J. S. Dearing

Tricks, Tips und Routinen für Taschenrechner der Serie HP-41

Deutsche Ausgabe von H. Dalkowski

Heldermann Verlag Berlin

J. S. Dearing

Tricks, Tips und Routinen für Taschenrechner der Serie HP-41

Deutsche Ausgabe von H. Dalkowski

Heldermann Verlag Berlin

John S. Dearing
Corvallis Software Inc.
P.O.Box 1412
Corvallis, OR 97339
U.S.A.

Heinz Dalkowski
Seidelbastweg 88a
D-1000 Berlin 47

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Dearing, John S.:
Tricks, Tips und Routinen für Taschenrechner der Serie HP-41 / J.S.Dearing. Dt. Ausg. von H. Dalkowski. - Berlin : Heldermann, 1984.
Einheitssacht.: Calculator tips and routines <dt.>
ISBN 3-88538-802-2
NE: Dalkowski, Heinz (Bearb.)

Das Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die des Nachdrucks, der photomechanischen Wiedergabe und der Speicherung in elektronischen Geräten bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Bei Vervielfältigung zu gewerblichen Zwecken, im Ganzen oder in Teilen, ist eine Vergütung an den Verlag zu bezahlen, deren Höhe mit dem Verlag zu vereinbaren ist.

© Heldermann Verlag Berlin
Herderstr. 6-7
D-1000 Berlin 41

ISBN 3-88538-801-4

Donald und Peggy gewidmet.

INHALTSVERZEICHNIS

Vorwort vi

Kapitel 1. Elementare Funktionen und Operationen 1

Kapitel 2. Tips zum Programmieren 10

Kapitel 3. Startbedingungen und Eingabeaufforderungen 19

Kapitel 4. Die Anzeige 26

Kapitel 5. Behandlung von Alpha-Ketten 33

Kapitel 6. Flags und Töne 37

Kapitel 7. Umgang mit dem Stapel 44

Kapitel 8. Speicher und 'Vorhang' 48

Kapitel 9. Datenregister 52

Kapitel 10. Block-Operationen 56

Kapitel 11. Matrizen und Datenverarbeitung 66

Kapitel 12. Sortieren 74

Kapitel 13. Zufallszahlen 78

Kapitel 14. Gebrochene Zahlen und Rundungsrechnung 81

Kapitel 15. Arithmetik und Algebra 86

Kapitel 16. Geometrie, Trigonometrie und Analysis 99

Kapitel 17. Zahlensysteme 107

Kapitel 18. Umrechnung von Maßeinheiten und Abkürzungsvorschläge 111

Kapitel 19. Statistik und Kombinatorik 116

Kapitel 20. Uhr, Kalender und Finanzen 124

Kapitel 21. Kartenleser und Lesestift 128

Kapitel 22. Der Drucker 134

Kapitel 23. Schrift-Fahnen 161

Kapitel 24. Auflösung von Gleichungen nach verschiedenen Variablen . . . 166

Kapitel 25. Synthetisches Laden von Bytes 171

Kapitel 26. Kryptographie 185

Kapitel 27. Anhang 189

Stichwortverzeichnis 199

Nachwort (Der neue HP-41CX) 209

Vorwort zur deutschen Ausgabe

Dieses Buch ist eine Sammlung von Tips und Routinen für Kleinrechner mit UPN-Logik, insbesondere für Taschenrechner der Serie HP-41. Es soll in erster Linie einen Ratgeber bilden, den man nach Bedarf befragen kann. Es ist nicht dazu gedacht, wie ein Lehr- oder Übungsbuch von vorn bis hinten durchgelesen zu werden. Voraussetzung zum Verständnis des Inhalts ist die Kenntnis der Benutzerhandbücher des Rechners und - sofern sie betroffen sind - der Peripheriegeräte. Auch läßt sich das Programmieren mit diesem Buch nicht erlernen. Indessen kann man seine schon bestehenden Fertigkeiten in dieser Kunst erheblich steigern, wenn man diese oder jene der Routinen sorgfältig analysiert. Es ist zu empfehlen, das Buch zunächst einmal durchzublättern, um einen Eindruck von dem gesammelten Material zu gewinnen. Um gezielt aus der Fülle auszuwählen und angestrebte Auskünfte finden zu können, bediene man sich dann des umfangreichen Stichwortverzeichnisses, dem alles einverleibt wurde, was irgend beim Suchen behilflich sein kann.

Man lasse sich beim ersten oberflächlichen Durchblättern nicht täuschen: Das Material wurde dankenswerterweise von John Dearing so zusammengetragen, daß vom Anfänger bis zum erfahrenen HP-41-Benutzer jeder aus den Kniffen und Routinen Nutzen ziehen kann. Und so stehen denn sehr elementare Dinge, die den Fortgeschrittenen zu einer Unterschätzung der Sammlung verleiten könnten, neben ganz verborgen blühenden äußerst pfiffigen Tips und zwischen Perlen aus der Hohen Schule der Synthetischen Programmierung. Dafür zwei kleine Beispiele: Wer schon kommmt von allein auf die unglaubliche elegante Auswertung des cos-Satzes, die unter 16-10 gegeben wird? Und wer sieht dem unscheinbaren Tip 16-7 auf den ersten Blick an, wieviele Bytes man durch ihn spart, wenn man in einem Programm die zugehörigen Fallunterscheidungen treffen muß?

Entstanden ist das Buch durch die schier herkulische Leistung John Dearings, der über Jahre hinweg die HP Key Notes, das PPC Calculator Journal, die Handbücher von HP, den PPC ROM und sonstige Quellen auswertete, die gefundenen Ideen, Tips und Routinen ordnete und schließlich gar von all den Autoren die Erlaubnis zur Publikation in seinem Buch einholte. Die deutsche Ausgabe ist gegenüber dem amerikanischen Original geringfügig erweitert und in einigen Details korrigiert worden. Insbesondere wurde ein ganz profanes aber recht nützliches, speziell für deutsche Leser gedachtes Programm unter 20-5 aufgenommen. Kapitel 26 ist ganz neu.

Fast alle Routinen sind als Programm-Auflistung abgedruckt. Das hat den Vorteil, daß Druckfehler in den Programmen so gut wie ausgeschlossen sind. Sobald synthetische Funktionen auftreten, können aber Druckbild und Anzeige voneinander abweichen. Auf Seite 178 findet man daher eine Gegenüberstellung der wechselseitig entsprechenden Symbole. Außerdem sind alle Routinen, die synthetische Funktionen enthalten, durch eine dies anzeigende Überschrift ausgewiesen, und selbstverständlich wird für alle

synthetischen Textzeilen, deren Druckbild ihren Inhalt verbirgt, die Byte-Folge in Dezimalzahlen angegeben.

Nur wenigen HP-41-Benutzern wird der Begriff der Synthetischen Programmierung unbekannt sein, jene phantastische, durch die unermüdliche Tätigkeit der PPC-Mitglieder gelungene Entdeckung verborgener Funktionen des HP-41, die Zugang zu den Systemregistern des Rechners verschafft, abenteuerliche Tastenzuweisungen gestattet, die Maschine wesentlich gesprächiger werden läßt und Programme in vielen Fällen schneller und kürzer macht. Standardwerk darüber ist das Buch 'Synthetische Programmierung auf dem HP-41C/CV' von William C. Wickes, erschienen beim Heldermann Verlag Berlin. Ein weiteres einschlägiges Buch, 'Synthetische Programmierung auf dem HP-41 - leicht gemacht' von Keith Jarett, einem anderen Pionier auf diesem Gebiet, erscheint in Kürze im selben Verlag. Dieses zweite Buch ist leichter zu lesen, erfaßt nicht alle Einzelheiten der synthetischen Programmierung, berücksichtigt dafür aber die X-Funktionen und den Time-Modul.

Trotz sorgfältig ausgeführter Korrekturen der Druckvorlagen kann es bei so fehleranfälligem Material wie dem vorliegenden nicht ausbleiben, daß sich Irrtümer und Druckfehler hartnäckig verborgen halten. Jeder Hinweis an den Übersetzer, auch auf kleinste Ungenauigkeiten, wird daher dankbar entgegengenommen. Leser, die selbst mit Tips und Routinen aufwarten können, welche ihrer Meinung nach in dieses Buch passen und in späteren Auflagen erscheinen sollten, werden gebeten, diese direkt an

John Dearing
Corvallis Software Inc.
P.O. Box 1412 Corvallis
OR 97339 USA

zu senden, es sei denn, sie sind allein auf deutsche Leser zugeschnitten wie z.B. 20-5.

Bis auf zwei Ausnahmen (6-16, 26-1/2) sind in den Routinen dieses Buches keine X-Funktionen enthalten. Sie bleiben einem möglicherweise später einmal erscheinenden zweiten Band vorbehalten. Lassen Sie sich jedoch deswegen nicht davon abhalten, auch Vorschläge mit solchen Funktionen schon jetzt an John Dearing zu schicken.

Wer dem PPC-Club beitreten möchte oder das PPC Calculator Journal bestellen will, wende sich an

PPC Calculator Journal
Department T & R
2545 W. Camden Place
Santa Ana CA 92704
USA

Eine kleine Plastik-Byte-Tabelle, die 'HP-41 Combined Hex/Decimal Byte Table' läßt sich unter der Adresse

Heldermann Verlag
Herderstraße 6-7
D - 1000 Berlin 41
Fed. Rep. Germany

anfordern. Diese Tabelle paßt in die Schutzhülle des Rechners und ist somit stets zur Hand.

Das in diesem Buch enthaltene Material ist mit keinerlei Verpflichtung oder Garantie irgendeiner Art verbunden. Die Verlage Corvallis Software Inc. und Heldermann sowie die beiden Herausgeber übernehmen keine Verantwortung und keine als Folge auftretende oder sonstige Haftung, die auf irgendeine Art aus der Benutzung dieses Materials oder Teilen davon entstehen könnte.

Kapitel 1

ELEMENTARE FUNKTIONEN UND OPERATIONEN

1-1 Standardzeichen ins X-Register bringen

Ein Drucker ist erforderlich. Vergewissern Sie sich, daß kein Alpha-Zeichen im Y-Register steht, und bringen Sie die 'ACCHR'-Nummer n ($1 \leq n \leq 127$) ins X-Register; dann führen Sie 'XEQ "BLDSPEC"' aus. Das gewünschte Zeichen entsteht im X-Register und kann von dort aus in jedes numerische oder Stapelregister gespeichert oder mit 'ARCL X' ins Alpha-Register gebracht werden. Beispielsweise bringt '40, XEQ "BLDSPEC"' die linke Klammer "(" ins X-Register; ähnlich ergeben '41' und '38' die Zeichen ")" und "&". Entnehmen Sie die anderen Zeichen, die sich erzeugen lassen, der Byte-Tabelle. Viele Zahlen ergeben allerdings nur das 'Vollzeichen' ▩.

1-2 Positionierung

Im RUN- oder PRGM-Modus positioniert 'SHIFT GTO.', gefolgt von einer *globalen Marke*, den HP-41 auf eben diese Marke im Programmspeicher (der '.' ist im RUN-Modus nicht nötig). 'SHIFT GTO.', gefolgt von einer *3-ziffrigen Zahl*, setzt den Zeiger auf eben diese Zeilen-Nummer in dem Programm, in welchem sich der Rechner gerade befindet (für Zeilen-Nummern von 1000 bis 1999 benutzen Sie 'EEX', gefolgt von den letzten 3 Ziffern der angesteuerten Zeile). Im RUN-Modus (*nicht* im PRGM-Modus!) setzt ein 'SHIFT GTO nm' ('nm' eine 2-ziffrige Zahl) den Zeiger auf eben diese *numerische Marke* in dem Programm, auf das der Rechner positioniert ist, und zwar (vorwärtssuchend) auf die Stelle, an der diese Marke, falls sie mehrfach verwendet wurde, das erste Mal erscheint. Ebenfalls nur im RUN-Modus setzt ein 'SHIFT RTN' den Zeiger auf die *Zeile 00* des gegenwärtigen Programms.

Verlust offener Rücksprünge

Sämtliche von 'RTN'- oder 'END'-Befehlen auszulösenden Rücksprünge gehen verloren, wenn ein Unterprogramm von Hand über das Tastenfeld aufgerufen, wenn 'SHIFT RTN' gedrückt oder wenn 'SIZE' ausgeführt wird.

Dezimalpunkte und Exponenten

Wenn (im RUN- oder PRGM-Modus) eine Zahl eingetastet werden soll, die mehr als 8 Ziffern und einen Exponenten von 10 enthält, muß spätestens vor die neunte Ziffer ein Dezimalpunkt gesetzt werden.

'R/S' verglichen mit 'STOP'

Ein laufendes Programm kann mit 'R/S' angehalten werden. Obwohl die 'STOP'-Funktion (zur Verwendung im USER-Modus) auch anderen Tasten zugewiesen werden kann, erfüllen

so belegte Tasten (im RUN-Modus) ihre neue Aufgabe nicht: sie können kein Programm stoppen. Umgekehrt unterbricht ein Druck auf die 'R/S'-Taste ein laufendes Programm in jedem Fall, auch wenn dieser Taste eine andere Funktion zugewiesen wurde und der Rechner sich im USER-Modus befindet. Wenn kein Programm läuft, startet ein 'R/S' das Programm auf der Zeile, auf die der Rechner positioniert ist, es sei denn, die 'R/S'-Taste ist mit einer anderen Funktion belegt, und der USER-Modus ist eingeschaltet.

- Quelle: HP-41 Bedienungshandbuch, © Copyright (Juni 1980) Hewlett-Packard Company. Abgedruckt mit Erlaubnis.

1-3 Symbolbezeichnungen

'Graugans': ⊦ ; 'verkehrte Graugans': ⊣ ; 'Text-T' oder hochgestelltes T: ᵀ ; Text-Folge-Zeichen oder 'faules T': ⊦ ; 'Vollzeichen': ▒ .

1-4 Verlangsamte Katalogdurchsicht

Hält man während einer Katalogdurchsicht eine beliebige Taste (außer 'R/S') niedergedrückt, verlangsamt sich das 'Durchblättern' des Katalogs um den Faktor 7. - Quelle: Richard Nelson (1) (PPC J, V6N5P32).

1-5 Indirekte Benutzung von Alpha-Marken

Obwohl Programm-Marken aus bis zu 7 Alpha-Zeichen (Komma, Punkt und Doppel-Punkt sind ausgeschlossen) bestehen können, müssen sie auf höchstens 6 Zeichen beschränkt werden, wenn sie indirekt adressiert werden sollen. Kürzere Marken sparen außerdem Bytes ein; es empfiehlt sich daher, sie kurz zu halten (maximal 6 Zeichen).

1-6 Fehlende bedingte Verzweigungen

Für 'X ≥ 0?' benutze man 'X ≠ 0?', 'X > 0?' aufeinanderfolgend; für 'X ≥ Y?' benutze man 'X ≠ Y?', 'X > Y?'.

1-7 Letzte zwei Ziffern bei 'SCI'-Darstellung sichtbar machen

'EEX nm' ('nm' der angezeigte Exponent; Vorzeichen berücksichtigen!), '/', 'FIX 9'. Alle 10 Ziffern sind daraufhin sichtbar. Rückkehr zur ursprünglichen Zahl mit 'LAST X', '*'. - Quelle: James Pittman (1002) (65 Notes, V3N9P15).

1-8 'Knapp passende' HP-67/97 Programme in einen nicht erweiterten HP-41C bringen

Setzen Sie die Anzahl der Datenregister herunter, laden Sie, packen Sie, und setzen Sie dann die Anzahl der Datenregister wieder herauf. - Quelle: Richard Nelson (1) (PPC J, V6N5P32).

1-9 Totallöschung

Um den gesamten Speicher zu löschen, halten Sie die Korrekturtaste gedrückt, während Sie den Rechner anschalten. Wenn Sie anschließend die Korrekturtaste loslassen, erscheint die Meldung "MEMORY LOST". Diese Meldung können Sie mit der Korrekturtaste löschen. Sollten Sie es sich im letzten Augenblick noch anders überlegen, lassen Sie die Korrekturtaste los, bevor Sie 'ON' loslassen. - Quelle: Bill Kolb (265).

1-10 Bytes

Der HP-41C hat in der Grundausstattung $63+\frac{4}{7}$ Register als Programmspeicher zur Verfügung. Er hat tatsächlich volle 64 Register (448 Bytes), doch enthalten die letzten 3 Bytes das ständige '.END.', so daß nur 445 Bytes für Programme übrig bleiben. Die lokalen Alpha-Marken A-J und a-e benötigen nur 2 Bytes, während die globalen Alpha-Marken mindestens 5 Bytes (4 + 1 für jedes Alpha-Zeichen) verbrauchen. Sind in einem Programm N-mal k gleiche Bytes hintereinander enthalten, erspart man sich $(N - 1) \cdot (k - 3) - 5$ Bytes, wenn man die k Bytes als Unterprogramm abarbeiten läßt. Bei dieser Formel ist vorausgesetzt, daß das Unterprogramm unter einer Kurzform-Marke (00-14) aufgerufen wird. - Quelle: HP Key Notes, V4N2P11. - Vgl. auch 1-27.

1-11 'USER'-Taste als 'Vorschalttaste'

Die 'USER'-Taste kann als echte Vorschalttaste (so wie 'SHIFT') eingesetzt werden, wenn der Routine, die einer Taste zugewiesen ist, ein 'CF 27' (USER-Modus unwirksam machen) eingefügt wird. Beispiel: 'LBL "SHIFT", FACT, CF 27, RTN'. Weisen Sie jetzt "SHIFT" der Taste 'EEX' zu, tasten Sie eine ganze Zahl zwischen 1 und 69 ein, und drücken Sie 'USER, EEX'. Die Fakultät der Zahl wird angezeigt, und der USER-Modus wird gelöscht. - Quelle: Jake Schwartz (1820) (PPC CJ, V6N7P4).

1-12 'Entern' ohne 'ENTER'

In einem Programm können Zahlen in den Stapel gebracht werden, ohne den Befehl 'ENTER' zu benutzen. Schalten Sie nach dem Eintasten der Zahl den USER-Modus ein und aus, um den Ziffern-Eintrag abzuschließen (ein 'ENTER', gefolgt von '←', tut es auch), und geben Sie anschließend die nächste Zahl ein. Das erspart Papier beim Auflisten der Programme; *jedoch:* der HP-41 setzt hinter jede Zeile, die einen Zahlen-Eintrag beinhaltet, ein nicht durch 'PACK' zu beseitigendes 'NULL' (nicht die Zahl '0'!), wenn eine weitere Zeile mit einem Zahlen-Eintrag folgt. Dieses 'NULL' beansprucht ein Byte; daher erspart man sich durch das Verfahren keine Bytes. - Quelle: Richard Nelson (1) (PPC CJ, V7N2P56).

1-13 HP-41 'verschließen' ("LO")

'LBL "LO", LBL 00, SF 11, OFF, GTO 00, END'. Wenn 'ON' gedrückt wird, erklingt 'TONE 7', und der HP-41 schaltet sich sofort wieder aus, wodurch Unbefugte, denen dieses Verfahren unbekannt ist, daran gehindert werden, im Rechner herumzupfuschen. Die Routine verhindert auch Stromverbrauch durch zufälliges Anschalten, wenn der Rechner mitgeführt wird. Man überlistet "LO", wenn man 'R/S' beim Einschalten gedrückt hält. - Quelle: HP Key Notes, V4N1P4; Bill Kolb (265).

1-14 'SIZE'-Sucher ("SZE")

Die folgende Routine findet die gegenwärtige Anzahl von Datenregistern. - Quelle: Richard Nelson (1) (PPC CJ, V7N1P0).

```
01♦LBL "SZE"      11 ISG X          21 INT          31 LASTX
02 0              12 ARCL IND X     22 X<>Y         32 INT
03 SF 25          13 FS? 25         23 4            33♦LBL 10
04 ARCL IND X     14 GTO 01         24 /            34 FIX 0
05 FC? 25         15 FRC            25 ,24          35 "SIZE= "
06 GTO 10         16 X=Y?           26 +            36 ARCL X
07 ,32001         17 GTO 09         27 +            37 AVIEW
08 ENTER↑         18 LASTX          28 SF 25        38 FIX 2
09 ,32064         19 DSE X          29 GTO 01       39 END
10♦LBL 01         20 ABS            30♦LBL 09
```

(80 Bytes)

1-15 Schneller 'SIZE'-Sucher ("SZ")

Quelle: Ron Knapp (618) (PPC CJ, V7N2P38).

```
01♦LBL "SZ"       08 2              15 CHS          22 "SIZE= "
02 9              09 /              16 DSE Z        23 ARCL X
03 ENTER↑         10 SF 25          17 GTO 01       24 AVIEW
04 -1             11 ST+ Y          18 X<0?         25 FIX 2
05 ENTER↑         12 ABS            19 CLX          26 END
06 512            13 ARCL IND Y     20 +
07♦LBL 01         14 FC?C 25        21 FIX 0
```

(51 Bytes)

1-16 Synthetischer 'SIZE'-Sucher ("?S")

Diese Routine schreibt die gesuchte Anzahl ins X-Register. Der ursprüngliche Wert aus X wird in Y, Z und T abgelegt, während im Alpha-Register 'Abfall' zurückbleibt. WARNUNG: Wie in anderen Programmen, die Ergebnisse aus dem Flag-Register benutzen, kann auch hier eine falsche Antwort entstehen: wenn man die Routine zwischen den Zeilen 6 und 9 mit 'SST' abarbeitet, wird zur Unzeit Flag 51 gesetzt, und wenn man bei angeschlossenem Drucker zwischen diesen Zeilen mit 'R/S' unterbricht und wieder startet, wird (ebenfalls zu Fehlern führend) Flag 55 gesetzt. - Zeile 04 hängt 4 'NULL'en an. - Quelle: Roger Hill (4940) (PPC CJ, V7N5P57).

```
01♦LBL "?S"     09 X<> d        17 *            25 RTN
02 RCL c        10 ENTER↑       18 INT          26 X<> L
03 STO [        11 INT          19 64           27 +
04 "├♦♦♦♦"      12 HMS          20 MOD          28 GTO 00
05 X<> [        13 7            21 SF 25        29 END
06 X<> d        14 *            22♦LBL 00
07 CF 02        15 +            23 RCL IND X
08 CF 03        16 -10          24 FC? 25
```

(56 Bytes)

1-17 Synthetischer Sucher für 'ΣREG', 'SIZE' und den 'Vorhang' ("Σ?", "S?" & "C?")

'XEQ "Σ?"', um die Nummer des ersten Registers des Statistik-Blocks zu finden. 'XEQ "S?"', um die Anzahl der Datenregister zu finden. 'XEQ "C?"', um die gegenwärtige Lage des Vorhangs (also die absolute Adresse des Datenregisters R_{00}) zu finden. - Zeile 26 lautet dezimal 244, 127, 0, 0, 1. - Quelle: Keith Jarett (4360), Roger Hill (4940) & Clifford Stern (4516) (PPC CJ, V7N10P15; PPC ROM).

```
01♦LBL "Σ?"     14 SF 25        27 X<> [        40 SF 11
02 CLA          15♦LBL 02       28 X<> d        41 FS?C 14
03 XEQ "C?"     16 RCL IND X    29 CF 01        42 SF 13
04 RCL \        17 FC? 25       30 CF 02        43 FS?C 15
05 XEQ 14       18 RTN          31 CF 04        44 SF 14
06 X<>Y         19 X<> L        32 CF 07        45 FS?C 16
07 -            20 +            33 FS?C 10      46 SF 15
08 RTN          21 GTO 02       34 SF 07        47 X<> d
09♦LBL "S?"     22♦LBL "C?"     35 FS?C 11      48  E2
10 XEQ "C?"     23 RCL c        36 SF 09        49 *
11 CHS          24♦LBL 14       37 FS?C 12      50 INT
12 64           25 STO [        38 SF 10        51 DEC
13 MOD          26 "├♦♦ˣ"       39 FS?C 13      52 END
```

(112 Bytes)

1-18 Synthetisches Aufheben und Wiedereinsetzen der Tastenzuweisungen ("SK" & "RK")

Um alle Tastenzuweisungen (System-Funktionen und Programme) aufzuheben, tasten Sie eine Registernummer ('n') ein, gefolgt von 'XEQ "SK"'; die Tastenzuweisungen werden dann in R_n und R_{n+1} abgelegt. Um sie wieder zur Verfügung zu haben, tasten Sie abermals 'n', jetzt gefolgt von 'XEQ "RK"', ein. Sie benötigen mindestens n + 2 Datenregister. Die vor dem Eintasten von 'n' in X, Y und Z liegenden Werte bleiben erhalten. - Zeile 24 lautet dezimal 243, 127, 15, 255. - Quelle: Keith Jarett (4360) & Roger Hill (4940) (PPC ROM).

```
01♦LBL "SK"     09 X<> e        17♦LBL "RK"     25 X<> \
02 SIGN         10♦LBL 14       18 SIGN         26 STO '
03 CLX          11 "*"          19 ARCL IND L   27 X<> [
04 X<> '        12 X<> [        20 "├♦"         28 STO e
05 XEQ 14       13 STO \        21 ISG L        29 RDN
06 ISG L        14 ASTO IND L   22 ""           30 CLA
07 ""           15 RDN          23 ARCL IND L   31 END
08 ,            16 RTN          24 "├‡"
```

(64 Bytes)

1-19 Beschriftung von Tastaturschablonen

Sog. Einreibe-Buchstaben sind ein empfehlenswerter Ersatz für die HP-eigenen selbstklebenden Etiketten zur Beschriftung der Tastaturschablonen; ein farbloser matter Sprühlack schützt die aufgebrachten Buchstaben vor Beschädigung.

1-20 Eintasten von Exponenten

Es ist überflüssig, im Anschluß an 'EEX' zwei Ziffern einzugeben, wenn der einzutastende Exponent nur eine Ziffer hat. - Quelle: Bill Kolb (265).

1-21 Synthetischer Blick auf die Tastenzuweisungen ("VK")

Diese Routine gibt Ihnen bekannt, welche Tasten mit System-Funktionen oder Programmen belegt sind. Fall I (kein Drucker angeschlossen): Die Tastenkodes der belegten Tasten werden in aufsteigender Reihenfolge (11 bis 84) unter Verwendung von 'PSE' angezeigt. Fall II (Drucker angeschlossen und eingeschaltet): Es wird 'PRKEYS' ausgeführt. Fall III (Drucker angeschlossen aber ausgeschaltet): Es wird die Meldung "PRINTER OFF" angezeigt. Darauf kann man folgendermaßen antworten: (1) Drucker anschalten, 'XEQ "VK"'. (2) Drucker entfernen, 'XEQ "VK"'. (3a) PRGM-Modus an, 3 mal 'SST', PRGM-Modus aus, 'R/S'. Daraufhin verhält sich das Programm wie im Fall I. (3b) PRGM-Modus an, 4 mal 'SST', PRGM-Modus aus, 'R/S'. Jetzt hält die Routine bei der Anzeige der einzelnen Tastenkodes an. 'R/S' führt zum nächsten Kode *). - Die Zeilen 11 und 16 lauten dezimal 243, 127, 16, 240 bzw. 243, 127, 32, 240. - Quelle: Roger Hill (4940) & Tom Cadwallader (3502) (PPC ROM).

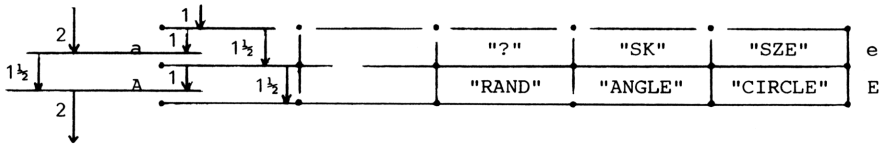
```
01♦LBL "VK"     25♦LBL 02       49 65           73 ABS          97♦LBL 06
02 SF 21        26 FC? IND \    50 RCL \        74 X<> [        98 STO d
03 FS? 55       27 FC? 50       51 INT          75 RDN          99 X<> _
04 PRKEYS       28 GTO 05       52 +            76 X<> \        100 AVIEW
05 FS? 55       29 X<> d        53 OCT          77 RDN          101 TONE 0
06 RTN          30 FC? IND \    54 1            78 " "          102 R↑
07 CF 21        31 FC? 50       55 ST+ Y        79 FC? 42       103 STO
08 8            32 GTO 05       56 %            80 "    -"      104 R↑
09 RCL '        33 X<> d        57 +            81 FS? 50       105 STO [
10 XEQ 07       34♦LBL 03       58 10           82 X<> d        106 R↑
11 "├9"         35 ISG \        59 MOD          83 X<>Y         107 R↑
12 X<> [        36 GTO 02       60 LASTX        84 FC? 50       108 X<> d
13 X<> d        37 DSE [        61 *            85 GTO 04       109 X<>Y
14 RCL e        38 GTO 01       62 INT          86 X<> d        110 FS? 42
15 XEQ 07       39 X<>Y         63 STO a        87 X<> _        111 X<> d
16 "├ "         40♦LBL 04       64 43           88 CLX          112 GTO 03
17 X<> Z        41 STO d        65 -            89 RCL d        113♦LBL 07
18 X<> [        42 CLST         66 ABS          90 FIX 0        114 CLA
19♦LBL 01       43 CLA          67 1            91 CF 29        115 X<> [
20 -27,00008    44 PSE          68 X<Y?         92 ARCL a       116 "├*****"
21 RCL [        45 CLD          69 ST+ a        93 ISG L        117 X<> \
22 -            46 RTN          70 FS? 42       94 GTO 06       118 X<> [
23 STO \        47♦LBL 05       71 FC? IND \    95 "├ -"        119 END
24 RDN          48 X<> d        72 CHS          96 ARCL a
```

(223 Bytes)

*) Hinweis des Übers.: (3a) und (3b) gelten nicht für den HP-IL-Drucker.

1-22 Zuweisungen auf die oberen Tasten dokumentieren

Häufig sind nur die obersten beiden Tastenreihen mit Zuweisungen belegt (oft mit den lokalen Marken A-J und a-e). Will man eine saubere Programm-Dokumentation erstellen, kann man die zugehörigen Zuweisungen folgendermaßen abbilden. Man setzt Punkte im Abstand von 11 Stellen (10 Stellen zwischen je 2 Punkten). Die Punkte können später mit Bleistift und Lineal verbunden werden:



Die Marken lassen sich jetzt leicht durch Abzählen ihrer Stellenzahl in den Kästchen zentrieren. Für Zuweisungen auf andere Tasten des Feldes umrandet man am besten eine Schablone und setzt die zugewiesenen Funktionen oder Marken auf bzw. über die abgezeichneten Tasten. - Quelle: John Dearing (2791).

1-23 GAUs

GAUs sind *g*rößte *a*nzunehmende *U*nfälle. Dabei friert die Anzeige ein, und einige oder alle Tasten werden funktionsunfähig. Solche Unfälle können eintreten, wenn der Rechner einen übermäßigen Stoß erhält oder synthetische Funktionen falsch angewendet werden. Gewöhnlich erwacht der Rechner wieder, ohne die Speicherinhalte zu verlieren, wenn die Batterien wenige Sekunden lang entfernt werden. In schlimmen Fällen kann es aber nötig sein, die Batterien eine ganze Nacht über oder gar 48 Stunden lang herauszunehmen. Für solche mißlichen Fälle schlägt Jim DeArras folgende Lösung vor: man stecke eine Karte in den Kartenleser und entferne die Batterien, sobald die Karte ihren halben Weg zurückgelegt hat. Man überlasse dann den Rechner etwa 2 Minuten lang sich selbst und setze danach die Batterien wieder ein. Ein "MEMORY LOST" kann die Folge sein. - Quelle: Bill Kolb (265).

1-24 Rangfolge der Funktionen auf den oberen Tastenreihen

Befindet sich der HP-41 im USER-Modus und bedient man eine Taste aus den oberen beiden Reihen (oder nach 'SHIFT' aus der obersten Reihe), so verfährt der Rechner wie folgt:

1. Trägt die Taste eine Zuweisung, wird die entsprechende Funktion ausgeführt.
2. Andernfalls findet (ausschließlich in dem Programm, auf das der Rechner positioniert ist) eine Suche nach der entsprechenden lokalen Marke (A-J, a-e) statt, und dort beginnt der Programmlauf.

3. Trifft beides nicht zu, wird die Funktion des NORMAL-Modus ausgeführt.

Aus diesem Grunde kann eine beträchtliche Zeit verstreichen, bevor eine auf den oberen Tasten liegende Funktion des NORMAL-Modus im USER-Modus ausgeführt wird. Diese Zeit läßt sich verkürzen, indem die Funktion entweder ihrer eigenen Taste zugewiesen wird oder ein 'GTO..' oder der NORMAL-Modus 'beste Suchverhältnisse' schaffen.

