "AXIS"
61 PROMPT
62 STO 08
63*LBL 04
64 "X MIN?"
65 PROMPT
66 STO 04
67 "X MAX?"
68 PROMPT
69 STO 05
70 RCL 04
71 X<>Y
72 X<=Y?
73 GTO 04
74 "X INC?"
75 PROMPT
76 X>0?
77 GTO 05
78 RCL 04
79 RCL 05
80 -
81 X<>Y
82 /
83*LBL 05
84 STO 06
85 RCL 00
86 RCL 11
87 +
88 RDN
89 "*ox*+="
90 ASTO [
91*LBL 06
92 ASTO Y
93 ASHF
94 ASTO X
95 ACX
96 " is Plot of: "
97 ARCL IND T
98 ACA
99 PRBUF
100 CLA
101 X<>Y
102 X<> [
103 ISG T
104 GTO 06
105 FIX 2
106 " Y: "
107 ARCL 01
108 "+ to "
109 ARCL 02
110 "+ ?"
111 PRA
112 " X:"
113 ARCL 04
114 "+ to "
115 ARCL 05
116 "+ ↓"
117 PRA
118 " Axis: "
119 ARCL 08
120 PRA
121 FIX 3
122 " Steps of "
123 ARCL 06
124 PRA
125 FIX 0
126 7
127 CHS
128 STO 12
129 RCL 03
130 161
131 /
132 X<> 07
133*LBL 15
134 " SPLIT "
135 ARCL X
136 AVIEW
137 SF 13
138 XEQ 14
139 RCL 00
140 RCL 01
```

```
141 -
142 RCL 03
143 X<>Y
144 X>Y?
145 CHS
146 RCL 07
147 /
148 RND
149 .1
150 +
151 X<0?
152 RCL 12
153 STO 09
154 RCL 04
155 STO 10
156 CLA
157 SF 03
158 RCL d
159 STO [
160 ASTO [
161 "⊢ "
162 RCL [
163 STO ]
164*LBL 16
165 RCL ]
166 X<> d
167 "⊣$004⊣"
168 STO J
169 RCL 00
170 STO \
171 RCL 09
172 STO 13
173*LBL 07
174 RCL \
175 RCL 11
176 +
177 RCL IND X
178 RCL 10
179 XEQ IND Y
180 RCL 01
181 -
182 RCL 03
183 X<>Y
184 X>Y?
185 CHS
186 RCL 07
187 /
188 RND
189 RCL [
190 FRC
191 +
192 X<0?
193 RCL 12
194 STO IND \
195  E2
196 ST/ [
197 ISG \
198 GTO 07
199 RCL 00
200 STO [
201 DSE [
202*LBL 08
203 ENTER↑
204 ABS
205 DSE X
206 BEEP
207 RCL IND X
208*LBL 09
209 RCL IND L
210 X<Y?
211 X<>Y
212 STO IND L
213 RDN
214 ISG L
215 GTO 09
216 STO IND Y
217 X<> Z
218 ISG X
219 GTO 08
220 RCL ]
221 X<> d
222 STO ]
223 RCL 12
224 STO \
225*LBL 10
226 RCL IND [
227 X<0?
228 GTO 11
229 ENTER↑
230 FRC
231 ST- Y
232  E2
233 *
234  E1
235 X≠Y?
236 GTO 05
237 RCL [
238 ISG X
239 CHS
240 X>0?
241 GTO 10
242 RDN
243 CLX
244 RCL IND T
245 INT
246 RCL Z
247 -
248 4
249 X<=Y?
250 GTO 10
251*LBL 12
252 RCL [
253 DSE X
254 BEEP
255 RCL IND X
256 STO \
257 STO IND [
258 GTO 11
259*LBL 10
260 X<> T
261 ENTER↑
262 X<> \
263 -
264 4
265 -
266 X<0?
267 GTO 11
268 SKPCOL
269 119
270 ACCOL
271 3
272 ST- \
273 GTO 11
274*LBL 05
275 X<> Z
276 ENTER↑
277 X<> \
278 -
279 RCL 12
280 +
281 X<0?
282 GTO 12
283 SKPCOL
284 X<>Y
285 ACCHR
286*LBL 11
287 ISG [
288 GTO 10
289 PRBUF
290 RCL 06
291 ST+ 10
292 RCL 05
293 RCL 10
294 X<=Y?
295 GTO 16
296 XEQ 14
297 RCL 03
298 ST+ 01
299 RCL 02
300 RCL 01
301 -
302 RCL 03
303 /
304 RND
305 CF 13
306 X>0?
307 GTO 15
308 BEEP
309 "End of "
310 "⊢Multiplotting"
311 PRA
312 ADV
313 ADV
314 ADV
315 ADV
316 RTN
317*LBL 14
318 3
319 SKPCOL
320 127
321 ACCOL
322 "--------"
323 RCL [
324 STO \
325 STO ]
326 "⊢--"
327 ACA
328 X<>Y
329 ACCOL
330 PRBUF
331 .END.

ZEILE 89
F6:0A:6F:78:2A:2B:01

ZEILE 167
F7:07:98:84:24.30:10:07
```

Autor: Robert Klauc

Derby (Pferderennen)

LBL DERBY	SIZE 030	Peripherie:
	1109 Bytes	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"
02 CLRG
03 FIX 0
04 CF 29
05 SF 12
06 "0<X<1"
07 PROMPT
08 STO 00
09 ADV
10 "DERBY"
11 PRA
12 ,02
13♦LBL 10
14 CF IND X
15 ISG X
16 GTO 10
17 1,008
18 STO 09
19♦LBL 09
20 16
21 XEQ 50
22 INT
23  E
24 X<>Y
25 X<=Y?
26 2
27 FS? IND X
28 GTO 09
29 SF IND X
30 STO IND 09
31 ISG 09
32 GTO 09
33 2,008
34 STO 09
35♦LBL 12
36 RCL 09
37 STO 10
38  E
39 -
40 RCL IND X
41♦LBL 13
42 RCL IND 10
43 X<Y?
44 X<>Y
45 STO IND 10
46 RDN
47 ISG 10
48 GTO 13
49 STO IND Y
50 ISG 09
51 GTO 12
52 1,008
53 STO 09
54 CF 12
55 CF 13
56♦LBL 16
57 RCL 09
58 ACX
59 XEQ IND 09
60 " "
61 ARCL IND 09
62 "├:1"
63 ACA
64 PRBUF
65 ISG 09
66 GTO 16
67 ADV
68 "POSITION: "
69 "├1,2 ODER 3"
70 PRA
71 "EINSATZ: "
72 "├BIS 800$"
73 PRA
74 20,029
75 STO 09
76♦LBL 18
77 ,
78 "PFERD NR. ?"
79 PROMPT
80 X=0?
81 GTO 17
82 STO IND 09
83 "EINSATZ ?"
84 PROMPT
85 RCL 09
86 19
87 -
88 "WETTE "
89 ARCL X
90 "├:"
91 PRA
92 RDN
93 CLA
94 ARCL X
95 "├$ AUF"
96 ACA
97 RCL IND 09
98 XEQ IND X
99 X<>Y
100  E3
101 /
102 FRC
103 ST+ IND 09
104 48
105 ACCHR
106 "POSITION ?"
107 PROMPT
108 3
109 X<Y?
110 3
111 CLA
112 ARCL Y
113 ACA
114 X<>Y
115  E4
116 /
117 ST+ IND 09
118 41
119 ACCHR
120 PRBUF
121 ISG 09
122 GTO 18
123♦LBL 17
124 ADV
125 SF 00
126 "START IN 1 "
127 "├MINUTE"
128 PRA
129 1,008
130 STO 09
131♦LBL 00
132  E1
133 RCL 09
134 +
135 LASTX
136 INT
137  E6
138 /
139 STO IND Y
140 ISG 09
141 GTO 00
142 21,025
143 STO 09
144♦LBL 27
145 11,018
146 STO 10
147 FIX 1
148♦LBL 19
149 4
150 XEQ 50
151 4
152 XEQ 50
153 +
154 2
155 /
156 RCL IND 10
157  E1
158 *
159 FRC
160  E5
161 *
162 X<>Y
163 RCL IND Y
164 LOG
165 +
166 RND
167 ST+ IND 10
168 ISG 10
169 GTO 19
170 12,018
171 STO 19
172♦LBL 26
173 RCL 19
174 STO 10
175  E
176 -
177 RCL IND X
178♦LBL 14
179 RCL IND 10
180 X<Y?
181 X<>Y
182 STO IND 10
183 RDN
184 ISG 10
185 GTO 14
186 STO IND Y
187 ISG 19
188 GTO 26
189 "DAS RENNEN IST"
190 "├ GESTARTET"
191 FS?C 00
192 PRA
193 11,018
194 STO 10
195 RCL 11
196 RND
197♦LBL 11
198 ST- IND 10
199 ISG 10
200 GTO 11
201 ADV
202 XEQ IND 09
203 "POS. PFERD  "
204 "├ DAHINTER"
205 PRA
206 1,008
207 STO 10
208 FIX 0
209♦LBL 15
210 RCL 10
211 ACX
212  E1
213 +
214 RCL IND X
215 ENTER↑
216 ENTER↑
217  E1
218 *
219 FRC
220  E5
221 *
222 XEQ IND X
223 FIX 1
224 RDN
```

```
225 RND
226 "  "
227 X≠0?
228 ARCL X
229 ACA
230 PRBUF
231 FIX 0
232 ISG 10
233 GTO 15
234 ISG 09
235 GTO 27
236 ADV
237 11,018
238 STO 09
239*LBL 29
240 RCL IND 09
241  E1
242 *
243 FRC
244  E5
245 *
246 RCL 09
247  E1
248 -
249 INT
250  E1
251 /
252 ST+ IND Y
253 ISG 09
254 GTO 29
255 ,
256 STO 12
257 20,029
258 STO 09
259*LBL 28
260 FIX 0
261 RCL 09
262 19
263 -
264 "WETTE "
265 ARCL X
266 "├:"
267 RCL IND 09
268 X=0?
269 GTO 32
270 PRA
271 INT
272 STO 10
273 RCL IND 09
274 FRC
275  E3
276 *
277 INT
278 STO 19
279 ST- 12
280 RCL IND 09
281  E3
282 *
283 FRC
284  E1
285 *
286 INT
287 STO 11
288 RCL IND 10
289 FRC
290  E1
291 *
292 X<=Y?
293 GTO 31
294 "ZERREISSE DEN "
295 "├WETTSCHEIN"
296 PRA
297*LBL 32
298 ISG 09
299 GTO 28
300 FIX 0
301 ADV
302 "DU "
303 RCL 12
304 X>0?
305 "├GEWANNST"
306 X<=0?
307 "├VERLORST"
308 "├ INSG. "
309 ABS
310 ARCL X
311 "├$"
312 PRA
313 STOP
314*LBL 31
315 RCL IND 09
316 RCL 19
317 RCL IND Y
318 INT
319 *
320 RCL 11
321 /
322 ST+ 12
323 "DU KASSIERST "
324 ARCL X
325 "├$"
326 PRA
327 GTO 32
328*LBL 50
329 RCL 00
330 9821
331 *
332 ,211327
333 +
334 FRC
335 STO 00
336 *
337 RTN
338*LBL 02
339 "  MAN O WAR   "
340 ACA
341 RTN
342*LBL 01
343 "  OROFINO     "
344 ACA
345 RTN
346*LBL 05
347 "  APRIL RUN   "
348 ACA
349 RTN
350*LBL 00
351 "  DERBYSTAR   "
352 ACA
353 RTN
354*LBL 07
355 "  PICADILLY   "
356 ACA
357 RTN
358*LBL 03
359 "  GALLANT FOX "
360 ACA
361 RTN
362*LBL 06
363 "  LANDSGIRL   "
364 ACA
365 RTN
366*LBL 04
367 "  GOLD RIVER  "
368 ACA
369 RTN
370*LBL 21
371 "KURZ NACH "
372 "├DEM START"
373 PRA
374 RTN
375*LBL 22
376 "NACH EINER "
377 "├1/4 MEILE"
378 PRA
379 RTN
380*LBL 23
381 "NACH DER "
382 "├HALBEN DISTANZ"
383 PRA
384 RTN
385*LBL 24
386 "VOR DER "
387 "├ZIELGERADE"
388 PRA
389 RTN
390*LBL 25
391 "ZIELEINLAUF"
392 PRA
393 END
```

– 7. –
Elektrotechnik

ELEKTROTECHNIK

Aufgenommene 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
Widerstandsthermometer	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

Rauschoptimierung

RF SIZE 023 USER
445 Bytes

00 = I_C = Kollektorstrom in mA
01 = B = Wechselspannungsverstärkungsfaktor ß
02 = R_{bb} = Basisbahnwiderstand in Ω
03 = R_e = äußere Emitterimpedanz in Ω
04 = R_G = Generatorwiderstand in Ω