1-25 SIGN

Die Funktion 'SIGN' ist mangelhaft. Sie sollte für das Argument 0 den Wert 0 statt 1 liefern. Um das zu erreichen, schlägt Leland Van Allen vor, 'STO L, X ≠ 0?, SIGN' statt 'SIGN' zu verwenden. - Quelle: Bill Kolb (265).

1-26 Synthetisch Tastenzuweisungen löschen ("KC")

'XEQ "KC"' löscht die Register ⊦ und e und die Tastenzuweisungsregister (sowohl für System-Funktionen wie für Benutzer-Marken). Eine Unterbrechung der Ausführung verursacht "MEMORY LOST"! - Zeile 4 lautet dezimal 246, 127, 1, 105, 11, 240, 0. - Quelle: Paul Lind (6157). - Vgl. auch 21-8.

```
01♦LBL "KC"     10 ENTER↑       19 192          28 DSE Z
02 RCL c        11  E           20 -            29 GTO 02
03 STO [        12♦LBL 01       21 ,            30 STO e
04 "⊦×ìλ♦"      13 FS? IND Z    22 R↑           31 STO '
05 RCL [        14 ST+ Y        23 X<> d        32 X<>Y
06 X<> d        15 ST+ X        24 X<> c        33 STO c
07 15,006       16 DSE Z        25 X<>Y         34 CLST
08 ENTER↑       17 GTO 01       26♦LBL 02       35 END
09 ,            18 RDN          27 STO IND Z
```

(72 Bytes)

1-27 Synthetisches Ermitteln der Anzahl eingesparter Bytes bei Verwendung eines Unterprogramms ("BS")

Diese Routine berechnet die Anzahl der Bytes, die man einspart, wenn man für mehrfach auftretende Befehlsfolgen ein Unterprogramm einsetzt (vgl. auch 1-10). Vor Ausführung ist der Stapel folgendermaßen zu laden:

Z: Anzahl R der mehrfach auftretenden Bytes, jedoch ohne die für das beabsichtigte Unterprogramm notwendigen Befehle 'LBL' und 'RTN'. Die Zahl R muß negativ sein, wenn die Aufrufe indirekt erfolgen.

Y: Anzahl C der Aufrufe des Unterprogramms. Die Zahl C muß negativ sein, wenn dem Unterprogramm eine Kurzform-Marke ('LBL 00'-'LBL 14') vorausgeht.

X: Anzahl A der Alpha-Zeichen der Marke, falls das Unterprogramm unter einer globalen Marke aufgerufen wird. Ist die Marke lokal (01-99, A-J, a-e), muß eine 0 ins X-Register.

Ausführung: 'R, ENTER, C, ENTER, A, XEQ "BS"'.

Es wird die Anzahl der eingesparten Bytes ausgegeben. Ist das Ergebnis negativ, verbraucht das beabsichtigte Unterprogramm mehr Bytes als die Wiederholung der Byte-Folge im Hauptprogramm. Bei indirekten Aufrufen berücksichtigt "BS" die für das Indexregister erforderlichen 7 Bytes ebenfalls; steht ein solches Register ohnehin zur Verfügung, muß noch 7 zur Ausgabe addiert werden. - Quelle: Charles Close (3878) (PPC CJ, V8N1P14).

Byte-Verbrauch:

XEQ numerisch: 3
XEQ indirekt: 2
XEQ α: 2 + 1 je Zeichen
LBL 00-14: 1
LBL 15-99: 2
LBL A-J, a-e: 2
LBL α: 4 + 1 je Zeichen
RTN: 1

```
01♦LBL "BS"     17 STO ]     33 RCL [     49 FS? 03
02 0            18 *         34 *         50 3
03 X<> c        19 FC? 01    35 FC? 01    51 +
04 RDN          20 SF 02     36 +         52♦LBL 01
05 %≠0?         21 FS? 03    37 FS? 01    53 FS? 01
06 SF 03        22 CF 02     38 RDN       54 7
07 STO [        23 X<>Y      39 -         55 FS? 01
08 RDN          24 X<> \     40 RCL [     56 +
09 X<0?         25 2         41 -         57 -
10 SF 04        26 FS? 02    42 RCL ]     58 RCL \
11 ABS          27 1         43 -         59 STO d
12 STO \        28 FS? 02    44 2         60 RDN
13 X<>Y         29 +         45 FS? 04    61 CLA
14 X<0?         30 X<>Y      46 GTO 01    62 END
15 SF 01        31 *         47 FC? 03
16 ABS          32 LASTX     48 1
```

(96 Bytes)

Kapitel 2

TIPS ZUM PROGRAMMIEREN

2-1 Sprünge/Löschen bis zum 'END' von beliebiger Zeile aus

(a) Auf das 'END' gelangt man am schnellsten mit 'GTO.999'; falls mehr als 1000 Zeilen übersprungen werden müssen, mit 'GTO .EEX 999'. (b) Löschen von der angezeigten Zeile (einschließlich) an bis zum 'END' (ausschließlich) erfolgt mit 'DEL 999' bzw. 'DEL EEX 999'. Der freigemachte Platz steht anderen Programmen allerdings erst nach einem 'PACK' zur Verfügung. - Quelle: Bill Kolb (265).

2-2 Programmänderungen

Wenn Sie sich bei Programmänderungen an den Zeilen-Nummern orientieren wollen, ist es sinnvoll, mit der *letzten* Zeile oder Zeilengruppe zu beginnen und sich von unten nach oben durchzuarbeiten, weil auf diese Weise die Befehle ihre Zeilen-Nummer bis zur Änderung bewahren, wodurch die Orientierung erhalten bleibt. Eine falsche Einzelzeile sollte man erst löschen, dann neu einfügen. - Quelle: Richard Nelson (1) (PPC J, V5N2P9).

2-3 'BST' beim Edieren

Es kann sich als bequem erweisen, beim Edieren 'BST' auf eine Taste zu legen, z.B. auf 'TAN', so daß das Umschalten entfällt. - Quelle: Richard Nelson (1) (PPC J, V6N5P32).

2-4 Wirkung einiger Funktionen auf die Register X und Y

Wenn Sie die untenstehenden Gleichungen berücksichtigen, können Sie die darin vorkommenden Funktionen oft als nützliche Abkürzungen in Programmen einsetzen. Um etwa die Quadratwurzel aus der Summe zweier in X und Y stehenden Werte zu finden, reicht es, 'R-P' aufzurufen. In den Gleichungen bezeichnet der Index 1 den Inhalt des jeweiligen Registers vor der Ausführung, der Index 2 den Registerinhalt nach der Ausführung. L = LASTX-Register. - Quelle: HP-41 Bedienungshandbuch, © Copyright (Januar 1979) Hewlett-Packard Company. Abgedruckt mit Erlaubnis.

'P-R'

Y_1 verloren

$Y_2 = X_1 \sin Y_1$

$X_2 = X_1 \cos Y_1$

$L_2 = X_1$

'R-P'

Y_1 verloren

$Y_2 = \arctan (Y_1/X_1)$

$X_2 = \sqrt{X_1^2 + Y_1^2}$

$L_2 = X_1$

'D-R'

$Y_2 = Y_1$

$X_2 = \frac{\pi X_1}{180}$

$L_2 = X_1$

'R-D'	'%'	'%CH'	'SIGN'
$Y_2 = Y_1$	$Y_2 = Y_1$	$Y_2 = Y_1$	$Y_2 = Y_1$
$X_2 = \frac{180\ X_1}{\pi}$	$X_2 = \frac{X_1 \cdot Y_1}{100}$	$X_2 = \frac{100(X_1 - Y_1)}{Y_1}$	$X_2 = -1, 0$ oder 1 *)
$L_2 = X_1$	$L_2 = X_1$	$L_2 = X_1$	$L_2 = X_1$

*) 'SIGN' liefert -1, wenn X_1 negativ ist, 0, wenn X Alpha-Zeichen enthält, und 1, wenn X_1 positiv oder 0 ist. - Vgl. auch 1-25.

2-5 Wiederholtes Ausführen des gegenwärtigen Programms

Der umständlichste Weg ist der, ausführlich 'XEQ, ALPHA, Programm-Name, ALPHA' zu tasten. Das Belegen einer Taste mit dem Programm für den Aufruf im USER-Modus ist bequemer. Wenn das Programm aber bis zur letzten Zeile, dem 'END', läuft, reicht es, 'R/S' zu drücken. Wird der Programmlauf mittendrin durch ein 'RTN' oder 'STOP' abgeschlossen, kann mit 'RTN', 'R/S' neu gestartet werden. - Vgl. auch 2-21.

2-6 Nicht-blinkende Pausen variabler Länge ("VP")

Es handelt sich nicht um echte 'PSE'n, denn das Programm läuft weiter. Zeile 02 nimmt die Zahl 'n' auf, welche die Länge der Pause bestimmt (n = 10 gibt etwa 1 Sekunde Pause; n = 100 etwa 10 Sekunden). Der ursprüngliche Inhalt von Register T wird durch 0 ersetzt. Man kann die Routine als Unterprogramm aufrufen oder die Schritte 02-08 in ein Programm einfügen. Größere Flexibilität erreicht man, wenn man Schritt 02 in 'RCL 00' ändert und 'n' vor der Ausführung von "VP" in R_{00} ablegt. - Quelle: Tom Cadwallader (3502) (PPC J, V6N6P21).

```
01♦LBL "VP"          04 VIEW X        07 GTO 14
02 'n' oder RCL 00   05♦LBL 14        08 CLD
03 RDN               06 DSE T         09 END
```

Wenn in einem Programm einer der Befehle 'AVIEW' oder 'VIEW' erscheint, verschwindet die fliegende Graugans, doch bleibt der PRGM-Indikator an: die Graugans, die normalerweise die Programmausführung anzeigt, wird durch den Inhalt des durch 'AVIEW' oder 'VIEW' angesprochenen Registers ersetzt. Sobald das Programm diese Ersatzanzeige durch 'CLD' löscht, kehrt die Graugans zurück; dies tut sie auch nach einer Unterbrechung des Programms. - Quelle: HP-41C Bedienungshandbuch, © (Jan. 1979) H-P. - Vgl. auch 4-2.

2-7 Funktion oder inverse Funktion in Abhängigkeit vom Test ausführen

Sei 'X(?)Y' ein beliebiger Test wie z.B. 'X = Y?'. (1) 'X(?)Y, CHS, +' führt zur *Subtraktion*, wenn die Bedingung erfüllt ist, dagegen zur *Addition*, wenn die Bedingung nicht erfüllt ist. (2) 'X(?)Y, 1/x, *' führt zur *Division* bei 'ja', zur *Multi-*

plikation bei 'nein'. (3) 'X(?)Y, 1/X, Y^X' zieht bei 'ja' die *x-te Wurzel* und erhebt bei 'nein' zur *x-ten Potenz*. - Quelle: Bill Kolb (265). - Vgl. auch 2-23.

2-8 *Zwei* Schritte bei erfüllter Bedingung ausführen

Wenn man bei nicht erfüllter Bedingung zwei Schritte überspringen will, muß man die Bedingung vor jedem der beiden Schritte abfragen. Z.B. setzt 'X = Y?, PSE, X = Y?, GTO 08' eine Pause *und* verzweigt zur Marke 08 genau dann, wenn x = y ist. Im Falle $x \neq y$ werden 'PSE' *und* 'GTO 08' übersprungen. Allerdings kann es Fehler geben, wenn der Schritt zwischen den Bedingungen den Stapel verändert, wie es beispielsweise durch '+' geschähe.

2-9 Das letzte 'RTN' oder 'END' ist entbehrlich

Ein Programm, welches an das Ende des Programmspeichers gesetzt wird, also ein Programm, das nach einem 'GTO..' eingetastet wird, muß nicht mit 'END' oder 'RTN' abgeschlossen werden. Das ständige '.END.' reicht aus. Es sei denn, es soll geschützt werden gegen das Überschreiben durch andere Programme, die über den Kartenleser oder den Lesestift in den Speicher gelangen. - Quelle: HP-41C Bedienungshandbuch, © (Jan. 1979) H-P. - Vgl. auch 2-30.

2-10 Aufruf einer lokalen Routine durch ein anderes Hauptprogramm

Ein Hauptprogramm "X" enthalte mehrere voneinander unabhängige Unterroutinen. Wenn diese mit numerischen Marken versehen werden und allen voran eine globale Marke (etwa "LA" für Leipziger Allerlei) gesetzt wird, lassen sie sich sämtlich durch Eingeben der numerischen Marke, gefolgt von 'XEQ "LA"', aufrufen. Das kann sowohl von Hand als auch *durch ein anderes Hauptprogramm* "Y" geschehen. Das 'RDN' befördert den Index ins Register T, so daß im allgemeinen keine Störung der zu verarbeitenden Daten zu erwarten ist.

'LBL "LA", RDN, GTO IND T, LBL 00, ... , RTN, LBL 01, ... , RTN, ... ,END'.

Quelle: William Cheeseman (4381) (PPC CJ, V7N5P9).

2-11 Indirektes 'XEQ' oder 'GTO' versagt bei lokalen Alpha-Marken (A-J, a-e)

Es gibt Programme, die Unterprogramme zum Druck einzelner Buchstaben enthalten. Dabei ist es sinnvoll, die jeweiligen Unterprogramme durch den zu druckenden Buchstaben selbst anzusteuern, also z.B. 'V' durch 'LBL "V"' drucken zu lassen. Man legt dazu "V" ins Register X und führt 'XEQ IND X' aus. Für die Marken 'LBL A' bis 'LBL J' schlägt dieses Verfahren jedoch fehl. Da man durch indirektes Adressieren aber sehr viel Speicherplatz spart, lohnt es, allen Unterprogrammen globale Marken voranzustellen, indem man den jeweiligen Buchstaben verdoppelt:

```
ASTO X      legt "A" ins X-Register,
ARCL X      verdoppelt "A" im Alpha-Register,
ASTO X      legt "AA" ins X-Register,
XEQ IND X   ruft die globale Marke "AA" auf.
```

Quelle: HP Key Notes, V4N1P11. - Bemerkung: Synthetisch lassen sich globale Marken 'LBL "A"' bis 'LBL "J"' herstellen, die sich (nach einem 'GTO..'!) auch indirekt ansteuern lassen.

2-12 Behandlung von übereinstimmenden Funktions- und Programmnamen

Eine Systemfunktion und ein Programm können identische Namen haben und dennoch *beide* über das Tastenfeld ausgeführt werden. Man braucht dazu nur die Funktion einer Taste zuzuweisen, *bevor* man ihren Namen als Programm-Marke verwendet. Das Programm kann dann einer anderen Taste zugewiesen oder auch mit 'XEQ' ausgeführt werden. Dasselbe gilt für mehrfach benutzte Programmnamen: man weist sie in der Reihenfolge, in der sie in den Speicher gelangen, verschiedenen Tasten zu. Der HP-41 bewahrt die Zuweisungen geordnet auf. - Quelle: HP Key Notes, V4N2P11. - Vgl. auch 27-5.

2-13 Programm-Segmentierung

Ein Programm kann in zwei Teilprogramme zerlegt werden, indem man an der 'Schnittstelle' ein 'END' einfügt: 'XEQ, ALPHA, E, N, D, ALPHA, ALPHA'. Man muß überdies sicherstellen, daß hinter der Schnittstelle eine globale Marke steht, damit sich das untere Teilprogramm auch unter einer solchen Marke auffinden läßt. Man kann auf diese Weise ein Programm segmentieren, die einzelnen Segmente auf Karten lesen und in neuer Reihenfolge wieder in den Rechner bringen. Löscht man anschließend die 'END's, hat man ein nach Wunsch umgeordnetes Programm, ohne einen einzigen Befehl neu geschrieben zu haben. - Quelle: John Dearing (2791) (PPC CJ, V8N1P14). - Bemerkung: Das Ganze läßt sich sinngemäß auch mit dem X-Memory statt mit dem Kartenleser durchführen.

2-14 Eine Schrittfolge innerhalb eines Programms mehrfach ausführen

Statt eines Unterprogramms kann man sich der folgenden Möglichkeiten bedienen. (1) Schleifenkontrolle:
'...,n, STO 00, LBL 01, Schrittfolge, DSE 00, GTO 01,...' arbeitet die Schrittfolge n-mal ab. (2) Flagbenutzung:
'...,SF 00, LBL 01, Schrittfolge, FS?C 00, GTO 01,...' arbeitet die Schrittfolge 2-mal ab. (3) 'XEQ' am Programmende:
'...,XEQ 01, XEQ 01, LBL 01, Schrittfolge, END' arbeitet die Schrittfolge 3-mal ab.
- Quelle: John Dearing (2791).

2-15 Simulation des Befehls 'NOP' (no operation)

'NOP' ist ein Befehl, der nichts oder jedenfalls nichts 'Schädliches' bewirkt. Der HP-41 hat kein echtes 'NOP'. Nichtsdestoweniger lassen sich viele Befehle so verwenden, daß sie ein 'NOP' simulieren. Setzt man beispielsweise - abhängig von der Situation - einen der Befehle 'DEG', 'RAD', 'GRAD', 'FIX', 'SCI', 'ENG', 'CF', 'SF' oder irgendein 'LBL' hinter 'ISG' oder 'DSE', werden diese 'bedingten Befehle' zu *unbedingten* Inkrement- bzw. Dekrement-Befehlen. Insbesondere entfällt die Beschränkung der Endwerte der Schleifen auf 999. 'LBL' ist unter den nicht-synthetischen 'NOP's der schnellste Befehl. Es lassen sich auch 'X<>Y' (mit) und 'STO X' (ohne Einwirkung auf den Stapel) als Ersatz-'NOP' verwenden. Der absolut schnellste, eleganteste und ein echtes 'NOP' perfekt simulierende Befehl ist allerdings das synthetisch erzeugte Programm-Byte F0 ('TEXT 0', dezimal 240), welches im Programmausdruck als "" erscheint.

2-16 Schleifenzähler für Iterationen

Um mitzuzählen, wie oft eine mit z.B. 'X = 0?' abgefragte Schleife durchlaufen wird, setzt man ein 'ISG nm' (R_{nm} ein Register, welches mit 0 initialisiert wird) in die Schleife und ein 'NOP' (2-15) unmittelbar dahinter. Nach Programmausführung liegt die gesuchte Zahl in R_{nm}. - Quelle: Richard Nelson (1) (65 Notes, V1N2P3).

2-17 Lokale Marken

Der Zugriff auf *globale Marken* innerhalb eines Programms mit 'GTO' oder 'XEQ' kann beschleunigt werden, indem man zusätzlich eine numerische Marke hinter die globale Marke einfügt (wenn man sie nicht gar ersetzen kann) und die jeweiligen 'GTO's und 'XEQ's auf die numerischen Marken springen läßt. Der Sprung auf numerische Marken ist nämlich (nach dem ersten Auffinden) wesentlich schneller als die stets von neuem beginnende Suche nach globalen Marken. Die Lage von *Kurzform-Marken* ('LBL 00-14') können sich 'GTO's nur innerhalb eines Abstandes von 112 Bytes merken. Für größere Abstände benutzt man 'LBL 15-99' (es sei denn, man erzeugt synthetisch 3-Byte-'GTO's zum abstandsunabhängigen Sprung auf die Kurzform-Marken). Weil sich *lokale Alpha-Marken* (A-J, a-e) weder durch 'GTO's noch 'XEQ's indirekt ansprechen lassen, kann man auch ihnen eine numerische Marke unmittelbar anschließen, die dann zum indirekten Ansprung verwendet wird. Durch 'XEQ' ausgelöste Verzweigungen unterliegen keiner Sprunglängenbegrenzung. - Vgl. auch 2-11.

2-18 Besser 'RDN, RCL' als 'CLX, RCL'

Dies vermeidet Stapel-Lift-Probleme, wenn die Programmausführung unmittelbar vor dem 'RCL'-Befehl von Hand unterbrochen wird. - Quelle: Keith Jarett (4360) (PPC CJ, V7N8P9).

2-19 Schutz gegen Stromausfall bei längerlaufenden Programmen

Die Befehlsfolge 'FS? 49, OFF' kann eingesetzt werden, um die Batterien vor völliger Entleerung zu bewahren. - Quelle: Richard Nelson (1).

2-20 'R↑' oder 'RDN'?

Wenn mit 'R↑' dasselbe erreicht werden kann wie mit 'RDN', ist es besser 'R↑' zu benutzen: 'RDN' dauert 17.4 ms; 'R↑' dauert 12.8 ms. Folglich ist es besser, 2 mal 'R↑' zu verwenden als 2 mal 'RDN', um den Inhalt von Z nach X zu bringen. Stapel-Umordnungen sollten daraufhin überprüft werden, ob 'RDN' durch 'R↑' ersetzt werden kann. - Quelle: Richard Nelson (1) (PPC CJ, V7N8P8). - Vgl. auch 7-11, 27-3.

2-21 Wiederholung eines Programmlaufs mit 'R/S'

Wenn die Programmausführung auf dem letzten Schritt anhält, reicht 'R/S' für den nächsten Lauf (vgl. 2-5). Um mit 'R/S' von einem internen 'RTN' oder 'STOP' aus zu starten, ist dahinter ein 'GTO' auf den Programmanfang einzufügen (am besten eine numerische Marke; vgl. 2-17).

2-22 Ein internes 'STOP' 'endgültig' machen

Wenn die Ausführung eines Programms mittendrin beendet wird und man sich gegen einen Start in den hinter 'STOP' (oder 'RTN') liegenden Programmteil durch ein versehentliches 'R/S' schützen will, kann man statt eines einfachen 'STOP' (oder 'RTN') die Folge 'LBL 14, STOP (oder RTN), GTO 14' verwenden.

2-23 Einen Wert nur bei falscher Bedingung addieren/subtrahieren

Ist 'X(?)Y' irgendeine einem Wert folgende Bedingung, leistet 'X(?)Y, CLX, +/-' das Gewünschte. Ähnlich arbeitet z.B. '5, FS? 00, CLX, +/-': 5 wird nur bei gelöschtem Flag 00 addiert/subtrahiert (vgl. auch 2-7).

2-24 Ausschalten im PRGM-Modus

Ältere Modelle können einen Fehler enthalten, der verhindert, daß die Sprunglängeninformationen in den Sprunganweisungen auf Null gesetzt werden, wenn ein Programm geändert wird. Dieser Fehler tritt dann auf, wenn man den HP-41 ausschaltet, *während er sich noch im PRGM-Modus befindet*. Um zu prüfen, ob Ihr Rechner diesen Fehler aufweist, befolgen Sie diese Anweisungen: 'LBL A, GTO 09, CHS, CHS, LBL 09, BEEP', PRGM aus, 'XEQ A', der 'BEEP' ertönt. PRGM an, 'STOP' hinter das zweite 'CHS' setzen; HP-41 ausschalten, *ohne den PRGM-Modus zu verlassen*. HP-41 anschalten, 'XEQ A'. Wenn der 'BEEP' jetzt ausbleibt, hat das 'GTO 09' mit altem Sprunglängenkode gearbeitet und folglich zum 'STOP' statt zum 'LBL 09' verzweigt. Das Fehlverhalten des Rechners

läßt sich einfach dadurch umgehen, daß man nach Änderungen den PRGM-Modus nur durch Druck auf die PRGM-Taste verläßt. - Quelle: Nach HP Key Notes, V4N1P3.

2-25 Programm-Meldungen

Länger laufende Programme oder Programmteile sollten eine Meldung enthalten. Legen Sie sie ins Alpha-Register (höchstens 12 Zeichen, z.B. "SORTIEREN"), gefolgt von 'AVIEW'; ans Ende des Programmteils setzen Sie 'CLD'. Sie sehen dann stets, was gerade abläuft.

2-26 Fünf Sekunden Pause

Wenn der Kartenleser eingesteckt ist, der Drucker jedoch nicht, läßt sich der Befehl '7PRTX' als lange Pause benutzen. - Quelle: PPC Melbourne Chapter.

2-27 Einfügen von Programmzeilen oberhalb von Schritt 01

Tasten Sie 'GTO.000', und geben Sie die beabsichtigten Zeilen ein. Sind Sie im RUN-Modus, reicht 'RTN' und der Übergang in den PRGM-Modus. - Quelle: Bill Kolb (265).

2-28 Alpha-Ketten für indirekte Adressierung

Eine Alpha-Kette kann ebensogut zur indirekten Adressierung benutzt werden wie eine Zahl. Diese Eigenschaft läßt sich gut für Adreßverzeichnisse oder in Wortspielen verwerten. - Quelle: Bill Kolb (265). - Bemerkung: Da man synthetisch auch 'LBL "A"' bis 'LBL "J"', also globale Marken mit dem Namen "A"-"J" erzeugen kann, läßt sich die Möglichkeit indirekter Adressierung einschränkungslos benutzen (vgl. auch die Bemerkung in 2-11).

2-29 Programme ohne Marken

Hat man ein Programm ohne globale Marke im Speicher, oder löscht man versehentlich seine globale Marke, kann man es mit 'CAT 1' wiederfinden: man startet den Katalog, stoppt ihn gleich auf der ersten Marke mit 'R/S' ab und arbeitet sich mit 'SST' durch, bis man auf zwei 'END's hintereinander stößt. Beim zweiten 'END' löscht man entweder das Programm ('XEQ, ALPHA, C, L, P, ALPHA, ALPHA, ALPHA'), oder man setzt eine Marke an den Programmanfang ('RTN', PRGM-Modus an, Marke eintasten). - Quelle: Bill Kolb (265).

2-30 Kein voreiliges 'END' setzen

Man setze kein 'END' an den Schluß eines Programms, bevor man ein anderes selbständiges Programm dahinter legen will. Das Setzen eines 'END' bewegt nämlich den Programmzeiger in einen neuen Speicherbereich, wodurch die Rückkehr in das ursprüngliche Programm unnötig erschwert wird. Um festzustellen, wieviele Register noch zur Verfü-

gung stehen, verwende man nicht 'GTO..', sondern 'GTO.000' oder 'GTO.mnl', wo 'mnl' größer als die Anzahl der Programmzeilen ist. - Quelle: Bill Wickes (3735). - Vgl. auch 2-9.

2-31 Alpha-Register ausdrucken ohne langatmige Anzeige

Will man den Inhalt des Alpha-Registers ausdrucken aber - insbesondere bei langen Alpha-Ketten - das zeitraubende Rollen der Buchstaben durch die Anzeige verhindern, benutze man statt 'AVIEW' die Befehlsfolge 'SF 25, PRA, CF 25'. Dann erfolgt der Ausdruck ohne Anzeige, und bei nichtangeschlossenem Drucker wird die Meldung "NONEXISTENT" vermieden. - Quelle: John Herzfeld (5428). Vgl. auch 3-12, 4-1, 6-14, 22-23.

2-32 Sprung auf Kurzform-Marken vermöge synthetischer 'GTO's mit vollem 'Sprunglängengedächtnis'

Nach gewöhnlicher Erkenntnis benutzt man die aufeinander bezogenen Kurzformen der 'LBL's und 'GTO's dann, wenn die Distanz zwischen ihnen 112 Bytes nicht übertrifft. Andernfalls werden die Langformen, 15-99, verwendet (vgl. 2-17). Paul Lind (6157) hat festgestellt, daß man synthetische 3-Byte-'GTO's für 00-14 erzeugen kann, so daß die Sprunglängenbegrenzung entfällt. Folglich spart man ein Byte ein, wenn man bei langer Distanz ein Kurzform-'LBL' mit einem Langform-'GTO' anspringt. Und die Ersparnis ist größer, sobald mehrfach auf das Kurzform-'LBL' verzweigt wird: mit 2-Byte-'GTO's innerhalb der 112-Byte-Zone und mit 3-Byte-'GTO's von außerhalb. Diese 3-Byte-'GTO's sind leicht synthetisch herzustellen (z.B. mit dem Programm "LB", Kapitel 25). Sie lauten dezimal 208, 0, n ($0 \leq n \leq 14$). Statt 0 an zweiter Stelle kann man auch irgendein Byte einsetzen. - Quelle: John Herzfeld (5428).

2-33 Synthetische Verlängerung und Verkürzung des Rücksprungstapels ("SR" & "LR")

Wenn "LR" ausgeführt wird, werden 6 Rückkehradressen in den Registern R_i und R_{i+1} abgelegt. Sobald Sie sich in der 5. Unterprogrammebene befinden und eine Verlängerung des Rückkehrstapels nötig ist, muß die Adresse des ersten Ablageregisters ins Register X gesetzt und "LR" aufgerufen werden, damit die nächsten 6 Rückkehradressen in die Zustandsregister a und b gelangen können. Fassen Sie *je 5* Rückkehradressen zu einer Gruppe zusammen: wenn Sie im Programm an eine Stelle gelangt sind, an der Sie eine Anzahl von Unterprogrammebenen verbraucht haben, die ein Vielfaches von 5 ist, und mindestens 2 weitere Ebenen benötigen, müssen Sie "LR" und "SR" einsetzen. Betrachten Sie dazu das untenstehende Musterbeispiel: verzweigten Sie von 'LBL 06'/ 'LBL 11' (Zeile 30/54) aus nicht in weitere Unterprogramme, erübrigten sich die Zeilen 23, 24, 26, 27/47, 48, 50, 51. Das Programm hält auf Zeile 05. Es hält ebenfalls auf Zeile 05, wenn Sie hinter 'LBL 06' ein 'RTN' setzen und die Zeilen 23, 24, 26, 27 löschen. Wenn Sie das 'RTN' aber stattdessen hinter 'LBL 07' (Zeile 34) setzen,

wird die Programmausführung auf Zeile 09 beim zweiten 'RTN' unterbrochen. Fügen Sie die Zeilen 23, 24, 26, 27 wieder ein, werden alle 'RTN's ausgeführt. Das Musterbeispiel bringt mit 'AVIEW' die Meldung "LBL 16" in die Anzeige und macht damit sichtbar, daß 'LBL 16' erreicht ist. Anschließend erscheint die '1', womit die Rückkehr aus allen Unterprogrammen angezeigt wird. Setzen Sie noch hinter jeden Zahleneintrag ein '+', ergibt sich zum Schluß erwartungsgemäß '136' (= 1 + 2 + ... + 16). - Quelle: PPC ROM. Harry Bertuccelli (3994) & Paul Lind (6157).

Musterbeispiel:

```
01♦LBL "X"      21 RTN          41 RTN          61 RTN
02 CLX          22♦LBL 05       42♦LBL 09       62♦LBL 13
03 XEQ 01       23 0            43 XEQ 10       63 XEQ 14
04 1            24 XEQ "LR"     44 10           64 14
05 RTN          25 XEQ 06       45 RTN          65 RTN
06♦LBL 01       26 0            46♦LBL 10       66♦LBL 14
07 XEQ 02       27 XEQ "SR"     47 2            67 XEQ 15
08 2            28 6            48 XEQ "LR"     68 15
09 RTN          29 RTN          49 XEQ 11       69 RTN
10♦LBL 02       30♦LBL 06       50 2            70♦LBL 15
11 XEQ 03       31 XEQ 07       51 XEQ "SR"     71 XEQ 16
12 3            32 7            52 11           72 16
13 RTN          33 RTN          53 RTN          73 RTN
14♦LBL 03       34♦LBL 07       54♦LBL 11       74♦LBL 16
15 XEQ 04       35 XEQ 08       55 XEQ 12       75 "LBL 16"
16 4            36 8            56 12           76 AVIEW
17 RTN          37 RTN          57 RTN          77 PSE
18♦LBL 04       38♦LBL 08       58♦LBL 12       78 CLD
19 XEQ 05       39 XEQ 09       59 XEQ 13       79 END
20 5            40 9            60 13
```

"SR" & "LR":

```
01♦LBL "SR"     12 ISG L        23 CLA          34 STO ]
02 SIGN         13 ""           24 STO b        35 ASTO IND L
03 SF 10        14 X<> IND L    25♦LBL "LR"     36 ISG L
04 RDN          15 STO \        26 SIGN         37 ""
05 RCL b        16 "⊢**"        27 RDN          38 "⊢*****"
06 STO [        17 X<> IND L    28 "+"          39 STO ]
07 RDN          18 STO ]        29 RCL a        40 ASTO IND L
08 FC?C 10      19 "⊢**"        30 STO \        41 RDN
09 RTN          20 X<> ]        31 RDN          42 CLA
10 "⊢***"       21 STO a        32 RCL b        43 END
11 RCL IND L    22 X<> \        33 X<> [
```

(95 Bytes)

2-34 Antworten auf Aufforderungen zur Eingabe 1-, 2- und 3-ziffriger Zahlen

Bekanntlich können die oberen beiden Tastenreihen zur Eingabe 2-ziffriger Zahlen benutzt werden, wenn der HP-41 dazu mit entsprechenden Unterstrichen (z.B. 'RCL__') auffordert. Diese Tasteneigenschaft läßt sich auch bei der Eingabe 1- oder 3-ziffriger Parameter ausnutzen. Beispiele: 'XEQ "LIST", LN' entspricht 'LIST 005', 'GTO., Σ+' entspricht 'GTO.001', 'XEQ "TONE", SIN' entspricht 'TONE 8'. Bemerkung: für 1-ziffrige Eingaben hat die Taste 'TAN' den Parameterwert '0', für 2- und 3-ziffrige Eingaben den Parameterwert '10'. - Quelle: Rolf R. Schmitt (4643).

Kapitel 3

STARTBEDINGUNGEN UND EINGABEAUFFORDERUNGEN

3-1 Unterprogramm für 'SIZE' und Programmüberschriften ("TITLE", "SIZE?" & "T+S")

(a) Programmüberschriften ("TITLE"):
In vielen Anwendungen sieht der Ausdruck freundlicher aus, wenn er eine Überschrift erhält. Die folgende Routine druckt die Überschrift in doppelter Buchstabenbreite und mit Vorschub. Man legt die Überschrift ins Alpha-Register und ruft auf:

'LBL "TITLE", ADV, SF 12, FS? 55, PRA, CF 12, ADV, RTN'. (20 Bytes)

(b) Prüfroutine für 'SIZE' ("SIZE?"):
Es ist sinnvoll, sich eine auf falscher Datenregister-Anzahl beruhende "NONEXISTENT"-Meldung rechtzeitig geben zu lassen, bevor man mit dem Eintasten von Werten beginnt. Die folgende Routine, am Programmanfang aufgerufen, löst das Problem:

'LBL "SIZE?", "SIZE>=", ARCL X, 1, -, SF 25, RCL IND X, RTN'. (25 Bytes)

Flag 25 ist das Fehler-Ignorierflag. Die erforderliche Datenregister-Anzahl muß vor dem Aufruf der Routine ins Register X gelegt werden. 'FC?C 25, PROMPT' muß dem Aufruf folgen. Beispiel: Für ein Programm, das 54 Datenregister beansprucht, lautet die eingangs stehende Befehlsfolge: '54, XEQ "SIZE?", FC?C 25, PROMPT'.

(c) Überschrift und Prüfroutine kombiniert ("T + S"): Weil häufig beide Routinen benötigt werden, kann man sie vereinigen:

```
01♦LBL "T+S"     05 PRA          09 ARCL X       13 RCL IND X
02 ADV           06 CF 12        10 1            14 RTN
03 SF 12         07 ADV          11 -
04 FS? 55        08 "SIZE>="     12 SF 25
```

(33 Bytes)

Beispiel: Für die Überschrift "F = MA" und eine Datenregister-Anzahl von 6 lautet die Befehlsfolge: '6, "F = MA", XEQ "T + S", FC?C 25, PROMPT'. - Quelle: Corvallis Division Column, PPC J, V6N7P19.

3-2 'SIZE' neu festsetzen ("RS")

Die Routine prüft, ob die gegenwärtige Datenregister-Anzahl ausreicht; falls nicht, fordert sie dazu auf, die Mindestanzahl festzusetzen. Diese vom Programm benötigte Mindestanzahl muß vor dem Aufruf von "RS" ins X-Register gesetzt werden.

```
01♦LBL "RS"     04 STO X        07 FS?C 25      10 ARCL Y
02 ENTER↑       05 SF 25        08 RTN          11 PROMPT
03 DSE X        06 VIEW IND X   09 "RESIZE: "    12 END
```

(33 Bytes)

3-3 'SIZE' prüfen

Die Anzahl der benötigten Datenregister wird vor Ausführung ins X-Register gelegt.

```
01♦LBL "?S"     04 DSE X        07 RCL IND X    10 END
02 "RESIZE: "   05 STO X        08 FC?C 25
03 ARCL X       06 SF 25        09 PROMPT
```

(31 Bytes)

- Quelle: John Dearing (2791).

3-4 'SIZE' synthetisch prüfen ("?S")

Die Routine zeigt die festzusetzende Anzahl ohne störende Zwischenzeichen (Punkt, Komma) an, stellt aber den ursprünglichen Anzeige-Modus wieder her.

```
01♦LBL "?S"     05 CF 29        09 ""           13 PROMPT
02 "RESIZE: "   06 ARCL Y       10 SF 25        14 END
03 RCL d        07 STO d        11 RCL IND Y
04 FIX 0        08 DSE Y        12 FC?C 25
```

(38 Bytes)

- Quelle: John Dearing (2791).

3-5 Synthetische 'SIZE'-Prüfung und -Festsetzung ("VS")

Um festzustellen, ob die vorhandene Datenregister-Anzahl groß genug ist, muß diese Anzahl eingegeben und "VS" aufgerufen werden. Die Routine fordert nur, wenn es nötig wird, dazu auf, 'SIZE' neu festzusetzen; nach Neufestsetzung geht es mit 'R/S' weiter. Die - vor Eingabe der erforderlichen Anzahl - in X, Y und Z befindlichen Werte bleiben erhalten; der Inhalt von T geht verloren. Um die folgende Fassung so zu ändern, daß zur etwaig notwendigen Neufestsetzung im Hauptprogramm aufgefordert wird, müssen Zeile 09 in 'FS? 25' geändert und Zeile 20 gelöscht werden. Die Verwendung der Routine in einem Programm geschieht dann mit der Befehlsfolge: '..., XEQ "VS", FC?C 25, PROMPT,...'.

```
01♦LBL "VS"     07 RCL IND T    13 R↑           19 RDN
02 SF 25        08 RDN          14 RCL d        20 PROMPT
03 INT          09 FS?C 25      15 FIX 0        21 END
04 RDN          10 RTN          16 CF 29
05 DSE T        11 "RESIZE = "  17 ARCL L
06 ""           12 TONE 3       18 STO d
```

(47 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM).

3-6 Eingabe-Routine ("IN")

Die Routine dient dazu, die Eingabe von Werten unter Verwendung von Texten anzufordern. Sie benutzt R_{00} als Zählregister. Wenn sie aufgerufen wird, erwartet sie einen Text von höchstens 5 Zeichen im Alpha-Register (höchstens 5, weil Zeile 06 ein Gleichheitszeichen anhängt und das Y-Register - angesprochen in Zeile 07 - alle Zeichen aufnehmen soll). Die Befehlsfolge, in die "IN" mehrfach eingebettet wird, lautet wie folgt, wenn R_{06}, R_{07} und R_{08} Eingabewerte aufnehmen sollen: '...,5, STO 00, "LG.", XEQ "IN", "HOEHE", XEQ "IN", "BR.", XEQ "IN",...'. Beachten Sie, daß die eingegebene Zahl (im Beispiel '5') um 1 kleiner ist als die Nummer des ersten Datenregisters, das einen Wert aufnehmen soll (im Beispiel R_{06}). Wenn Sie mit R_{01} beginnen wollen, starten Sie mit '0'. "IN" ist sehr bequem für andere Benutzer Ihrer Programme. Wenn das Programm einmal durchlaufen wurde, kann es erneut abgearbeitet werden, indem Sie lediglich die Werte eintasten, die Sie zu ändern wünschen: 'R/S' ohne Eingabe läßt die Werte aus der vorangegangenen Runde bestehen. Dies erlaubt eine schnelle Untersuchung eines Problems mit gezielt abgeänderten Ausgangswerten. Flag 22 wird bei Rückkehr aus "IN" gesetzt, wenn der Benutzer eine Eingabe tätigt; ohne Eingabe bleibt es gelöscht. Diese Tatsache können Sie u.U. noch anderweitig verwenden. Natürlich arbeitet die folgende (erste) Version von "IN" nicht richtig, wenn der Drukker angeschlossen aber ausgeschaltet ist. - Quelle: Corvallis Division Column, PPC J, V6N7P18.

```
01♦LBL "IN"      07 ASTO Y      13 ARCL Y      19 ARCL X
02 CF 22         08 "⊢?"        14 STOP        20 PRA
03 1             09 CF 21       15 STO IND 00  21 END
04 ST+ 00        10 AVIEW       16 FS? 22
05 RCL IND 00    11 SF 21       17 FC? 55
06 "⊢="          12 CLA         18 RTN
```

(44 Bytes)

In der nächsten Version sind die Zeilen 07-14 durch ein 'PROMPT' ersetzt. Das Fragezeichen in der Eingabe-Aufforderung fällt fort, aber die Begrenzung der Textteile auf 5 Zeichen entfällt ebenfalls. Zugleich ist die Verwendung bei angeschlossenem aber ausgeschaltetem Drucker gesichert, wenn vorab Flag 21 gelöscht wird.

```
01♦LBL "IN"      05 RCL IND 00    09 FS? 22      13 PRA
02 CF 22         06 "⊢="          10 FC? 55      14 END
03 1             07 PROMPT        11 RTN
04 ST+ 00        08 STO IND 00    12 ARCL X
```

(31 Bytes)

Die letzte Version druckt sowohl die unveränderten Werte (deren Eingabe mit 'R/S' übersprungen wurde), als auch die neuen. Wenn kein Drucker angeschlossen ist, werden die mit Text versehenen Werte (alte und neue) mit 'PSE' angezeigt. Auch hier ist 'CF 21' auszuführen, falls der Drucker eingesteckt aber ausgeschaltet ist. - Quelle: John Dearing (2791).

```
01♦LBL "IN"     05 "⊢="         09 FS? 55       13 AVIEW
02 1            06 PROMPT       10 PRA          14 PSE
03 ST+ 00       07 STO IND 00   11 FS? 55       15 END
04 RCL IND 00   08 ARCL X       12 RTN
```

(31 Bytes)

3-7 Ausgabe-Routine ("OUT")

Die Routine versieht Ausgabewerte mit Text und druckt sie oder zeigt sie an. Der Ausgabewert muß ins X-Register gelegt werden und der zugehörige Text ins Alpha-Register, bevor "OUT" aufgerufen wird. Es wird Flag 21 gesetzt. Beispiel für die Benutzung: '..., RCL 06, "LAENGE", XEQ "OUT", RCL 07, "HOEHE", XEQ "OUT", RCL 08, "BREITE", XEQ "OUT",...'.

```
'LBL "OUT", SF 21, "⊢=", ARCL X, AVIEW, RTN'.
```

(16 Bytes)

- Quelle: Corvallis Division Column, PPC J, V6N7P18.

3-8 Routine zur Abfrage 'ja' oder 'nein' ("YN")

Häufig ist es angebracht, den Benutzer zu befragen und ihn eine von zwei möglichen Antworten geben zu lassen. Fast immer ist es dabei möglich, die Frage so zu stellen, daß 'ja' oder 'nein' als Antwort paßt. Die Antwort läßt sich in Gestalt gesetzter oder gelöschter Flags notieren. Die Routine "YN" hilft bei einer solchen benutzerfreundlichen Programmgestaltung. (1) Sie fügt dem im Alpha-Register abgelegten Text (er darf aus höchstens 6 Zeichen bestehen) die Zeichen "? J/N" an. (2) Sie druckt das Ergebnis der Befragung aus, wenn ein Drucker angeschlossen (und eingeschaltet) ist. (3) Ohne Drucker wird das Ergebnis mit 'PSE' angezeigt (vorab 'CF 21', falls doch eingesteckt aber ausgeschaltet). (4) Das durch den Inhalt von X bestimmte Flag wird gesetzt bzw. gelöscht (Flag 3 wenn '3' in X liegt). (5) Der vorhandene Flagzustand wird aufrechterhalten, wenn der Benutzer die Frage nicht beantwortet und mit 'R/S' weiterspringt. (6) Der Alpha-Modus wird selbständig ein- und ausgeschaltet. Beispiel: ein Programm möge den Benutzer fragen, ob die zu verwendenden Einheiten metrisch (SI) oder englisch sein sollen: '..., 0, "METRIC", XEQ "Y/N",...'; bei 'metrisch' bedient man die 'J'-Taste, und Flag 00 wird gesetzt; bei 'N' wird Flag 00 gelöscht. Dieses Flag kann später im Programm nach Bedarf abgefragt werden. Bemerkung: Natürlich kann die Routine leicht anderen Bedürfnissen angepaßt werden, etwa 'Links oder Rechts', 'Oben oder Unten', 1 oder 2 usw. - Quelle: Corvallis Division Column, PPC J, V6P7N19.

```
01♦LBL "YN"     09 RTN          17 CLA          25 AVIEW
02 CF 23        10 CF IND X     18 ARCL L       26 FC? 21
03 ASTO L       11 RDN          19 "⊢: "        27 PSE
04 "⊢? J/N"     12 ASTO X       20 FS? IND T    28 FC? 21
05 AON          13 "J"          21 "⊢JA"        29 RTN
06 PROMPT       14 ASTO Y       22 FC? IND T    30 PRA
07 AOFF         15 X=Y?         23 "⊢NEIN"      31 END
08 FC? 23       16 SF IND T     24 FC? 21
```

(70 Bytes)

3-9 Kurze ja/nein Entscheidungen

Eine besonders kurze Befehlsfolge für 'ja/nein' ist '...,"(Frage)?", CF 23, AON, PROMPT, AOFF,...,F(Test)? 23,...'. Bei 'ja' ist "J" (oder ein beliebiges anderes Zeichen), dann 'R/S' zu tasten, bei 'nein' sofort 'R/S'. Der Zustand von Flag 23 (Alpha-Eingabe Flag) entspricht der Antwort: bei 'ja' ist Flag 23 gesetzt, bei 'nein' gelöscht (es sei denn, versehentlich ist - von Hand - ein Alpha-Eintrag vorgenommen worden). Flag 23 kann für Entscheidungen getestet werden. Die Text-Frage kann z.B. "DRUCKEN?" lauten; die Beschränkung auf 6 Buchstaben entfällt. - Quelle: Valentin Albillo (4747).

3-10 Meldungen bei Startvorbereitungen

Längere Startvorbereitungen sollten mit einem "FERTIG" und/oder akustischem Signal abgeschlossen werden. Noch besser ist es, eine symbolische Aufforderung in die Anzeige zu legen, so daß dem Benutzer klar ist, was er tun soll: z.B. "P↑Q↑F: A", wenn drei Konstanten p, q, f mit 'ENTER' in den Stapel zu bringen sind und der Programmlauf mit der Taste 'A' gestartet werden soll.

3-11 Ein Flag bestimmt die Meldung

Die Befehlsfolge '..., CF 00,..., "MLDG1", FS? 00, "MLDG2", PROMPT, ...' bewirkt folgendes: ist Flag 00 gelöscht, wird "MLDG1" angezeigt, ist Flag 00 gesetzt, erscheint "MLDG2". Beispiel (Flag 00 eingangs gelöscht): 'LBL C, SF 00, LBL B, "NEIGUNG?", FS? 00, "WINKEL?", PROMPT, FS?C 00, TAN, ...'. Drücken Sie 'B', erscheint "NEIGUNG?"; bei 'C' wird "WINKEL?" gefragt und 'TAN' berechnet; Flag 00 ist anschließend in jedem Fall gelöscht. - Vgl. auch 4-7, 4-19.

3-12 Anzeige von Registerinhalten mit Unterbrechung und ohne Druck

Will man den Inhalt eines Stapel- oder Datenregisters (z.B. R_{mn}) anzeigen, ohne zu drucken, das Programm aber zur Anzeige unterbrechen, kann man die Aufgabe mit passend zusammengestellten Befehlen 'VIEW mn', 'STOP' und 'CF/SF 21' lösen. Besser jedoch ist die Befehlsfolge 'CLA, ARCL mn, PROMPT'. Sie leistet nämlich genau das Geforderte, unabhängig davon, ob der Drucker abgekoppelt, angeschlossen und angeschaltet oder angeschlossen und ausgeschaltet ist. Ein u.U. für diese Folge geänderter Anzeige-Modus (auf z.B. 'FIX 0') kann zurückgesetzt werden, indem (z.B.) 'FIX 2' unmittelbar vor 'PROMPT' eingefügt wird. - Quelle: John Dearing (2791) (PPC CJ, V7N9P28). - Hinweis: Die hier vorgeschlagene Technik wird z.B. im Programm "DFD" (14-2) angewendet. - Vgl. auch 2-31, 4-1, 4-18, 6-4, 6-6, 6-14, 22-23.

3-13 'Ist eine neue Zahl eingegeben worden?', falls 'nein', alten Wert verwenden

Die Befehlsfolge '..., FS?C 22, STO 01, RCL 01, ...' bewirkt genau diese Entscheidung, vorausgesetzt Flag 22 wird vor der möglichen Eingabe gelöscht.

3-14 Test auf numerische Eingabe

Gewöhnlich verwendet man '..., CF 22, "(Frage)?", PROMPT, F(TEST)? 22, ...'. Bei Zahleneingabe wird Flag 22 gesetzt, bei bloßem 'R/S' bleibt Flag 22 gelöscht. Anschließend wird es für Entscheidungen getestet. Wenn '0' unter Ihren Eingaben nicht auftreten kann, gibt es eine Methode, die 2 Bytes einspart: '..., 0, "(Frage)?", PROMPT, X(Test)0?, ...'. Nach einer Zahleneingabe ist die Antwort auf - z.B. - 'X=0?' 'nein', bei 'R/S' ohne Eintrag 'ja'. - Quelle: Valentin Albillo (4747).

3-15 Alten Eintrag betrachten, bevor neuer eingetastet wird

Man füge 'RCL mn' unmittelbar vor 'PROMPT' ein, wenn der auf die Eingabe-Aufforderung folgende Eintrag in R_{mn} abgelegt werden soll. Beispiel: '..., "HOEHE?", RCL 01, PROMPT, STO 01, ...'. Auf diese Weise kann der alte Wert durch Betätigung der Korrekturtaste betrachtet werden. Soll er abermals verwendet werden, einfach 'R/S', andernfalls neuen Wert eintasten, dann 'R/S'. - Quelle: Robert McDonald (5460).

3-16 Eingabe und Berechnung in verschiedenen Maßeinheiten

Verwendet man eine Eingabe-Routine (wie "IN", 3-6), die vorhandene Werte benutzt, wenn keine neuen eingegeben werden, bei der jedoch die Eingaben in einer anderen Maßeinheit getätigt werden als die in der Berechnung benötigte (etwa Eingabe in 'Fuß', Berechnung in 'Metern'), kann man so vorgehen: man nutzt die Registerarithmetik zur Umwandlung, ruft die Eingabe-Routine auf und bedient sich dann der Registerarithmetik ein zweites Mal zur Rückwandlung für das Programm. Beispiel: die Befehlsfolge

'..., 3, STO 00, "L. FUSS", .3048, ST/ 04, XEQ "IN", .3048, ST* 04, ...'

wandelt den Inhalt von R_{04} in 'Fuß' und fordert zur Eingabe in 'Fuß' auf. Bei unverändertem Wert, 'R/S', sonst neuen Wert eintasten, dann 'R/S'. In beiden Fällen steht zum Schluß der Wert in 'Metern' in R_{04}.

3-17 Aufruf verschiedener Funktionen

Programme, die - je nach Umstand - verschiedene Benutzerfunktionen (keine Systemfunktionen) aufrufen, können nach dem Namen der Funktion fragen, ihn ablegen und die Funktion dann indirekt aufrufen. Der oder die betreffenden Namen können auch von Hand oder durch das Programm vorher passend abgelegt werden. Die Namen dürfen höchstens 6 Zeichen enthalten. Beispiel:

'..., "FUNKT.-NAME?", AON, PROMPT, AOFF, ASTO 03, ..., XEQ IND 03, ...'

3-18 Aufforderungen zur Eingabe ohne Programm-Stop ("NUM?" & "WRD?")

Die beiden folgenden Unterroutinen können dazu benutzt werden, Eingaben zu tätigen, ohne daß das Programm zu diesem Zweck echt unterbricht. Das aufrufende Programm setzt eine Meldung ins Alpha-Register, und die Unterroutine wartet - die Meldung anzeigend - 'standhaft' auf die Eingabe. 'XEQ "WRD?"' behandelt Alpha-Eingaben, 'XEQ "NUM?"' behandelt Zahleneingaben. Wenn der Eintrag erfolgt ist (Zahlen ins X-Register, Zeichen ins Alpha-Register), wird ins Hauptprogramm zurückverzweigt.

```
01♦LBL "NUM?"    03 CF 22      05 PSE        07 GTO 01
02 AVIEW         04♦LBL 01     06 FC?C 22    08 END
```

(20 Bytes)

```
01♦LBL "WRD?"    04 AVIEW      07 FC?C 23    10 END
02 CF 23         05♦LBL 02     08 GTO 02
03 AON           06 PSE        09 AOFF
```

(22 Bytes)

- Quelle: HP Key Notes, V4N1P6.

Kapitel 4

DIE ANZEIGE

4-1 Blick ins Alpha-Register ("VA")

Diese Routine unterbricht - im Gegensatz zu 'AVIEW' - ein Programm in keinem Falle. Hinter ihr kann bei Bedarf 'STOP' oder 'PSE' eingefügt werden (u.U. gesteuert durch Flags). "VA" druckt den Inhalt des Alpha-Registers, wenn der Drucker angeschlossen und eingeschaltet und Flag 21 gesetzt ist.

```
01♦LBL "VA"     04 SF 25        07 AVIEW        10 END
02 SF 25        05 FS?C 21      08 FC?C 25
03 PRA          06 CF 25        09 SF 21
```

(24 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM). Vgl. auch 2-31, 3-12, 4-18, 6-4, 6-6, 6-14, 22-23.

4-2 '(A)VIEW' verdrängt die Graugans

Die fliegende Graugans verschwindet aus der Anzeige, wenn in einem Programm 'VIEW' oder 'AVIEW' ausgeführt wird; der Inhalt des angesprochenen Registers verbleibt solange in der Anzeige, bis das Programm auf ein 'CLD', ein neues 'VIEW' oder 'AVIEW' oder ein 'STOP' trifft. 'CLD' oder 'STOP, R/S' bringt die Graugans zurück. - Quelle: HP-41 Bedienungshandbuch, © Copyright (Juni 1980) Hewlett-Packard Company. - Vgl. auch 2-6.

4-3 Anzeige-Modus speichern und rückrufen ("DSPS" & "DSPR")

"DSPS" speichert die Information über die Anzahl der Nachkommastellen und des Anzeige-Formats. Der Inhalt des T-Registers wird durch '8' ersetzt. Die Inhalte von X, Y und Z bleiben unverändert. "DSPR" holt die Information zurück, ohne den Stapel zu behelligen. Die Routinen benutzen die Flags 05 und 06 und das Register 00. Sie sind nützlich, wenn ein Unterprogramm bestimmte abweichende Modi verwendet und nach Ablauf des Unterprogramms der alte Zustand wiederhergestellt werden soll. - Quelle: Scott Morrison (4360) (PPC J, V6N5P31).

```
01♦LBL "DSPS"   11 ST+ 00     21 CF 06        31 FS? 06
02 0            12 RDN        22 FS? 40       32 ENG IND 00
03 STO 00       13 4          23 SF 05        33 FC?C 05
04 RDN          14 FS? 37     24 FS? 41       34 FS?C 06
05 1            15 ST+ 00     25 SF 06        35 FS? 05
06 FS? 39       16 RDN        26 RDN          36 SCI IND 00
07 ST+ 00       17 8          27 RTN          37 END
08 RDN          18 FS? 36     28♦LBL "DSPR"
09 2            19 ST+ 00     29 FS? 05
10 FS? 38       20 CF 05      30 FIX IND 00
```

(75 Bytes)

4-4 Synthetisches Speichern und Rückrufen des Anzeige-Modus ("SD" & "RD")

"SD" speichert den gegenwärtigen Anzeige-Modus in das Register, dessen Adresse in X liegt; "RD" ruft ihn daraus zurück. Beide Routinen lassen die Inhalte von X, Y und Z, die vor Eingabe der Register-Adresse vorhanden waren, unversehrt. "SD" speichert den Zustand der Flags 16-55.

```
01♦LBL "SD"     08 "*"          15 SIGN         22 STO \
02 SIGN         09 X<> [        16 ARCL IND L   23 "┣*****"
03 RDN          10 STO \        17 RDN          24 X<> \
04 RCL d        11 ASTO IND L   18 RCL d        25 STO d
05 STO [        12 RDN          19 STO \        26 RDN
06 "┣♦♦"        13 RTN          20 "┣**"        27 CLA
07 X<> [        14♦LBL "RD"     21 X<> ]        28 END
```

(66 Bytes)

- Quelle: Keith Jarett (4360) (PPC ROM).

4-5 Synthetisches Bewahren des Anzeige-Modus bei kurzfristiger Änderung

Man benutze das Flagregister d: es wird unmittelbar vor einer kurzfristigen Änderung aufgerufen und sobald als möglich in den alten Zustand zurückversetzt. Beispiel: um den Inhalt von X für eine Verwendung im Alpha-Register im Modus 'FIX 0, CF 29' zur Verfügung zu haben und anschließend den ursprünglichen Modus zurückzugewinnen, reicht die Befehlsfolge:

'..., RCL d, FIX 0, CF 29, ARCL Y, STO d, ...'

Im X-Register bleibt 'Abfall' zurück.

4-6 Längen von Alpha-Ketten

Bei der Verwendung von Alpha-Ketten, Eingabe-Aufforderungen eingeschlossen, beachte man folgende Längen-Begrenzungen:

6: Anzahl der Zeichen, die in einem numerischen oder Stapel-Register abgelegt werden können.

9: (a) Anzahl der Zeichen, die in einer Programmzeile mit 2-ziffriger Zeilen-Nummer (01-99) ohne 'Laufschrift' angezeigt werden können.
(b) Anzahl der Zeichen, die einer vollen, 15 Zeilen enthaltenden Programmzeile ohne Zeichen-Verlust angehängt werden können.

12: Anzahl der Zeichen, die ein 'PROMPT' ohne Laufschrift anzeigen kann (hierbei können nichtbenachbarte Kommas, Punkte und Doppelpunkte unberücksichtigt bleiben).

15: Anzahl der Zeichen, die eine Programmzeile höchstens aufnehmen kann.

24: Anzahl der Zeichen, die das Alpha-Register höchstens aufnehmen kann.

4-7 Text-Anzeige in Abhängigkeit von einem Flagzustand oder Test

Die Befehlsfolge '"TEXT1", TEST, "TEXT2"' kann häufig sehr gut verwendet werden. Beispiel: '"RICHTIG", X=Y? , "FALSCH"' setzt "FALSCH" ins Alpha-Register, wenn x=y gilt, jedoch "RICHTIG", wenn x≠y ist. Diese Technik kann auch dazu benutzt werden, längere Meldungen platzsparender einzusetzen. Beispiel: 'CLA, X≠Y?, "UN", "├RICHTIG"' meldet "RICHTIG" bei x=y und "UNRICHTIG" bei x≠y. - Quelle: Valentin Albillo (4747). - Vgl. auch 3-11, 4-19.

4-8 Anzeige-Modus synthetisch bestimmen ("DS")

Diese Routine versieht den HP-41 mit einer 'Anzeige-Funktion'. "DS" beläßt den Rechner im 'FIX'-, 'SCI'- oder 'ENG'-Format, aber der absolute Wert des ganzzahligen Teils der im X-Register befindlichen Zahl bestimmt die Anzahl der Nachkommastellen, sofern diese Zahl zwischen -9 und 9 liegt; andernfalls wird "DATA ERROR" angezeigt. Beispiel: der 'FIX 2'-Modus ändert sich durch '6, XEQ "DS"' in den 'FIX 6'-Modus. Die Routine zerstört die Inhalte der Register T und L.

```
01♦LBL "DS"      06 SCI IND L     11 STO [      16 CLA
02 SIGN          07 X<> d         12 "├**"      17 END
03 RDN           08 STO [         13 X<> \
04 RCL d         09 "├*****"      14 STO d
05 STO ]         10 X<> ]         15 RDN
```

(42 Bytes)

- Quelle: Keith Jarett (4360) (PPC ROM).

4-9 Änderung des Anzeige-Formats ("7F", "7S" & "7E")

Die Routine erlaubt es, das Format zu ändern, beläßt aber die Anzahl der Nachkommastellen. Sie benutzt keine numerischen Register. Ebensowenig werden die Zustände der Flags berührt, mit Ausnahme natürlich der Flags 40 & 41, die das Format festlegen.
- Quelle: Heinz Dalkowski.

```
01♦LBL "7F"      08 RTN           15 1          22 +
02 XEQ 07        09♦LBL "7E"      16 +          23 FS? 36
03 FIX IND X     10 XEQ 07        17 FS? 38     24 8
04 RTN           11 ENG IND X     18 2          25 +
05♦LBL "7S"      12♦LBL 07        19 +          26 END
06 XEQ 07        13 CLST          20 FS? 37
07 SCI IND X     14 FS? 39        21 4
```

(56 Bytes)

4-10 Synthetisches Anzeige-Format 'FIX/ENG'

Sind die Flags 40 & 41 zugleich gesetzt, befindet sich der HP-41 im sog. 'FIX/ENG'-Format. Im gewöhnlichen 'FIX'-Format (Flag 40 gesetzt, Flag 41 gelöscht) weicht die Darstellung von Zahlen, die zur Anzeige in diesem Format zu groß oder zu klein sind, ins 'SCI'-Format aus; im 'FIX/ENG'-Format jedoch schwenkt die Anzeige in diesen Fällen ins 'ENG'-Format um. - Quelle: William Wickes (3735) ('Synthetische Programmierung auf dem HP-41 C/CV', S. 106).

4-11 Fast-stetige Laufschrift

Fast-stetige Laufschrift nach links kann durch aufeinanderfolgende Registeraufrufe simuliert werden: '..., ARCL 01, ARCL 02, ARCL 03, ARCL 04, AVIEW, ARCL 05, ARCL 06, AVIEW, ARCL 07, ARCL 08, AVIEW,...'. Regel: die größte Stetigkeit wird erreicht, wenn die Anzahl der durch jede Gruppe von 'ARCL's im Alpha-Register belegten Bytes genau 12 beträgt und keine Trennungszeichen (Komma, Punkt, Doppelpunkt) enthalten sind; die erste Gruppe darf 12 - 24 Bytes belegen. - Quelle: Richard J. Nelson (1) (PPC J, V6N5P32).

4-12 Bessere Lesbarkeit der Laufschrift

Um eine durchlaufende Anzeige von mehr als 12 Zeichen nicht sofort nach links verschwinden zu lassen, kann man eine oder zwei Leerstellen ('SPACE') davorsetzen. - Quelle: Richard J. Nelson (1) (PPC J, V6N5P32).

4-13 Linksseitige Laufschrift ("SCE")

"SCE" ist insbesondere für Zeichenketten von mehr als 24 Zeichen geeignet. (1) Die volle Meldung wird in Teile zu je 13 Zeichen zerlegt (Leerzeichen zählen mit; nichtbenachbarte Trennungszeichen bleiben unberücksichtigt). Der letzte Teil darf weniger als 13 Zeichen umfassen. (2) Eine Marke - falls erwünscht - führt den Text an. (3) Die ersten *13* Zeichen kommen in eine Programmzeile. (4) 'ALPHA, ALPHA, ALPHA', um den Eintrag abzuschließen und eine neue Textzeile vorzubereiten. (5) Die nächsten *11* Zeichen werden mit 'APPEND' angeschlossen (13+11=24). (6) 'AVIEW'. (7) Nun fährt man mit Schritt (3) fort, und zwar mit dem ersten Zeichen des zuletzt in Schritt (5) verwendeten Textteiles. (8) Die Schritte (3) - (7) werden wiederholt, bis die volle Meldung verarbeitet ist. Beispiel: Die folgende Routine schiebt den Text "**HP-41C/V**" wiederholt von rechts nach links durch die Anzeige. Schritt 02 besteht aus 12 Leerzeichen, gefolgt von einem "*"; Schritt 05 endet mit 2 Leerzeichen; Schritt 06 fügt 11 Leerzeichen an. - Quelle: David Walker (1840) (PPC J, V6N7P3).

```
01♦LBL "SCE"        03 "┣**HP-41C/V**"   05 "**HP-41C/V**  "   07 AVIEW
02 "          *"    04 AVIEW             06 "┣          "      08 GTO "SCE"
```

(68 Bytes)

4-14 Rechtsseitige Laufschrift (Ersetzen der Graugans) ("SCR")

Die folgende Routine beruht auf der Ausnutzung eines kleinen sehr willkommenen HP-41-Fehlers, der das Fehler-Ignorierflag betrifft. Man lege das oder die gewünschten Ersatzzeichen ins Alpha-Register, setze Flag 25, befehle 'AVIEW' und gebe eine *beliebige* Anweisung, die normalerweise ein "NONEXISTENT" auslöst, z.B. 'SF 99'. Setzt man etwa die Befehlsfolge '"-", SF 25, AVIEW, CF 99' vor eine Schleife, so wird die Graugans durch einen Bindestrich ersetzt. Hier eine Routine zur Vorführung der behaupteten Erscheinung. Sie werden zur Eingabe der Ersatzzeichen aufgefordert. Wählen Sie z.B. "GRAUGANS", um die 'wirkliche' Gans durch ihre Bezeichnung zu ersetzen:

```
01♦LBL "SCR"        04 STOP       07 AVIEW      10 5
02 "LAUFZEICHEN?"   05 AOFF       08 CF 88      11 SIN
03 AON              06 SF 25      09♦LBL 01     12 GTO 01
```

(33 Bytes)

Offenkundig wird der Inhalt des Alpha-Registers durch die Anzeige geschoben. Versuchen Sie es mit "AAAAAAAAAAAATESTTESTTEST" (12 'A's und 3 'TEST's). Das volle Alpha-Register meldet sich mit einem Warnton. Der Text wird zunächst nach links, dann nach rechts durch die Anzeige befördert. Die 'A's verschwinden, und dann wandert nur noch der Textrest "TESTTESTTEST". Beachten Sie auch, daß bei diesem verbleibenden 'Rundlauf' alle 12 beteiligten Zeichen ständig in der Anzeige zu sehen sind. Folgerung: "SCR" rollt die letzten 12 Zeichen einer Alpha-Kette von mehr als 12 Zeichen nach rechts durch die Anzeige, sobald der normale Lauf nach links beendet worden ist. Wenn Sie 6 mal hintereinander das Zeichenpaar ":." eintasten, entsteht ein ungewöhnlicher optischer Effekt. - Quelle: Richard Nelson (1) (PPC J, V6N8P24).