Taste A optimiert auf minimales Rauschen, Taste B errechnet Rauschfaktor bei vorgegenem Kollektorstrom

$$Ø6 = 25{,}9 \frac{Ø2}{ØØ}$$

$$Ø7 = \frac{25{,}9}{ØØ} + (Ø3)$$

$$Ø8 = Ø2 + Ø4$$

$$Ø9 = \frac{Ø8 + Ø6}{Ø6}$$

$$U_R = \frac{Ø9^2}{Ø4} (Ø7 + \frac{Ø8}{Ø9})$$

$$RF = 20 \times \lg (U_R)$$

```
01*LBL "RF"
02 ADV
03 XEQ 11
04 SF 12
05 "RAUSCHFAKTOR"
06 AVIEW
07 CF 12
08 XEQ 11
09*LBL B
10 CF 00
11 "Ic ?"
12 PROMPT
13 STO 00
14 SF 00
15*LBL A
16 FIX 0
17 " ?"
18 ASTO 13
19 RCL 01
20 "B: "
21 ASTO 17
22 ARCL X
23 ARCL 13
24 PROMPT
25 STO 01
26 RCL 02
27 "Rbb: "
28 ASTO 18
29 ARCL X
30 ARCL 13
31 PROMPT
32 STO 02
33 RCL 03
34 "Re: "
35 ASTO 19
36 ARCL X
37 ARCL 13
38 PROMPT
39 STO 03
40 RCL 04
41 "RG: "
42 ASTO 20
43 ARCL X
44 ARCL 13
45 PROMPT
46 STO 04
47 RCL 10
48 "b: "
49 ASTO 21
50 ARCL X
51 "⊢ KHZ?"
52 PROMPT
53 STO 10
54 RCL 11
55 FIX 1
56 "Ue: "
57 ASTO 22
58 ARCL X
59 "⊢ MV ?"
60 PROMPT
61 STO 11
62 25,9
63 RCL 01
64 *
65 STO 05
66 RCL 02
67 RCL 04
68 +
69 STO 08
70 CLX
71 FC? 00
72 STO 00
73 STO 16
74  E3
75 STO 15
76 ,1
77 STO 14
78 FIX 2
79 FC? 00
80 GTO 01
81*LBL 00
82 RCL 05
83 RCL 00
84 CF 21
85 VIEW X
86 /
87 STO 06
88 RCL 01
89 /
90 RCL 03
91 +
92 STO 07
93 RCL 08
94 ENTER↑
95 ENTER↑
96 RCL 06
97 +
98 LASTX
99 /
100 STO 09
101 /
102 RCL 07
103 +
104 RCL 09
105 X↑2
106 RCL 04
107 /
108 *
109 FS? 00
110 X<>Y
111 FC?C 00
112 RTN
113 GTO 04
114*LBL 03
115 STO 16
116 ST+ 00
117 XEQ 00
118 ENTER↑
119 X<> 15
120 X<>Y
121 RTN
122*LBL 01
123 RCL 14
124 XEQ 03
125 X<Y?
126 GTO 01
127  E1
128 CHS
129 ST/ 14
130*LBL 02
131 RCL 14
132 XEQ 03
133 X<Y?
134 GTO 02
135*LBL 04
136 X<>Y
137 LOG
138  E1
139 *
140 STO 12
141 CF 00
142 SF 21
143 TONE 9
144 CLA
145 FC? 55
146 GTO 05
147 FIX 0
148 ADV
149 ARCL 17
150 ARCL 01
151 PRA
152 CLA
153 ARCL 18
154 ARCL 02
155 PRA
156 CLA
157 ARCL 19
158 ARCL 03
159 PRA
160 CLA
161 ARCL 20
162 ARCL 04
163 PRA
164 FIX 2
165 CLA
166 ARCL 21
167 ARCL 10
168 "⊢ KHZ"
169 PRA
170 CLA
171 ARCL 22
172 ARCL 11
173 "⊢ MV"
174 PRA
175*LBL 05
176 RCL 12
177 "F: "
178 ARCL X
179 "⊢ db"
180 AVIEW
181 RCL 00
182 RCL 16
183 -
184 "Ic: "
185 ARCL X
186 "⊢ MA"
187 AVIEW
188 1,606 E-11
189 RCL 04
190 *
191 RCL 10
192 *
193 SQRT
194 XEQ 06
195 RCL 12
196 ÷
197 "RP: "
198 ARCL X
199 "⊢ db"
200 AVIEW
201 CHS
202 RCL 11
203 XEQ 06
204 +
205 "RA: "
206 ARCL X
207 "⊢ db"
208 AVIEW
209 XEQ 11
210 GTO A
211*LBL 06
212 775
213 /
214 LOG
215 20
216 *
217 RTN
218*LBL 11
219 FC? 55
220 RTN
221 "------"
222 ACA
223 ACA
224 ACA
225 ACA
226 ADV
227 END
```

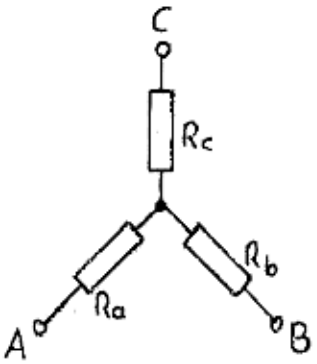
Stern-Dreieck-Umwandlung

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"



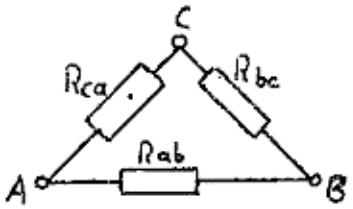
XEQ DR

Ra=100.00E0
Rb=200.00E0
Rc=300.00E0

Rbc=1.1000E3
Rca=550.00E0
Rab=366.67E0

Dreieckschaltung

Rbc
ENTER
Rca
ENTER
Rab
XEQ"ST"



XEQ ST

Rbc=1.1000E3
Rca=550.00E0
Rab=366.67E0

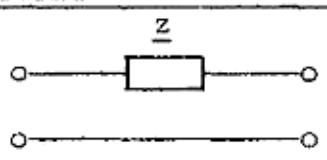
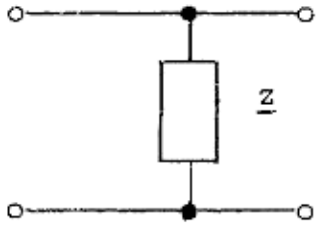
Ra=100.00E0
Rb=200.00E0
Rc=300.00E0

```
01*LBL "DR"
02 CF 01
03 FS? 00
04 GTO 03
05 XEQ 09
06 FC? 55
07 GTO 03
08*LBL 01
09 ADV
10 "Ra="
11 ARCL Z
12 AVIEW
13 "Rb="
14 ARCL Y
15 AVIEW
16 "Rc="
17 ARCL X
18 AVIEW
19 FS? 01
20 RTN
21*LBL 03
22 STO L
23 RDN
24 ST+ L
25 RDN
26 ST* L
27 RDN
28 X<> L
29 RDN
30 STO L
31 RDN
32 ST* L
33 RDN
34 RDN
35 ST+ L
36 X<> L
37 RDN
38 1/X
39 RDN
40 1/X
41 RDN
42 1/X
43 X<>Y
44 *
45 Rt
46 LASTX
47 *
48 Rt
49 LASTX
50 *
51 FS? 00
52 RTN
53*LBL 02
54 ADV
55 "Rbc="
56 ARCL Z
57 AVIEW
58 "Rca="
59 ARCL Y
60 AVIEW
61 "Rab="
62 ARCL X
63 AVIEW
64 RTN
65*LBL "ST"
66 SF 01
67 FS? 00
68 GTO 04
69 XEQ 00
70 FS? 55
71 XEQ 02
72*LBL 04
73 STO L
74 RDN
75 ST+ L
76 RDN
77 ST+ L
78 RDN
79 X<> L
80 RDN
81 STO L
82 RDN
83 ST* L
84 X<> L
85 RDN
86 ST* L
87 X<> L
88 RDN
89 RDN
90 ST* L
91 X<> L
92 Rt
93 X<>Y
94 RDN
95 /
96 X<> Z
97 LASTX
98 /
99 X<>Y
100 LASTX
101 /
102 FS? 00
103 RTN
104 GTO 01
105*LBL 00
106 CF 12
107 CF 13
108 ENG 4
109 END
```

Betriebsdämpfungsmaß und Betriebsdämpfungswinkel von Vierpolen

BB	SIZE 034	Drucker MAN	Synthetik
ST	1002 Bytes	zwei Memorys	User
KO			

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.

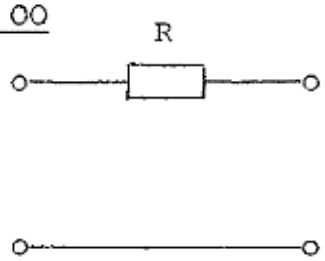
SCHALTUNG	KETTENMATRIX
a) 	$A = \begin{pmatrix} \underline{A}_{11} = 1 & \underline{A}_{12} = \underline{z} \\ \underline{A}_{21} = 0 & \underline{A}_{22} = 1 \end{pmatrix}$
b) 	$A = \begin{pmatrix} \underline{A}_{11} = 1 & \underline{A}_{12} = 0 \\ \underline{A}_{21} = 1/\underline{z} & \underline{A}_{22} = 1 \end{pmatrix}$

Werden mehrere Vierpole in Reihe geschaltet, so müssen die einzelnen Kettenmatrizen multipliziert werden, was nach den Regeln der Matrizenrechnung 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.

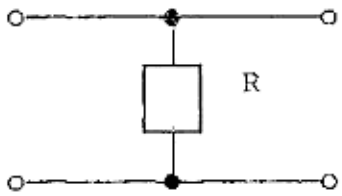
Mit dem hier vorgestellten Programm lassen sich folgende unvollkommene, aber natürlich auch vollkommene, Vierpole verarbeiten. Es können aber auch einige hinzugefügt oder ersetzt werden, dazu dienen Label 23, 24, ...

Die Zahlenangaben der folgenden Vierpole geben die Nummern des Labels an, unter denen der entsprechende Vierpol gefunden wird, um dessen Kettenmatrix aufzustellen.

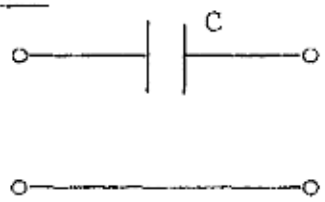
LBL 00



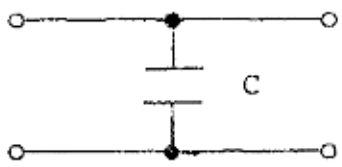
LBL 01



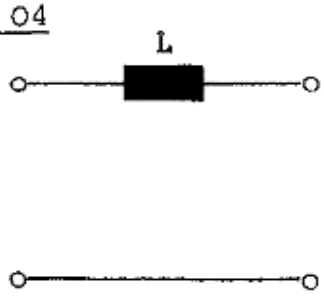
LBL 02



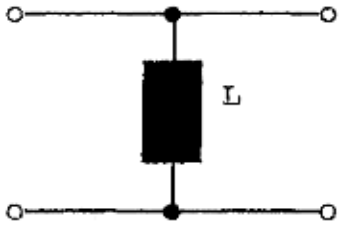
LBL 03



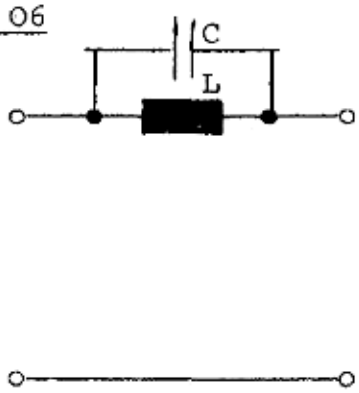
LBL 04



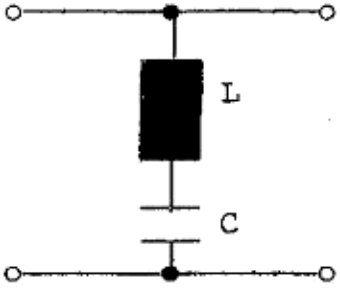
LBL 05



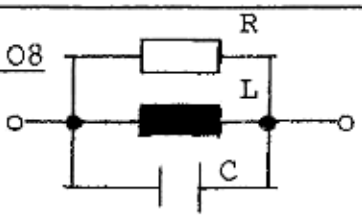
LBL 06



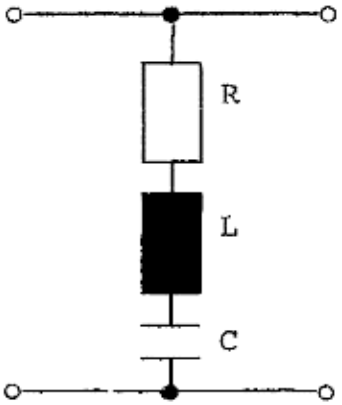
LBL 07



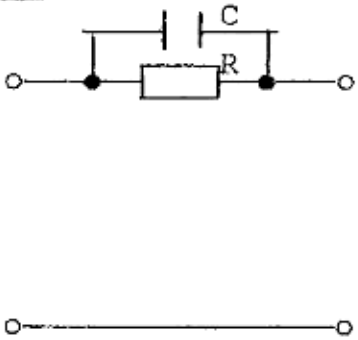
LBL 08



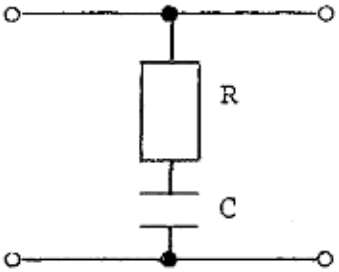
LBL 09



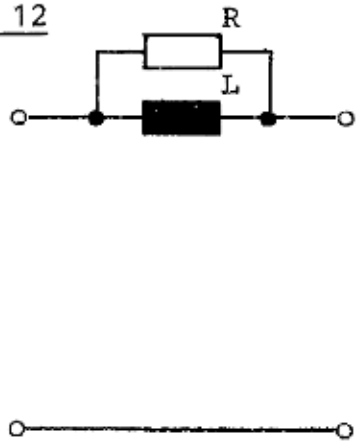
LBL 10



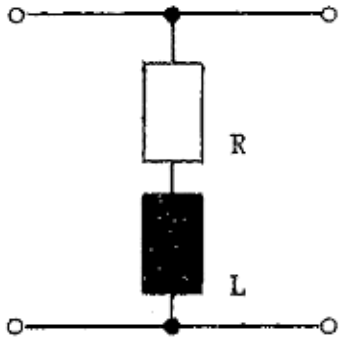
LBL 11



LBL 12



LBL 13



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:	2xPIxf = ω
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 b_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_i, 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