4-15 Synthetisches Ersetzen der Graugans

Man setze - mit einer Kette von höchstens 12 Zeichen im Alpha-Register - die Befehlsfolge 'RCL d, AVIEW, STO d' vor eine Schleife, dann wird die Graugans durch den Inhalt des Alpha-Registers ersetzt, der nun an ihrer Stelle durch die Anzeige zieht. - Quelle: William Wickes (3735) ('Synthetische Programmierung auf dem HP-41 C/CV', S. 121).

4-16 Register X & Y zugleich anzeigen ("XY" & "X?Y")

Diese Routinen sind nützlich, wenn das Ergebnis einer Berechnung ein *Zahlenpaar* ist, das angezeigt werden soll (z.B. komplexe Zahlen oder Koordinaten). "XY" verwendet den bestehenden Anzeige-Modus; "X?Y" formatiert entsprechend der in R_{00} stehenden Zahl 'm.n', wodurch die Inhalte von X und Y im Format 'FIX m' bzw. 'FIX n' ausgegeben werden.

```
01♦LBL "XY"         03 ARCL X     05 ARCL Y     07 END
02 CLA              04 "  "       06 AVIEW
```

(18 Bytes)

```
01♦LBL "X?Y"        05 "⊢ "        09 *              13 AVIEW
02 CLA              06 RCL 00      10 FIX IND X      14 END
03 FIX IND 00       07 FRC         11 RDN
04 ARCL X           08 10          12 ARCL Y
```

(29 Bytes)

- Quelle: Peter Ladrach (5060) (PPC CJ, V7N4P6).

4-17 Synthetische Prüfung der Anzeige-Segmente ("DT")

Um zu prüfen, ob sämtliche Segmente und Indikatoren der Anzeige ihre Funktion erfüllen, 'XEQ "DT"'. Die Routine zeigt zunächst 12 Kommas, kurz darauf alle 12 Vollzeichen und Doppelpunkte sowie sämtliche Indikatoren der Fußzeile (BAT, USER usw.). Um zum Ausgangszustand zurückzukehren, muß der PRGM-Modus ausgeschaltet und dann 'R/S' gedrückt werden. Die Routine verwendet die Stapelregister T & L. Die Zeilen 02 und 10 lauten dezimal 247, 248, 0, 0, 16, 0, 33, 232 bzw. 246, 128, 58, 128, 58, 128, 58.

```
01♦LBL "DT"                 07 AON        13 ARCL L     19 RDN
            02 "♦♦θ♦!"      08 PSE        14 ARCL L     20 CLD
03 RCL [                    09 AOFF       15 X<> d      21 END
04 ",,,,,,"                 10 ":::"      16 AVIEW
05 ASTO L                   11 ASTO L     17 STOP
06 ARCL L                   12 ARCL L     18 X<> d
```

(56 Bytes)

- Quelle: William Wickes (3735) & Valentin Albillo (4747) (PPC ROM).

4-18 'AVIEW'-Ersatz Routinen ("AV" & "AVN")

Die Routinen können von Programmen aufgerufen werden, um 'AVIEW' zu ersetzen. "AV" ('Alpha View') bewirkt nur dann eine Programmunterbrechung, wenn der Drucker ausgeschaltet ist. "AV" hat folgende Eigenschaften: (a) Kein Drucker - 'AVIEW' ohne Unterbrechung. (b) Drucker ausgeschaltet und Meldung "PRINTER OFF" *) - Drucker anschalten und 'R/S', woraufhin der Inhalt des Alpha-Registers gezeigt und gedruckt wird. (c) Drucker eingeschaltet - es wird gedruckt und angezeigt, ohne zu unterbrechen. (d) Flag 21 - Flag 21 *steuert nicht* den Druckvorgang; sein Zustand bleibt erhalten. "AV" soll den Bedürfnissen des Benutzers genügen, der von seinem HP-41-System ganz einfach verlangt: der Drucker soll drucken, wenn er angeschlossen ist, und sich melden, falls er dabei ausgeschaltet ist. "AV" zeigt überdies eine ungewöhnliche Flagbenutzung. Die Zeilen 03 und 09 sprechen Flag 14 (vorgesehen zum Beschreiben geschützter Karten) an, um den Zustand von Flag 21 festzuhalten. Die Zeilen 09 und 10 stellen beider ursprünglichen Zustand wieder her. Die Gefahr, die Routine anzuhalten, um ausgerechnet eine geschützte Karte zu beschreiben, ist so gut wie ausgeschlossen.

---

*) Hinweis des Übers.: der HP-IL-Drucker meldet sich mit "TRANSMIT ERR"

"AVN" unterbricht im Gegensatz zu "AV" den Programmlauf in keinem Fall, auch dann nicht, wenn ein angeschlossener Drucker nicht druckbereit ist. Es ähnelt "VA" (4-1), doch läßt "AVN" selbst dann drucken, wenn - bei druckbereitem Drucker - Flag 21 gelöscht ist, während "VA" in diesem Fall nur anzeigt.

```
01♦LBL "AV"     07 CF 21
02 FS? 21       08 AVIEW
03 SF 14        09 FS?C 14
04 SF 21        10 SF 21
05 FS? 55       11 END
06 PRA
```

(26 Bytes)

```
01♦LBL "AVN"    07 CF 25
02 FS? 21       08 CF 21
03 SF 14        09 AVIEW
04 SF 21        10 FS?C 14
05 SF 25        11 SF 21
06 PRA          12 END
```

(29 Bytes)

- Quelle: HP Key Notes, V5N1P7 & V5N2P12. Vgl. auch 2-31, 3-12, 4-1, 6-4, 6-6, 6-14, 22-23.

4-19 Verschiedene Alpha-Texte in einer Schleife ausgeben

Gelegentlich sollen bei einem Schleifendurchlauf verschiedene Texte angezeigt und/oder gedruckt werden. Das folgende Beispiel zeigt eine Byte-sparende und sehr schnelle Technik, die häufig mit Gewinn verwendet werden kann:

```
'..., 1.009, STO mn, LBL 14, SF IND mn, ...
 ..., FS?C 01, "TEXT1", FS?C 02, "TEXT2", ...
 ..., FS?C 09, "TEXT9", AVIEW, ISG mn, GTO 14, ...'
```

Natürlich müssen die Flags (im voranstehenden Beispiel 01-09) gelöscht sein, bevor die Schleife durchlaufen wird. - Quelle: Heinz Dalkowski.

4-20 Ergebnisse anzeigen und Berechnung fortsetzen ("DSP")

Diese Routine vermag die Zeit, die der Benutzer auf gleichartige Ausgaben wartet, zu verkürzen. Man verwendet "DSP" statt 'STOP' oder 'PROMPT' für die Ausgabe. Flag 04 muß vor dem ersten Aufruf von "DSP" gelöscht sein; Flag 03 wird so benutzt: 'SF 03', um 'VIEW X' auszuführen; 'CF 03', um 'AVIEW' auszuführen. Das Hauptprogramm berechnet den ersten Wert, der zur Ausgabe ins X- oder ins Alpha-Register gelegt wird. Dann wird nach "DSP" verzweigt, um den Inhalt des X- oder Alpha-Registers anzuzeigen; "DSP" setzt anschließend Flag 04. Während man das erste Ergebnis notiert, rechnet der HP-41 geschäftig das zweite Ergebnis aus. Der PRGM-Indikator bleibt gesetzt, bis "DSP" zum zweiten Mal aufgerufen wird. Weil Flag 04 jetzt gesetzt ist (es darf nicht vom Hauptprogramm gelöscht werden!), wird die Ausführung unterbrochen. Auf 'R/S' hin wird nun *sofort* der zweite Wert angezeigt, und während man diesen Wert notiert, berechnet das Hauptprogramm bereits den dritten, usw. Warnung: 'R/S' darf nicht gedrückt werden, solange der PRGM-Indikator noch gesetzt ist! Flag 21 muß gelöscht werden, wenn ein Drucker angeschlossen aber ausgeschaltet ist.

```
'LBL "DSP", FS? 04, STOP, FS? 03, VIEW X, FC? 03, AVIEW, SF 04, RTN'
```

(20 Bytes)

- Quelle: HP Key Notes, V5N2P13.

Kapitel 5

BEHANDLUNG VON ALPHA-KETTEN

5-1 Alpha-Ketten speichern und rückrufen ("AM" & "MA")

'XEQ "AM"' legt den Inhalt des Alpha-Registers in Datenregister: 'XEQ "MA"' holt ihn von dort ins Alpha-Register zurück; "AM" löscht überdies das Alpha-Register. Im X-Register muß in beiden Fällen zur Steuerung eine Zahl der Form 'abc.lmn' liegen. Die Adresse des letzten Datenregisters (lmn) darf nicht um mehr als 3 größer sein als die des ersten Registers (abc), es sei denn, man benutzt eine vollständige Steuerungszahl der Form 'abc.lmnik' (ik>1). '17.020, XEQ "AM"' z.B. legt alle 24 Zeichen des Alpha-Registers in R_{17}-R_{20} ab; benutzt man stattdessen '17.018', werden nur die ersten 12 Zeichen gespeichert.

```
01♦LBL "AM"      05 ISG X        09 CLA          13 GTO 02
02♦LBL 01        06 GTO 01       10♦LBL 02       14 END
03 ASTO IND X    07 RTN          11 ARCL IND X
04 ASHF          08♦LBL "MA"     12 ISG X
```

(32 Bytes)

- Quelle: Keith Jarett (4360) (PPC ROM).

5-2 Beschränkungen des Alpha-Ketten-Vergleichs *auf frühen Modellen*

Wenn Sie zwei Alpha-Ketten vergleichen wollen, die ursprünglich mehr als 6 Zeichen enthalten, müssen Sie folgendermaßen vorgehen, um die Ketten richtig zu kürzen und korrekte Vergleichsergebnisse zu erzielen (Ketten können nur mit 'X = Y?' oder 'X ≠ Y?' getestet werden). (1) Speichern Sie die (erste/zweite) Kette mit 'ASTO mn' in irgendein Datenregister R_{mn}. Ist die Kette nicht länger als 6 Zeichen, führen Sie gleich Schritt 4 bzw. 6 aus. (2) Löschen Sie das Alpha-Register mit 'CLA'. (3) Holen Sie die Kette mit 'ARCL mn' aus R_{mn} ins Alpha-Register. (4) Speichern Sie die (erste) Kette mit 'ASTO X' ins X-Register. (5) Wiederholen Sie die Schritte 1-3 für die zweite Alpha-Kette. (6) Speichern Sie die (zweite) Kette mit 'ASTO Y' ins Y-Register. (7) Testen Sie 'X = Y?' oder 'X ≠ Y?'. - Quelle: HP Key Notes, V4N1P3.

5-3 Vorsicht beim Edieren von Leerstellen kombiniert mit Trennungszeichen (. : ,)

Die Korrekturtaste, welche die zur Eingabe auffordernde Unterstreichung bewegt, kann diese Unterstreichung gelegentlich um zwei Zeichenstellen zurücksetzen, obwohl nur ein Zeichen gelöscht wird! Probieren Sie im Alpha-Modus diese Schrittfolge aus: 'CLA, "Z", SPACE, ":", ←, ←, AVIEW'. Sie sehen zwar "Z" aus der Anzeige verschwinden, doch das 'AVIEW' (oder auch 'ALPHA, ALPHA') bringt es wieder zum Vorschein. Wenn Sie ein

noch verwirrenderes Beispiel für diesen Effekt sehen wollen, setzen und löschen Sie mehrere Paare 'SPACE, ":"' hinter "Z". Ganz dramatisch wird es, wenn Sie dies mit der größtmöglichen Anzahl solcher Paare tun: die Unterstreichung verschwindet links und taucht 3 '←'s weiter plötzlich rechts wieder auf; beim vierten '←' gar kehrt der ganze restliche Alpha-Inhalt in die Anzeige zurück. - Quelle: Charles Harris (1959) (PPC CJ, V7N3P28).

5-4 Synthetisches Löschen des letzten Alpha-Zeichens ("AD")

Die Routine löscht das letzte Zeichen einer Kette im Alpha-Register. Sie benutzt den Stapel.

```
01♦LBL "AD"     07 RCL \        13 RDN          19 ASHF
02 RCL ↑        08 CLA          14 STO \        20 ARCL Z
03 RCL ]        09 STO [        15 RDN          21 END
04 ,            10 ASTO X       16 STO ]
05 X<> \        11 RDN          17 ,1
06 "┣******"    12 STO [        18 STO ↑
```

(47 Bytes)

- Quelle: Gerard Westen (4780) (PPC ROM).

5-5 Synthetisches Aussondern und Ersetzen von Zeichen ("NC" & "SU")

"NC" bezieht sich auf eine positive Zahl im X-Register, deren ganzzahliger Anteil 'n' zwischen 1 und 10 liegt, und ersetzt eine beliebig lange Kette im Alpha-Register durch ihr n-tes Zeichen, gerechnet von rechts aus; dieses Zeichen wird zugleich ins X-Register gelegt. Die vor Eintasten von 'n' in X und Y befindlichen Werte rücken nach Y und Z. - "SU" versieht den Benutzer mit einer wirkungsvollen Möglichkeit, Alpha-Ketten zu behandeln: befinden sich eine positive Zahl, deren ganzzahliger Anteil 'n' zwischen 1 und 10 liegt, im X- und eine Alpha-Kette im Y-Register, so wird das - von rechts aus gesehen - n-te Zeichen der Alpha-Kette im Alpha-Register durch das ganz rechterhand stehende Zeichen der Alpha-Kette im Y-Register ersetzt. Die vor Eintasten von 'n' in X und Y befindlichen Zeichen und Werte bleiben erhalten. Ist 'n' um 1 größer als die Anzahl der Zeichen im Alpha-Register, wird das aus Y verwendete Zeichen der Kette im Alpha-Register links vorangestellt.

```
01♦LBL "NC"   09 X<>Y       17 FS? 25     25 X<> Z   33 X<> L   41 CLX     49 RDN
02 CF 25      10 -          18 RCL ↑      26 STO ↑   34 10↑X    42 ISG L    50 RTN
03 GTO 14     11 RCL d      19 "┣*"       27 RDN     35 RCL d   43 CLX     51♦LBL 14
04♦LBL "SU"   12 SCI IND Y   20 FC?C 25    28 X<> ]   36 FIX 0   44 X<> ↑   52 X<> ]
05 SF 25      13 ARCL Y     21 GTO 14     29 X<> \   37 CF 29   45 STO \   53 CLA
06♦LBL 14     14 STO d      22 X<> Z      30 STO [   38 ARCL Y  46 CLX     54 STO [
07 INT        15 RDN        23 STO ]      31 DSE L   39 STO d   47 X<> ]   55 ASTO X
08 E1         16 X<> ]      24 "┣******"  32 CLX     40 RDN     48 STO [   56 END
```

(112 Bytes)

- Quelle: William Wickes (3735) & Carter Buck (4783) (PPC ROM).

5-6 Synthetische Umwandlungen Zeichen ⇄ Dezimalzahl ("CD" & "DC")

"CD" wandelt ein im Alpha-Register liegendes Zeichen in die entsprechende Dezimalzahl (0-255) aus der Byte-Tabelle um. Liegen mehrere Zeichen im Alpha-Register, wird das ganz rechts stehende Zeichen umgewandelt. Ist Flag 10 gelöscht, wird das Zeichen aus dem Alpha-Register entfernt; bei gesetztem Flag bleibt es erhalten. - "DC" verwendet den ganzzahligen Anteil der im X-Register liegenden Zahl modulo 256, um ihn in das entsprechende Zeichen der Byte-Tabelle umzusetzen und dem Text im Alpha-Register rechterhand anzufügen. Die Inhalte von X und Y bleiben erhalten, die von Z und T gehen verloren.

```
01♦LBL "CD"      17 ST+ ]         33 LASTX         49 RCL \
02 "⊢♦↓****"     18 RDN           34 +             50 "⊢*"
03 RCL [         19 6             35 OCT           51 X<> ]
04 FS? 10        20 ST* ]         36 X<> d         52 X<>Y
05 "⊢*"          21 RDN           37 FS?C 11       53 STO \
06 STO [         22  E1           38 SF 12         54 X<> ↑
07 CLX           23 ST* L         39 FS?C 10       55 "⊢*"
08 X<> ]         24 X<> L         40 SF 11         56 STO ↑
09 SIGN          25 ST+ ]         41 FS?C 09       57 RDN
10 CLX           26 CLX           42 SF 10         58 X<> ]
11 X<> \         27 X<> ]         43 FS? 07        59 X<> \
12 "⊢***"        28 RTN           44 SF 09         60 STO [
13 X<> [         29♦LBL "DC"      45 FS? 06        61 RDN
14 X<> L         30 INT           46 SF 08         62 END
15 X<> \         31 256           47 X<> d
16 INT           32 MOD           48 X<> [
```

(129 Bytes)

- Quelle: William Wickes (3735) & Roger Hill (4940) (PPC ROM).

5-7 Synthetische Umwandlung Hexadezimalzahl ⇄ nicht-normalisierte Zahl ("HN" & "NH")

Eine nicht-normalisierte Zahl (NNZ) heißt eine solche Zahl, deren Vorzeichen-Nybble weder 0 (positive Zahl) noch 1 (Alpha-Kette) noch 9 (negative Zahl) ist oder deren Vorzeichen-Nybble zwar 0 oder 9 lautet, die aber hexadezimale Ziffern der Werte A-F enthält. - "HN" wandelt eine im Alpha-Register stehende Hexadezimalzahl (von bis zu 14 Hexadezimalziffern) in eine NNZ um, die ins X-Register kommt und deren zugehöriger Alpha-Wert zum Schluß im Alpha-Register steht. Die Werte von X, Y und Z gelangen nach Y, Z und T; der Inhalt von L wird zu '0'. - "NH" wandelt umgekehrt eine im X-Register stehende NNZ in ihre 14 Hexadezimalziffern um und setzt diese ins Alpha-Register. Die NNZ in X bleibt bestehen. Liegt eine Alpha-Kette im X-Register, so wird diese in ihre 14 Hexadezimalziffern umgewandelt. Sie bleibt ebenfalls in X liegen. Die Routine endet im Alpha-Modus. Die Inhalte von T und L gehen verloren.

```
01♦LBL "NH"     30 FC? 06       59 AON          88 FS? 04
02 CLA          31 FS? 05       60 RTN          89 SF 08
03 STO [        32 FC? 04       61♦LBL "HN"     90 FC? 01
04 SIGN         33 GTO 14       62 7            91 GTO 14
05 X<> d        34♦LBL 13       63 SIGN         92 SF 08
06 "⊢♦♦♦"       35 SF 01        64♦LBL 02       93 FC?C 11
07 ,            36 CF 02        65 RDN          94 SF 11
08 X<> [        37 CF 03        66 RCL \        95 FS? 11
09 "⊢♦♦"        38 CF 04        67 X<> d        96 GTO 14
10 X<> [        39 FS?C 07      68 CF 11        97 FC?C 10
11 FIX 9        40 GTO 14       69 CF 10        98 SF 10
12 ARCL \       41 SF 07        70 FC?C 09      99 FC? 10
13 "⊢**"        42 FS?C 06      71 GTO 14       100 SF 09
14 ARCL ]       43 GTO 14       72 SF 12        101♦LBL 14
15 X<> ]        44 SF 06        73 FC?C 15      102 X<> d
16 FIX 3        45 CF 05        74 SF 15        103 X<> ]
17 ARCL ]       46♦LBL 14       75 FS? 15       104 "⊢*"
18 STO ]        47 X<> d        76 GTO 14       105 STO ↑
19 "⊢*"         48 STO [        77 FC?C 14      106 "⊢*"
20 RDN          49 "⊢*"         78 SF 14        107 X<> ↑
21 STO d        50 RDN          79 FC? 14       108 STO ]
22 CLX          51 DSE X        80 SF 13        109 DSE L
23 FS? 10       52 GTO 01       81♦LBL 14       110 GTO 02
24 GTO 12       53♦LBL 12       82 FS? 07       111 CLA
25 RDN          54 STO ↑        83 SF 11        112 STO [
26 14           55 X<> ]        84 FS? 06       113 AOFF
27♦LBL 01       56 X<> \        85 SF 10        114 END
28 RCL ]        57 STO [        86 FS? 05
29 X<> d        58 X<> L        87 SF 09
```

(227 Bytes)

- Quelle: Roger Hill (4940), William Wickes (3735), John McGechie (3324) & Richard Hall (4803).

5-8 Eine Zahl (von höchstens 6 Ziffern) in X in ihrer Alpha-Form nach X bringen ("N-A")

'LBL "N-A", CLA, ARCL X, ASTO X, RTN'. - Quelle: Jake Schwartz (1820) (PPC J, V6N8P26).

5-9 Verlust von Zeichen durch 'ARCL'

Sobald dem Alpha-Inhalt Zeichen mit 'ARCL' angefügt werden, ertönt kein Warnton, wenn die äußersten linken Zeichen verlorengehen. - Quelle: William Wickes (3735).

Kapitel 6

FLAGS UND TÖNE

6-1 Flagschaltungen ("FT")

Die Befehlsfolge 'FC?C nm, SF nm' setzt ein Flag, wenn es gelöscht war und löscht es, wenn es gesetzt war. Sie kann gewinnbringend in Schleifen eingesetzt werden. "FT" schaltet das Flag, dessen Nummer (0-29) in X liegt, um: 'LBL "FT", FC?C IND X, SF IND X, RTN'. - Quelle: Ron Knapp (618) (PPC J, V6N5T6) & Jake Schwartz (1820) (PPC J, V6N8P26).

6-2 Flag setzen oder löschen in Abhängigkeit vom Wert in X

'LBL A, SF 01, X = 0?, CF 01,...'. - Quelle: Bill Kolb (265) (BP 67/97).

6-3 Massenlöschung von Flags ("CFX" & "CFA")

"CFX" löscht den durch die im X-Register liegende Kontrollzahl 'abc.lmn' bestimmten 'Block' von Flags, z.B. die Flags 5-10 durch '5.01, XEQ "CFX"'. "CFA" stellt 0.025 voran, um die Flags 0-25 zu löschen. Sämtliche Daten- und Stapelregister (bis auf T) bleiben unversehrt.

```
'LBL "CFA", .025, LBL "CFX", CF IND X, ISG X, GTO "CFX", RDN, RTN'
```

(29 Bytes)

- Quelle: William Cheeseman (4381) (PPC CJ, V7N5P7).

6-4 Synthetisches Umschalten von Flags ("IF")

Diese Routine schaltet das Flag, dessen Nummer in X liegt, um. Sie ist anwendbar auf die Flags 00-29, 31-44, 47-50, 52 und 55. Flag 47 steuert den SHIFT-Modus, Flag 48 den ALPHA-Modus, Flag 52 den PRGM-Modus; Flag 50 friert die Graugans ein (mit '←' oder 'R/S' wird sie 'erlöst'). '52, XEQ "IF"' schaltet in den PRGM-Modus, und zwar auf die Zeile 00 des aufrufenden Programms bzw. auf dessen letzte, falls es mit 'RTN, RTN' oder 'RTN, END' schließt. Die Werte aus X und Y (vor Eintasten der Flag-Nummer) bleiben erhalten, die aus Z und T gehen verloren. Beim Umschalten von Flag 55 ist zu beachten: "IF" setzt Flag 55, wenn der Drucker abgekoppelt ist, aber es löscht dieses Flag ohne Dauerwirkung, wenn der Drucker angeschlossen ist: Druckbefehle setzen es nämlich wieder und werden dann ausgeführt.

```
01♦LBL "IF"     09 RCL d        17 X<> d        25 SCI IND X
02 ABS          10 X<> [        18 FC?C IND ]   26 ARCL X
03 24           11 INT          19 SF IND ]     27 X<> ]
04 +            12 SCI IND X    20 X<> d        28 STO d
05 STO [        13 ARCL X       21 STO [        29 RDN
06 8            14 X<>Y         22 RDN          30 CLA
07 ST/ [        15 X<> ]        23 12           31 RTN
08 MOD          16 X<> \        24 -            32 END
```

(58 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM).

6-5 Synthetischer Blick auf die Flags ("VF")

'XEQ "VF"' zeigt Ihnen, welche Flags gesetzt sind. "VA" ruft in Zeile 04 die Routine "IF" (aus 6-4) auf. Zeile 08 lautet dezimal 245, 4, 168, 0, 128, 1.

```
01♦LBL "VF"     15 X<> d        29 RTN          43♦LBL 03
02 50           16 CLA          30♦LBL 02       44 X<> d
03 FC? 50       17♦LBL 01       31 "⊢ "         45 RTN
04 XEQ "IF"     18 FS? IND L    32 X<> d        46♦LBL "VA"
05 "BP♦♦♦♦"     19 XEQ 02       33 ARCL L       47 SF 25
06 RCL [        20 ISG L        34 CF 24        48 PRA
07 SIGN         21 GTO 01       35 DSE T        49 SF 25
08 "α      ♦z"  22 X<> d        36 GTO 03       50 FS?C 21
09 X<> [        23 FC?C 24      37 XEQ "VA"     51 CF 25
10 4            24 XEQ "VA"     38 TONE 6       52 AVIEW
11 RDN          25 ADV          39 CLA          53 FC?C 25
12 "FLAGS SET:" 26 BEEP         40 4            54 SF 21
13 X<> d        27 X<> d        41 RDN          55 END
14 XEQ "VA"     28 RDN          42 SF 24
```

(131 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM).

6-6 Synthetisches Umschalten von Flag 55 ("55")

Mit 'XEQ "55"' kann das Druckerflag 55 angeschaltet werden, wenn der Drucker abgekoppelt ist. Sobald Flag 55 gesetzt ist, unterbricht '(A)VIEW' nicht die Programmausführung, und zwar auch dann nicht, wenn Flag 21 gesetzt wird, wodurch ROM-Routinen als Unterprogramm *ohne Ausgabeunterbrechung* genutzt werden können. Der Stapel bleibt unbehelligt; nur Flag 55 wird umgeschaltet. Datenregister werden nicht benutzt. Das Alpha-Register bleibt gelöscht zurück. Die Ausführungszeit beträgt weniger als 1 Sekunde.

```
01♦LBL "55"     06 STO [        11 SF 15        16 STO d
02 CLA          07 "⊢*****"     12 X<> d        17 X<> ]
03 STO \        08 X<> [        13 STO [        18 CLA
04 CLX          09 STO d        14 "⊢**"        19 END
05 RCL d        10 FC?C 15      15 X<> \
```

(47 Bytes)

- Quelle: Valentin Albillo (4747). Vgl. auch 2-31, 3-12, 4-1, 4-18, 6-14, 22-23.

6-7 Synthetisches Rücksetzen der Flags ("RF")

Diese Routine versetzt die Flags in den Zustand, den sie nach einer Totallöschung ("MEMORY LOST") hätten, mit Ausnahme des Anzeige-Modus, der 'FIX 2' statt 'FIX 4' erhält. Beachten Sie, daß Flag 27 (USER-Modus) dabei gelöscht wird. Das Alpha-Register wird gelöscht. Die Flags 26, 28, 29, 38 und 40 werden gesetzt. Die Zeile 02 lautet dezimal 244, 44, 2, 128, 0.

```
01♦LBL "RF"          03 ASTO d          05 CLA
02 ",x̄♦"             04 CF 03           06 END
```

(19 Bytes)

- Quelle: Valentin Albillo (4747) & Carter Buck (4783) (PPC ROM).

6-8 Synthetische Massensteuerung von Flags

Eine synthetische Zeile von 7 Zeichen, gefolgt von einem 'RCL M' setzt eine NNZ (nicht-normalisierte Zahl; vgl. 5-7) ins X-Register. Eine wichtige Verwendung so hergestellter NNZs ist die Massensteuerung von Flags durch die Speicherung der NNZ ins Flagregister d, so daß alle 56 Flags auf einen Schlag in den gewünschten Zustand versetzt werden können. Die zugrundeliegende Befehlsfolge lautet '"XXXXXXX", RCL M, STO d', worin "XXXXXXX" die synthetische Zeile zur Herstellung der NNZ bedeuten soll. Diese Befehlsfolge benutzt nur 12 Bytes, also ebensoviele, wie für 6 Befehle der Art 'SF mn' oder 'CF mn' nötig sind; mithin werden Bytes eingespart, sobald mehr als 6 Flags in einen bestimmten Zustand versetzt werden sollen. - Um die den gewünschten Flagzustand herstellende synthetische Textzeile festzulegen, schreibt man die Standorte aller 56 Flags als binäre 56-Bit-Zahl auf, mit Einsen für gesetzte und Nullen für gelöschte Flags, und gruppiert dann die Bits in hexadezimale Bytes zu je 8 Bits. Das folgende Beispiel setzt die Flags 1, 2, 3, 26 (Tonsignal), 28 (Dezimalpunkt), 29 (Zifferngruppierung), 38, 39, 40 und 41 ('FIX/ENG 3'-Format; vgl. 4-10), 43 (RAD-Modus), 44 (Dauereinschaltung); alle anderen Flags werden gelöscht. Flag 00 ist das äußerste linke, Flag 55 das äußerste rechte Bit.

0111	0000	0000	0000	0000	0000	0010	1100	0000	0011	1101	1000	0000	0000
7	0	0	0	0	0	2	C	0	3	D	8	0	0

Die benötigte Textzeile lautet also 'F7 70 00 00 2C 03 D8 00', das ist dezimal 247, 112, 0, 0, 44, 3, 216, 0. - Quelle: William Wickes (3735) ('Synthetische Programmierung auf dem HP-41C/CV', S. 106/107). Vgl. auch 25-6.

6-9 Zuweisung von 'TONE' auf die obersten zwei Tastenreihen

Weist man 'TONE' jeder Taste der beiden obersten Reihen zu, bringt ein doppelter Druck im USER-Modus die Töne 1-5 bzw. 6-0 hervor. - Quelle: George Donaldson (3825) (PPC J, V6N5P19). - Dieselbe Technik ist auch auf andere Systemfunktionen anwendbar,

z.B. 'FIX' auf '1/x' für einfach auszuführendes 'FIX 2'. Mit Hilfe synthetischer Techniken (s. das Tastenzuweisungsprogramm "KA" in W. Wickes, 'Synthetische Programmierung auf dem HP-41C/CV', S. 69-71) lassen sich solche Tastenzuweisungen sogar so vornehmen, daß längere Tastenfolgen (beispielsweise STO + IND X) auf einen Tastendruck beschränkt werden.