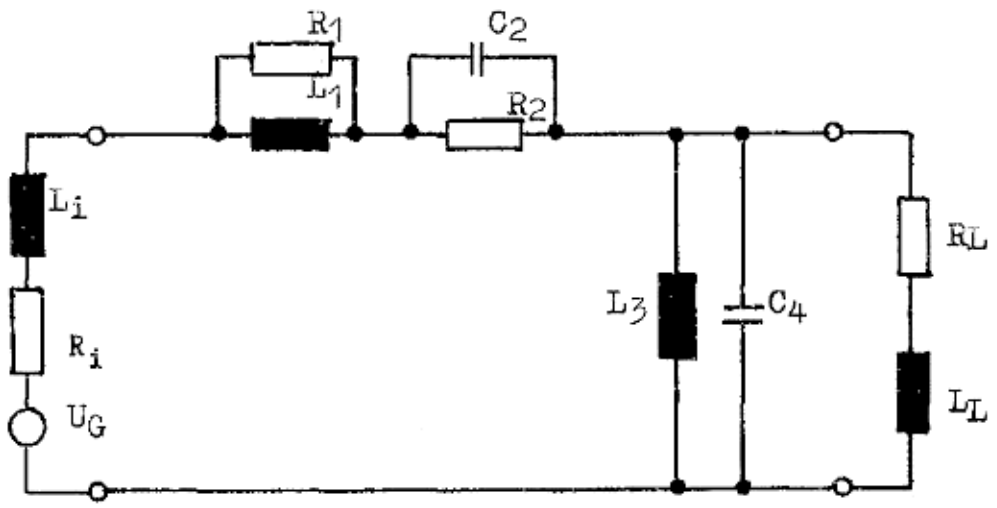
Soll das Programm in dieser Form verwendet werden, ist ein Drucker notwendig. Steht dieser aber nicht zur Verfügung, müssen die Druckbefehle gelöscht werden (gemeint ist: PRA). Programmzeile 102 kann durch einen AVIEW-Befehl, und Zeilen 193 und 197 durch je einen PROMPT-Befehl ersetzt werden. Sind diese 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 A
2	Eingabe des Generator-innenwiderstandes, des Lastwiderstandes und die Werte der Bauelemente Vorbereitung Achtung Falls Innenkapazität oder Lastkapazität vorhanden, müssen diese mit neg. Vorzeichen eingegeben werden. Sind alle Schaltelemente eingegeben, weiter mit 4	 R_i L_i bzw.-C_i R_L L_L bzw.-C_L r,l oder C_1 usw.	XEQ"BB" R/S R/S R/S R/S R/S usw.	RI? LI,-CI? RL? LL,-CL? R,L,C?1 usw.
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	Frequenzeingaben Startfrequenz Frequenzschrittweite Endfrequenz Berechnungsstart Rechnung endet mit	 f f f_{max}	XEQ"ST" R/S R/S R/S	F? dF? Fmax? f_{max} + f




```
RI= 600 OHM
LI= 1 E-6 H
RL= 600 OHM
LL= 0,5 E-3 H

R1= 10000 OHM
L1= 2,387 H
R2= 10000 OHM
C2= 15,79 E-9 F
L3= 5,68 E-3 H
C4= 6,63 E-6 F
```

BANDPASS

```
123*LBL A
124 XEQ 12
125 XEQ 10
126 XEQ 05
127 XEQ 03
128 RCL 30
```

```
R1?=600.0000000E0
LI,-CI?=1.000000000E-6
RL?=600.0000000E0
LL,-CL?=500.0000000E-6


R,L,C?1=10.00000000E3
R,L,C?2=2.387000000E0
R,L,C?3=10.00000000E3
R,L,C?4=15.97000000E-9
R,L,C?5=5.680000000E-3
R,L,C?6=6.630000000E-6

F?=760.0000000E0 HZ
dF?=20.00000000E0 HZ
F MAX?=900.0000000E0 HZ
```

```
F=760.0000000E0 HZ
aB=30.77797780E0 dB
bB=-70.76079148E0 GRAD

F=780.0000000E0 HZ
aB=27.69521014E0 dB
bB=-62.56698329E0 GRAD

F=800.0000000E0 HZ
aB=23.72786874E0 dB
bB=-44.05666164E0 GRAD

F=820.0000000E0 HZ
aB=20.81697244E0 dB
bB=-551.0328894E-3 GRAD

F=840.0000000E0 HZ
aB=23.53203216E0 dB
bB=42.85409582E0 GRAD
```

```
F=860.0000000E0 HZ
aB=27.28356704E0 dB
bB=61.31158556E0 GRAD

F=880.0000000E0 HZ
aB=30.14967348E0 dB
bB=69.50731768E0 GRAD

F=900.0000000E0 HZ
aB=32.33727274E0 dB
bB=73.89173730E0 GRAD
```

```
                PRP "BB"


 01♦LBL "BB"
ENG 9  DEG  "R1?"
PROMPT  "⊢="  ARCL X
PRA  STO 27  "LI,-CI?"
PROMPT  "⊢="  ARCL X
PRA  STO 31  "RL?"
PROMPT  "⊢="  ARCL X
PRA  STO 29  "LL,-CL?"
PROMPT  "⊢="  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  "⊢="
ARCL X  PRA  STO IND 26
 E  ST+ 26  ST+ 25
GTO 17

 49♦LBL "ST"
SF 29  ADV  "F?"  PROMPT
"⊢="  ARCL X  "⊢ HZ"
PRA  "dF?"  PROMPT  "⊢="
ARCL X  "⊢ HZ"  PRA
STO 21  -  STO 20
"F MAX?"  PROMPT  "⊢="
ARCL X  "⊢ 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  "⊢ 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
"⊢ dB"  PRA  "bB="
ARCL Y  "⊢ 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

398♦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

538*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 "KO"
FIX 0  "KOR.:"  ARCL X
PRA  "R/L/C?"  ARCL X
32  +  ENG 9  PROMPT
"⊢="  ARCL X  PRA
STO IND Y  ADV  GTO 17
END
```

Impedanz - Anpassung

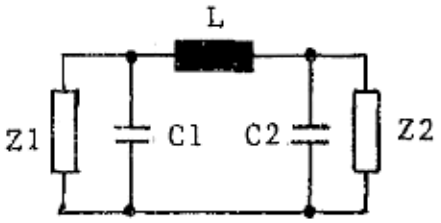
IM SIZE 006
105 Bytes

Transformierende Tiefpässe werden zur Anpassung von verschiedenen 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"
02 ENG 3
03 "Z1?"
04 PROMPT
05 XEQ 01
06 STO 00
07 "Z2?"
08 PROMPT
09 XEQ 01
10 STO 01
11 "F?"
12 PROMPT
13 XEQ 01
14 STO 02
15 "<>F?"
16 PROMPT
17 XEQ 01
18 STO 03
19 PI
20 *
21 STO 04
22 RCL 01
23 RCL 00
24 *
25 SQRT
26 STO 05
27 RCL 01
28 +
29 *
30 1/X
31 "C2="
32 XEQ 01
33 RCL 04
34 RCL 05
35 RCL 00
36 +
37 *
38 1/X
39 "C1="
40 XEQ 01
41 RCL 03
42 4
43 PI
44 *
45 RCL 02
46 X↑2
47 *
48 /
49 RCL 05
50 ST+ X
51 RCL 00
52 +
53 RCL 01
54 +
55 *
56 "L="
57♦LBL 01
58 ARCL X
59 AVIEW
60 FC? 55
61 STOP
62 END
```

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"

```
F?1.000E3
A1?1.362E0
B1?618.0E-3

C1?10.00E-9
C2?10.00E-9

R1=23.38E3
R2=17.53E3

C1?5.700E-9
C2?10.00E-9

R1=32.19E3
R2=22.34E3
```

XEQ"TP"

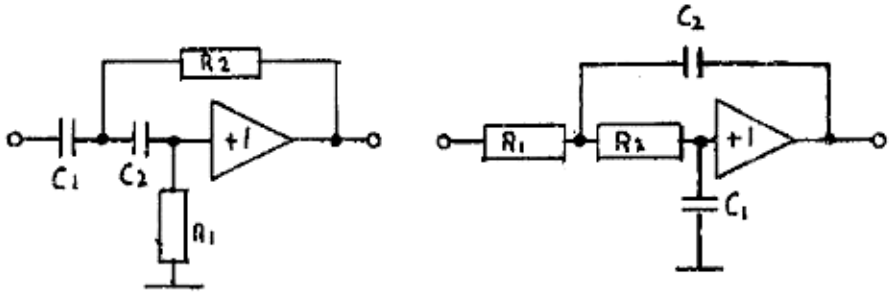
```
F?1.000E3
A1?1.362E0
B1?618.0E-3

C1?10.00E-9
C2?10.00E-9

R1=7.885E3
R2=13.79E3
C2=14.48E-9

C1?1.000E-9
C2?1.500E-9

R1=72.22E3
R2=144.5E3
C2=1.500E-9
```



```
01♦LBL "HP"
02 SF 01
03 GTO 00
04♦LBL "TP"
05 CF 01
06♦LBL 00
07 ADV
08 ENG 3
09 "F?"
10 PROMPT
11 XEQ 01
12 STO 01
13 "A1?"
14 PROMPT
15 XEQ 01
16 STO 04
17 "B1?"
18 PROMPT
19 XEQ 01
20 STO 05
21♦LBL A
22 ADV
23 "C1?"
24 PROMPT
25 XEQ 01
26 STO 02
27 "C2?"
28 PROMPT
29 XEQ 01
30 STO 03
31 ADV
32♦LBL B
33 FS? 01
34 GTO 03
35 RCL 04
36 X↑2
37 RCL 03
38 X↑2
39 *
40 4
41 RCL 05
42 *
43 RCL 02
44 *
45 RCL 03
46 *
47 -
48 X<0?
49 GTO 02
50 SQRT
51 STO 06
52 RCL 04
53 RCL 03
54 *
55 STO 00
56 X<>Y
57 -
58 4
59 PI
60 *
61 RCL 01
62 *
63 RCL 02
64 *
65 RCL 03
66 *
67 STO Z
68 /
69 "R1="
70 XEQ 01
71 RCL 06
72 RCL 00
73 +
74 RCL Z
75 /
76 "R2="
77 XEQ 01
78 RCL 03
79 "C2="
80 GTO 01
81♦LBL 02
82 1/2
83 ST* 03
84 GTO B
85♦LBL 03
86 2
87 PI
88 *
89 RCL 01
90 *
91 STO 06
92 RCL 04
93 *
94 1/X
95 RCL 02
96 1/X
97 RCL 03
98 1/X
99 +
100 *
101 "R1="
102 XEQ 01
103 RCL 04
104 RCL 05
105 RCL 06
106 *
107 /
108 RCL 02
109 RCL 03
110 +
111 1/X
112 *
113 "R2="
114♦LBL 01
115 ARCL X
116 AVIEW
117 FC? 55
118 STOP
119 END
```

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 / 10.80	PT 100 DIN 43 760 / 10.80
T=130.00 C R=183.345 OHM	T=150.00 C R=157.315 OHM
T=-60.00 C R=69.520 OHM	T=850.00 C R=390.263 OHM
T=50.26 C R=129.265 OHM	T=-200.00 C R=18.493 OHM
R=70.258 OHM T=-58.42 C	R=18.500 OHM T=-199.98 C