6-10 Synthetische Ton-Programme ("T1" - "T6")

"T1": Raumschiff Enterprise feuert mit der Laser-Kanone. - "T2": BEEP 2. - "T3": Toccata von J. S. Bach. - "T4": Tonfolge, deren Rhythmik dem amerikanischen Werbeslogan 'Shave & a Haircut 2 Bits' entlehnt ist ('2 bits' ist Slang und bedeutet 24 cents). - "T5": Alarm. - "T6": Wer diese Tonfolge hört, dem 'begegnet zum ersten Mal die dritte Art'! - Quellen: "T1", Cary Reinstein (2046); "T3", Nicholas Peros (2392); "T2", "T4"-"T6", Gary Tenzer (1816) (PPC CJ, V7N2P49). Die im folgenden Programmausdruck dazugesetzten Zahlen geben den Dezimalwert des zweiten Bytes des jeweiligen synthetischen Tones an. Synthetische Töne bestehen aus 2 Bytes: der Dezimalwert des ersten Bytes ist immer 159, der des zweiten eine Zahl zwischen 0 und 127. Zur Herstellung der synthetischen Töne kann das Programm "LB" (25-1) benutzt werden.

```
01♦LBL "T1"        20 TONE 6          39♦LBL 01          58 X<>Y
02 TONE 7  (57)    21 TONE 4          40 X<> X           59♦LBL 02
03 TONE 7  (57)    22 TONE 8          41 X<> X           60 TONE 7  (57)
04 TONE 7  (57)    23 TONE 9          42 RTN             61 TONE 2  (72)
05 TONE 9  (89)    24 RTN             43♦LBL "T4"        62 TONE 7  (57)
06 TONE 9  (89)    25♦LBL "T3"        44 TONE X  (115)   63 TONE 2  (72)
07 TONE 9  (89)    26 TONE 1          45 TONE 1          64 TONE 7  (57)
08 TONE 7  (57)    27 TONE 0          46 TONE 1          65 TONE 2  (72)
09 TONE 9  (89)    28 TONE 3  (33)    47 TONE 8  (98)    66 TONE 7  (57)
10 TONE 9  (89)    29 XEQ 01          48 TONE Z  (113)   67 TONE 2  (72)
11 TONE 7  (57)    30 XEQ 01          49 TONE b  (124)   68 RTN
12 TONE 9  (89)    31 XEQ 01          50 TONE 3  (83)    69♦LBL "T6"
13 TONE 9  (89)    32 TONE 0          51 RTN             70 TONE 0
14 TONE 9  (89)    33 TONE 5  (15)    52♦LBL "T5"        71 TONE 1
15 RTN             34 TONE 4  (94)    53 XEQ 02          72 TONE 5  (15)
16♦LBL "T2" "      35 TONE H  (109)   54 XEQ 02          73 TONE b  (124)
17 TONE 1          36 TONE b  (124)   55 X<>Y            74 TONE 4  (94)
18 TONE 5          37 TONE 3  (13)    56 X<>Y            75 .END.
19 TONE 3          38 RTN             57 X<>Y
```

(168 Bytes)

Nachtrag: Das kürzestmögliche und schnellste Dauer-Tonprogramm ist dieses: 'LBL "T7", RCL b, TONE 9 (89), STO b'; vor dem ersten Aufruf 'PACK' ausführen.

6-11 Mozart synthetisch ("MOZ" & "MO")

"MOZ" spielt eine Passage aus 'Eine kleine Nachtmusik', begleitet von einer passend bewegten Anzeige. Die Stapelregister werden gelöscht. "MO" ist eine Version ohne Anzeige und ohne Stapelbenutzung. *)

*) Der Übersetzer glaubt nicht, daß man die Tonfolge der Zeilen 30-36 (bzw. 23-29) irgendwo bei Mozart finden kann.

```
01♦LBL "MOZ"        11♦LBL              21 TONE   (109)   31♦LBL 01
02 CLST             12 TONE H (109)     22♦LBL 01         32 TONE 0 (80)
03 CF 21            13♦LBL 01           23 TONE 0   (0)   33 TONE 0 (80)
04 ")()MOZART()("   14 TONE 6 (96)      24♦LBL 01         34♦LBL 01
05 SF 25            15 +                25 TONE 2   (2)   35 TONE 6 (66)
06 AVIEW            16♦LBL 01           26♦LBL 01         36 TONE 3 (83)
07 SF 99            17 TONE H (109)     27 TONE 3   (3)   37 "***MOZART***"
08♦LBL 01           18♦LBL 01           28 +              38 PROMPT
09 TONE 6 (96)      19 TONE 0 (0)       29♦LBL 01         39 GTO "MOZ"
10 +                20♦LBL 01           30 TONE 6   (66)  40 .END.
```

(93 Bytes)

```
01♦LBL "MO"         09♦LBL 01           17♦LBL 01         25 TONE 0 (80)
02 TONE 6  (96)     10 TONE H (109)     18 TONE 2 (2)     26 TONE 0 (80)
03♦LBL 01           11♦LBL 01           19♦LBL 01         27♦LBL 01
04♦LBL 01           12 TONE 0 (0)       20 TONE 3 (3)     28 TONE 6 (66)
05 TONE H  (109)    13♦LBL 01           21♦LBL 01         29 TONE 3 (83)
06♦LBL 01           14 TONE H (109)     22♦LBL 01         30 .END.
07 TONE 6  (96)     15♦LBL 01           23 TONE 6 (66)
08♦LBL 01           16 TONE 0 (0)       24♦LBL 01
```

(51 Bytes)

- Quelle: Robert Swanson (5993).

6-12 'Mary has a little lamb' ("MARY")

Dieses hübsche Liedchen verlangt 12 Register. 'XEQ "MARY"', Aufforderung zum 'R/S' abwarten; dann 'R/S' so oft, wie eine Wiederholung gewünscht wird. - Quelle: Bill Kolb (265) (PPC CJ, V7N1P13 & V7N4P13).

```
01♦LBL "MARY"       23 ASTO 08          45 TONE 3     67 TONE 1
02 FS? 55           24 "  A$"           46 XEQ 02     68 VIEW 01
03 CF 21            25 ASTO 09          47 XEQ 01     69 TONE 0
04 "HOLD ON..."     26 " SNOW"          48 X<>Y       70 VIEW 02
05 AVIEW            27 ASTO 10          49 VIEW 05    71 TONE 1
06 " MARY"          28 " "              50 TONE 2     72 VIEW 03
07 ASTO 00          29 ASTO 11          51 VIEW 06    73 TONE 2
08 " HAD"           30 "OK-PRESS R/S"   52 TONE 1     74 TONE 2
09 ASTO 01          31 PROMPT           53 VIEW 07    75 VIEW 04
10 "  A"            32♦LBL 00           54 TONE 1     76 TONE 2
11 ASTO 02          33 XEQ 01           55 VIEW 08    77 RTN
12 "LITTLE"         34 XEQ 02           56 TONE 2     78♦LBL 02
13 ASTO 03          35 VIEW 03          57 VIEW 09    79 X<>Y
14 " LAMB"          36 TONE 1           58 TONE 1     80 X<>Y
15 ASTO 04          37 TONE 1           59 VIEW 10    81 X<>Y
16 " ITS"           38 VIEW 04          60 TONE 0     82 X<>Y
17 ASTO 05          39 TONE 1           61 VIEW 11    83 X<>Y
18 "FLEECE"         40 XEQ 02           62 STOP       84 X<>Y
19 ASTO 06          41 VIEW 03          63 GTO 00     85 END
20 " WAS"           42 TONE 2           64♦LBL 01
21 ASTO 07          43 TONE 3           65 VIEW 00
22 "WHITE"          44 VIEW 04          66 TONE 2
```

(228 Bytes)

6-13 Flag 12 stets bei Bedarf setzen

Flag 12 veranlaßt - wenn gesetzt - den Drucker, in doppelter Breite zu drucken. Da dieses Flag gelöscht wird, wenn man den HP-41 ausschaltet, ist es sinnvoll, Flag 12 stets unmittelbar vor den jeweiligen Druckbefehlen zu setzen statt 'global' am Programmanfang. So vermeidet man, durch unvorhergesehene Unterbrechungen, bei denen der HP-41 ausgeschaltet wird, unerwünschte Normalausdrucke zu erhalten. - Quelle: HP Key Notes, V5N1P7. - Flag 13 sollte ebenso behandelt werden.

6-14 Flag 21

Flag 21 gibt dem Benutzer die Möglichkeit, den Drucker bezüglich seiner Antworten auf '(A)VIEW' zu steuern. Flag 21 wird automatisch dem Zustand von Flag 55 angepaßt, es sei denn, Flag 55 wird synthetisch verändert (vgl. 6-4 und 6-6). Flag 55 ist bei angeschlossenem Drucker gesetzt, andernfalls gelöscht. Sein Zustand läßt sich - außer durch synthetische Eingriffe - nicht beeinflussen. Flag 21 hingegen kann nach Belieben eingestellt werden. Wirkungen von Flag 21 bei '(A)VIEW': *(a) Flag 21 gesetzt:* Die Programmausführung wird beim '(A)VIEW' unterbrochen, es sei denn, der Drucker kann ungehindert arbeiten, weil er angeschlossen und eingeschaltet ist. Dagegen erfolgt eine Unterbrechung beim '(A)VIEW', wenn entweder der Drucker überhaupt nicht angeschlossen oder aber angeschlossen doch ausgeschaltet ist. *(b) Flag 21 gelöscht:* '(A)VIEW' verursacht weder eine Druckausgabe noch eine Programmunterbrechung. Die vermutlich verwirrendste Lage entsteht, wenn man bei angeschlossenem doch ausgeschalteten Drucker auf ein '(A)VIEW' stößt. Das Programm hält 'stur' an. Es gibt 3 'Lösungen': (1) Drucker anschalten und 'R/S'. (2) 'CF 21' und 'R/S'. (3) HP-41 ausschalten, Drucker entfernen, HP-41 wiedereinschalten, 'R/S'. Der Einsatz der Flags 55 und 21 muß sorgfältig geplant werden, damit die beabsichtigten Kombinationen von Anzeige und Druck auch wirklich nach Wunsch arbeiten. - Quelle: HP Key Notes, V5N1P7. Vgl. auch 2-31, 3-12, 4-1, 4-18, 6-4, 6-6, 22-23.

6-15 Beliebiges Flag synthetisch setzen oder löschen ("SET" & "CLR")

Die Routinen rufen das synthetische Umschaltprogramm "IF" (6-4) auf und setzen bzw. löschen das Flag, dessen Nummer in X liegt.

```
'LBL "SET", FC? IND X, XEQ "IF", RTN'        (16 Bytes)
'LBL "CLR", FS? IND X, XEQ "IF", RTN'        (16 Bytes)
```

6-16 Beliebigen synthetischen Ton anhören ("TO" & "TON")

Mit "TO" läßt sich jeder beliebige synthetische Ton anhören. 'n, XEQ "TO"', und 'TONE n' ($0 \leq n \leq 127$) 'erklingt'. Eingabefehler werden von der Routine "TO" korrigiert. Zeile 28 lautet dezimal 243, 127, 206, 124. - "TON" ist wesentlich kürzer, weil der Befehl 'XTOA' aus dem X-Funktionen-Modul verwendet wird. Zeile 02 lautet dezimal 241,

159; Zeile 04 stimmt überein mit Zeile 28 aus "TO". Die Eingabe der Zahl 'n' für den Ton n muß bei "TON" korrekt sein: $0 \leq n \leq 127$.

```
01♦LBL "TO"     09 CHS          17 FS? 07       25 "┣↑↑"
02 ABS          10 X<> d        18 SF 09        26 RCL \
03 INT          11 FS?C 11      19 SF 07        27 STO [
04 128          12 SF 12        20 SF 05        28 "┣¿"
05 MOD          13 FS?C 10      21 SF 04        29  E6
06 LASTX        14 SF 11        22 X<> d        30 X<> b
07 +            15 FS?C 09      23 CLA          31 END
08 OCT          16 SF 10        24 STO [
```

(64 Bytes)

```
01♦LBL "TON"    03 XTOA         05  E6          07 END
02 ""           04 "┣¿"         06 X<> b
```

(22 Bytes)

6-17 'Logisches Und' und 'logisches Oder' für bedingte Verzweigungen

1. Sollen zwei Bedingungen, die durch logisches Und verbunden sind, abgefragt werden, muß die zweite Abfrage invertiert werden. Beispiel: Soll "SUB" genau dann ausgeführt werden, wenn 'X = 0' gilt *und gleichzeitig* Flag 05 gesetzt ist, kann dies mit der Befehlsfolge

'..., X=0? FC? 05, GTO 00, XEQ "SUB", LBL 00,...'

oder mit der logisch gleichwertigen Befehlsfolge

'..., FS? 05, X ≠ 0?, GTO 00, XEQ "SUB", LBL 00,...'

erreicht werden.

2. Sollen zwei Bedingungen, die durch logisches Oder (also 'nichtausschließendes Oder', nicht zu verwechseln mit dem umgangssprachlichen 'entweder-oder', bei dem eine Bedingung die andere ausschließt) verbunden sind, abgefragt werden, muß die erste Abfrage invertiert werden. Beispiel: Soll "SUB" genau dann ausgeführt werden, wenn *mindestens eine* der beiden Bedingungen 'X = 0' oder 'Flag 05 gesetzt' ist erfüllt ist (logisches Oder), kann dies mit der Befehlsfolge

'..., X ≠ 0?, FS? 05, XEQ "SUB",...'

oder mit der logisch gleichwertigen Befehlsfolge

'..., FC? 05, X=0?, XEQ "SUB",...'

erreicht werden. - Quelle: HP Key Notes V5N2P12.

Kapitel 7

UMGANG MIT DEM STAPEL

7-1 X operiert auf X

'ST + X' verdoppelt den Inhalt von X. Es ist schneller als '2, *' und läßt den Rest des Stapels unberührt. *'ST - X'* wirkt wie 'CLX', veranlaßt aber die Anhebung des Stapels bei der nächsten Eingabe; es verbraucht jedoch 1 Byte mehr. *'ST * X'* wirkt wie 'X↑2', verbraucht 1 Byte mehr, läßt aber L unberührt. *'ST / X'* ersetzt den Inhalt von X durch '1' (solange er ≠ 0 ist) und läßt den Rest des Stapels unversehrt. - Quelle: Valentin Albillo (4747).

7-2 Inhalt von X bei gelöschtem Flag mit/durch Konstante multiplizieren/dividieren

gute Lösung: '..., FS? 00, GTO 14, 5, * (oder /), LBL 14, ...'
bessere Lösung: '..., 5, FC? 00, * (oder /), FS? 00, RDN, ...'
beste Lösung: '..., 5, FS? 00, SIGN, * (oder /), ...',
(letzte Lösung mit der Beschränkung auf positive Konstanten).

7-3 x-y ins X-Register bei unverändertem Y-Register

(a) '%CH, %'; Rundungsfehler sind möglich. (b) 'RCL Y, -'; der Inhalt von T geht verloren. - Quelle: Joseph Horn (1537) (PPC CJ, V7N4P13).

7-4 RCL X, Y, Z oder T

Alle 4 Funktionen belegen 2 Bytes. *'RCL X'* wirkt wie 'ENTER' (XYZT→XXYZ), doch wird beim nächsten Eintrag der Stapel angehoben. *'RCL Y'* wirkt ähnlich wie 'X<>Y' (XYZT→YXYZ) doch geht T verloren. *'RCL Z'* (XYZT→ZXYZ) beseitigt ebenfalls den Inhalt von T. *'RCL T'* wirkt wie 'R↑' (XYZT→TXYZ), außer nach einem 'ENTER' oder 'CLX', wonach der Inhalt von X überschrieben wird. - Vgl. auch 7-11.

7-5 Den Inhalt von X in '1' ändern, ohne den Stapel zu beeinflussen

'ST/X' bewirkt das Beabsichtigte für jede Zahl außer für '0'. *'SIGN, ABS'* ändert *jede* Zahl in '1', ersetzt aber den Inhalt von L durch das Ergebnis von 'SIGN'. - Quelle: PPC Melbourne Chapter.

7-6 X und Y durch 10 dividieren

ordinär: '10, /, X<>Y, LAST X, /, X<>Y'; elegant: '10, ST/ Z, /'.

7-7 Stapel-Analyse ("SA")

Die Routine "SA" kann dazu benutzt werden, die Wirkung einer anderen Routine auf den Stapel sichtbar zu machen. Zunächst 'XEQ "SA"', dann die zu testende Routine (z.B. 'LASTX, STO T, X < > Y') ausführen. Anschließend Stapel betrachten. - Vgl. auch 7-11.

```
01♦LBL "SA"     04 "T"        07 ASTO Z     10 "X"
02 "L"          05 ASTO T     08 "Y"        11 ASTO X
03 ASTO L       06 "Z"        09 ASTO Y     12 END
```

(29 Bytes)

7-8 Automatischer Blick in den Stapel ohne Drucker ("ST")

'XEQ "ST"' (vorab 'CF 21', falls der Drucker angeschlossen ist), und zwar zu beliebigem Zeitpunkt, um den Inhalt des Stapels, L eingeschlossen, zu betrachten. "ST" benutzt das Alpha-Register.

```
01♦LBL "ST"     07 ARCL X     13 PSE        19 ARCL T
02 "L= "        08 AVIEW      14 "Z= "      20 AVIEW
03 ARCL L       09 PSE        15 ARCL Z     21 PSE
04 AVIEW        10 "Y= "      16 AVIEW      22 CLD
05 PSE          11 ARCL Y     17 PSE        23 END
06 "X= "        12 AVIEW      18 "T= "
```

(50 Bytes)

- Quelle: Bruce Clark (5795).

7-9 Stapel-Austausch, -Ablage und -Rückruf ("STX", "STS" & "STR")

"STX" vertauscht die Inhalte von L, X, Y, Z und T mit denen der Register R_{00}-R_{04}. "STS" legt die Inhalte von L bis T in R_{00}-R_{04} ab. "STR" bring die Inhalte der Register R_{00}-R_{04} nach L bis T.

```
01♦LBL "STX"    10 RDN        19 RDN        28 RCL 00
02 X<> L        11 X<> 04     20 STO 02     29 STO L
03 X<> 00       12 RDN        21 RDN        30 RCL 04
04 X<> L        13 RTN        22 STO 03     31 RCL 03
05 X<> 01       14♦LBL "STS"  23 RDN        32 RCL 02
06 RDN          15 X<> L      24 STO 04     33 RCL 02
07 X<> 02       16 STO 00     25 RDN        34 END
08 RDN          17 X<> L      26 RTN
09 X<> 03       18 STO 01     27♦LBL "STR"
```

(64 Bytes)

- Quelle: Bill Carter (2998) (PPC CJ, V7N7P15).

7-10 Indirektes Ablegen und Rückrufen des Stapelinhaltes ("SM" & "MS")

"SM" legt den Inhalt des Stapels (X, Y, Z, T und L) in den 5-Register-Block, der durch den Wert in R_{00} bestimmt wird. Der Inhalt von L bleibt dabei in L liegen, der Rest des Stapels geht jedoch verloren (kann aber mit dem folgenden "MS" wiedergewonnen

werden). "MS" ruft den Inhalt des durch den Wert in R_{00} bestimmten 5-Register-Blocks in den Stapel zurück, und zwar in der Ordnung X, Y, Z, T und L. "MS" kann insbesondere nach einem zuvor erfolgten "SM" aufgerufen werden.

```
01♦LBL "SM"     09 ST- 00       17 RTN          25 DSE 00
02 XEQ c        10 RTN          18♦LBL "MS"     26 RCL IND 00
03 XEQ c        11♦LBL c        19 4            27 DSE 00
04 XEQ c        12 STO IND 00   20 ST+ 00       28 RCL IND 00
05 XEQ c        13 RDN          21 RCL IND 00   29 DSE 00
06 LASTX        14 1            22 SIGN         30 STO X
07 STO IND 00   15 ST+ 00       23 DSE 00       31 RCL IND 00
08 4            16 RDN          24 RCL IND 00   32 END
```

(68 Bytes)

- Quelle: PPC CJ, V7N10P7 (PPC ROM).

7-11 Stapel-Umordnungen ("STACK")

Diese Routine kann dazu benutzt werden, die Wirkung verschiedener Kombinationen von Stapelfunktionen ('X < > Y, RCL T, STO Z, RDN' usw.) zu beobachten. 'XEQ "STACK"' zeigt "X - Y - Z - T" an und füllt die jeweiligen Stapelregister mit ihren Namen. Dann werden die Stapel-Funktionen aufgerufen. Anschließend zeigt 'R/S' ihre Wirkung (z.B. "YXZT" nach einem 'X < > Y'). Mit 'R/S' beginnt man von vorn. Bei angeschlossenem Drucker muß vorab 'CF 21' ausgeführt werden. Zur schnelleren Ausführung kann man Stapelfunktionen wie 'X < > _ _' und 'R↑' Tasten zuweisen. ('X < > Y' und 'RDN' werden übrigens im USER-Modus schneller ausgeführt, wenn sie ihren eigenen Tasten zugewiesen werden. Vgl. 1-24). - Quelle: John Dearing (2791). - Vgl. auch 7-7.

```
01♦LBL "STACK"  06 "Z"          11 PROMPT       16 ARCL T
02 "X"          07 ASTO Z       12 CLA          17 AVIEW
03 ASTO X       08 "T"          13 ARCL X       18 END
04 "Y"          09 ASTO T       14 ARCL Y
05 ASTO Y       10 "X-Y-Z-T"    15 ARCL Z
```

(47 Bytes)

Spielart 1 (59 Bytes): Setzt man hinter Zeile 17 die Befehle 'PSE, PSE, SF 11, OFF, GTO "STACK"', zeigt der Rechner zwei Pausen lang die Umordnung, schaltet sich dann aus und beginnt die neue Runde auf 'ON' hin.

Spielart 2 (50 Bytes): Wenn Sie die Wirkung auf den Stapel einzelschrittweise verfolgen wollen, ersetzen Sie Zeile 17 ('AVIEW') durch 'PROMPT, GTO 00' und fügen hinter Zeile 11 ('PROMPT') 'LBL 00' ein. Dann können Sie im Anschluß an jeden einzelnen Befehl aus einer Folge von Stapelfunktionen die neue Zusammensetzung mit 'R/S' abrufen. Die folgende Tabelle gibt Ihnen einige Möglichkeiten an die Hand, den Stapel nach Wunsch umzuordnen (eine vollständige Tabelle finden Sie unter 27-3). Die Funktionen, die zur Umordnung benutzt werden können, sind: R↑, RDN, ENTER, X < > Y, X < > Z, X < > T, STO Y, STO Z, STO T, RCL X, RCL Y, RCL Z und RCL T. In den meisten Fällen gibt es

mehrere gangbare Wege, eine bestimmte Neuordnung zu erreichen. Gewöhnlich wählt man den 'byteärmsten' Weg, also z.B. 'R↑' (1 Byte) statt 'RCL T' (2 Bytes).

Das Zeichen '-' steht für den Registeraustausch ('X - Y' z.B. bedeutet 'X < > Y').

XYZT	(Ausgangsordnung)	YXZT	X-Y	ZXYT	X-Y, X-Z	TXYZ	R↑
XYTZ	X-Z, RDN, X-Y	YXTZ	X-Z, RDN	ZXTY	X-Y, RDN, X-Y	TXZY	X-Y, X-T
XZYT	RDN, X-Y, R↑	YZXT	X-Z, X-Y	ZYXT	X-Z	TYXZ	X-Y, R↑
XZTY	X-Y, RDN	YZTX	RDN	ZYTX	RDN, X-Y	TYZX	X-T
XTYZ	R↑, X-Y	YTXZ	RDN, X-Y, RDN	ZTXY	RDN, RDN	TZXY	RDN, RDN, X-Y
XTZY	RDN, RDN, X-Z	YTZX	X-T, X-Y	ZTYX	X-Y, RDN, RDN	TZYX	RDN, X-Z

7-12 x*y oder x+y ohne Veränderung des Wertes in L

'ST* (bzw. +) Y, RDN' multipliziert (bzw. addiert) die Inhalte von X und Y, ohne den Inhalt von L zu verändern. Nützlich, wenn L z.B. eine Kontrollzahl enthält, die nur durch Inkrementfunktionen verändert werden darf. - Quelle: Kiyoshi Akima (3456).

Kapitel 8

SPEICHER UND 'VORHANG'

8-1 Synthetisches Versetzen des Vorhangs ("CU")

"CU" nimmt die Zahl 'n' aus dem X-Register und addiert sie auf die im Register c liegende Adresse von R_{00}. Wenn 'n' positiv ist, werden die Datenregister R_{00} bis R_{n-1} zu Programmregistern, weil der imaginäre 'Vorhang', der im Speicher die Daten von den Programmen trennt, aus seiner Anfangslage unterhalb von R_{00} an eine neue Stelle unmittelbar unterhalb von R_n versetzt wird; R_n wird zum neuen R_{00}. Wenn 'n' negativ ist, wandert der Vorhang im Speicher nach unten, so daß 'n' bisherige Programmregister zu Datenregistern werden. All dies geschieht ohne Veränderung oder Bewegung der Inhalte der in den Vorgang einbezogenen Register.

Um Speicherplatz zu sparen, ist es sinnvoll, Programme auf die Datenregister R_{00}-R_{15} zugreifen zu lassen. Daher ist es die Regel, mehrere Programme im Speicher zu haben, deren jedes den gleichen Block von Datenregistern benutzt, so daß die Ausführung eines Programms im allgemeinen die von anderen Programmen verwendeten oder abgelegten Daten zerstört. "CU" löst dieses Problem.

Sie geben eine ganze Zahl 'n' ins X-Register und führen "CU" aus. Für positives 'n' wird R_n das neue R_{00} (Vorhang nach oben verschoben); für negatives 'n' wird R_{00} das neue R_{-n} (Vorhang nach unten verschoben). Sämtliche anderen Datenregister-Nummern werden sinngemäß verschoben.

Beispiel: Nehmen Sie an, es werde ein 'Programm 1', welches in R_{00}-R_{50} Daten für spätere Verarbeitung zurückläßt, ausgeführt. Doch zwischenzeitlich soll ein 'Programm 2', welches sich der Register R_{00}-R_{25} bedienen will, laufen. Damit 'Programm 2' nichts anrichtet, wird vorher '51, XEQ "CU"' ausgeführt (die Mindestanzahl von Datenregistern muß 77 betragen). Nach Ausführung von 'Programm 2' kann der Rechner mit '-51, XEQ "CU"' auf einen zweiten Lauf von 'Programm 1' vorbereitet werden.
***WARNUNG**:* Der Versuch, den Vorhang über den Kopf des Speichers hinaus zu versetzen ('n' größer als die Anzahl der verfügbaren Datenregister) oder auf Adressen vom Hexadezimalwert kleiner als '0C0', verursacht "MEMORY LOST".

Der HP-41 arbeitet auch dann normal, wenn die durch ein 'SIZE' festgelegte Stellung des Vorhangs durch "CU" verändert wurde. Allerdings darf der Speicher bei nach oben versetztem Vorhang nicht durch 'PACK' verdichtet werden, weil dies im allgemeinen die unterhalb des Vorhangs abgelegten Daten unwiderruflich durch Beseitigen der Null-Bytes zerstört. Diese Gefahr kann durch ein vorher an den Kopf des Programmspeichers gesetztes 'END', gefolgt von einem 'PACK' gebannt werden. Wenn der Vorhang anschlie-

ßend nach oben versetzt wird, sind die entstehenden Programmregister gegen 'PACK's durch das 'END', welches ein verdichtetes Programm vortäuscht, geschützt.

```
01♦LBL "CU"      13 X<> L       25 FC?C IND Y      37 GTO 00
02 ABS           14 INT         26 SF IND Y        38♦LBL 14
03 RDN           15 X=0?        27 FC? IND Y       39 X<> J
04 RCL c         16 GTO 14      28 CHS             40 X<> d
05 STO [         17 2           29 X>0?            41 STO [
06 "^♦♦♦♦"       18 /           30 GTO 13          42 "ÞABC"
07 11            19 RCL [       31 FC? IND Y       43 X<> \
08 X<> [         20 X<>Y        32 CHS             44 X<> c
09 X<> d         21 FRC         33 DSE Y           45 RDN
10 STO J         22 X=0?        34 GTO 01          46 CLA
11♦LBL 00        23 GTO 13      35♦LBL 13          47 END
12 "ÞA"          24♦LBL 01      36 DSE [
```

(87 Bytes)

- Quelle: William Wickes (3735) ('Synthetische Programmierung auf dem HP-41C/CV', S. 110/111) (PPC ROM).

8-2 Beschränkungen beim Löschen von Programmen

Wenn Sie ein längeres Programm (von mehr als 233 Zeilen) löschen wollen, müssen Sie den Drucker - sofern er angeschlossen ist - in den MAN-Modus schalten, bevor Sie 'CLP' aufrufen. Programme von mehr als 1089 Zeilen müssen mit 'DEL' gelöscht werden. Beispiel: 'DEL EEX 980' löscht ein Programm von 1980 Zeilen; das 'END' bleibt allerdings zurück. - Quelle: HP Key Notes, V4N1P3. - Für den HP-IL-Drucker bestehen keine Beschränkungen.

8-3 Ermitteln der freien Register

Im Anschluß an den letzten eingetasteten Befehl eines Programms kann man die verbleibenden freien Register mit 'SST' ermitteln: die Anzeige weist '.END. REG lmn' vor, worin 'lmn' die gesuchte Anzahl ist. Das nächste 'SST' setzt dann den Zeiger auf die Zeile 01 des gegenwärtigen Programms, ein 'BST' jedoch auf die letzte Programmzeile, so daß weitere Befehle angefügt werden können. Von beliebiger Stelle eines beliebigen Programms aus können Sie die noch verfügbare Register-Anzahl mit 'GTO.000' ermitteln: die Anzeige weist in diesem Fall '00 REG lmn' vor, und der Zeiger steht auf Zeile 00 des Programms. - Quelle: HP-41 Bedienungshandbuch, © Copyright (Juni 1980) Hewlett-Packard Company. Abgedruckt mit Erlaubnis.

8-4 Synthetischer Zugriff auf das '.END.' ("EN")

Gewöhnlich ist ein Programm, welches gerade geschrieben wird, das letzte im Programmspeicher und enthält somit das ständige '.END.'. Wird der Adreßzeiger zwischendurch auf ein anderes Programm gesetzt, gibt es nur zwei Möglichkeiten, ihn wieder auf das letzte zurückzuführen: die Verwendung von 'GTO' zusammen mit dem 'Buchstabieren' ei-

ner globalen Marke dieses Programms (sofern eine solche Marke vorhanden ist) oder den Aufruf von 'CAT 1', um an das Ende des Programmspeichers zu gelangen (zeitraubend, falls viele globale Marken vorhanden sind). 'XEQ "EN"' eröffnet eine dritte Möglichkeit: der Adreßzeiger wird auf Zeile 00 des Programms gesetzt, welches das '.END.' enthält.

```
01♦LBL "EN"      06 X<> d      11 X<> d      16 STO b
02 RCL c         07 CF 00      12 CLA        17 END
03 STO [         08 CF 01      13 STO [
04 "⊢♦♦♦♦♦"      09 SF 02      14 "⊢♦♦"
05 X<> [         10 SF 03      15 X<> \
```

(45 Bytes)

- Quelle: William Wickes (3735) ('Synthetische Programmierung auf dem HP-41C/CV, S. 91).

8-5 Synthetischer Sucher für den 'Vorhang' ("C?")

'XEQ "C?"' ermittelt die gegenwärtige Lage des Vorhangs (die absolute Adresse von R_{00}).
- Zeile 04 lautet dezimal 244, 127, 0, 0, 65.

```
01♦LBL "C?"      09 CF 04      17 FS?C 13    25 X<> d
02 RCL c         10 CF 07      18 SF 11      26  E38
03 STO [         11 FS?C 10    19 FS?C 14    27 /
04 "⊢♦♦A"        12 SF 07      20 SF 13      28 INT
05 X<> [         13 FS?C 11    21 FS?C 15    29 DEC
06 X<> d         14 SF 09      22 SF 14      30 END
07 CF 01         15 FS?C 12    23 FS?C 16
08 CF 02         16 SF 10      24 SF 15
```

(66 Bytes)

- Quelle: Keith Jarett (4360), Roger Hill (4940) & Clifford Stern (4516) (PPC CJ, V7N10P15; PPC ROM). Vgl. auch 1-17.