```
01♦LBL "PT"
02 XEQ 16
03 ,390802
04 STO 01
05 580195 E-10
06 CHS
07 STO 02
08 42735 E-12
09 STO 03
10  E2
11 CHS
12 /
13 STO 04
14 GTO 15
15♦LBL "NI"
16 XEQ 16
17 ,5485
18 STO 01
19 665 E-6
20 STO 02
21 2885 E-12
22 STO 03
23 SF 03
24♦LBL 15
25 FC? 55
26 GTO 01
27 ADV
28 SF 12
29 FC? 03
30 "   PT"
31 FS? 03
32 "   NI"
33 "├ 100"
34 PRA
35 CF 12
36 "   DIN 43 760 /"
37 "├ 10.00"
38 PRA
39♦LBL 01
40 "T?"
41  E
42 PROMPT
43 FC?C 22
44 GTO 00
45 FC? 03
46 CF 00
47 FC? 03
48 850,01
49 FS? 03
50 180,01
51 X<=Y?
52 GTO 01
53 RDN
54 FC? 03
55 -200,01
56 FS? 03
57 -60,01
58 X>Y?
59 GTO 01
60 RDN
61 ADV
62 XEQ 06
63♦LBL 03
64 FS? 03
65 GTO 10
66 X<0?
67 SF 00
68♦LBL 10
69 ENTER↑
70 ENTER↑
71 FS? 03
72 GTO 11
73 RCL 04
74 *
75 RCL 03
76 +
77 *
78 RCL 02
79 +
80 FC?C 00
81 X<> L
82 GTO 12
83♦LBL 16
84 CF 01
85 SF 02
86 CF 22
87 CF 03
88 RTN
89♦LBL 11
90 RCL 03
91 *
92 *
93 RCL 02
94 +
95♦LBL 12
96 *
97 RCL 01
98 +
99 *
100  E2
101 +
102 FS? 01
103 GTO 04
104 GTO 07
105♦LBL 00
106 "R?"
107 ,
108 FS? 03
109 STO 04
110 FC? 03
111 STO 05
112 PROMPT
113 FC?C 22
114 GTO 01
115 FC? 03
116 390,266
117 FS? 03
118 223,23
119 X<=Y?
120 GTO 00
121 RDN
122 FC? 03
123 18,491
124 FS? 03
125 69,45
126 X>Y?
127 GTO 08
128 RDN
129 ADV
130 XEQ 07
131 STO 00
132♦LBL 05
133 FS? 03
134 SF 01
135  E2
136 FS? 03
137 GTO 13
138 X>Y?
139 SF 01
140♦LBL 13
141 -
142 RCL 02
143 /
144 RCL 01
145 LASTX
146 /
147 2
148 /
149 X↑2
150 LASTX
151 RDN
152 +
153 SQRT
154 R↑
155 FS? 03
156 -
157 FS? 03
158 GTO 14
159 +
160 CHS
161 FC? 01
162 GTO 06
163♦LBL 14
164 GTO 03
165♦LBL 04
166 RCL 00
167 -
168 ABS
169  E-4
170 X>Y?
171 GTO 06
172 LASTX
173 FC? 03
174 ST+ 05
175 FS? 03
176 ST+ 04
177 RCL 00
178 FC? 03
179 RCL 05
180 FS? 03
181 RCL 04
182 -
183 GTO 05
184♦LBL 06
185 "T="
186 FS?C 01
187 X<> T
188 FIX 2
189 RND
190 ARCL X
191 "├ C"
192 ,
193 GTO 08
194♦LBL 07
195 "R="
196 FIX 3
197 RND
198 ARCL X
199 "├ OHM"
200  E
201♦LBL 08
202 X<>Y
203 AVIEW
204 FS?C 02
205 RTN
206 SF 02
207 GTO IND Y
208 END
```

Addition von zwei Sinusschwingungen gleicher Frequenz

AD SIZE 004 User
086 Bytes

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
"∠1?" PROMPT STO 02
"∠2?" 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="
ARCL X PROMPT

 34*LBL C
RCL 02 SIN RCL 00 *
RCL 03 SIN RCL 01 *
+ RCL 02 COS RCL 00
* RCL 03 COS RCL 01
* + / ATAN "∠="
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

ICH ZIEHE
VON 42 NACH 53

Abb. 1

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.

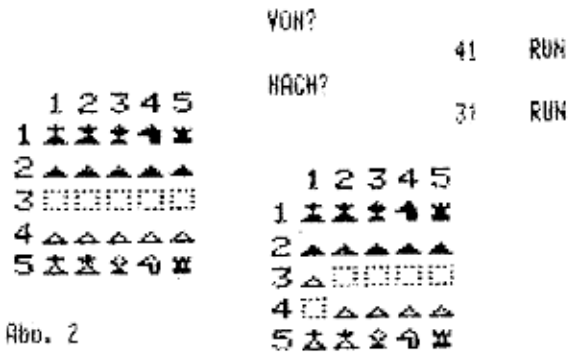
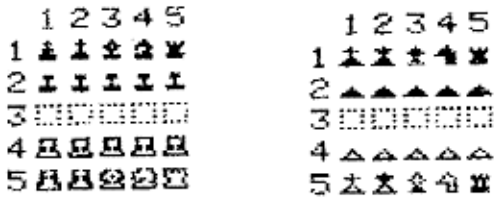


Abb. 2

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=  16,031
R27=  -1,000
```



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

Bedienung des Programms:

- SIZE 083 einstellen
- wenn Sie das Brett ausdrucken wollen, laden Sie bitte die Programme P und 5 und führen GTO.. aus
- laden Sie die Datenkarte (2 Seiten)
  Wenn Sie jetzt den Kartenleser abstecken und dafür den Drucker einstecken wollen, empfehle ich Ihnen, hinter RDTAX die Befehle SF 11 und OFF einzufügen
- Sie spielen immer die weißen Figuren, während der HP 41 C(V) die schwarzen spielt.
- HP ZUERST?, wenn Sie wollen, daß Ihr Rechner den ersten Zug macht, drücken Sie R/S.
  Wenn Sie den ersten Zug machen wollen, drücken Sie N; R/S
- jetzt ziehen Sie und der Rechner abwechselnd.

```
 01*LBL "NS"
CLRG  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  CLΣ
ASTO 69  FS? 55  XEQ "P"
CF 23  AON  "HP ZUERST?"
PROMPT  AOFF  FC?C 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 06
"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
"⊢ NACH "  RCL 13
XEQ 09  RCL 00  FRC  10
*  FRC  X≠0?
"⊢, 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
"⊢ HABE"  FC? 04  "⊢N"
"⊢ GEWONNEN"  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? 10
GTO 00  X<6?  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? 10  1  X=0?  XEQ 12
RCL 02  8  XEQ 09
FS? 10  CLX  X>0?
XEQ 12  RCL 02  6
XEQ 09  FS? 13  RTN
X<=0?  RTN

295*LBL 12
CF 00  STO 07  FRC  1 E2
*  STO 06  RCL Z  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 00  FS? 10
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 00  8  XEQ 00
RCL 00  6

402*LBL 00
XEQ 09  FS? 10  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 00  FS? 10
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? 10  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? 10  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 10  LASTX  .END.
```

Beispiel:

```
 01♦LBL "P"
ADV  SF 12  9  SKPCOL
49,053  STO 13

 08♦LBL 00
ACCHR  2  SKPCOL  X<>Y
ISG X  GTO 00  PRBUF
50,054  STO 15

 18♦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 [  "┣12"  RCL \
STO d  END
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ ♘ ♖
2 ▲ ▲ ▲ ▲ ▲
3 □ □ □ □ □
4 △ △ △ △ △
5 ♙ ♙ ♔ ♘ ♖
```

```
HP ZUERST?
H                    RUN
VON?
                41   RUN
NACH?
                31   RUN
ICH ZIEHE
VON 22 NACH 31
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ ♘ ♖
2 ▲ □ ▲ ▲ ▲
3 ▲ □ □ □ □
4 □ △ △ △ △
5 ♙ ♙ ♔ ♘ ♖
```

```
VON?
                42   RUN
NACH?
                32   RUN
ICH ZIEHE
VON 23 NACH 32
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ ♘ ♖
2 ▲ □ □ ▲ ▲
3 ▲ ▲ □ □ □
4 □ □ △ △ △
5 ♙ ♙ ♔ ♘ ♖
```

```
VON?
                43   RUN
NACH?
                32   RUN
ICH ZIEHE
VON 21 NACH 32
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ ♘ ♖
2 □ □ □ ▲ ▲
3 ▲ ▲ □ □ □
4 □ □ □ △ △
5 ♙ ♙ ♔ ♘ ♖
```

```
VON?
                54   RUN
NACH?
                33   RUN
ICH ZIEHE
VON 24 NACH 33
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ ♘ ♖
2 □ □ □ □ ▲
3 ▲ ▲ ▲ □ □
4 □ □ □ △ △
5 ♙ ♙ ♔ □ ♖
```

```
VON?
                55   RUN
NACH?
                54   RUN
ICH ZIEHE
VON 33 NACH 44
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ ♘ ♖
2 □ □ □ □ ▲
3 ▲ ▲ □ □ □
4 □ □ □ ▲ △
5 ♙ ♙ ♔ ♖ □
```

```
VON?
                54   RUN
NACH?
                44   RUN
ICH ZIEHE
VON 14 NACH 33
```

```
  1 2 3 4 5
1 ♙ ♙ ♙ □ ♖
2 □ □ □ □ ▲
3 ▲ ▲ ♘ □ □
4 □ □ □ ♖ △
5 ♙ ♙ ♔ □ □
```

```
VON?
                52   RUN
NACH?
                32   RUN
ICH ZIEHE
VON 12 NACH 32
```

```
  1 2 3 4 5
1 ♙ □ ♙ □ ♖
2 □ □ □ □ ▲
3 ▲ ♙ ♘ □ □
4 □ □ □ ♖ △
5 ♙ □ ♔ □ □
```

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 Warheitswertbelegungen 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 (O,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

$\vdash (A \wedge B \rightarrow C) \overset{?}{\leftrightarrow} (B \wedge A \rightarrow C)$ mit A=1, B=2, C=3

Eingabe des Problems als Programm:

```
LBL P
1
ENTER
2
XEQ AND
3
XEQ IF
2
ENTER
1
XEQ AND
3
XEQ IF
XEQ EQUI
```

Die Formel enthält 3 Elemente (A, B, C). Also nach XEQ LOGIC wird die Frage "ELEMENTE ?" mit 3 beantwortet. Der Rechner permutiert nun sämtliche Wahrheitswertbelegungen durch. Findet er keine Belegung, bei der das zu prüfende Gesetz falsch wäre, kündigt er dies durch BEEP und "WAHR" an.

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. $2 \nsim 5$
4. $(5 \wedge 3) \nsim (\neg 5 \wedge \neg 3)$
5. $4 \rightarrow (3 \wedge 1)$

Verständlicherweise müssen die fünf Aussagen noch miteinander verknüpft werden, da sie ja als Gesamtheit Gültigkeit haben. Dies geschieht mit der -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. keinen 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"
02 ΣREG 01
03 CF 04
04 CF 05
05♦LBL 15
06 FIX 0
07 CF 00
08 CF 01
09 CF 02
10 CF 03
11 CF 06
12 CF 22
13 CF 29
14 CLRG
15 12
16 "ELEMENTE?"
17 TONE 9
18♦LBL 01
19 AVIEW
20 PSE
21 FC? 22
22 GTO 01
23 X≠0?
24 X>Y?
25 GTO 15
26 STO 00
27 STO 18
28  E3
29 /
30  E
31 +
32 STO 16
33 FS? 05
34 RTN
35 GTO 17
36♦LBL 16
37 19,9
38 STO 14
39 RCL 16
40 STO 15
41 CLA
42♦LBL 02
43 ARCL IND 15
44 RCL IND 15
45 X≠0?
46 SF 03
47 X≠0?
48 GTO 00
49 FS? 03
50 SF 06
51♦LBL 00
52 ISG 15
53 GTO 02
54 FC?C 03
55 SF 00
56  E
57 FC? 04
58 RCL 17
59 FC?C 06
60 STO 17
61 AVIEW
62 TONE 0
63 XEQ "P"
64 ASTO 13
65 ASHF
66 ASTO 14
67 FS?C 01
68 GTO 03
69 RCL IND X
70 X≠0?
71 GTO 17
72 SF 03
73 "F:"
74 TONE 2
75 GTO 00
76♦LBL 03
77 RCL 15
78 X=0?
79 GTO 17
80 "LOESUNG"
81 BEEP
82♦LBL 00
83 CF 05
84 AVIEW
85 PSE
86 CLA
87 FS?C 03
88 "---"
89 ARCL 13
90 ARCL 14
91 PROMPT
92 SF 02
93 FC?C 00
94 GTO 17
95♦LBL 04
96 CF 02
97 CF 04
98 "END"
99 PROMPT
100 GTO 04
101♦LBL 05
102 FS?C 01
103 GTO 06
104 "W"
105 BEEP
106 GTO 00
107♦LBL 06
108 "UNLOESBAR"
109 TONE 2
110♦LBL 00
111 PROMPT
112 FS?C 00
113 GTO 04
114 SF 02
115♦LBL 17
116 FC?C 05
117 FS? 00
118 GTO 05
119 1,9
120 STO 13
121 FS? 02
122 GTO 00
123 CLΣ
124 ΣREG 07
125 CLΣ
126 ΣREG 01
127 RCL 18
128 STO 15
129  E
130♦LBL 07
131 STO IND 15
132 DSE 15
133 GTO 07
134♦LBL 00
135 XEQ "PM"
136 SF 02
137 GTO 16
138♦LBL "PM"
139 FS? 02
140 GTO 09
141♦LBL 08
142 RCL 00
143 FACT
144 RCL 18
145 FACT
146 /
147 RCL 00
148 RCL 18
149 -
150 FACT
151 /
152 STO 17
153 RTN
154♦LBL 09
155 RCL IND 13
156 ISG 13
157 RCL IND 13
158 X<Y?
159 GTO 08
160 X≠Y?
161 SF 03
162 RCL 00
163 RCL 13
164 X<Y?
165 GTO 09
166 GTO 12
167♦LBL 08
168  E
169 STO IND 13
170 ST- 13
171 CLX
172 STO IND 13
173 STO 15
174 FC?C 03
175 GTO 12
176 RCL 13
177 INT
178  E
179 -
180  E3
181 /
182 STO 14
183 ISG 14
184♦LBL 10
185 RCL IND 14
186 X≠0?
187 GTO 11
188  E
189 STO IND 14
190 ST+ 15
191 ISG 14
192 GTO 10
193♦LBL 11
194  E
195 ST- 13
196 CLX
197 STO IND 13
198 DSE 15
199 GTO 11
200♦LBL 12
201 CF 03
202 DSE 17
203 RTN
204 CF 02
205 DSE 18
206 GTO 17
207 CLX
208 STO IND 00
209 RTN
210♦LBL "PS"
211 SF 04
212 SF 05
213 XEQ 15
214  E
215 RCL 01
216 STO 18
217♦LBL 13
218 RDN
219 CLA
220 ARCL Y
221 "ł.WERT"
222 CF 22
223 TONE 9
224♦LBL 14
225 AVIEW
226 PSE
227 FC? 22
228 GTO 14
229 X>Y?
230 GTO 13
231 STO IND Z
232 ST+ 18
233 ISG Z
234 GTO 13
235 XEQ 00
236 GTO 16
237♦LBL "XOR"
238 SF 03
239♦LBL "EQUI"
240 STO 13
241 X<> IND 13
242 STO IND 13
243 X<>Y
244 STO 13
245 X<> IND 13
```

```
246 STO IND 13
247 X>Y?
248 X<>Y
249 X≠0?
250 GTO 00
251 X<>Y
252 X=0?
253 GTO 00
254 +
255 FC?C 03
256 CLX
257 GTO 18
258*LBL 00
259 X<> 13
260 +
261 FS?C 03
262 CLX
263 GTO 18
264*LBL "IF"
265 STO 13
266 X<> IND 13
267 X≠0?
268 GTO 00
269 X<> IND 13
270 X<>Y
271 STO 13
272 X<> IND 13
273 X=0?
274 GTO 00
275 X<> IND 13
276 RDN
277 CLX
278 GTO 18
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
305 X<> IND 13
306 ST* IND 14
307 X<> IND 13
308 X<>Y
309 GTO 19
310*LBL "NAND"
311 XEQ "AND"
312*LBL "NON"
313 STO 13
314 X<> IND 13
315 X≠0?
316 GTO 00
317 X<> IND 13
318 GTO 18
319*LBL 00
320 X<> IND 13
321 CLX
322*LBL 18
323 STO IND 14
324 RDN
325*LBL 19
326 RCL 14
327 ISG 14
328 RTN
329*LBL "Σ"
330 SF 01
331 19,9
332 STO 14
333 RCL IND Y
334 ST* 15
335 X=0?
336 GTO 17
337 END
```

— 9. —
Sonstiges

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.
WOBEI AAA=ANFANGSREG.
      EEE=ENDREGISTER

DAS PROGRAMM VERWENDET
NUR STACK UND LAST X-REG
NACH DEM SORTIEREN STEHT
DIE HOECHSTE ZAHL IM
KLEINSTEN REGISTER.
BEI ALPHA-DATEN ER-
SCHEINT >ALPHA DATA<
```



```
   6152
```

Happy
Programming
Ing.(grad.) Hasso Kleep (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 R↑
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 21 0,001 - eingibt. Andreas

P.S.: Nähere Informationen zum Sortieren finden sich in CHIP 7/81

Das Programm noch einmal in groß

```
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 R↑
18 ISG X
19 GTO 01
20*LBL 04
21 RCL L
22 FS?C 00
23 GTO 01
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. -zulagen sind steuer- und sozialversicherungsfrei) Wenn ja, 'J eingeben, Rechner hält im ALPHA-Modus an.
14	R/S	SOZV.=?	Zulage sozialversicherungspflichtig? Wenn ja, 'J eingeben, Rechner hält im ALPHA-Modus an. Rechner geht nach Schritt 9.
15	R/S	VWL AN/AG?	Vermögenswirksame Leistungen: Anteil Arbeitnehmer, ENTER , Anteil 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,- Sozialversicherungsfreibetrag berechnet.
17	R/S	SONSTIGES?	Sonstige Bezüge, die zum Bruttolohn gehören, eingeben (z.B. zusätzliches Urlaubsgeld, Lohnfortzahlung usw.). Rechner fragt anschließend, ob der Betrag sozialversicherungs- bzw. steuerpflichtig ist. Ist der Betrag nur teilweise sozialversicherungs- oder steuerpflichtig, 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 sozialversicherungspflichtig? Wenn ja, 'J eingeben, Rechner hält im ALPHA-Modus an. Weiter bei Schritt 17.
20	R/S	FREIBETRAG?	Steuerfreibetrag laut Lohnsteuerkarte eingeben (nicht den Weihnachtsfreibetrag, Altersentlastungsbetrag, 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=... oder L=...	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 Bruttogehalt 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 Monatstarifs, 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 Vorsorgepauschale
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-Württemberg, Hamburg, Niedersachsen, Schleswig-Holstein DM 0,60, in Hessen DM 0,30, in den anderen Ländern kein Mindestbetrag
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, Ø 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:

F00: Angestellter
F01: Steuerklasse I
F02: Steuerklasse II
F03: Steuerklasse III
F04: Steuerklasse IV
F05: Steuerklasse V
F06: Steuerklasse VI
F07: benutzt
F08: benutzt
F15: rk. Kirchensteuerpflicht
F16: ev. Kirchensteuerpflicht
F17: rk. und ev. Kirchensteuerpflicht
F18: keine Kirchensteuerpflicht
F19: Altersentlastungsbetrag berücksichtigen
F20: keine Arbeitslosenversicherung
F22: benutzt
F27: gesetzt
F29: gesetzt

Datenspeicher:

R00: Bruttolohn
R01: Ecklohn
R02: Steuerpflichtiger Betrag
R03: Sozialversicherungspflichtiger Betrag
R04: VWL
R06: benutzt
R07: benutzt
R08: benutzt
R09: Anzahl Kinder
R10: Steuerklasse, benutzt

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
"
02 0
03 STO d
04 CLRG
05 SF 29
06 SF 27
07 FIX 2
08 "STPFL="
09 ASTO 06
10 "SOZV.*"
11 ASTO 07
12 6
13 "ST.KL?"
14 PROMPT
15 INT
16 X<=0?
17 LOG
18 X>Y?
19 ASIN
20 SF IND X
21 STO 10
22 CF 22
23 "KINDER?
"
24 PROMPT
25 FC?C 22
26 0
27 INT
28 X<0?
29 LOG
30 STO 09
31 SF 18
32 "RELIGIO
N?"
33 AON
34 PROMPT
35 AOFF
36 ASTO 00
37 "  "
38 ARCL 00
39 ASTO 00
40 ASHF
41 ASTO Z
42 " "
43 ARCL 00
44 ASHF
45 ASTO X
46 "K"
47 ASTO Y
48 X=Y?
49 SF 15
50 X=Y?
51 CF 18
52 "E"
53 ASTO Y
54 X=Y?
55 SF 16
56 X=Y?
57 CF 18
58 FS? 18
59 GTO 07
60 "K"
61 FS? 15
62 "E"
63 ASTO Y
64 RDN
65 X=Y?
66 GTO 07
67 SF 15
68 SF 16
69 SF 17
70*LBL 07
71 CLX
72 "ALTER?"
73 PROMPT
74 INT
75 63
76 X<Y?
77 SF 19
78 X<=Y?
79 SF 20
80 CF 22
81 "GEHALT?
"
82 PROMPT
83 FS? 22
84 STO 08
85 FS? 22
86 SF 00
87 FS?C 22
88 GTO 08
89 "ECKLOHN
?"
90 PROMPT
91 X<0?
92 LOG
93 STO 01
94 CF 22
95 "GSTD.*?
"
96 ASTO 08
97 PROMPT
98 FC?C 22
99 0
100 X<0?
101 LOG
102 *
103 STO 00
104 CLA
105 ARCL 08
106 ARCL X
107 PROMPT
108*LBL 09
109 "Z-STD=?
"
110 ASTO 08
111 PROMPT
112 FC?C 22
113 GTO 08
114 RCL 01
115 "% ?"
116 PROMPT
117 FC?C 22
118 GTO 08
119 %
120 RND
121 "%="
122 ARCL X
123 PROMPT
124 RCL Z
125 *
126 RND
127 CLA
128 ARCL 08
129 ARCL X
130 PROMPT
131 XEQ 13
132 GTO 09
133*LBL 08
134 "VWL ANT
AG?"
135 PROMPT
136 FC?C 22
137 GTO 10
138 X<0?
139 LOG
140 ST+ 00
141 +
142 LASTX
143 X<>Y
144 X<Y?
145 ASIN
146 52
147 X<>Y
148 X>Y?
149 ASIN
150 STO 04
151*LBL 10
152 "GRATI.?
"
153 PROMPT
154 FS?C 22
155 XEQ 11
156*LBL 12
157 "SONSTIG
ES?"
158 PROMPT
159 FS? 22
160 XEQ 13
161 FS?C 22
162 GTO 12
163 "FREIBET
RAG?"
164 PROMPT
165 FS?C 22
166 ST- 02
167 RCL 00
168 ST+ 02
169 ST+ 03
170 ,4
171 *
172 250
173 X>Y?
174 X<>Y
175 FS?C 19
176 ST- 02
177 CLA
178 ARCL 06
179 ARCL 02
180 PROMPT
181 CLA
182 ARCL 07
183 ARCL 03
184 PROMPT
185 "BRUTTO="
"
186 ARCL 00
187 PROMPT
188 RCL 02
189 XEQ 14
190 12
191 *
192 STO 07
193 9
194 %
195 4752
196 X>Y?
197 X<>Y
198 STO 08
199 GTO IND
10
200*LBL 11
201 ST+ 00
202 600
203 ST- 02
204 6
205 /
206 X>Y?
207 X<>Y
208 ST- 03
209 RTN
210*LBL 13
211 RDN
212 RDN
213 AON
214 CLA
215 ARCL 06
216 "+?"
217 PROMPT
218 ASTO X
219 "J"
220 ASTO Y
221 X=Y?
222 SF 07
223 CLA
224 ARCL 07
225 "+?"
226 PROMPT
227 ASTO X
228 X=Y?
229 SF 08
230 RDN
231 RDN
232 ST+ 00
233 FC?C 07
234 ST- 02
235 FC?C 08
236 ST- 03
237 AOFF
238 RTN
239*LBL 14
240 3
241 +
242 4,5
243 /
244 INT
245 4,5
246 *
247 1,49
248 +
249 RTN
250*LBL 01
251 0
252 STO 09
253*LBL 04
254 1314
255 GTO 15
256*LBL 02
257 RCL 09
258 X#0?
259 SF 07
260 2136
261 ENTER↑
262 2178
263 FS?C 07
264 +
265 GTO 15
266*LBL 03
267 1584
268*LBL 15
269 STO 10
270 XEQ 16
271 SF 07
272 XEQ 16
273 RCL Z
274 +
275 XEQ 18
276 RCL 10
277 +
278 CHS
279 RCL 07
280 +
281 FS? 03
282 XEQ 19
283 FC? 03
284 XEQ 20
285 GTO 21
286*LBL 05
287*LBL 06
288 SF 09
289 0
290 STO 09
291 RCL 07
292 FS? 05
293 1044
294 FS? 06
295 18
296 -
297 XEQ 18
298 STO 07
299 STO 10
300 ,22
301 ST* 10
302 RDN
303 1,5
304 *
305 XEQ 19
306 X<> 07
307 2,5
308 *
309 XEQ 19
310 RCL 07
311 -
312 RCL 10
313 INT
314 X<=Y?
```

```
315 X<>Y
316*LBL 21
317 1,2
318 /
319 INT
320 10
321 /
322 "LST="
323 ARCL X
324 PROMPT
325 STO 01
326 RCL 09
327 X=0?
328 SF 07
329 1
330 -
331 150
332 *
333 20
334 -
335 50
336 X>Y?
337 X<>Y
338 RDN
339 FS? 04
340 XEQ 17
341 FS?C 07
342 CLX
343 -
344 8
345 %
346 XEQ 22
347 X<=0?
348 SF 18
349 ,6
350 X<=Y?
351 RDN
352 FS?C 18
353 CLX
354 FS? 17
355 XEQ 17
356 XEQ 22
357 "E KST="
358 ARCL X
359 FS? 16
360 PROMPT
361 "K KST="
362 ARCL X
363 FS? 15
364 PROMPT
365 ENTER↑
366 FS? 17
367 +
368 ST+ 01
369 RCL 03
370 XEQ 14
371 2,25
372 -
373 STO 03
374 440
375 X>Y?
376 SF 07
377 10
378 *
379 X<=Y?
380 STO 03
381 4
382 /
383 3
384 *
385 FS? 00
386 XEQ 23
387 X>Y?
388 X<>Y
389 5,5
390 FS? 07
391 CLX
392 FS? 08
393 CLX
394 %
395 RND
396 ST+ 01
397 "G="
398 ARCL X
399 PROMPT
400 RCL 03
401 9,25
402 FS? 07
403 CLX
404 %
405 RND
406 ST+ 01
407 "K="
408 FS? 00
409 "L="
410 ARCL X
411 PROMPT
412 CLX
413 1,5
414 FS?C 07
415 CLX
416 FS?C 20
417 CLX
418 %
419 RND
420 ST+ 01
421 "M="
422 ARCL X
423 PROMPT
424 RCL 04
425 ST+ 01
426 "VWL="
427 ARCL X
428 PROMPT
429 "ABZUEGE
=?"
430 ASTO 10
431 PROMPT
432 FS?C 22
433 ST+ 01
434 CLA
435 ARCL 10
436 "HE="
437 ARCL 01
438 PROMPT
439 RCL 00
440 RCL 01
441 -
442 "NETTO="
443 ARCL X
444 PROMPT
445 181,5
446 X<>Y
447 "ERS.K="
448 ARCL Y
449 FS? 08
450 PROMPT
451 FS?C 08
452 +
453 RCL 09
454 2
455 X<Y?
456 SF 07
457 RDN
458 RDN
459 RCL 04
460 FC? 07
461 ,3
462 FS?C 07
463 ,4
464 *
465 "SPARZUL
.="
466 ARCL X
467 PROMPT
468 +
469 "ERSTATT
UNG ?"
470 PROMPT
471 FS?C 22
472 +
473 "AUSZAHL
EN="
474 ARCL X
475 PROMPT
476*LBL 16
477 4200
478 FS? 07
479 XEQ 17
480 FC? 03
481 XEQ 17
482 600
483 FS?C 07
484 XEQ 17
485 FS? 04
486 XEQ 17
487 RCL 09
488 *
489 +
490 RCL 08
491 X>Y?
492 X<>Y
493 RTN
494*LBL 17
495 2
496 /
497 RTN
498*LBL 19
499 2
500 /
501 XEQ 20
502 2
503 *
504 RTN
505*LBL 18
506 54
507 /
508 INT
509 54
510 *
511 RTN
512*LBL 22
513 1 E2
514 *
515 INT
516 1 E2
517 /
518 RTN
519*LBL 23
520 X<=Y?
521 SF 08
522 RTN
523*LBL 20
524 FC? 09
525 XEQ 18
526 STO 05
527 1 E4
528 /
529 6
530 X<>Y
531 X>Y?
532 SF 07
533 1,8
534 -
535 STO 06
536 4,2
537 -
538 FS? 07
539 STO 06
540 FS?C 07
541 GTO 24
542 18 E3
543 RCL 05
544 X<=Y?
545 GTO 26
546 RCL 06
547 3,05
548 *
549 73,76
550 -
551 RCL 06
552 *
553 695
554 +
555 RCL 06
556 *
557 2208
558 +
559 RCL 06
560 *
561 3034
562 +
563 INT
564 RTN
565*LBL 24
566 13 E4
567 RCL 05
568 X>Y?
569 GTO 25
570 RCL 06
571 ,09
572 *
573 5,45
574 -
575 RCL 06
576 *
577 88,13
578 +
579 RCL 06
580 *
581 5040
582 +
583 RCL 06
584 *
585 20018
586 +
587 INT
588 RTN
589*LBL 25
590 RCL 05
591 ,56
592 *
593 14837
594 -
595 INT
596 RTN
597*LBL 26
598 4213
599 X<>Y
600 X>Y?
601 GTO 27
602 0
603 RTN
604*LBL 27
605 ,22
606 *
607 926
608 -
609 INT
610 END
```