8-6 Synthetisches Verschieben von Datenregisteradressen; Versetzen des Vorhangs mit 'ΣREG' ("HD", "UD" & "ΣC")

Die Mindestanzahl verfügbarer Datenregister muß 6+k betragen. Mit 'k, XEQ "HD"' wird dann der 'Vorhang' um k Register nach oben versetzt. In das vorherige R_k (jetzt R_{00}) gelangt eine Alpha-Kette. Um den Vorhang in die ursprüngliche Lage zurückzubringen, genügt 'XEQ "UD"'. Anschließend steht die Alpha-Kette in R_k (also dem Register, das bei nach oben versetztem Vorhang R_{00} war). ["HD" benutzt die Routine "ΣC", welche die Adressen von 'ΣREG' und R_{00} im Register c vertauscht, welche also die Nummer des ersten Statistikregisters als Bestimmungszahl für die Versetzung des Vorhangs benutzt. Laden Sie "Σ?" (aus 1-17), um festzustellen, wie "ΣC" arbeitet: 'SIZE 010, ΣREG 03, XEQ "ΣC", XEQ "S?"' (Sie sehen 7), 'XEQ "Σ?"' (Sie sehen -3), 'XEQ "ΣC", XEQ "S?"' (Sie sehen 10), 'XEQ "Σ?"' (Sie sehen 3).] Das Beispiel unten links zeigt, wie "HD" mit k=5 und

Datenregisteranzahl 11 arbeitet. Nach Versetzen des Vorhangs ist R_{05} zu R_{00} geworden (es enthält die erwähnte Alpha-Kette), R_{06} zu R_{01} usw. Die Werte der alten Register R_{00}-R_{04} sind in den Programmspeicher 'getaucht'. Nach 'XEQ "UD"' hat der Vorhang seine alte Lage. Die 5 ersten Werte sind 'wieder da', doch in R_{05} bleibt die Alpha-Kette zurück, und es wurde 'ΣREG 01' festgelegt.

vorher		nachher
R_{10}	= 10.00	= R_{05}
R_{09}	= 9.00	= R_{04}
R_{08}	= 8.00	= R_{03}
R_{07}	= 7.00	= R_{02}
R_{06}	= 6.00	= R_{01}
R_{05}	= Alpha-Kette	= R_{00}
R_{04}	= 4.00	
R_{03}	= 3.00	
R_{02}	= 2.00	
R_{01}	= 1.00	
R_{00}	= 0.00	

```
01♦LBL "HD"          17 "┣A"          33 RDN           49 "┣K"
02 SIGN              18 CLX           34 RCL c         50 X<> L
03 RDN               19 STO \         35 ΣREG 00       51 STO [
04 RCL c             20 "┣B"          36 X<> c         52 "┣L"
05 "0"               21 STO [         37 STO [         53 X<> \
06 X<> [             22 "┣CD"         38 "┣F"          54 X<> c
07 "┣**"             23 X<> ]         39 RDN           55 RDN
08 STO \             24 X<> d         40 RCL \         56 CLA
09 RDN               25 SF 08         41 "┣GHI"        57 RTN
10 ASTO IND L        26 X<> d         42 STO L         58♦LBL "UD"
11 ΣREG IND L        27 X<> \         43 RDN           59 CLA
12♦LBL "ΣC"          28 "┣E"          44 RCL c         60 ARCL 00
13 CLA               29 X<> d         45 STO [         61 ASTO c
14 RCL c             30 SCI IND \     46 X<> ]         62 ΣREG 01
15 STO ]             31 X<> d         47 "┣J"          63 END
16 STO [             32 STO ]         48 STO [
```

(141 Bytes)

- Quelle: Keith Jarett (4360) (PPC ROM).

Kapitel 9

DATENREGISTER

9-1 Datenregister laden und betrachten ("LD" & "RV")

'XEQ "LD"', um Daten zu speichern; Sie werden zur Eingabe, beginnend mit R_{01}, aufgefordert. 'XEQ "RV"' zum Betrachten der Inhalte von R_{01} an. R_{00} ist Steuerregister für beide Routinen. Bei "RV" erfolgt die Ausgabe druckend, sofern das möglich ist; sie wird mit 'R/S' oder durch ein "NONEXISTENT" unterbrochen.

```
01♦LBL "LD"    09 "├=?"        17 1,4         25 FIX 2
02 1,4         10 PROMPT       18 STO 00      26 SF 29
03 STO 00      11 STO IND 00   19♦LBL 02      27 ARCL IND 00
04 FIX 0       12 ISG 00       20 FIX 0       28 AVIEW
05 CF 29       13 GTO 01       21 CF 29       29 ISG 00
06♦LBL 01      14 RTN          22 "R"         30 GTO 02
07 "R"         15♦LBL "RV"     23 ARCL 00     31 END
08 ARCL 00     16 SF 21        24 "├="
```

(69 Bytes)

- Quelle: John Dearing (2791) (PPC CJ, V7N4P7).

9-2 Registerinhalt aufrufen und auf 0 oder 1 setzen

Auf 0 setzen mit 'RCL nm, ST- nm'; auf 1 setzen mit 'RCL nm, ST/ nm' (solange der Inhalt von R_{nm} nicht 0 ist). - Quelle: Bill Kolb (265) (PB 67/97). - Schnellere Lösung ohne Einschränkung des Wertebereichs: '0 (oder 1), X< >nm'. - Quelle: PPC Melbourne Chapter.

9-3 0 - 1 Umschaltungen

Um den Inhalt von R_{nm} von 0 in 1 bzw. von 1 in 0 umzuändern: '..., RCL nm, 1, -, ABS, STO nm, RDN,...'. - Quelle: Joseph Holmes (3673) (PPC CJ, V7N5P7).

9-4 'XEQ IND' für die Wiedergewinnung von Information ("PHONE")

Im folgenden Beispiel wird eine Telefon-Nr. als Antwort auf einen Namen (von höchstens 6 Zeichen) ausgegeben. 'XEQ "PHONE"', Aufforderung mit dem Namen beantworten, 'R/S', Ausgabe der Telefon-Nr.

```
01♦LBL "PHONE" 06 ASTO X      11 "533-555-1212" 16♦LBL "FELIX"
02 "NAME?"     07 XEQ IND X   12 RTN            17 "222-2791"
03 AON         08 AVIEW       13♦LBL "TILL"     18 END
04 PROMPT      09 STOP        14 "753-555-6767"
05 AOFF        10♦LBL "JULJA" 15 RTN
```

(90 Bytes)

- Quelle: HP Key Notes, V4N1P11.

9-5 'STO' und anschließender Registerarithmetik-Befehl

'STO mn, ST+ mn': legt in 2 Schritten das Doppelte des Inhalts von X in R_{mn}, ohne den Stapel zu verändern.

'STO mn, ST- mn': löscht R_{mn}, ohne den Stapel zu verändern.

'STO mn, ST mn'*: legt das Quadrat des Wertes in X nach R_{mn}, ohne den Stapel zu verändern.

'STO mn, ST/ mn': legt, sofern der Wert in X nicht 0 ist, 1 nach R_{mn}, ohne den Stapel zu verändern.

9-6 Löschen von Datenregistern mit höheren Adressen

Es ist zu empfehlen, bei der Benutzung von Datenregistern, die wichtigsten Daten in Register mit niedrigeren Adressen zu legen. Notizen hingegen in höher adressierte Register. Man kann dann ohne Schaden die oberen Register mit 'SIZE' löschen: 'SIZE 20, SIZE 41' z.B. löscht die Register R_{20} bis R_{40}. - Quelle: Richard Nelson (1).

9-7 Verdichten und Auflösen von Registerinhalten ("PR" & "UR")

"PR" kann dazu verwendet werden, Zahlen in verdichteter Form in Datenregistern abzulegen. "UR" löst die Verdichtung auf und stellt die ursprüngliche Darstellung der Zahlen wieder her. Die Zahlen werden bezüglich einer (frei zu wählenden) Basis b dargestellt. Diese Technik erlaubt es, mehrere Zahlen in einem Register unterzubringen. Beide Routinen erwarten in R_{10} die Basis b und in R_{11} die Adresse des Registers, das die verdichteten Zahlen aufnehmen soll bzw. dessen Inhalt aufzulösen (zu 'entschlüsseln') ist.

R_{10}: Basis b R_{11}: Registeradresse

Um z.B. die Zahl 'n' an k-ter Stelle (Position k) in R_{pq} (pq in R_{11}) abzulegen, muß 'n, ENTER, k, XEQ "PR"' befohlen werden; 'n' unterliegt dabei der Einschränkung $0 \leq n \leq b-1$. "PR" ruft "UR" auf und hinterläßt 'Schutt' im Stapel. Um die in R_{pq} (pq in R_{11}) auf Position k stehende Zahl wiederzugewinnen, muß 'k, XEQ "UR"' befohlen werden. "UR" legt die abgerufene Zahl ins X-Register.

Tabelle der zulässigen Basen und Positionen

Datenbereich	Basis b	Pos.-Nr.	Datenbereich	Basis b	Pos.-Nr.
0-1	2	1-30	0-20	21	1-7
0-2	3	1-19	0-36	37	1-6
0-3	4	1-15	0-99	100	1-5
0-4	5	1-13	0-214	215	1-4
0-6	7	1-11	0-1413	1414	1-3
0-9	10	1-10	0-99999	100000	1-2
0-13	14	1-8			

Den größten Nutzen zieht man aus diesem Verfahren, wenn die größten Werte auf die niedrigsten Positionen und die kleinsten Werte auf die höchsten Positionen gelegt

werden. In vielen Fällen kann man dann über die Werte der Tabelle hinausgehen. Wenn es für Sie auf den Datenbereich ankommt, richten Sie sich bei Ihrer Wahl von b nach der ersten Spalte; wenn die Anzahl der 'künstlichen Speicher' für Sie von größerem Wert ist, richten Sie sich bei der Wahl von b nach der dritten Spalte (Positionen). Bei gegebener Basis b lautet die Formel für die Anzahl der mit Sicherheit zur Verfügung stehenden Positionen

$$\text{INT}\left[\frac{\ln(b \cdot 999.999.999)}{\ln b}\right]$$

Beispiel: der Tabelle läßt sich entnehmen, daß man im Falle b = 21 bis zu 7 Zahlen in ein Datenregister legen kann, vorausgesetzt, diese Zahlen liegen im Bereich 0 bis 20. Benutzen Sie "PR", um die Zahlen 13, 19, 14, 15, 8, 18 und 16 in R_{12} unterzubringen. Holen Sie sie anschließend mit "UR" ins X-Register zurück. Zunächst also: '21, STO 10, 12, STO 11'; dann '13, ENTER, 1, XEQ "PR", 19, ENTER, 2, XEQ "PR"' usw. bis '16, ENTER, 7, XEQ "PR"'. Ein 'RCL 12' zeigt Ihnen die Zahl 1.447.473.103. Wenn Sie diese Zahl zur Basis 21 darstellen, ergibt sich, daß die abgelegten Zahlen gerade die Koeffizienten der Potenzen von 21 sind:

$$1.447.473.103 = 16*21^6 + 18*21^5 + 8*21^4 + 15*21^3 + 14*21^2 + 19*21^1 + 13.$$

Benutzen Sie nun "UR" für die Wiedergewinnung der Koeffizienten: '1, XEQ "UR"' legt '13' ins X-Register, '2, XEQ "UR"' legt 19 ins X-Register, usw. Ausführungszeit für "PR" 2 Sekunden, für "UR" 1 Sekunde.

```
01♦LBL "UR"    07 RCL IND 11   13 MOD         19 *
02 1           08 X<>Y         14 RTN         20 ST- IND 11
03 -           09 ST/ Y        15♦LBL "PR"    21 X<>Y
04 RCL 10      10 X<>Y         16 XEQ "UR"    22 ST+ IND 11
05 X<>Y        11 INT          17 X<>Y        23 END
06 Y↑X         12 RCL 10       18 ST* Z
```

(43 Bytes)

- Quelle: John Kennedy (918) (PPC ROM).

9-8 Doppel-Speicherung

Wenn zwei Zahlen vorliegen - vorzugsweise eine größer als 1 und eine kleiner als 1 - und zugleich keine Datenregister 'zu verschenken' sind, Programmspeicherplatz aber noch zur Verfügung steht, dann speichern Sie beide Zahlen in einem Register. Z.B. können '127' und '0.35' als '127.35' zusammen in R_{00} abgelegt werden. Zur Rückgewinnung benutzt man 'RCL 00, INT' bzw. 'RCL 00, FRC'. Sind beide Zahlen größer (oder kleiner) als 1, müssen sie durch Stellenverschiebung 'zurechtgestutzt' werden. - Quelle: John Martellaro (1896) (PPC J, V5N1P16).

9-9 Uneingeschränkter Registeraustausch und volle Registerarithmetik

Die folgenden Regeln sind auf alle numerischen und Stapelregister anwendbar.

Inhalt zweier beliebigen Register R und S austauschen ('R <> S'):

Regel: 'X<>R, X<>S, X<>R'.

Beispiel: 'X<>13, X<>15, X<>13' tauscht den Inhalt von R_{13} gegen den Inhalt von R_{15} aus.

*Register R mit 'STO' ('ST+, -, *, /') ins Register S bringen:*

Regel: 'X<>R, STO(ST+, -, *, /) S, X<>R'.

Beispiel: 'X<>11, ST+ Z, X<>11' addiert den Inhalt von R_{11} auf Z.

*Register R mit 'ST+' ('ST-, *, /') ins Register X bringen:*

Regel: 'X<>R, ST+(ST-, *, /) R, X<>R'.

Beispiel: 'X<>20, ST/ 20, X<>20' dividiert den Inhalt von X durch den Inhalt von R_{20}.

- Quelle: John Dearing (2791).

9-10 Register R in Register S (Adresse in X) kopieren

(a) 'RCL R, STO IND Y, RDN'. Der Inhalt von T wird dabei durch den Inhalt von R ersetzt. (b) 'X<>R, STO IND R, X<>R'. Diese Methode läßt den Stapel unberührt. - Beispiel: in X liege 3, dann wird der Inhalt von R_{04} (er sei 7) durch 'RCL 04, STO IND Y, RDN' oder durch 'X<>04, STO IND 04, X<>04' in R_{03} kopiert:

X	3	7	7	3
R_{04}	7	3	3	7
R_{03}	?	?	7	7

9-11 Indirektes Speichern und Rückrufen ("SI" & "RI")

Diese Routinen arbeiten ähnlich wie 'STO' und 'RCL', doch sind sie auf alle Datenregister, also auch von R_{100} an aufwärts anwendbar. *"SI"*: abzuspeichernde Zahl eintasten, 'ENTER', Registernummer eintasten, 'XEQ "SI"'; T und L werden benutzt. *"RI"*: Registernummer eintasten, 'XEQ "RI"'; L wird benutzt.

'LBL "SI", SIGN, RDN, STO IND L, RTN, LBL "RI", SIGN, RDN, RCL IND L, END'

(24 Bytes)

Kapitel 10

BLOCK-OPERATIONEN

10-1 Alle Register eines Blockes mit demselben Wert laden

Man lege 'abc.lmn' nach Y und den Wert nach X. Beispiel: '1.013, ENTER, 8, LBL 14, STO IND Y, ISG Y, GTO 14' lädt die Register R_{01}-R_{13} mit '8'.

10-2 'Eigen-Ladung' ("SLD")

Diese Routine lädt jedes numerische Register mit seiner eigenen Adresse. Der Ladevorgang wird mit 'R/S' oder durch eine "NONEXISTENT"-Meldung (Register nicht vorhanden) unterbrochen. "SLD" ist nützlich beim Testen vieler Block-Operationen. - Quelle: John Dearing (2791) (PPC CJ, V7N5P7). Vgl. auch 10-27.

'LBL "SLD", 0, LBL 00, STO IND X, 1, +, GTO 00, RTN' (16 Bytes)

10-3 Block-Eingabe ("INBL")

Diese Routine fordert zur Eingabe von Daten für einen Registerblock auf. Der Block wird durch Eingabe einer Kontrollzahl 'abc.lmn' in X festgelegt; dann 'XEQ "INBL"'.

```
01♦LBL "INBL"     07 X=0?         13 "⊢=?"        19 FIX 2
02 FIX 0          08 GTO 13       14 PROMPT       20 SF 29
03 CF 29          09♦LBL 14       15 STO IND Z    21 END
04 ENTER↑         10 "R"          16 RDN
05 INT            11 ARCL Y       17 ISG Y
06 "R0"           12♦LBL 13       18 GTO 14
```

(45 Bytes)

- Quelle: John Dearing (2791). Vgl. auch 9-1.

10-4 Synthetische Block-Eingabe ("INB")

Die Routine fordert zur Eingabe von Daten für einen durch 'abc.lmn' (in X) festgelegten Registerblock auf. Der Anzeigemodus bleibt erhalten.

```
01♦LBL "INB"      07♦LBL 14       13 STO d        19 RDN
02 ENTER↑         08 "R"          14 RDN          20 ISG Y
03 INT            09 RCL d        15♦LBL 13       21 GTO 14
04 "R0"           10 FIX 0        16 "⊢=?"        22 CLX
05 X=0?           11 CF 29        17 PROMPT       23 END
06 GTO 13         12 ARCL Z       18 STO IND Z
```

(46 Bytes)

- Quelle: John Dearing (2791).

10-5 Verbesserte synthetische Block-Eingabe ("IB")

Die den Registerblock bestimmende Steuerungszahl 'abc.lmn' muß vor Aufruf von "IB" in X liegen; sie wird anschließend nach X zurückgegeben. Bei Bedarf kann zur Eingabe von Zeichen der Alpha-Modus gewählt werden. Wechsel von numerischer und Alpha-Eingabe ist möglich. Der HP-41 verbleibt in dem vom Benutzer gewählten Anzeige-Modus.

```
01♦LBL "IB"     09 RCL d      17 ENTER↑      25 RDN
02 ENTER↑       10 FIX 0      18 ASTO X      26 FS? 23
03 INT          11 CF 29      19 CF 23       27 ASTO IND Y
04 "R0"         12 ARCL Z     20 PROMPT      28 ISG Y
05 X=0?         13 STO d      21 FC? 23      29 GTO 14
06 GTO 13       14 RDN        22 STO IND T   30 LASTX
07♦LBL 14       15♦LBL 13     23 FC? 23      31 AOFF
08 "R"          16 "├=?"      24 RDN         32 END
```

(60 Bytes)

- Quelle: John Dearing (2791).

10-6 Doppelt inkrementierte Block-Eingabe ("BI")

Vor Aufruf von "BI" muß die den Block bestimmende Kontrollzahl 'abc.lmnik' (ik inkrementiert die Registeradressen) in Z liegen, der Eingabewert für das erste Register des Blocks in Y und das Inkrement (positiv oder negativ) für die weiteren Eingabewerte in X. Beispiel: '5.00902, ENTER, 50, ENTER, 10, XEQ "BI"' legt 50 nach R_{05}, 60 nach R_{07} und 70 nach R_{09}.

```
01♦LBL "BI"     04 LASTX      07 ISG Y
02 -            05 +          08 GTO 10
03♦LBL 10       06 STO IND Y  09 END
```

(19 Bytes)

- Quelle: PPC ROM.

10-7 Block betrachten ("VB")

Diese Routine zeigt/druckt den Inhalt aller Datenregister eines Blockes, die einen von '0' verschiedenen Wert haben. Die Kontrollzahl 'abc.lmn' wird ins X-Register gegeben; dann 'XEQ "VB"'. Die Routine benutzt '0, X = Y?' statt 'X = 0?', um die "ALPHA DATA"-Meldung (die Alpha-Ketten auslösen würden) zu vermeiden. Um auch R_{00} mitanzuzeigen, sind einige Schritt mehr als eigentlich notwendig enthalten. Passen Sie u.U. die Zeilen 28 und 35 Ihren eigenen Bedürfnissen an. Wenn die Routine ohne Drucker abläuft, können Sie bei jedem Halt, das einen Registerinhalt anzeigt, beliebige Register mit beliebigen Werten laden, insbesondere löschen, und dann zur nächsten Anzeige mit 'R/S' fortfahren.

```
01♦LBL "VB"     11 "R00"        21 LASTX        31 CLD
02 SF 21        12 X=0?         22 10           32♦LBL 10
03 CF 29        13 GTO 11       23 X>Y?         33 ISG L
04 STO L        14♦LBL 12       24 "├0"         34 GTO 12
05 0            15 0            25 ARCL L       35 FIX 2
06 RCL IND L    16 RCL IND L    26♦LBL 11       36 SF 29
07 X=Y?         17 X=Y?         27 "├= "        37 CLST
08 GTO 10       18 GTO 10       28 FIX 2        38 END
09 RCL L        19 FIX 0        29 ARCL IND L
10 INT          20 "R"          30 AVIEW
```

(72 Bytes)

- Quelle: John Dearing (2791). Vgl. auch 9-1.

10-8 Synthetisches Betrachten eines Blockes ("VB")

Die voranstehende Routine 10-7 kann in eine synthetische Version abgeändert werden, bei der der vor Aufruf bestehende Anzeige-Modus die Anzeige der Registerinhalte bestimmt und erhalten bleibt: (beachten Sie 2-2) Zeilen 36, 35, 28 und 25 löschen; hinter Zeile 24 'RCL d, CF 29, FIX 0, ARCL L, STO d' einfügen; Zeilen 19 und 03 löschen.

10-9 Synthetisches Betrachten eines Blockes ("BV")

Kontrollzahl 'abc.lmn' zur Festlegung des Blockes eintasten; dann 'XEQ "BV"'. Alle Register des Blockes, deren Inhalt nicht '0' ist, werden angezeigt und - sofern der Drucker angeschlossen und Flag 21 gesetzt ist - gedruckt. Soll länger angezeigt werden, muß man Flag 09 vor Aufruf setzen (dies bewirkt ein 'PSE' nach jedem 'AVIEW'). Mit 'SF 10' kann man die Routine zum 'STOP' nach jedem 'AVIEW' veranlassen. Zeile 29 ist der synthetische Ton 38. Der ursprüngliche Anzeige-Modus bleibt erhalten.

```
01♦LBL "BV"     12 RCL d        23 FS? 09       34♦LBL "VA"
02 ,            13 CF 29        24 PSE          35 SF 25
03 ENTER↑       14 FIX 0        25 LASTX        36 PRA
04♦LBL 00       15 ARCL Y       26 ,            37 SF 25
05 CLX          16 STO d        27 ENTER↑       38 FS?C 21
06 RCL IND Z    17 "├: "        28♦LBL 01       39 CF 25
07 X=Y?         18 R↑           29 TONE 8(38)   40 AVIEW
08 GTO 01       19 ARCL X       30 ISG Z        41 FC?C 25
09 X<> Z        20 XEQ "VA"     31 GTO 00       42 SF 21
10 INT          21 FS? 10       32 TONE 6       43 .END.
11 CLA          22 STOP         33 RTN
```

(83 Bytes)

- Quelle: Richard Schwarz (2289) (PPC ROM).

10-10 Block invertieren ("RB")

Diese Routine invertiert jeden Block numerischer Datenregister, der durch eine Kontrollzahl 'abc.lmn' in X festgelegt wurde. Z.B. bewegt '4.008, XEQ "RB"' die Inhalte der betroffenen Datenregister wie folgt: R_{08} nach R_{04}, R_{07} nach R_{05}, R_{05} nach R_{07}, R_{04} nach R_{08}, R_{06} bleibt unverändert. Die Routine ist besonders nützlich, wenn eine

aufsteigende Sortierung in eine absteigende geändert werden soll, oder umgekehrt. Sie benutzt selbst keine Datenregister und ist sehr schnell: 100 Register werden in 15 Sekunden invertiert.

```
01♦LBL "RB"     07 ST+ Y     13 /          19 X<> IND Y
02 ENTER↑       08 X<>Y      14 +          20 ISG Z
03 ENTER↑       09 2         15 X<>Y       21 DSE Y
04 FRC          10 /         16♦LBL 00     22 GTO 00
05 1 E3         11 INT       17 X<> IND Y  23 END
06 *            12 1 E3      18 X<> IND Z
```

(41 Bytes)

- Quelle: Valentin Albillo (4747).

10-11 Block duplizieren ("DUP")

Diese Routine kopiert die Daten eines Blockes in einen zweiten Block. Die Blöcke dürfen sich nicht überlappen. Die den ersten Block festlegende Steuerungszahl 'abc.lmn' gehört ins Y-Register, die Adresse des ersten Registers des zweiten Blockes ins X-Register.

'LBL "DUP", LBL 00, RCL IND Y, STO IND Y, RDN, 1, +, ISG Y, GTO 00, RTN'

(20 Bytes)

- Quelle: Bill Kolb (265) (PPC CJ, V7N4P17).

10-12 Block-Inhalt bewegen ("BM")

Diese Routine bewegt den Inhalt eines ganzen Datenregisterblockes in die Register eines 'Zielblockes'. 'Sende-' und 'Zielblock' dürfen sich überlappen. Die Adresse des ersten Registers des Zielblocks darf kleiner sein als die des ersten Registers des Sendeblockes. Benutzung: Adresse des ersten Registers des Sendeblockes eingeben, 'ENTER', Adresse des ersten Registers des Zielblockes eingeben, 'ENTER', Blocklänge eingeben, 'XEQ "BM"'. "BM" simuliert die X-Funktion 'REGMOVE'.

```
01♦LBL "BM"     07 ST+ Z     13♦LBL 04     19 ST+ Z
02 SIGN         08 +         14 R↑         20 ST+ Y
03 RDN          09 -1        15♦LBL 05     21 DSE L
04 X<Y?         10 ST+ Z     16 RCL IND Z  22 GTO 05
05 GTO 04       11 ST+ Y     17 STO IND Z  23 END
06 LASTX        12 RDN       18 RDN
```

(41 Bytes)

- Quelle: John Kennedy (918) (PPC ROM).

10-13 Block-Inhalte austauschen ("XB")

Diese Routine tauscht die Inhalte der Register zweier gleichlangen Blöcke gegeneinander aus. Die Blöcke dürften sich nicht überlappen. Benutzung: Adresse des ersten Registers des einen Blockes eingeben, 'ENTER', Adresse des ersten Registers des anderen Blockes eingeben, 'ENTER', Blocklänge eingeben, 'XEQ "XB"'.

```
01♦LBL "XB"     04 RCL IND Z    07 RDN          10 DSE L
02 SIGN         05 X<> IND Z    08 ST+ Z        11 GTO 02
03♦LBL 02       06 STO IND T    09 ST+ Y        12 END
```

(26 Bytes)

- Quelle: John Kennedy (918) und Richard Schwartz (2289) (PPC CJ, V7N10P7).

10-14 Block-Inhalte austauschen ("BE")

Die Inhalte zweier Blöcke gleicher Länge werden gegeneinander ausgetauscht. Benutzung: Kontrollzahl 'abc.lmn' für ersten Block eingeben, 'ENTER', Adresse des ersten Registers des zweiten Blockes eingeben, 'XEQ "BE"'. Welcher der beiden Blöcke als erster und welcher als zweiter angesehen wird ist gleichgültig. "BE" simuliert die X-Funktion 'REGSWAP'.

```
01♦LBL "BE"     04 X<> IND Y    07 ISG X        10 GTO 01
02♦LBL 01       05 STO IND Z    08 STO X        11 END
03 RCL IND Y    06 RDN          09 ISG Y
```

(25 Bytes)

- Quelle: Bill Kolb (265) (PPC CJ, V7N4P17; PPC ROM).

10-15 Synthetischer 'primär-sekundär' Tausch ("P-S")

Diese Routine simuliert die Funktion 'P $\gtrless$ S' des HP-67/97, welche die Inhalte der Register R_{00}-R_{09} gegen die der Register R_{10}-R_{19} austauscht. Der Stapel bleibt unberührt; das Alpha-Register wird benutzt. Ausführungszeit 3.5 Sekunden.

```
01♦LBL "P-S"    08 19           15 RDN          22 R↑
02 STO [        09 ENTER↑       16 DSE Y        23 RCL ]
03 RDN          10 9            17 DSE X        24 RCL \
04 STO \        11♦LBL 01       18 GTO 01       25 RCL [
05 RDN          12 RCL IND X    19 RCL 00       26 END
06 STO ]        13 X<> IND Z    20 X<> 10
07 RDN          14 STO IND Y    21 STO 00
```

(48 Bytes)

- Quelle: David Bartholomew (3666) (PPC CJ, V7N8P8).

10-16 Block-Rotation ("BLR")

Diese Routine vertauscht die Inhalte der Register eines Blockes, welcher durch eine Kontrollzahl 'abc.lmn' in X bestimmt wird, zyklisch. Beispiel: mit '2.004, XEQ "BLR"' gelangt der Inhalt von R_{02} nach R_{03}, der von R_{03} nach R_{04} und der von R_{04} nach R_{02}. Die Werte aus X und Y wandern nach Z bzw. T.

```
01♦LBL "BLR"    05 RCL IND L    09 STO IND Y    13 GTO 07
02 ENTER↑       06♦LBL 07       10 RTN          14 END
03 ABS          07 ISG L        11♦LBL 08
04 RDN          08 GTO 08       12 X<> IND L
```

(28 Bytes)

- Quelle: John Kennedy (918) (PPC CJ, V7N10P9).

10-17 Block-Rotation mit Richtungsangabe ("BR")

Für diese Routine wird die Adresse des ersten Block-Registers nach Y gelegt und die Blocklänge 'n' nach X. 'XEQ "BR"' vertauscht die Registerinhalte zyklisch um ein Register, und zwar bestimmt das Vorzeichen von 'n' dabei die 'Rotationsrichtung': ist 'n' positiv, wird um ein Register nach rechts, also im Uhrzeigersinn verschoben; ist 'n' negativ, geschieht die Verschiebung nach links, also dem Uhrzeigersinn entgegen.

```
01♦LBL "BR"     09 -            17 ST+ Y        25 +
02 CHS          10 SIGN         18 DSE L        26 STO Y
03 X<0?         11♦LBL 06       19 GTO 06       27 -1
04 GTO 07       12 RCL IND Z    20 RTN          28 ST+ Z
05 RCL Y        13 X<> IND Z    21♦LBL 07       29 GTO 06
06 X<>Y         14 STO IND T    22 CHS          30 END
07 1            15 RDN          23 1
08 ST+ Z        16 ST+ Z        24 -
```

(51 Bytes)

- Quelle: John Kennedy (918) & Richard Schwarz (2289) (PPC ROM).

10-18 Extremwerte eines Blockes synthetisch finden ("BX")

Man lege die den Block bestimmende Kontrollzahl 'abc.lmn' ins X-Register und führe 'XEQ "BX"' aus. Anschließend liegt der kleinste Wert des Blockes in X und der größte in Y. Bei gesetztem Flag 10 wird nur mit absoluten Werten gerechnet. Die Kontrollzahl befindet sich am Schluß im Register O, die Adresse des den kleinsten Wert enthaltenden Registers in N und die Adresse des den größten Wert enthaltenden Registers in M. Der Block-Inhalt bleibt insgesamt unzerstört. Sind im Block größte und kleinste Werte mehrfach vertreten, wird jeweils das erste Register (niedrigste Adresse), welches einen solchen Extremwert enthält, in N bzw. M notiert. Die Routine kann als Matrix-Hilfsprogramm dienen, weil sie zur Bestimmung von Pivot-Elementen verwendbar ist (vgl. 11-4). - Statt der synthetischen Register O, N und M können auch gewöhnliche Datenregister beschickt werden. Die Zeilen 37, 32, 04, 03 und 02 dürfen sogar ganz entfallen.

```
01♦LBL "BX"     12 CLX          23 ISG Z        34♦LBL 11
02 STO [        13 RCL IND Z    24 GTO 08       35 CLX
03 STO \        14 FS? 10       25 X<>Y         36 RCL T
04 STO ]        15 ABS          26 R↑           37 STO \
05 RCL IND X    16 X>Y?         27 RTN          38 X<>Y
06 FS? 10       17 GTO 10       28♦LBL 10       39 RDN
07 ABS          18 R↑           29 X<>Y         40 GTO 09
08 ENTER↑       19 X>Y?         30 CLX          41 END
09 ENTER↑       20 GTO 11       31 RCL Z
10 RDN          21 RDN          32 STO [
11♦LBL 08       22♦LBL 09       33 GTO 09
```

(65 Bytes)

- Quelle: Richard Schwartz (2289) (PPC ROM).

10-19 R_{00} bis R_{mn} löschen ("CLRGX")

Mit 'mn' in X setzt diese Routine die Register R_{00}-R_{mn} auf '0'.

'LBL "CLRGX", STO 00, CLX, LBL 14, STO IND 00, DSE 00, GTO 14, RTN'

(19 Bytes)

- Quelle: John Dearing (2791) (PPC CJ, V7N5P7).

10-20 Block löschen ("EB")

Diese Routine setzt die Inhalte aller Datenregister, die in dem durch die Kontrollzahl 'abc.lmn' bestimmten Block liegen, auf Null. Beispiel: '5.009, XEQ "EB"' löscht die Register R_{05}-R_{09}.

'LBL "EB", 0, LBL 13, STO IND Y, ISG Y, GTO 13, RTN' (15 Bytes)

- Quelle: John Dearing (2791) (PPC CJ, V7N4P22). - Vgl. auch 10-22.

10-21 Registerlöschung mit 'CLΣ'

'CLΣ' löscht 6 benachbarte Datenregister, beginnend mit dem durch 'ΣREG' festgelegten Register. Z.B. löscht 'ΣREG 10, CLΣ, ΣREG 13, CLΣ' (6 Bytes) die Register R_{10}-R_{18}. Achtung: möglicherweise muß nach Anwendung dieses Verfahrens 'ΣREG' anschließend auf den alten Wert zurückgesetzt werden. - Quelle: HP Key Notes, V4N1P5.