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, ^{T}AD und ^{T}DA können irgendwelchen Tasten zugeordnet werden

Ausführung: 1. ASA-Wert eingeben ($3 \leq x \leq 12800$), XEQ ^{T}AD
2. DIN-Wert eingeben ($6 \leq x \leq 42$), XEQ ^{T}DA

Listing:

```
001  LBL'DA      DIN → ASA
     FIX 0       Status festlegen
     CF 29
     INT         etwaiger Dezimalteil
     STO 00      unterdrücken, ab-
                 speichern
     3
     MOD
     GTO IND X   die passende Reihe
                 aussuchen
009  LBL 00      1. Reihe
     12800       Anfangswert
     XEQ 03
     INT
013  LBL 04      Anzeige-Routine
     CLA
     ARCL 00
     ⊢'DIN='
     ARCL X
     ⊢'ASA'
     GTO 06
019  LBL 02      2. Reihe
     1 E 4       Anfangswert
     XEQ 03
     700
023  LBL 05      Routine, die die
     x<>y        Unregelmäßigkeit der
     x>y?        Reihen ausgleicht
     GTO 04
     1,024
     *
     GTO 04
031  LBL 01      3. Reihe
     8 E 3       Anfangswert
     XEQ 03
     70
     GTO 05
036  LBL 03      ASA-Wert be-
     2           rechnen
     RCL 00
     3           Anfangswert
     /           ------------------
     14          2^INT|DIN/3 - 14|
     -
     ABS
     INT
     y^x
     /
     RTN
048  LBL'AD      ASA → DIN
     FIX 0       Status fest-
                 legen
     CF 29
     STO 00
     1 E 2       Umrechnung
     /
     LN              ln ASA/100
     2           3 ---------- + 21
     LN              ln 2
     /
     3
     *
     21
     +
     RND
     CLA
     ARCL 00
     ⊢'ASA='
     ARCL X
     ⊢ DIN'
068  LBL 06      Standardmodus
     AVIEW       herstellen
     SF 29
     FIX 2
072  END
```

Olaf Gursch (120)
Grüntenstraße 29 b
1000 Berlin 42

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 Umrechnerei 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 Ø wird als Zwischenspeicher benutzt. Für die Benutzung mit anderen Währungen (Ausland) ist das Programm UMR als Unterprogramm erforderlich.

1.2 CHANGE)
1.3 WECHS): 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.

1.4 EINH: Mit EINH wird der Austausch der Währungseinheiten bzw. Kurse zwischen den von UMR benutzten Rechenspeichern 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 ØØ setzen. Dies verkürzt die Ausführung. Ist FLAG ØØ 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 Fahrtkosten 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 Kilometerschil, 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 Geschwingigkeitskorrektur 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,O4	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 "WIEVIEL DM?"
08 PROMPT
09 ENTER↑
10 "WIEVIEL "
11 ARCL 07
12 PROMPT
13 /
14 STO 08
15 "1 "
16 ARCL 07
17 "├="
18 ARCL 08
19 "├DM"
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 WAEHRUNG"
04 "├?"
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 "WIEVIEL "
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 "├:"
26 ARCL 15
27 "├="
28 ARCL X
29 PROMPT
30 END
```

```
01♦LBL "RMU"
02 RCL 08
03 1/X
04 STO 09
05 "WIEVIEL DM?"
06 PROMPT
07 STO 00
08 RCL 09
09 *
10 ARCL 00
11 "├DM="
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 "├EN: "
21 ARCL X
22 "├DM"
23 AVIEW
24 PSE
25 RCL 26
26 X<>Y
27 -
28 "RESTBETAG: "
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 "├KM/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 "├L"
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 "├L"
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 "├DM"
13 AVIEW
14 PSE
15 FIX 2
16 RCL 00
17 *
18 "AUTOKOSTEN:"
19 ARCL X
20 "├DM"
21 PROMPT
22 END
```

3. TIMER: s. auch HP KEY-NOTESVol.5 No.1

```
Ø1 LBL'TIMER'          Ø9  1              18  *
Ø2 LBL A               1Ø  ST+ ØØ         19  1/X
Ø3  'SICHER FAHREN'    11  GTO ØØ         2Ø  'AV='
Ø4  AVIEW              12 LBL B           21  ARCL X
Ø5 LBL Ø1              13  RCL ØØ         22  |- KM/H
Ø6  STO ØØ             14  1ØØØØ         23  AVIEW
Ø7 LBL ØØ              15  /              24  GTO Ø1
Ø8  PSE                16  HR             25  RTN
                       17  1,5            26  END
```

PROGRAMMBEISPIEL:

```
EINGABE                FUNKTION       ANZEIGE
                       SIZE Ø27
EINGABE DES PROGRAMMPAKETES
                       CLRG
                       ASN'TIMER'11
                       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'KMKO'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
25ØØ                   R/S            1 OES=Ø,14 DM
                       XEQ'CHANGE'(15) EINHEIT?
LIRE                   R/S            WIEVIEL DM?
1Ø3,75                 R/S            WIEVIEL LIRE
5ØØØØ                  R/S            1 LIRE=2,Ø8E-3DM
                       XEQ'TANKEN'(23)DM?=1
1                      R/S            1 L KOSTET?
1,32                   R/S            WIEVIEL LITER?
31,83                  R/S            42,Ø2
                                      KM-STAND?
83Ø65                  R/S            AUF 1ØØ KM:8,51L
                                      GEFAHRENE KM 374,ØØ
                                      DURCHSCHNITTLIC
                                      HER VERBRAUCH:8,51
                       XEQ'URLAUB'(13)PROGRAMM?
2                      R/S            ERNAEHRUNG
                                      DM? =1
1                      R/S            DM-BETRAG?
15                     R/S            15,ØØ
                       XEQ'NACH'(24)  NICHT VOLL
                                      GETANKT? =Ø
Ø                      R/S            DM? =1
1                      R/S            1 L KOSTET?
1,489                  R/S            WIEVIEL LITER?
1Ø,Ø7                  R/S            14,99
                       XEQ'URLAUB'(13)PROGRAMM?
1                      R/S            UNTERKUNFT
                                      DM? =1
Ø                      R/S            ZUR ZEIT: LIRE
                                      AENDERUNG? ≠ Ø
2                      R/S            WELCHE WAEHRUNG
OES                    R/S            WIEVIEL OES
175                    R/S            25,36
                       R/S            PROGRAMM?
2                      R/S            ERNAEHRUNG
                                      DM? =1
Ø                      R/S            ZUR ZEIT: OES
                                      AENDERUNG? /Ø
Ø                      R/S            WIEVIEL OES
```

```
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'KMKO
END           75 BYTES
LBL'TIMER
END           67 BYTES
```

EINGABE	FUNKTION	ANZEIGE
1ØØ	R/S	14,49
	R/S	PROGRAMM?
4	R/S	FAHRTKOSTEN
		DM? =1
5	R/S	ZUR ZEIT:OES
		AENDERUNG? ≠Ø
Ø	R/S	WIEVIEL OES
17Ø	R/S	24,63
	SF ØØ	
	XEQ'TANKEN'(23)	DM? =1
2	R/S	1 L KOSTET?
1Ø,84	R/S	WIEVIEL LITER?
28,Ø4	R/S	44.Ø4
		KM-STAND?
836Ø8	R/S	AUF 1ØØ KM:7,Ø2 L
		GEFAHRENE KM 917,ØØ
		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 ØØ	
	XEQ'EINH'(34)	WELCHE WAEHRUNG?
LIRE	R/S	2,Ø8 -Ø3
	XEQ'TANKEN'(23)	DM? =1
Ø	R/S	1 L KOSTET?
96Ø	R/S	WIEVIEL LITER?
1Ø,6	R/S	21,56
		KM-STAND?
8372Ø	R/S	AUF 1ØØ KM:9,46 L
		GEFAHRENE KM 1Ø29,ØØ
		DURCHSCHNITTLIC
		HER VERBRAUCH: 7,83 L
	XEQ'UMR'(21)	ZUR ZEIT: LIRE
		AENDERUNG? ≠Ø
6	R/S	WIEVIEL LIRE
45ØØ	R/S	LIRE:45ØØ,ØØ=9,34
	XEQ'RMU'(22)	WIEVIEL DM?
112	R/S	112,ØØDM=53975,ØØ LIRE
	XEQ'SUM'(14)	HOECHSTBETRAG?
5ØØ	R/S	BISHER AUSGEGEB
		EN: 2Ø4,9Ø DM
		RESTBETRAG: 295,1Ø
	XEQ'KMKO'(33)	KM-STAND?
8372Ø	R/S	GEFAHRENE KM:1Ø29,ØØ
		1 KM KOSTETE:Ø,3783 DM
		AUTOKOSTEN: 389,32

BLEIBT NACHZUTRAGEN, WAS IHR EUCH LÄNGST GEDACHT HABT, IN KLAMMERN SIND DIE TASTEN-NUMMERN ANGEGEBEN LAUT ZUORDNUNG.

Guido Studer
Turnhallenweg 7a
CH-4143 Dornach

Programm: Entfernungsberechnung mit Koordinaten

a) Beschreibung

Dieses Entfernungsberechnungsprogramm ist ein speziell auf dem HP-41C entwickeltes Anwenderprogramm, das nach der Formel

$$E_{(miles)} = \cos^{-1}\left[\sin(NL_1)\ \sin(NL_2) + \cos(NL_1)\ \cos(NL_2)\ \cos(OB_2 - OB_1)\right] \times 60$$

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 0-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 *".

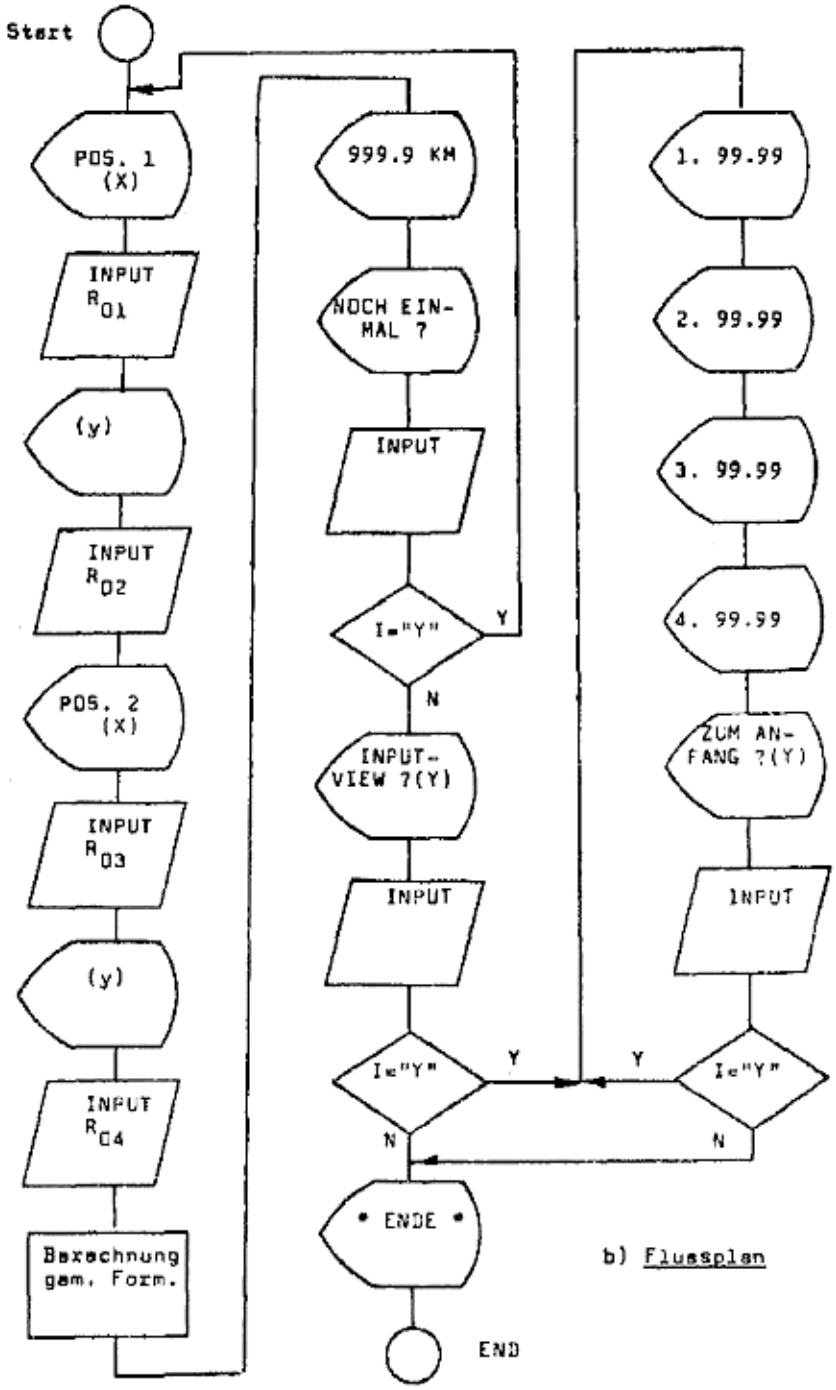
Das im LBL 07 verwendete Zeichen "Y" kann natürlich auch mit der deutschen Eingabe "J" ausgewechselt werden. Ich persönlich ziehe jedoch "Y" vor, weil es unter dem "*"-Zeichen sehr günstig liegt und einfach zu merken ist.

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.

Das ganze Programm kommt auf ~~33 belegte Programmzeilen~~ 240 Bytes, 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

```
01*LBL "ENTF"
02*LBL 01
03 " POS. 1 (X) ?"
04 PROMPT
05 STO 01
06 " (Y) ?"
07 PROMPT
08 STO 02
09 " POS. 2 (X) ?"
10 PROMPT
11 STO 03
12 " (Y) ?"
13 PROMPT
14 STO 04
15 " BERECHNUNG"
16 AVIEW
17 RCL 02
18 -
19 COS
20 RCL 03
21 COS
22 *
23 RCL 01
24 COS
25 *
26 RCL 01
27 SIN
28 RCL 03
29 SIN
30 *
31 +
32 ACOS
33 60
34 *
35 1,8544
36 *
37 STO 00
38 " "
39 ARCL 00
40 "+ KM"
41 PROMPT
42 "NOCH EINMAL?"
43 XEQ 07
44 X=Y?
45 GTO 01
46 "INPUT-VIEW ?"
47 XEQ 07
48 X≠Y?
49 GTO 03
50*LBL 02
51 "KONTROLLE :"
52 AVIEW
53 PSE
54 "1. "
55 ARCL 01
56 AVIEW
57 PSE
58 "2. "
59 ARCL 02
60 AVIEW
61 PSE
62 "3. "
63 ARCL 03
64 AVIEW
65 PSE
66 "4. "
67 ARCL 04
68 AVIEW
69 PSE
70 "ZUM ANFANG ?"
71 XEQ 07
72 X=Y?
73 GTO 01
74*LBL 03
75 " ** ENDE **"
76 PROMPT
77*LBL 07
78 AON
79 PROMPT
80 ASTO Y
81 AOFF
82 "Y"
83 ASTO X
84 END
```

Programm: Entfernungsberechnung mit Koordinaten

d) Beispiel

X = Nördliche Länge Y = Oestliche Breite
-X = Südliche Länge -Y = Westliche Breite

Paris : 48.833 / 2.333 Basel : 47.566 / 7.583

Eingabe:	Anzeige:	
XEQ "ENTF"	POS. 1 (X)	
48.833 (R/S)	(Y)	
2.333 (R/S)	POS. 2 (X)	
47.566 (R/S)	(Y)	
7.583 (R/S)	BERECHNUNG	(ca. 2 sec.)
	413.98 KM	
(R/S)	NOCH EINMAL?	
N (R/S)	INPUT-VIEW?	
Y (R/S)	KONTROLLE :	) jeweils ca. 1 sec.
	1. 48.83	)
	2. 2.33	)
	3. 47.57	)
	4. 7.58	)
	ZUM ANFANG ?	
N (R/S)	** ENDE **	

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 (°) 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 (°), Geschwindigkeit - (nm) und Richtung (°) 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 (nur bei letzter Flugteilstrecke)	Std,Min.

6. Berechnung der Windkomponenten durch XEQ F
   Eingabe der Start-/Landebahnrichtung (°), der Richtung (°) 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 "FN"
CF 01  SF 27  CLRG  CLX
XEQ 12  BEEP

 08*LBL 01
CLA  ARCL 45  ARCL 44
"+?"  PROMPT  STO 00  32
X<>Y  X>Y?  GTO 01

 19*LBL 02
RCL 00  33  X=Y?  GTO 0
RDN  FIX 0  CLA  ARCL 46
"+ "  ARCL 44  ARCL X
AVIEW  STOP  FIX 3
VIEW X  STO IND 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  RDTAX  SF 01  CLX
GTO "FN"

 54*LBL 0
CLA  ARCL 47  ARCL 48
AVIEW  PSE  CLA  ARCL 45
ARCL 44  "+?"  PROMPT
STO 00  RCL IND 00
FIX 4  STOP  STO IND 00
STOP

 71*LBL C
1  ST- 00  RCL 00  X=0?
GTO 05  GTO 02

 78*LBL 03
RCL 00  32  X<>Y  X>Y?
GTO 04  RCL IND 00  9
X=Y?  GTO 04  RCL 00
FIX 0  CLA  ARCL 44
ARCL 46  "+ "  ARCL X
AVIEW  PSE  FIX 3
RCL IND 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 49
FS? 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 IND 00  X=0?  GTO 08
ADV  ADV  SF 12  CLA
ARCL 44  ARCL 00  XEQ 11
RCL IND 00  INT  STO 41
CLA  ARCL 51  "+TRACK "
ARCL 53  ARCL X  AVIEW
RCL IND 00  FRC  1 E3  *
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
X<>Y  RCL 35  RCL 34
P-R  RDN  -  RDN  -  R^
X<>Y  R-P  "GROUND"
ARCL 57  ARCL 53  ARCL X
AVIEW  STO 39  RCL 42
X<>Y  /  ST+ 48  HMS
FIX 2  CLA  ARCL 48
ARCL 58  ARCL 52  ARCL X
AVIEW  1  ST+ 00  FIX 0
GTO 07

275*LBL 06
TONE 0  TONE 5  TONE 9
CLA  ARCL 46  "+ ERROR"
AVIEW  GTO E

284*LBL 08
ADV  "END OF LEGS"
AVIEW  ADV  RCL 48  HMS
FIX 2  "TOTAL FL."
ARCL 58  ARCL X  AVIEW
ADV  ADV  STOP

299*LBL 09
360  MOD  RTN

303*LBL 10
CF 01  1.032  RDTAX  RTN

306*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  X<>Y
"+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 53

405*LBL 13
"TION"  ASTO 54  "DIREC"
ASTO 55  "WIND"  ASTO 56
"SPEED"  ASTO 57  .END.
```