10-22 Block löschen ("CR")

Diese Routine ist eine Spielart von 10-20. 'abc.lmn, XEQ "CR"'.

'LBL "CR", 0, STO IND Y, RDN, ISG X, GTO "CR", RTN' (17 Bytes)

- Quelle: John Burkhart (4382).

10-23 Block löschen ("BC")

'XEQ "BC"' löscht alle Register des durch die Kontrollzahl 'abc.lmn' in X festgelegten Blockes. Will man jedes zweite Register eines Blockes löschen, braucht man sich nur einer vollständigen Kontrollzahl 'abc.lmnik' zu bedienen: '10.02002, XEQ "BC"' z.B. löscht die Register R_{10}, R_{12}, R_{14}, R_{16}, R_{18} und R_{20}. Y, Z und T werden nicht benutzt. In Schritt 02 wurde 'SIGN' statt 'STO L' verwendet, um 1 Bytes zu sparen.

'LBL "BC", SIGN, CLX, LBL 13, STO IND L, ISG L, GTO 13, RTN' (16 Bytes)

- Quelle: John Kennedy (918) (PPC ROM).

10-24 Auswahl ohne Wiederholungen ("SE")

Die Routine kann 'Lotto spielen'. Sie wählt ein Register aus einem zusammenhängenden Block von Registern zufallsabhängig aus und zeigt dessen Inhalt an. Bei anschließen-

den Auswahlen gibt es keine Wiederholungen: es wird stets aus den übrigbleibenden Registern ausgewählt. Das Programm kann gewissermaßen als 'Kartenmischer' (Registermischer) angesehen werden, der nach und nach alle 'Karten' (alle Registerinhalte) zufallsbedingt ausgibt. "SE" ruft den Zufallsgenerator "RN" (13-3) auf. Als Startwerte braucht man die Adresse des ersten Registers des Blockes in R_{06}, die Gesamtanzahl der Register des Blockes in R_{07} und eine Zahl zwischen 0 und 1 als Anfangswert für den Zufallsgenerator in einem passend gewählten Register R_{mn}. Dann startet man mit 'mn, XEQ "SE"'. Bei häufigem Aufruf von "SE" sollte R_{07} u.U. vor jedem 'XEQ "SE"' auf '0' getestet werden. Sobald R_{07} '0' enthält, sind alle Register des Blockes ausgewählt; die Inhalte der Register bleiben in ihrer Gesamtheit im Block erhalten, sie erfahren jedoch eine Umordnung. Und zwar liegen sie im Anschluß an eine vollständig durchgeführte Auswahl in einer solchen Ordnung im Block, die der erfolgten Auswahl entgegengesetzt ist, also der Inhalt des zuerst (zuletzt) ausgewählten Registers im letzten (ersten) Register des Blockes. Will man den ganzen Vorgang wiederholen, muß R_{07} wieder mit der Blocklänge gefüllt werden. Die Routine benutzt den Stapel; sie verwendet keine Flags. Die Ausführung beträgt etwa 1 1/2 Sekunden. [Die in "RN" liegenden Zeilen 16 und 22 können auch aus direkten Befehlen (etwa 'RCL 05' und 'STO 05') bestehen; dann muß vor Beginn der Anfangswert für "RN" ebenfalls direkt eingegeben werden, und man braucht ihn nicht vor jedem 'XEQ "SE"' aufs neue ins X-Register zu legen.]

Beispiel: (1) Man lade Daten, wie sie unten in der linken Spalte aufgeführt sind. Als Anfangswert für "RN" wähle man '0.141592654' und lege ihn in R_{09} ab. Dann '10, STO 06, 5, STO 07'. (2) Auf '9, XEQ "SE"' hin erscheint "SONJA"; der Block-Inhalt ist jetzt so wie die mittlere Spalte geordnet. (3) Wenn Sie Schritt (2) noch 4 mal wiederholen, werden alle Namen nacheinander angezeigt, und der Block-Inhalt hat die in der rechten Spalte gezeigte Ordnung. (4) Will man aus diesem neu geordneten Block auswählen, muß R_{07} auf 5 zurückgesetzt werden; dann kann man mit Schritt (2) fortfahren. Natürlich arbeitet die Routine mit Zahlen ebenso wie mit Alpha-Ketten.

R_{10}: KATJA		R_{10}: KATJA			R_{10}: LUTZ	
R_{11}: ULF	R_{07}	R_{11}: ULF	R_{07}		R_{11}: KNUT	R_{07}
R_{12}: LUTZ	enth.	R_{12}: LUTZ	enth.	...	R_{12}: ULF	enth.
R_{13}: SONJA	5	R_{13}: KNUT	4		R_{13}: KATJA	0
R_{14}: KNUT		R_{14}: SONJA			R_{14}: SONJA	

R_{mn}: Anfangswert für "RN"; R_{06}: erstes Register; R_{07}: Blocklänge.

```
01♦LBL "SE"     07 DSE 07       13 STO IND Y    19 ,211327
02 XEQ "RN"     08 STO X        14 RTN          20 +
03 RCL 07       09 RCL 07       15♦LBL "RN"     21 FRC
04 *            10 +            16 RCL IND X    22 STO IND Y
05 RCL 06       11 RCL IND X    17 9821         23 END
06 ST+ Y        12 X<> IND Z    18 *
```

(55 Bytes)

- Quelle: Bill Kolb (265) & John Kennedy (918) (PPC ROM).

10-25 Austausch zwischen geraden und ungeraden Registern ("OE")

Die Routine tauscht in einem durch die Kontrollzahl 'abc.lmn' festgelegten Block die Inhalte der geraden Register gegen die Inhalte der ungeraden Register aus. Dies geht nach folgenden Regeln vor sich: Ist die Blocklänge gerade, so sind R_{abc} das erste bzw. R_{lmn} das letzte Register, deren Inhalte umgetauscht werden. Z.B. tauscht '1.004, XEQ "OE"' R_{01} gegen R_{02} und R_{03} gegen R_{04}; '2.005, XEQ "OE"' tauscht R_{02} gegen R_{03} und R_{04} gegen R_{05}. Ist dagegen die Blocklänge ungerade, wird das erste auf den Block folgende Register in den Austausch-Vorgang mit einbezogen: '2.004, XEQ "OE"' tauscht R_{02} gegen R_{03} und R_{04} gegen R_{05}. Die Werte in X und Y bleiben erhalten.

```
01♦LBL "OE"     05 X<>Y         09 X<> IND Y    13 ISG L
02 2 E-5        06 +            10 STO IND L    14 GTO 00
03 +            07♦LBL 00       11 RDN          15 RDN
04 1,4          08 RCL IND L    12 ISG X        16 END
```

(34 Bytes)

- Quelle: John Herzfeld (5428).

10-26 Block betrachten und edieren ("B?")

Mit einer einfachen (oder zu inkrementierenden) Kontrollzahl 'abc.lmn(ik)' in X kann 'XEQ "B?"' dazu dienen, die Inhalte eines Blockes zu betrachten und nach Bedarf zu ändern. In der Anzeige erscheinen die Register und ihre gegenwärtigen Inhalte. Soll der Inhalt unverändert bleiben, einfach 'R/S'; tastet man eine neue Zahl ein und erst dann 'R/S', wird der alte Inhalt durch die neue Eingabe ersetzt. Fügt man hinter Zeile 17 ('RDN') zusätzlich ein 'FC? 22' ein, wird ein geänderter Inhalt erst noch angezeigt, bevor die Routine zum nächsten Register übergeht. Der vor Eingabe der Kontrollzahl in X stehende Wert bleibt erhalten.

```
01♦LBL "B?"     07 ARCL X       13 FS? 22       19 GTO 14
02♦LBL 14       08 SCI 5        14 STO IND Y    20 RDN
03 ENTER↑       09 CF 22        15 FS? 22       21 FIX 2
04 INT          10 "⊢ "         16 RDN          22 END
05 FIX 0        11 ARCL IND X   17 RDN
06 "R"          12 PROMPT       18 ISG X
```

(43 Bytes)

- Quelle: Larry Trammell (6824).

10-27 'Eigen-Ladung' mit Kontrollzahl ("SLC")

Diese Routine lädt jedes Register eines durch eine Kontrollzahl 'abc.lmn' in X bestimmten Blockes mit seiner eigenen Adresse. Beispiel: '5.007, XEQ "SLC"' lädt '5' in R_{05}, '6' in R_{06} und '7' in R_{07}. Die Inhalte von X, Y und Z bleiben erhalten. - Quelle: John Dearing (2791). Vgl. auch 10-2.

'LBL "SLC", INT, LBL 08, STO IND X, ISG X, LBL 08, ISG L, GTO 08, RDN, RTN' (20 Bytes)

10-28 Block-Multiplikation ("B*")

Diese Routine berechnet die Summe der Produkte von einander entsprechenden Werten aus zwei Blöcken. Liegt die einen Block von y-Werten bestimmende Kontrollzahl 'abc.lmn' in Y und die Adresse des ersten Registers eines Blockes von x-Werten in X, liefert 'XEQ "B*"' die Summe $\Sigma x \cdot y$ nach X. Die durch 'ΣREG' festgelegten Statistik-Register werden nicht benutzt (vgl. dagegen 19-11).

```
01♦LBL "B*"     05 RCL IND Y     09 ST+ L      13 ISG Y
02 0            06 RCL IND Y     10 RDN        14 GTO 00
03 +            07 ST* Y         11 ISG X      15 LASTX
04♦LBL 00       08 RDN           12♦LBL 00     16 END
```

(30 Bytes)

- Quelle: Kiyoshi Akima (3456). - Vgl. auch 19-11.

Kapitel 11

MATRIZEN UND DATENVERARBEITUNG

11-1 Matrix-Routinen ("M1"-"M5")

Jede der 5 Matrix-Routinen erfordert 2 Startwerte; der eine Wert ist das Anfangsregister der Matrix, der andere Wert ist ihre Spaltenanzahl. Von den Matrizen wird angenommen, daß jede ihrer Zeilen in einem zusammenhängenden Registerblock gespeichert ist. Daher entspricht die Anzahl der Spalten der Blocklänge der Zeilen, und die ganze Matrix liegt Zeile für Zeile in aufeinanderfolgenden Blöcken. R_{07} enthält die Adresse des ersten Matrix-Registers; R_{08} enthält die Anzahl ihrer Spalten. Zeilen- und Spaltennummern beginnen mit 1.

"M1" tauscht zwei Zeilen in der Matrix gegeneinander aus; Eingabewerte sind die Nummern der beiden auszutauschenden Zeilen ["M1" kann auch als Teil der 'Datenbasis'-Routinen "IR" und "DR" (siehe 11-2) betrachtet werden, weil man sie dazu benutzen kann, zwei 'Sätze' gegeneinander auszutauschen; die zugehörige Eingabe besteht in diesem Fall aus den beiden Satznummern.]

"M2" multipliziert die Zeile einer Matrix mit einer Konstanten; Eingabewerte sind die Konstante und die Zeilennummer. Als Teil eines 'Datenbasissystems' kann "M2" dazu dienen, einen numerischen Satz mit einer Konstanten, 0 eingeschlossen, zu multiplizieren (die Eingabe besteht dann aus der Konstanten und der Satznummer).

"M3" addiert das Vielfache einer Zeile auf eine andere Zeile. Die Zeile, auf die addiert wird, ändert sich; die Zeile, deren Vielfaches berechnet wird, bleibt unverändert. Eingabewerte sind die Nummer der Zeile, auf die addiert wird, die Nummer der Zeile, die vervielfacht wird, und die Konstante. "M3" kann als Teil eines 'Datenbasissystems' betrachtet werden: sobald Sätze aus numerischen Einträgen bestehen (z.B. Preisspalten), kann "M3" dazu benutzt werden, das Vielfache eines Satzes auf einen anderen Satz zu addieren.

"M4" liefert die Zeilennummer i und die Spaltennummer j eines Matrixelementes (i,j), wenn man die Adresse des Registers, das den Wert dieses Elementes enthält, eingibt; i liegt nach Ausführung in Y und j in X. In einem 'Datenbasissystem' kann "M4" dazu behilflich sein, 'Feld' und 'Satz' einer Aufzeichnung zu bestimmen.

"M5" ist die Umkehrung von "M4". Gibt man Zeilennummer i und Spaltennummer j eines Matrixelementes (i,j) ein, bekommt man die Adresse des Registers geliefert, das den Wert von (i,j) enthält. Als Teil von "IR"/"DR" läßt sich "M5" dazu verwenden, das Register ausfindig zu machen, welches ein bestimmtes Feld (mit bekannter Nummer) eines bestimmten Satzes (mit bekannter Nummer) enthält.

Mustermatrix: Die links unten stehende 6 × 5 Matrix sei in den Registern R_{15}-R_{44} abgelegt. '21' ist der Wert des Elementes (1,1) in der linken oberen Ecke (Zeile 1, Spalte 1). Die zugehörigen Register mit ihren Inhalten sind rechterhand aufgeführt.

21	35	55	74	83	:	R15: 21	R21: 93	R27: 32	R33: 62	R39: 82
11	93	56	36	29	:	R16: 35	R22: 56	R28: 27	R34: 97	R40: 23
65	78	32	27	75	:	R17: 55	R23: 36	R29: 75	R35: 54	R41: 45
53	94	46	62	97	:	R18: 74	R24: 29	R30: 53	R36: 39	R42: 77
54	39	61	67	82	:	R19: 83	R25: 65	R31: 94	R37: 61	R43: 15
23	45	77	15	25	:	R20: 11	R26: 78	R32: 46	R38: 67	R44: 25

Die Matrix beginnt in R_{15}, und die Anzahl ihrer Spalten ist 5; also muß wie folgt gespeichert werden:

R_{07}: Startadresse 15; R_{08}: Spaltenanzahl 5.

Keine der fünf Routinen verändert die Inhalte von R_{07} und R_{08}. Mit den wie vorstehend abgespeicherten Matrixwerten und den Startwerten in R_{07} und R_{08} kann man die folgenden Beispiele durchrechnen:

(1) "M1": Um zwei Zeilen der Matrix gegeneinander auszutauschen, lege man die beiden Nummern der Zeilen (ihre Reihenfolge hat keinen Einfluß auf das Ergebnis) nach Y und X und führe 'XEQ "M1"' aus. '2, ENTER, 4, XEQ "M1"' z.B. tauscht die Zeilen 2 und 4.

(2) "M2": Um alle Werte einer Zeile 'i' mit der Kontanten 'k' zu multiplizieren: 'k, ENTER, i, XEQ "M2"'. Z.B. verdoppelt '2, ENTER, 6, XEQ "M2"' alle Werte der 6-ten Zeile.

(3) "M3": Um das 'k'-fache der Zeile 'i' der Zeile 'j' zuzuschlagen: 'j, ENTER, i, ENTER, k, XEQ "M3"'. Will man z.B. das Doppelte der Zeile 3 von Zeile 4 abziehen, muß man '4, ENTER, 3, ENTER, 2, CHS, XEQ "M3"' ausführen. Anschließend lautet die Zeile 4: -77, -62, -18, 8, -53.

(4) "M4": Will man wissen, welches Matrixelement (i,j) im Register 'r' liegt: 'r, XEQ "M4"'. Die Matrix selbst bleibt unverändert. Die Zeilennummer i gelangt nach Y, die Spaltennummer j nach X. Beispiel: Um Zeilen- und Spaltennummer des in R_{38} liegenden Wertes der obigen Mustermatrix zu finden: '38, XEQ "M4"'. 4 gelangt nach X, 5 nach Y, also enthält R_{38} den Wert des Elementes (5,4).

(5) "M5": Um die Registeradresse des Matrixelementes (i,j) zu finden: 'i, ENTER, j, XEQ "M5"'. Die Registeradresse gelangt nach X. Beispiel: '2, ENTER, 3, XEQ "M5"' liefert '22', d.h. der Wert des Matrixelementes (2,3) liegt in R_{22}. Den Wert selbst, '56', erhält man dann mit 'RCL IND X'.

```
01♦LBL "M2"     24 ""          47 RCL X       70 ST+ 08
02 XEQ 00       25 ISG Z       48 RCL 08      71 X<> L
03 X<>Y         26 GTO 02      49 ST- Z       72 X<> 08
04♦LBL 01       27 RTN         50 SIGN        73 1
05 ST* IND Y    28♦LBL "M1"    51 -           74 -
06 ISG Y        29 XEQ 00      52  E3         75 RCL 07
07 GTO 01       30 X<>Y        53 /           76 +
08 RTN          31 XEQ 00      54 +           77 RTN
09♦LBL "M3"     32♦LBL "BE"    55 RTN         78♦LBL "QR"
10 STO [        33 RCL IND Y   56♦LBL "M4"    79 X<>Y
11 RDN          34 X<> IND Y   57 RCL 07      80 STO ]
12 XEQ 00       35 STO IND Z   58 -           81 X<>Y
13 X<>Y         36 RDN         59 RCL 08      82 MOD
14 XEQ 00       37 ISG X       60 XEQ "QR"    83 ST- ]
15 RCL [        38 ""          61 ISG Y       84 LASTX
16 SIGN         39 ISG Y       62 ""          85 ST/ ]
17♦LBL 02       40 GTO "BE"    63 ISG X       86 CLX
18 RDN          41 RTN         64 ""          87 X<> ]
19 RCL IND Y    42♦LBL 00      65 RTN         88 X<>Y
20 LASTX        43 RCL 08      66♦LBL "M5"    89 END
21 *            44 *           67 X<> 08
22 ST+ IND Y    45 RCL 07      68 ST- 08
23 ISG Y        46 +           69 *
```

(171 Bytes)

Will man die synthetischen Befehle vermeiden, müssen folgende Änderungen vorgenommen werden: Register M (Zeilen 10 und 15) muß durch ein geeignetes gewöhnliches Register, etwa R_{00}, ersetzt werden. Dasselbe gilt für Register O (Zeilen 80, 83, 85 und 87); z.B. durch R_{01} ersetzbar. Die synthetischen 'NOP's 'Text 0' (Zeilen 24, 38, 62 und 64) können durch beliebige gewöhnliche 'NOP's, z.B. 'STO X', ersetzt werden (vgl. auch 2-15). Schließlich muß noch der Kurzform-Exponent 'E3' (Zeile 52) in '1 E3' geändert werden. - Quelle: John Kennedy (918) (PPC ROM).

11-2 Sätze einfügen und löschen ("IR" & "DR")

Diese Routinen können als Teil eines 'Datenbasissystems' angesehen werden. Sie sind anwendbar auf 'Dateien', die aus 'Sätzen' fester Länge bestehen, wobei jeder Satz in einem zusammenhängenden Block von Datenregistern liegt. Die ganze Datei besteht aus solchen aneinandergereihten Blöcken. "IR" ist eine Routine, die Platz zwischen zwei Dateisätzen schafft, indem sie die Inhalte von Blöcken versetzt, damit ein neuer Satz eingefügt werden kann. "DR" löscht einen Satz und rückt die verbliebenen Sätze der Datei zusammen, so daß der aktuelle Datenbereich so kompakt wie möglich bleibt. Für den Aufruf von "IR" oder "DR" muß die Datei im Arbeitsspeicher liegen, und die Register R_{07}-R_{09} müssen folgendermaßen geladen sein:

R_{07}: Adresse des ersten Dateiregisters
R_{08}: Anzahl der Register je Satz (entspricht der Spaltenanzahl in 11-1)
R_{09}: Gesamtzahl der Sätze in der Datei (entspricht der Zeilenanzahl in 11-1).

Um Platz für einen neuen k-ten Satz zu schaffen: 'k, XEQ "IR"'. "IR" bewegt den k-ten Satz und alle folgenden Sätze in höher numerierte Register, so daß die Daten eines neuen Satzes eingefügt werden können; "IR" setzt auch R_{09} um 1 höher, um dieses Register auf den neuesten Stand zu bringen. Man achte darauf, daß das Einfügen eines neuen k-ten Satzes die Umnumerierung des alten k-ten Satzes und aller Folgesätze nach sich zieht. Man beachte auch, daß "IR" natürlich nicht die neuen Daten einfügt, sondern nur Register dafür freimacht.

Um den k-ten Satz aus der Datei zu löschen: 'k, XEQ "DR"'. Die dem k-ten Satz folgenden Sätze werden alle um einen Block zurückgesetzt, und R_{09} wird um 1 vermindert, damit dort die gültige Gesamtzahl der Sätze auffindbar ist. Man berücksichtige, daß anschließend alle dem k-ten Satz folgenden Sätze eine neue Nummer tragen.

Beispiel: Man nehme sich die folgende Musterdatei und bereite die Einfügung eines neuen 3-ten Satzes vor:

#1: Josef Eibner	#2: Mike Johnson	#3: Jane Hamilton	#4: Paul Jones
354-1662	363-5648	261-2347	745-3254
Gary, IN	Boston, MA	Fresno, CA	Denver, CO

Die Musterdatei wird in R_{10}-R_{33} abgelegt; sie besteht aus 4 Sätzen zu je 6 Registern:

R10: JOSEF	R16: MIKE	R22: JANE	R28: PAUL
R11: EIBNER	R17: JOHNSO	R23: HAMILT	R29: JONES
R12:	R18: N	R24: ON	R30:
R13: 354.1662	R19: 363.5648	R25: 261.2347	R31: 745.3254
R14: GARY	R20: BOSTON	R26: FRESNO	R32: DENVER
R15: IN	R21: MA	R27: CA	R33: CO

Außerdem müssen R_{07}-R_{09} passend gefüllt werden:

R_{07}: Startregister 10
R_{08}: Anzahl der Register je Satz 6
R_{09}: Gesamtzahl der Sätze 4.

Nun führe man '3, XEQ "IR"' aus. Anschließend befinden sich der 3-te und 4-te Satz in den Registern R_{28}-R_{33} bzw. R_{34}-R_{39}; somit ist der Block R_{22}-R_{27} aufnahmebereit für einen neuen Satz. (Jedoch sind die Inhalte von R_{22}-R_{27} unverändert die alten!) R_{09} enthält jetzt 5.

Will man den (neuen) 3-ten Satz in R_{22}-R_{27} löschen: '3, XEQ "DR"'. Danach besteht die Datei wieder in ihrer ursprünglichen Form, und R_{09} enthält wiederum 4. (Allerdings sind die Inhalte von R_{34}-R_{39} unverändert zurückgeblieben.)

```
01♦LBL "IR"     13 +          25 ""         37 RDN
02 ISG 09       14 STO Y      26♦LBL "BM"   38♦LBL 04
03 ""           15 RCL 09     27 SIGN       39 R↑
04 XEQ 03       16 R↑         28 RDN        40♦LBL 05
05 ST- T        17 -          29 X<Y?       41 RCL IND Z
06 *            18 RCL 08     30 GTO 04     42 STO IND Z
07 GTO "BM"     19 RTN        31 LASTX      43 RDN
08♦LBL 03       20♦LBL "DR"   32 ST+ Z      44 ST+ Z
09 RCL 07       21 XEQ 03     33 +          45 ST+ Y
10 RCL 08       22 ST- Z      34 -1         46 DSE L
11 RCL Z        23 *          35 ST+ Z      47 GTO 05
12 *            24 DSE 09     36 ST+ Y      48 END
```

(89 Bytes)

Die synthetischen 'NOP's (Zeilen 03 und 25) können durch 'STO X' ersetzt werden. - Quelle: John Kennedy (918) (PPC ROM).

11-3 Matrizen-Eingabe/Ausgabe ("MIO")

Diese Routine fordert zur Eingabe der Werte einer Matrix (ausschließlich numerische Werte) auf; die Aufforderung erfolgt über die Elementadressen (i,j) (Zeile, Spalte); Begriffe und Größen stimmen mit den in 11-1 ("M1"-"M5") verwendeten überein. Die Matrizen werden Zeile für Zeile gespeichert, wobei jede Zeile einen zusammenhängenden Block von Datenregistern belegt; die ganze Matrix besetzt eine Folge von Registerblöcken. Ein Teil der Routinen aus 11-1 muß im Speicher liegen, denn "MIO" ruft "M4" und "M5" auf und "M4" ihrerseits "QR". Außerdem benutzt "MIO" die lokalen Marken 'A', 'B' und 'C', die im USER-Modus mit folgender Bedeutung zur Verfügung stehen:

'A': Matrix eingeben 'B': Matrix betrachten 'C': (i,j) aufrufen

Folgende Flags und Register werden benutzt:

Flags: 09: benutzt
29: gelöscht

Register: R_{03}: Zähler, R_{04}: Zeiger, R_{05}: Nachkommastellen, R_{07}: Startregisteradresse, R_{08}: Spaltenanzahl, R_{09}: Zeilenanzahl

Beispiel 1: Um die Lösung des folgenden linearen Gleichungssystems (vgl. "RRM" aus 11-4) vorzubereiten, speichere man die zugehörige Matrix.

$$\begin{aligned} -5x + 10y + 15z &= 5 \\ 2x + y + z &= 6 \\ x + 3y - 2z &= 13 \end{aligned} \quad = \quad \left[\begin{array}{ccc:c} -5 & 10 & 15 & 5 \\ 2 & 1 & 1 & 6 \\ 1 & 3 & -2 & 13 \end{array}\right]$$

Man setze den HP-41 auf das Programm "MIO" und drücke (im USER-Modus) 'A'. Die erste Aufforderung (Startregister) wird mit '10, R/S' beantwortet (falls der Drucker angeschlossen aber ausgeschaltet ist, erst 'CF 21'); die zweite Aufforderung (Zeilenanzahl ↑ Spaltenanzahl) mit '3, ENTER, 4, R/S'. Anschließend wird Koeffizient für Koeffizient eingegeben und jede Eingabe mit 'R/S' beendet. Nach der letzten Eingabe ['13' - Element (3,4)] ertönt der 'BEEP'. Will man die Dateneingabe überprüfen, fülle

man R_{05}, um die Anzahl der Nachkommastellen festzusetzen, und drücke 'B'. Jetzt wird die ganze Matrix Element für Element angezeigt. Bei angeschlossenem Drucker werden die Werte auch gedruckt. Zieht man die 'SCI'-Anzeige vor, muß Zeile 44 in 'SCI IND 05' abgeändert werden. (Soll die Routine - bei abgekoppeltem Drucker - bei jeder Anzeige anhalten, muß man Flag 21 setzen.) Nach der letzten Anzeige ertönt der 'BEEP'.

Will man feststellen, welchem Register ein bestimmtes Element entspricht, muß man 'C' tasten. Eingabewerte sind Zeilen- und Spaltennummer. Z.B. führt '3, ENTER, 2' und anschließendes 'C' zur Ausgabe "R19.00" und "(3,2)=3.00". Also liegt der Wert des Elementes (3,2) im Register R_{19} und lautet '3'. Bemerkung: Wenn bei der Eingabe der Matrix (im Anschluß an 'A') ein falscher Wert in die Register gelangt, lasse man diesen Fehler ruhig durchgehen und fahre mit der Eingabe fort; nach Beendigung der Eingaben kann man dann mit 'C' das Register, welches den falschen Wert enthält, ausfindig machen und den richtigen Wert von Hand speichern.

Beispiel 2: Um die Lösung des folgenden linearen Gleichungssystems und die Berechnung der Inversen und der Determinante der Koeffizienten-Matrix (vgl. 11-4) vorzubereiten, speichere man die zugehörige Matrix:

$$\begin{aligned} 14x + 2y - 6z &= 9 \\ -4x + y + 9z &= 3 \\ 6x - 4y + 3z &= -4 \end{aligned} \quad = \quad \left[\begin{array}{ccc:ccc:c} 14 & 2 & -6 & 1 & 0 & 0 & 9 \\ -4 & 1 & 9 & 0 & 1 & 0 & 3 \\ 6 & -4 & 3 & 0 & 0 & 1 & -4 \end{array}\right]$$

Die abzuspeichernde Matrix besteht aus der ursprünglichen Koeffizientenmatrix, erweitert um die sog. Einheitsmatrix und die Spalte der Konstanten. Dies ergibt insgesamt eine 3×7 Matrix (s. oben rechts). Man setze den HP-41 auf das Programm "MIO", drücke (im USER-Modus) 'A' und gebe '10' als Startregister und '3, ENTER, 7' als Zeilen bzw. Spaltenanzahl ein. Anschließend werden alle Matrixwerte eingetastet [(1,1)='14', (1,2)='2', ..., (3,7)='-4']. Mit 'B' lassen sich die Werte - falls nötig - überprüfen und mit 'C' korrigieren.

```
01♦LBL "MIO"      17 CF 09       33 ARCL X       49 STO X
02 STOP           18♦LBL 01      34 "⊢)="        50 DSE 03
03♦LBL A          19 CF 29       35 FC? 09       51 GTO 02
04 "START REG?"   20 RCL 07      36 GTO 03       52 BEEP
05 AVIEW          21 STO 04      37 "⊢?"         53 RTN
06 STOP           22 RCL 08      38 AVIEW        54♦LBL C
07 STO 07         23 RCL 09      39 TONE 0       55 XEQ "M5"
08 "DIM: Z?↑S?"   24 *           40 STOP         56 " R"
09 AVIEW          25 STO 03      41 STO IND 04   57 ARCL X
10 STOP           26♦LBL 02      42 GTO 04       58 AVIEW
11 STO 08         27 RCL 04      43♦LBL 03       59 STO 04
12 X<>Y           28 XEQ "M4"    44 FIX IND 05   60 1
13 STO 09         29 FIX 0       45 ARCL IND 04  61 STO 03
14 SF 09          30 " ("        46 AVIEW        62 CF 09
15 GTO 01         31 ARCL Y      47♦LBL 04       63 GTO 02
16♦LBL B          32 "⊢,"        48 ISG 04       64 END
```

(133 Bytes)

- Quelle: John Kennedy (918).

11-4 Determinante und Inverse einer Matrix; Lösung linearer Gleichungssysteme ("RRM")

Diese Routine berechnet Determinanten und Inverse von Matrizen und löst lineare Gleichungssysteme. Sie behandelt diese drei Matrix-Rechenprobleme einzeln oder gleichzeitig und benutzt eine Technik, die unter dem Namen partielle Pivotierung bekannt ist und hier dazu dient, Rundungsfehler zu verkleinern. Die einzige Beschränkung für die behandelten Matrizen liegt in der Anzahl der verfügbaren Datenregister. "RRM" kann sogar auf mehr als eine Matrix im Arbeitsspeicher angewendet werden. "RRM" ruft mehrere andere Routinen auf, die daher ebenfalls im Programmspeicher liegen müssen: "M1"-"M5" (11-1), "BE" (10-14 oder 11-1), "BX" (10-18) und "QR" (15-12 oder 11-1). Die Matrizen-Eingabe/Ausgabe-Routine (11-3) ist nützlich zum Laden, Betrachten und Korrigieren der Werte und wird in den unten vorgeführten Beispielen verwendet. Vor dem Aufruf von "RRM" muß die Adresse des Startregisters (10 oder größer) in R_{07} abgelegt werden, die Anzahl der Spalten in R_{08} und die Anzahl der Zeilen in R_{09}. Falls Sie "MIO" einsetzen, sind diese Voraussetzungen erfüllt.

Folgende Flags und Register werden benutzt:

Flag 10: benutzt	Register: R_{01}: Determinante		R_{07}: Startregisteradresse
	R_{02}: benutzt		R_{08}: Spaltenanzahl
	R_{03}: "		R_{09}: Zeilenanzahl
	R_{04}: "		Alpha-Register: benutzt

Beispiel 1: Zunächst wird Beispiel 1 aus 11-3 mit "MIO" durchgeführt. Wenn die Matrix richtig geladen ist, 'XEQ "RRM"'; der Programmlauf benötigt etwa 50 Sekunden; wenn er zuende ist, geht man nach "MIO" und drückt 'B', um sich die 'Ergebnis-Matrix' anzeigen zu lassen. Sie lautet:

$$\left[\begin{array}{ccc:c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$$

x=2, y=3, z=-1 ist die Lösung des Gleichungssystems.

Die Determinante der quadratischen Koeffizientenmatrix liegt in R_{01}: 150.00.