RECHEN-
BEISPIEL :

```
DATA LEG 1.
                  28.100
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.
WINDDIRECTION ?
                      60.
TRUE AIRSPEED ?
                      95.




TAKE OFF/
LANDING

DIRECTION ?
                     110.
WINDDIRECTION ?
                      60.
WINDSPEED ?
                      10.

+HEAD/-TAILWIND
                     6.8
+RIGHT/-LEFT
CROSSWIND
                    -7.3
```

```
LEG 1.
------

TRUE TRACK      28.
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   183.
MAG. HEADING   186.
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

NAME	LABEL	QUELLE	AUTOR
Mondphasen/ Finsternisse	PH FI	Prisma 1980	Günter Lelarge
Ephemeride	EP	Uwe Hommelsheim	Uwe Hommelsheim

Nicht aufgenommenes Programm

NAME	LABEL	QUELLE	AUTOR
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 sowie 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"
02 FC? 04
03 XEQ 05
04 SF 00
05 CF 03
06 "PHAS: "
07 GTO 04
08*LBL "FI"
09 FC? 04
10 XEQ 05
11 "FINST: "
12 SF 03
13 CF 00
14*LBL 04
15 FIX 2
16 CF 29
17 SF 28
18 ADV
19 SF 12
20 ACA
21 CF 12
22 ACX
23 PRBUF
24 ADV
25 STO 26
26 RCL 23
27 *
28 RCL 21
29 -
30 STO 00
31 GTO 03
32*LBL 05
33 RDTA
34 SF 04
35 RTN
36*LBL 12
37 CF 02
38 RCL 09
39 RCL 04
40 SIN
41 *
42 RCL 05
43 ENTER↑
44 +
45 RCL 04
46 -
47 SIN
48 7
49 /
50 +
51 ABS
52 RCL 05
53 COS
54 X<0?
55 SF 02
56 FS? 00
57 GTO 01
58 RCL 00
59 *
60 -
61 RCL 02
62 COS
63 RCL 06
64 *
65 RCL 07
66 +
67 -
68 X>0?
69 GTO 14
70 LASTX
71 RCL 21
72 FRC
73 *
74 /
75 FS? 02
76 "MONDF:"
77 FC?C 02
78 "SONNF:"
79*LBL 01
80 FIX 2
81 RCL 00
82 INT
83 ENTER↑
84 ENTER↑
85 RCL 22
86 -
87 RCL 23
88 /
89 INT
90 STO 25
91 RCL 23
92 *
93 INT
94 -
95 ENTER↑
96 ENTER↑
97 RCL 24
98 /
99 INT
100 STO 20
101 RCL 24
102 *
103 INT
104 -
105 12
106 RCL 20
107 E
108 -
109 X<=Y?
110 GTO 02
111 X<>Y
112 -
113 ISG 25
114*LBL 02
115 E
116 %
117 R↑
118 +
119 RCL 25
120 E6
121 /
122 +
123 RCL 22
124 FRC
125 +
126 STO Y
127 E2
128 *
129 FRC
130 E4
131 *
132 RCL 26
133 INT
134 X<Y?
135 STOP
136 RCL Z
137 FS? 02
138 "VOLLM:"
139 FS? 03
140 GTO 11
141 FC?C 02
142 "NEUM:"
143*LBL 11
144 "⊢ AM "
145 STO Y
146 INT
147 SF 29
148 FIX 0
149 ARCL X
150 R↑
151 FRC
152 E2
153 *
154 FIX 4
155 ARCL X
156 PRA
157 RCL Z
158*LBL 12
159 RCL 00
160 FRC
161 24
162 *
163 HMS
164 FIX 2
165 FS? 01
166 RTN
167 "UM "
168 ARCL X
169 "⊢ UHR"
170 PRA
171 CF 01
172*LBL 14
173 RCL 01
174 RCL 02
175 SIN
176 +
177 ST+ 00
178*LBL 03
179 9,1
180 STO 25
181 XEQ 00
182 ENTER↑
183 ISG 25
184 RCL IND 25
185 +
186 ,2
187 P-R
188 ISG 25
189 RCL IND 25
190 +
191 X<>Y
192 STO 04
193 *
194 +
195 STO 05
196 XEQ 00
197 STO 03
198 XEQ 00
199 STO 02
200 -
201 +
202 +
203 SIN
204 32
205 %
206 +
207 RCL 04
208 +
209 ST+ 03
210 RCL 03
211 RCL 04
212 RCL 24
213 D-R
214 /
215 +
216 ST+ 02
217 RCL 02
218 ,7
219 ST* 04
220 P-R
221 9
222 +
223 *
224 ST+ 03
225 RCL 03
226 RCL 05
227 +
228 3
229 LN
230 P-R
231 2
232 D-R
233 -
234 *
235 ST+ 03
236 RCL 03
237 RCL 04
238 XEQ 00
239 -
240 +
241 STO 04
242 ENTER↑
243 +
244 SIN
245 9
246 /
247 -
248 ST+ 05
249 RCL 06
250 RCL 05
251 TAN
252 ATAN
253 ABS
254 X<=Y?
255 GTO 12
256 LASTX
257 RCL 02
258 COS
259 8
260 -
261 *
262 E
263 %
264 ST+ 00
265 GTO 03
266*LBL 00
267 ISG 25
268 RCL IND 25
269 RCL 00
270 *
271 ISG 25
272 RCL IND 25
273 +
274 END
```

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. $\Sigma+$ und $\Sigma-$ 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 ($<5{,}729577951\ E{-}99$) 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 angeschlossen 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≠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 00, ENTER, LBL 00, 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 00 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, ← , ALPHA und GTO .001; 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, ← , GTO .001; wieder befindet sich der Rechner im ersten KA-Register.

11. Mit folgenden Schritten verändert man das Flag-Register: GTO .., Σ REG 00 (SIZE 005), CL Σ, EEX 67, STO 02, EEX -97, STO 05, SF 25, SDEV oder MEAN speichert in Register d die Zahl 99 99 9? ?? ?9 99 99 (Hex), wobei ? bedeutet, daß diese Halbbytes nicht verändert werden. Die 9en setzen also die Flags 00, 03, 04, 07, 08, 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 Zeilennumerierung führen. Beispiel: GTO .., +,+,+,+,+, STOP, LN, nun SIZE maximal wählen und außerhalb von PRGM SF 25, GTO .001. Nun in das Programm irgendeinen Befehl einfügen (z.B. ENTER), nach dem PACKING erscheint in der Anzeige: 02 LN

13. Wenn X oder Y negativ sind, weichen die Ergebnisse von X MOD Y und von Y - X·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↑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 5o und die Anzeige. Mit SF 25 und irgendeiner 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 NONEXITENT), LBL OO, PI, SIN, GTO OO; evtl. angeschlossenen Drucker einschalten oder CF 21.

2o. 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, O.
Diese werden also so eingegeben:

8 BLDSPEC, 8 BLDSPEC, 62 BLDSPEC, 8 BLDSPEC, 8 BLDSPEC, O BLDSPEC.
Vorsicht: Die letzte O darf nicht weggelassen werden !

COL					COL					COL					COL				
1→	99:	0:	127:	20:	1→	- :	- :	- :	- :	1→	128:	128:	- :	- :	1→	127:	127:	127:	0:
2→	20:	0:	2:	20:	2→	- :	56:	- :	- :	2→	72:	72:	49:	68:	2→	65:	65:	65:	28:
3→	6:	0:	20:	21:	3→	- :	68:	- :	68:	3→	127:	127:	73:	68:	3→	85:	85:	93:	62:
4→	20:	20:	32:	62	4→	14:	68:	- :	124:	4→	0:	0:	73:	93:	4→	65:	73:	65:	127:
5→	99:	99:	127:	64:	5→	17:	56:	17:	64	5→	88:	0:	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:	0:

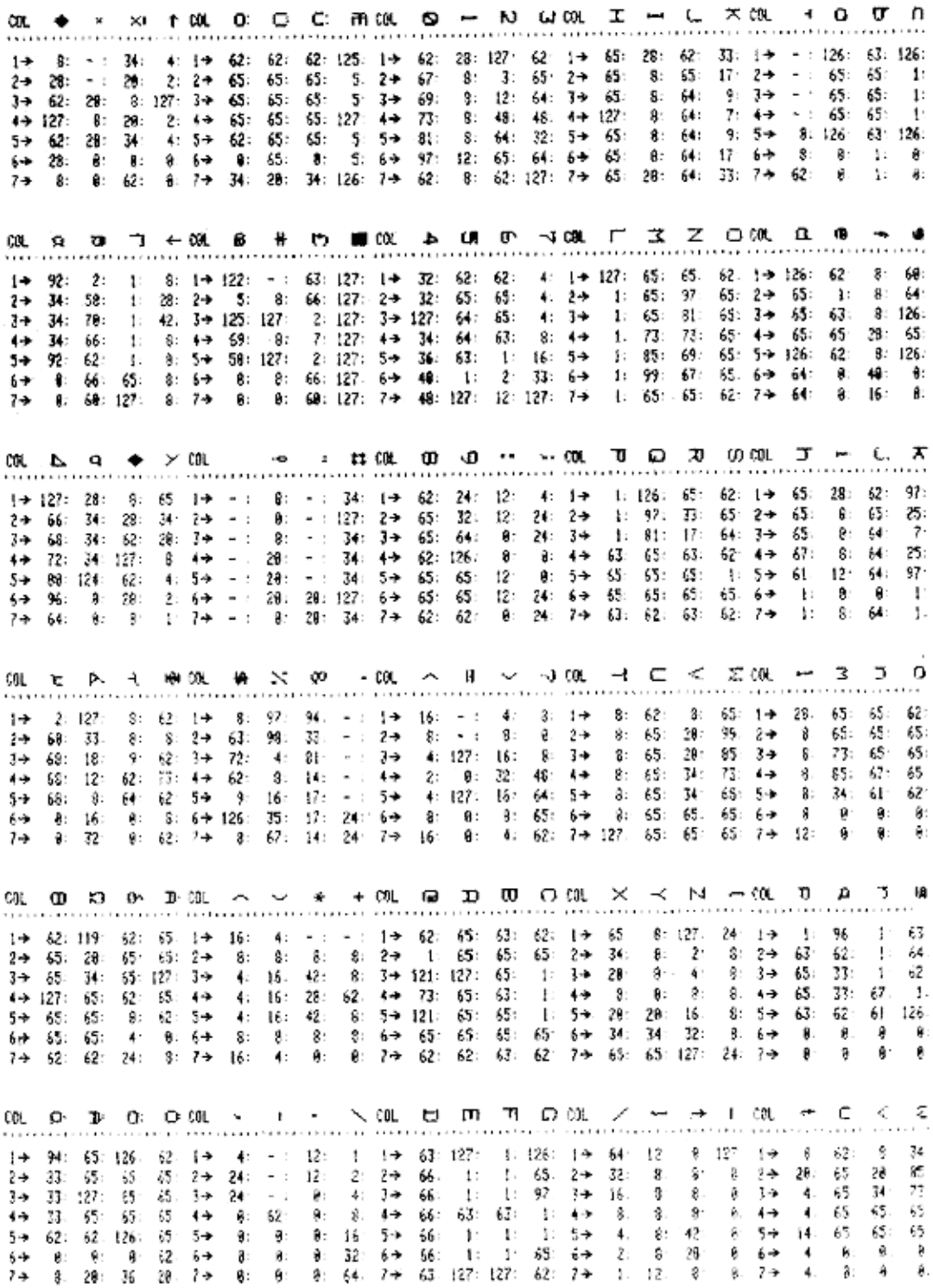
COL					COL					COL					COL				
1→	0:	16:	127:	1:	1→	- :	- :	- :	- :	1→	0:	0:	36:	34:	1→	14:	56:	28:	- :
2→	0:	32:	66:	1:	2→	- :	- :	- :	- :	2→	24:	4:	34:	17:	2→	31:	60:	28:	- :
3→	0:	127:	4:	1:	3→	- :	116:	- :	84:	3→	48:	4:	34:	17:	3→	62:	38:	95:	28:
4→	0	32:	0:	127:	4→	- :	84:	- :	84:	4→	0:	0:	36:	34:	4→	124:	127:	127:	62:
5→	0:	16:	4:	1:	5→	29:	92:	21:	124:	5→	18:	16:	48:	60	5→	62:	38:	95:	62:
6→	0:	0:	66:	1:	6→	21:	0:	21:	0	6→	12:	16:	48:	60:	6→	31:	60:	28:	127:
7→	0:	0:	127:	1:	7→	23:	0:	31:	0:	7→	0:	0:	36:	34:	7→	14:	56:	28:	127:

COL					COL					COL					COL				
1→	- :	- :	- :	- :	1→	- :	- :	- :	- :	1→	- :	- :	- :	28:	1→	- :	127:	65:	- :
2→	- :	- :	- :	- :	2→	- :	- :	- :	- :	2→	42:	60:	81:	34:	2→	- :	127:	65:	- :
3→	- :	100:	- :	16:	3→	- :	28:	- :	92:	3→	42:	60:	82:	28:	3→	28:	62:	34:	- :
4→	- :	84:	- :	56:	4→	- :	16:	- :	84:	4→	42:	84:	84:	0:	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:	88:	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					COL				
1→	- :	- :	- :	- :	1→	- :	- :	- :	- :	1→	3:	- :	- :	0:
2→	- :	- :	- :	- :	2→	- :	- :	- :	- :	2→	13:	65:	- :	4:
3→	- :	16:	- :	32:	3→	- :	124:	- :	4:	3→	49:	127:	28:	60:
4→	- :	16:	- :	16:	4→	- :	84:	- :	4:	4→	65:	65:	42:	60:
5→	4:	16:	0:	0:	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					COL				
1→	- :	- :	- :	- :	1→	- :	- :	- :	- :	1→	- :	3:	56:	28:
2→	- :	- :	- :	- :	2→	- :	- :	- :	- :	2→	4:	12:	60:	34:
3→	- :	56:	- :	- :	3→	- :	124:	- :	28:	3→	120:	16:	64:	93:
4→	- :	60:	- :	60:	4→	- :	84:	- :	28:	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:	60:	34:
7→	0:	0:	14:	0:	7→	31:	0:	31:	0:	7→	0:	3:	56:	28:

COL					COL					COL				
1→	- :	16:	- :	120:	1→	- :	- :	- :	- :	1→	1:	127:	127:	127:
2→	- :	16:	64:	72:	2→	- :	- :	- :	- :	2→	15:	65:	65:	65:
3→	7:	32:	64:	127:	3→	- :	100:	- :	12:	3→	121:	65	69:	69
4→	5:	127:	62:	0	4→	- :	16:	- :	112:	4→	0:	73:	65:	73:
5→	7:	1:	1:	72:	5→	27:	100:	3	12:	5→	48:	65	01	01
6→	0:	1:	1:	48:	6→	4:	0:	20	0	6→	0:	65:	65	65:
7→	0:	1:	0	72:	7→	27:	0:	5	0	7→	120:	127:	127	127:



Best of

PRISMA

CCD - Computerclub Deutschland e. V.