Beispiel 2: Man führt Beispiel 2 aus 11-3 mit "MIO" aus. Sobald die Matrix geladen ist, 'XEQ "RRM"'; anschließend wieder nach "MIO" zurück und 'B'. Jetzt findet man die Matrix:

$$\left[\begin{array}{ccc:ccc:c} 1 & 0 & 0 & 0.0631 & 0.0291 & 0.0388 & 0.5000 \\ 0 & 1 & 0 & 0.1068 & 0.1262 & -0.1650 & 2.0000 \\ 0 & 0 & 1 & 0.0162 & 0.1100 & 0.0356 & 0.3333 \end{array}\right]$$

Die letzte Spalte enthält die Lösung des Systems: x=1/2, y=2, z=1/3. Die Determinante der Koeffizientenmatrix liegt in R_{01} und hat den Wert 618; ihre Inverse ist die 3×3 Matrix in der Mitte. Mit "DF" (14-1) kann man die Dezimaldarstellung dieser Wer-

te (sie liegen in den Registern R_{13}-R_{15}, R_{20}-R_{22} und R_{27}-R_{29}) in Bruchdarstellung umwandeln:

$$\begin{bmatrix} 13/206 & 3/103 & 4/103 \\ 11/103 & 13/103 & -17/103 \\ 5/309 & 34/309 & 11/309 \end{bmatrix}$$

Ist man lediglich an der Determinante einer Matrix interessiert, braucht man für "RRM" nur die Eingabewerte einer quadratischen Matrix. Will man die Inverse einer Matrix berechnen, gibt man die Werte so wie in Beispiel 2 von 11-3 ein, jedoch ohne die rechterhand stehende letzte Spalte der Konstanten. "RRM" transformiert die ursprünglich eingegebenen nicht notwendig quadratischen Matrizen in ihre sog. zeilenreduzierte Staffelform. Die Ergebnismatrix ist daher zeilenäquivalent zu der Ausgangsmatrix. Aus diesem Grunde können Systeme, die keine oder unendlich viele Lösungen haben, leicht mit "RRM" untersucht werden: der reduzierten Matrix läßt sich nämlich entnehmen, ob und wo Parameter eingesetzt werden können, so daß man sofort alle Lösungen bestimmen kann, wenn solche existieren. Allerdings ist eine Warnung nötig: Durch Rundungsfehler können die Determinanten bereits sehr einfacher Matrizen zu Unrecht einen von 0 verschiedenen Wert erhalten, wodurch fehlerhafte Analysen und Ergebnisse entstehen. Beispiel: "RRM" berechnet die Determinante der Matrix $\begin{vmatrix} 3 & 4 \\ 15 & 20 \end{vmatrix}$ zu $-1{,}5\cdot10^{-8}$, obwohl diese Determinante verschwindet. Dagegen ergibt sich für $\begin{vmatrix} 3 & 15 \\ 4 & 20 \end{vmatrix}$ das richtige Ergebnis 0. Bei Determinanten, deren Wert nahe bei 0 liegt, ist also Vorsicht geboten.

```
01♦LBL "RRM"     19 X>Y?          37 1/X           55 ""
02 ,             20 RTN           38 RCL [         56 RCL 09
03 STO 03        21 RCL 04        39 INT           57 RCL 02
04 STO 04        22 XEQ "M5"      40 XEQ "M4"      58 X>Y?
05 SIGN          23 X<> Z         41 RDN           59 GTO 05
06 STO 01        24 XEQ "M5"      42 STO 02        60 RCL 03
07 SF 10         25  E3           43 XEQ "M2"      61 X=Y?
08♦LBL 05        26 /             44 RCL 02        62 GTO 07
09 ISG 03        27 +             45 ST- 02        63 RCL 02
10♦LBL 06        28 RCL 08        46 RCL 03        64 RCL 04
11 ISG 04        29  E5           47 X=Y?          65 XEQ "M5"
12 ""            30 /             48 GTO 07        66 RDN
13 RCL 08        31 +             49 XEQ "M1"      67 RCL IND T
14 RCL 04        32 XEQ "BX"      50 RCL 01        68 CHS
15 X>Y?          33 RCL IND [     51 CHS           69 XEQ "M3"
16 RTN           34 ST* 01        52 STO 01        70 GTO 07
17 RCL 09        35 X=0?          53♦LBL 07        71 END
18 RCL 03        36 GTO 06        54 ISG 02
```

(120 Bytes)

- Quelle: John Kennedy (918).

Kapitel 12

SORTIEREN

12-1 Sortierprogramm ("QS")

Dieses Programm sortiert die numerischen Daten der Register R_{01} - R_{mn}. Vor Aufruf von "QS" muß 'mn' ins X-Register gelegt werden. "QS" löscht Flag 21, setzt 'FIX 3' fest und benutzt Flag 4. Bereits sortierte Daten werden besonders zeitaufwendig behandelt! Der gerade in der Verarbeitung befindliche Block wird, während das Programm läuft, in Form der Kontrollzahl 'abc.lst' angezeigt. "QS" benötigt für Notizzwecke einige oberhalb von R_{mn} liegende Register: um 'N' Zahlen zu sortieren, werden (R_{00} mitgerechnet) höchstens $N + 1 + INT(\log_2 N)$ Register benötigt. Will man z.B. die Daten der Register R_{01}-R_{32} sortieren, führt man '32, XEQ "QS"' aus. Wenn der 'BEEP' ertönt und "GETAN" erscheint, zeigt eine Durchsicht der Datenregister, daß in R_{00} und R_{33}-R_{36} Kontrollzahlen liegen (bei bereits sortierten Daten werden allerdings nicht alle diese Notiz-Register benötigt), die Daten in R_{01}-R_{32} in aufsteigender Folge sortiert wurden und die Inhalte der Register von R_{37} an aufwärts unverändert sind. 'Je größer die Ordnung' der Ausgangsdaten, desto länger die Ausführungszeit: 32 Zahlen werden, wenn sie stark zufallsbedingt angeordnet sind, in etwa 1 1/2 Minuten sortiert; ein erneutes Sortieren der bereits sortierten Daten nimmt dagegen gut 3 Minuten in Anspruch.

```
01♦LBL "QS"     25 X>Y?        49 STO IND T   73 /           97 X>Y?        121 GTO 27
02 STO 00       26 XEQ 99      50 X<>Y        74 +           98 XEQ 22      122 ISG Z
03 CF 21        27 FS?C 04     51 STO IND Z   75 X<> IND 00  99♦LBL 24     123 BEEP
04 FIX 3        28 GTO 24      52 GTO 20      76 ISG 00      100 DSE 00     124 STO IND Z
05 CF 04        29 RCL IND Z   53♦LBL 19      77 STOP        101 GTO 15     125 RDN
06 RCL 00       30 R↑          54 R↑          78 STO IND 00  102 "GETAN"    126 X<>Y
07 1 E3         31 RCL L       55 RCL IND Y   79 ENTER↑      103 AVIEW      127 2
08 /            32 X<>Y        56 X<=Y?       80 RTN         104 BEEP       128 -
09 1            33♦LBL 16      57 GTO 21      81♦LBL 23      105 RTN        129 X≠0?
10 +            34 R↑          58♦LBL 20      82 1           106♦LBL "IST"  130 GTO 26
11 ST+ 00       35 CLX         59 DSE Z       83 -           107♦LBL 99     131 STO Z
12 STO IND 00   36 RCL IND Z   60 CLA         84 RCL IND 00  108 SF 04      132♦LBL 27
13♦LBL 15       37 R↑          61 RDN         85 INT         109 ISG IND 00 133 CLX
14 RCL IND 00   38 X<=Y?       62 RDN         86 X<>Y        110♦LBL 25     134 1
15 VIEW X       39 GTO 18      63 X<=Y?       87 X>Y?        111 RCL IND 00 135 ST+ Z
16 INT          40♦LBL 17      64 GTO 23      88 XEQ 22      112 RCL IND X  136 RDN
17 LASTX        41 ISG T       65 GTO 19      89 RDN         113 X<>Y       137 STO IND Y
18 FRC          42 STOP        66♦LBL 21      90 RDN         114 1          138 ISG IND 00
19 1 E3         43 RDN         67 STO IND T   91 1           115 -          139 GTO 25
20 *            44 RDN         68 X<>Y        92 +           116 INT        140 END
21 STO Z        45 X<=Y?       69 STO IND Z   93 RCL IND 00  117♦LBL 26
22 X<>Y         46 GTO 23      70 GTO 17      94 FRC         118 X<>Y
23 -            47 GTO 16      71♦LBL 22      95 1 E3        119 RCL IND Y
24 15           48♦LBL 18      72 1 E3        96 *           120 X<=Y?
```

(249 Bytes)

- Quelle: Mike Hale (4457) (PPC CJ, V7N2P39).

12-2 Stapel-Sortierung ("S1")

Diese Routine sortiert die Daten in den Stapelregistern X, Y, Z und T. Sie arbeitet nicht, sobald in einem der Register eine Alpha-Kette liegt. Wird eine absteigende Ordnung (größter Wert in X, kleinster in T) gewünscht, muß Flag 10 gelöscht werden; bei gesetztem Flag 10 ergibt sich eine aufsteigende Ordnung. "S1" selbst löscht Flag 10; will man dies vermeiden, muß Zeile 29 ('FS?C 10') in 'FS? 10' geändert werden.

```
01♦LBL "S1"     10 R↑          19 X<=Y?       28♦LBL 01
02 X>Y?         11 X<=Y?       20 GTO 03      29 FS?C 10
03 X<>Y         12 GTO 01      21 X<>Y        30 RTN
04 RDN          13 X<>Y        22♦LBL 03      31 RDN
05 X>Y?         14 RDN         23 RDN         32 X<> Z
06 X<>Y         15 X<=Y?       24 RDN         33 END
07 R↑           16 GTO 02      25 GTO 01
08 X>Y?         17 X<>Y        26♦LBL 02
09 X<>Y         18 RDN         27 R↑
```

(46 Bytes)

- Quelle: PPC ROM. Vgl. auch 12-6.

12-3 Synthetische Sortierprogramme ("S2" & "S3")

Diese Routinen arbeiten mit (vor)sortierten Daten etwa genauso schnell wie mit zufallsbedingt geordneten. Sämtliche Register oberhalb des Blockes, dessen Daten sortiert werden, bleiben unzerstört. Beide Routinen verwenden eine den Block festlegende Kontrollzahl 'abc.lmn', die vorher ins X-Register gelegt werden muß. Diese Kontrollzahl liegt nach Beendigung des Programmlaufs wieder in X. Stapel- und Alpha-Register werden benutzt; ebenso Flag 10. *"S2" (geringere Datenanzahl)* arbeitet schneller als "S3", solange die Anzahl der beteiligten Register höchstens 32 beträgt. Die Routine benutzt überhaupt keine Datenregister und sortiert daher *jeden* Block von höchstens 32 Registern. Sie wird von "S3" als Unterprogramm benutzt. *"S3" (größere Datenanzahl)* arbeitet schneller als "S2", sobald die Anzahl der beteiligten Register 32 übersteigt. Diese zweite Routine benutzt die Register R_{01} und R_{02}, so daß mit ihr also nur Blöcke ab Register R_{03} verarbeitet werden können. Ein Vergleich der Ausführungszeiten von "S2" und "S3" ergibt: bei 5 Registern etwa 4 Sekunden gegen etwa 9 Sekunden; bei 32 Registern je etwa 50 Sekunden; bei 96 Registern etwa 5 1/2 Minuten gegen etwa 3 1/2 Minuten.

```
01♦LBL "S3"     47 X<> IND T     93  E3          139 ISG \
02 STO ]        48 ISG T         94 /           140 GTO 00
03 SF 10        49 ""            95 RCL ]       141 GTO 02
04♦LBL 09       50 DSE [         96 INT         142♦LBL 01
05 STO 02       51 GTO 11        97 +           143 X<> ↑
06♦LBL 05       52♦LBL 06        98 SF 10       144 X<Y?
07 STO 01       53 X<> IND Z     99 GTO 09      145 GTO 01
08 FRC          54 FRC          100♦LBL 07      146 RDN
09  E3          55 R↑           101  E          147 X<> ↑
10 *            56 INT          102 -          148 RDN
11 STO \        57 +            103  E3         149 ISG \
12 RCL 01       58♦LBL 10       104 /          150 GTO 00
13 INT          59 ENTER↑       105 +          151 GTO 02
14 -            60 FRC          106 GTO 10     152♦LBL 01
15  E           61  E3          107♦LBL 08     153 X<> T
16 +            62 *            108 LASTX      154 X<Y?
17 STO [        63 X<>Y         109 BEEP       155 X<>Y
18 RCL 01       64 -            110 RTN        156 R↑
19  E-4         65 31           111♦LBL "S2"   157 X<> ↑
20 +            66 X>Y?         112 CF 10      158 X<> T
21 ,            67 GTO 01       113 STO [      159 ISG \
22 RCL X        68 LASTX        114 INT        160 GTO 00
23♦LBL 14       69 FS?C 10      115  E99       161♦LBL 02
24 RCL IND Z    70 GTO 09       116 STO ↑      162 R↑
25 +            71 GTO 05       117 ENTER↑     163 X<> IND [
26 ISG Y        72♦LBL 01       118 ENTER↑     164 X<> ↑
27 ""           73 LASTX        119 ENTER↑     165 ISG [
28 ISG Z        74 XEQ "S2"     120♦LBL 04     166 X<> IND [
29 GTO 14       75 INT          121 X<> [      167 R↑
30 X<>Y         76 RCL ]        122 STO \      168 ISG [
31 /            77 INT          123 X<> [      169 X<> IND [
32 RCL \        78 X=Y?         124♦LBL 00     170 R↑
33 X<>Y         79 GTO 08       125 X<> IND \  171 ISG [
34 RCL 01       80 RDN          126 X<Y?       172 X<> IND [
35 STO T        81 RCL 01       127 GTO 01     173 FS?C 10
36♦LBL 13       82 INT          128 ISG \      174 GTO 03
37 X<> IND \    83 X<>Y         129 GTO 00     175 R↑
38♦LBL 11       84 X>Y?         130 SF 10      176 ISG [
39 X<Y?         85 GTO 07       131 GTO 02     177 X<> IND [
40 GTO 12       86 RCL 02       132♦LBL 01     178♦LBL 03
41 DSE \        87 INT          133 R↑         179 ISG [
42 ""           88 X<>Y         134 X<Y?       180"GTO 04
43 DSE [        89 X>Y?         135 GTO 01     181 LASTX
44 GTO 13       90 GTO 07       136 X<> ↑      182 END
45 GTO 06       91  E           137 R↑
46♦LBL 12       92 -            138 R↑
```

(282 Bytes)

- Quelle: Ray Evans (4928) (PPC ROM).

12-4 Kleinste oder größte von 3 Zahlen finden ("SORT")

Diese Routine ermittelt die kleinste von 3 Zahlen; will man die größte finden, muß 'X>Y?' durch 'X<Y?' ersetzt werden (Schritte 04 und 07). Soll die Untersuchung auf 4 oder mehr Zahlen ausgedehnt werden, muß die Schrittfolge 04-06 für jede weitere Zahl eingesetzt werden.

```
'LBL "SORT", RCL 01, RCL 02, X>Y?, X<>Y, RCL 03, X>Y?, X<>Y, RTN'
```

(16 Bytes)

- Quelle: Bill Kolb (265) (BP 67/97).

12-5 Synthetisch alphabetisieren ("AL")

Diese Routine alphabetisiert zwei in X und Y liegende Alpha-Ketten, und zwar in der durch 'ACCHR' gegebenen Ordnung (vgl. Byte-Tabelle). Die Inhalte von Z und T gehen verloren. Die Routine läßt sich auch indirekt verwenden: legt man nämlich die Adressen von zwei Registern, welche Alpha-Ketten enthalten, nach X und Y, dann gelangt die Kette, welche nach der 'ACCHR'-Ordnung die erste ist, in das Register, dessen Adresse in X lag; die andere Kette kommt in das Register, dessen Adresse in Y war. Zweimal 'RDN' führt die beiden Adressen nach X und Y zurück.

```
01♦LBL "AL"     14 X<=Y?        27 R↑           40 ASTO L
02 CLA          15 RTN          28 RTN          41 ARCL L
03 CF 10        16 X<> IND T    29♦LBL 13       42 "⊢♦♦♦♦♦"
04 XEQ 14       17 X<> IND Z    30 R↑           43 ,
05 XEQ 14       18 X<> IND T    31 R↑           44 FC? 10
06 X=Y?         19 RTN          32 SF 10        45 GTO 14
07 XEQ 13       20♦LBL 12       33 XEQ 14       46 STO \
08 CLA          21 X<=Y?        34♦LBL 14       47 "⊢♦♦"
09 SF 10        22 GTO 12       35 SF 25        48♦LBL 14
10 X<=Y?        23 R↑           36 ARCL IND Y   49 STO [
11 CF 10        24 X<> Z        37 FC? 25       50 "⊢♦♦"
12 FC?C 25      25♦LBL 12       38 ARCL Y       51 X<> \
13 GTO 12       26 R↑           39 "⊢♦♦♦♦♦"     52 END
```

(107 Bytes)

Quelle: William Wickes (3735), Keith Jarett (4360) & William Cheeseman (4381) (PPC ROM).

12-6 Schnelle Stapel-Sortierung ("QSS")

Diese Routine arbeitet genauso wie die Routine "S1" aus 12-2; sie ist aber schneller und kürzer und benutzt überdies keine lokalen Marken. Soll "QSS" ebenso wie "S1" Flag 10 nach jedem Lauf automatisch löschen, muß Zeile 21 ('FC? 10') in 'FC?C 10' geändert werden.

```
01♦LBL "QSS"    07 R↑           13 RDN          19 FS? 10
02 X<Y?         08 X<Y?         14 X<Y?         20 X<> Z
03 X<>Y         09 X<>Y         15 X<>Y         21 FC? 10
04 R↑           10 RDN          16 R↑           22 R↑
05 X<Y?         11 X<Y?         17 X<Y?         23 END
06 X<>Y         12 X<>Y         18 X<>Y
```

(34 Bytes)

Eine noch kürzere Version, die nur absteigend sortiert, erhält man durch Löschung der Zeilen 19-21; diese kürzere Version kann aufsteigend sortieren, wenn alle 'X<Y?' (Zeilen 02, 05, 08, 11, 14 und 17) in 'X>Y?' abgeändert werden. - Quelle: Jeff R. Reasoner.

Kapitel 13

ZUFALLSZAHLEN

13-1 Zufallsgenerator (π + Keim)^3 ("RAN")

Vor Aufruf von "RAN" muß ein sog. 'Keim', d.i. eine die Folge von Zufallszahlen anführende Zahl zwischen 0 und 1, nach R_{10} gelegt werden. Beispiel: '0.05101975, Sto 10, XEQ "RAN"' liefert (in 'FIX 4'-Anzeige) '0.5416'; mit 'R/S' gelangt man zu '0.9649'; usw.

'LBL "RAN", RCL 10, PI, +, ENTER, X↑2, *, FRC, STO 10, RTN' (16 Bytes)

- Quelle: Jim Butterfield (1076) (65 Notes, V4N8P4).

13-2 Kürzester und schnellster Zufallsgenerator

Die Befehlsfolge 'LBL 14, RCL 10, R-D, FRC, STO 10, RTN' bildet einen außerordentlich kurzen und schnellen Zufallsgenerator. Ein zweites 'R-D' kann, falls gewünscht, eingefügt werden. Der 'Keim' für die Folge muß vorher ins Register R_{10} gelegt werden. Jede beliebige Zahl (außer 0 oder Vielfache von π) kann den Keim bilden, insbesondere also auch eine ganze Zahl. Der Generator erzeugt gleichmäßig verteilte Zahlen zwischen 0 und 1. - Quelle: Valentin Albillo (4747) (PPC CJ, V7N6P35).

13-3 Zufallsgenerator ("RN")

"RN" ist ein Zufallsgenerator, der gleichmäßig verteilte (Pseudo-)Zufallszahlen erzeugt, die im Bereich $0<r<1$ liegen. Vor Aufruf müssen ein 'Keim' (zwischen 0 und 1) in ein Register R_{mn} und dessen Adresse 'mn' ins Register X gelegt werden. Die erzeugte Zufallszahl liegt anschließend in X und im 'Keimregister' R_{mn}.

```
01♦LBL "RN"   06 +
02 RCL IND X  07 FRC
03 9821       08 STO IND Y
04 *          09 END
05 ,211327
```

(27 Bytes)

Eingabe:	Ausgabe:
T: T	T: Y
Z: Z	Z: Y
Y: Y	Y: Registeradresse
X: Registeradresse	X: nächste Zufallszahl

- Quelle: Don Malm (1362) (65 Notes, V4N8P4) (PPC ROM).

13-4 Gauß'scher Zufallsgenerator ("GN")

Diese Routine liefert Zufallszahlen, die einer Gauß'schen Verteilung (Glockenkurve) unterliegen. Mittelwert und Standardabweichung müssen (in R_{06} und R_{07}) durch den Benutzer festgelegt werden. "GN" ruft "RN" (13-3) auf und muß daher eine Registeradresse

in X vorfinden. Auch ein 'Keim' ist vor dem ersten Aufruf in das durch den Inhalt von X bestimmte Register zu setzen. "GN" hinterläßt *zwei* Zufallszahlen im Stapel: eine in X und eine in Y. Die Routine verlangt den DEG-Modus.

```
01♦LBL "GN"  10 *
02 XEQ "RN"  11 R↑
03 LN        12 RCL 07
04 ST+ X     13 *
05 CHS       14 P-R
06 SQRT      15 RCL 06
07 X<>Y      16 ST+ Z
08 XEQ "RN"  17 +
09 360       18 END
```

(35 Bytes)

Eingabe:
T: T
Z: Z
Y: Y
X: Registeradresse
L: L
R_{06}: Mittelwert
R_{07}: Standardabweichung

Ausgabe:
T: Registeradresse
Z: Registeradresse
Y: Zufallszahl Nr. 2
X: Zufallszahl Nr. 1
L: Mittelwert

- Quelle: Kiyoshi Akima (3456) & John Kennedy (918) (PPC CJ, V7N8P11) (PPC ROM).

Will man die Beschränkung auf den DEG-Modus beseitigen, muß man Zeile 09 ('360') durch '1, ASIN, *, 4' ersetzen. Diese Änderung kostet lediglich 1 Byte. In jedem trigonometrischen Modus werden dann der Funktion 'P-R' die Werte im richtigen Format zur Verfügung gestellt, also stets aus dem entsprechend repräsentierten Vollkreis stammend. - Quelle: Larry Trammell (6824).

13-5 Indirekte Adressierung zum Test von Zufallsgeneratoren ("TR" & "RNG")

Die Funktion 'ISG IND X' kann dazu benutzt werden, Zufallsgeneratoren, die Zahlen aus dem Bereich $0<r<1$ abliefern, zu testen. Die erzeugte Zufallszahl wird mit 10 multipliziert, um die erste Dezimalziffer nach dem Komma vor das Komma zu schieben; dann setzt 'ISG IND X' das dieser Ziffer entsprechende Register R_{0k} $(0 \leq k \leq 9)$ um jeweils 1 hoch. Diese Zählregister R_{00}-R_{09} müssen zu Anfang gelöscht werden. "TR" ist ein solches Testprogramm. Es löscht die Register R_{00}-R_{12} und fordert zur Eingabe der Anzahl auf, die bestimmt, wie oft der Zufallsgenerator (hier "RNG") aufgerufen werden soll, sowie zur Eingabe eines Keimes, der in R_{10} abgelegt wird. Während des Programmlaufs unterrichtet die Anzeige über die Anzahl der bereits erzeugten Zahlen. Anschließend ertönt der 'BEEP', und der Rechner schaltet sich aus. Das Register R_{0k} $(0 \leq k \leq 9)$ enthält nach Abschluß eine Zahl, die besagt, wie häufig k als erste Nachkommastelle in den durch "RNG" erzeugten Zufallszahlen aufgetreten ist. Statt "RNG" kann auch ein beliebiger anderer Generator verwendet werden. Die Ergebnisse eines Tests von "RNG" für 100 bzw. 1000 Zahlen sind unten rechts aufgeführt; in beiden Fällen lautet der Keim '0.2579846319'.

```
01♦LBL "TR"          17♦LBL 13
02 CF 21             18 XEQ "RNG"
03 FIX 0             19 10
04 12                20 *
05 STO 00            21 ISG IND X
06 CLX               22 BEEP
07♦LBL 14            23 1
08 STO IND 00        24 ST+ 12
09 DSE 00            25 VIEW 12
10 GTO 14            26 DSE 11
11 "ANZ. WUERFE ?"   27 GTO 13
12 PROMPT            28 BEEP
13 STO 11            29 PSE
14 "KEIM-ZAHL ?"     30 OFF
15 PROMPT            31 END
16 STO 10
```

(78 Bytes)

```
01♦LBL "RNG"
02 RCL 10
03 997
04 *
05 FRC
06 STO 10
07 END
```

(17 Bytes)

Reg.	'Würfe'	
	100	1000
R00	13	103
R01	6	94
R02	14	118
R03	13	99
R04	8	105
R05	9	92
R06	9	88
R07	8	107
R08	9	104
R09	11	90

R_{10}: Keim R_{11}: Schleifen-Zähler R_{12}: Anzahl durchgeführter 'Würfe'.

- Quelle: Bill Kolb (265) & John Dearing (2791) (PPC CJ, V7N4P16).

Kapitel 14

GEBROCHENE ZAHLEN UND RUNDUNGSRECHNUNG

14-1 Umwandlung Dezimalzahl → Bruch ("DF")

Diese Routine nimmt die in X liegende Dezimalzahl und gibt sie als Näherungsbruch p/q aus: p in Y, q in X. Die Güte der Näherung hängt von der in R_{07} gespeicherten Zahl 'n' ($0 \leq n \leq 9$) ab: das Ergebnis der Division von p durch q stimmt auf mindestens 'n' Nachkommastellen mit dem Eingabewert überein.

```
01♦LBL "DF"     13♦LBL 07     25 STO 10     37 X≠0?
02 STO 08       14 RDN        26 RCL 08     38 GTO 07
03 INT          15 1/X        27 *          39 RCL Z
04 0            16 ENTER↑     28 FIX 0      40♦LBL 08
05 STO 09       17 INT        29 RND        41 RCL 10
06 1            18 -          30 STO Z      42 SIGN
07 STO 10       19 RCL 09     31 RCL 10     43 ST* 10
08 RCL 08       20 RCL 10     32 /          44 *
09 R↑           21 STO 09     33 RCL 08     45 RCL 10
10 X=Y?         22 LASTX      34 -          46 END
11 GTO 08       23 *          35 FIX IND 07
12 ST- Y        24 +          36 RND
```

(61 Bytes)

- Quelle: John Kennedy (918) (PPC CJ, V7N8P11; PPC ROM).

14-2 Hilfsprogramm für "DF" ("DFD")

Dieses Programm 'bedient' die Routine "DF" (aus 14-1). Sie arbeitet sowohl ohne als auch mit Drucker (ein- oder ausgeschaltet) (Anwendung des in 3-12 gegebenen Tips). "DFD" fordert zur Eingabe von 'n' für die Approximationsgüte und zur Eingabe der umzuwandelnden Dezimalzahl auf, führt "DF" aus, formatiert die beiden Ergebniswerte und zeigt das Resultat als Bruch an; anschließendes 'R/S' gibt dann noch den Wert des Näherungsbruches dezimal bekannt. Mit '←' gelangt man ins 'FIX 2'-Format; mit 'R/S' beginnt man eine neue Runde. Sollen 'n' oder die Dezimalzahl unverändert bleiben, können die entsprechenden Aufforderungen mit 'R/S' übersprungen werden. Beispiel: für n = 5 und die Dezimalzahl π = 3.141592654 ergibt sich der auf (mindestens) 5 Nachkommastellen (tatsächlich sogar 6 Stellen) genaue Näherungsbruch 355/133 = 3.141592920.

```
01♦LBL "DFD"    06 RCL 08       11 CF 29      16 FIX 9      21 ARCL X
02 RCL 07       07 "DEZIMAL?"   12 CLA        17 SF 29      22 FIX 2
03 "N ?"        08 PROMPT       13 ARCL Y     18 PROMPT     23 PROMPT
04 PROMPT       09 XEQ "DF"     14 "├/"       19 /          24 GTO "DFD"
05 STO 07       10 FIX 0        15 ARCL X     20 CLA        25 END
```

(61 Bytes)

- Quelle: John Dearing (2791).

14-3 Kürzen von Brüchen ("RED")

Tastet man 'Zähler, ENTER, Nenner, XEQ "RED"', so weist die Routine zunächst den größten gemeinsamen Teiler vor (will man dies ausschließen, müssen die Zeilen 16-19 gelöscht werden) und hält dann mit der Anzeige des gekürzten Bruches an.

```
01♦LBL "RED"    08 MOD          15 CF 29        22 "⊢/"
02 STO 01       09 X≠0?         16 "GGT="       23 ARCL 01
03 X<>Y         10 GTO 01       17 ARCL X       24 AVIEW
04 STO 00       11 +            18 AVIEW        25 FIX 2
05 X<>Y         12 ST/ 00       19 PSE          26 SF 29
06♦LBL 01       13 ST/ 01       20 CLA          27 CLX
07 STO Z        14 FIX 0        21 ARCL 00      28 END
```

(53 Bytes)

- Quelle: John Dearing (2791) (PPC CJ, V7N4P32).

14-4 Rundung mit vorgegebener Genauigkeit ("NF")

Man lege die zu rundende Zahl 'z' nach Y und die Zahl 'g', welche die Genauigkeit der Rundung vorschreibt, nach X; dann 'XEQ "NF"'. Ist 'z' negativ, muß 'g' ebenfalls negativ gewählt werden. Das Rundungsergebnis 'r' erfüllt die Ungleichung ABS[z-r] $\leq$ ABS[g/2]. Beispiele: '24.56, ENTER, 1, XEQ "NF"' liefert '25.00'; '2.29, ENTER, .5, XEQ "NF"' liefert '2.50'; '12.79, CHS, ENTER, 3, 1/X, CHS, XEQ "NF"' liefert '-12.67'.

```
'LBL "NF", /, LASTX, X<>Y, .5, +, INT, *, RTN'
```

(15 Bytes)

14-5 Rundung auf den nächsten Bruchteil

Eine Rundung auf den nächsten Bruchteil verläuft nach folgender Regel: (1) Multiplikation mit dem Kehrwert des Bruchteils; (2) Addition (Subtraktion) von 0.5, wenn das Ergebnis positiv (negativ) ist; (3) ganzen Anteil wählen und durch den Kehrwert des Bruchteils dividieren. Beispiele:

Positive Zahl auf die nächste ganze Zahl runden: '..., .5, +, INT, ...'.
Positive Zahl auf die nächste Hälfte runden: '..., 2, *, .5, +, INT, 2, /, ...'.
Negative Zahl auf das nächste Drittel runden: '..., 3, *, .5, -, INT, 3, /, ...'.
Positive oder negative Zahl auf das nächste Viertel runden:

```
'..., CF 14, X<0?, SF 14, 4, *, .5,
FC? 14, +, FS?C 14, -, INT, 4, /, ...'.
```

14-6 Treppenfunktionen herstellen

Eine positive Zahl wird (im 'FIX'-Modus) durch die Befehlsfolge '..., .99, +, INT, ...' auf die nächste ganze Zahl aufgerundet, sobald ihr gebrochener Anteil mindestens 0.01 beträgt: '13.00'→'13', '13.01'→'14'. Im 'FIX 3'-Modus wählt man '.999', im 'FIX 4'-Modus '.9999' usw.

Soll eine positive Zahl - unabhängig vom Anzeige-Modus - aufgerundet werden, sobald ihr gebrochener Anteil ≠ 0 ist, leistet die Befehlsfolge '..., ENTER, FRC, X≠0?, SIGN, +, INT, ...' das Gewünschte. Negative Zahlen rundet die Folge, sofern ein gebrochener Anteil existiert, ab.

14-7 Bruchrechnung ("F+", "F-", "F*", "F/", "RE" & "MX")

Dieses Programm setzt sich aus einer Anzahl von Routinen zusammen, die zur Addition, Subtraktion, Multiplikation, Division und Kürzung von Brüchen vorgesehen sind, einschließlich der Verwandlung eines unechten Bruches in seine Gestalt als 'ganze Zahl + echter Bruch'.

"F+", "F-", "F", "F/":* Addition, Subtraktion, Multiplikation und Division von Brüchen. Die Eingabe besteht aus den Zählern (nach T, Y) und Nennern (nach Z, X), die Ausgabe aus Zähler p (in Y) und Nenner q (in X); ist Flag 10 gesetzt, wird p/q in der Anzeige vorgewiesen. Beispiel: um '2/5 : -3/4' zu berechnen, tastet man '2, ENTER, 5, ENTER, 3, CHS, ENTER, 4, XEQ "F/"'. Anschließend liegt der Nenner 15 in X und der Zähler -8 in Y; bei gesetztem Flag 10 wird "-8/15" angezeigt.

"RE": Kürzung von Brüchen. Die Eingabe besteht aus Zähler und Nenner (nach Y und X), die Ausgabe liegt genauso; ist Flag 10 gesetzt, wird der gekürzte Bruch in der Anzeige vorgewiesen. Beispiel: um '45/925' zu kürzen, tastet man '45, ENTER, 925, XEQ "RE"'. Anschließend liegt 185 in X und 9 in Y; bei gesetztem Flag 10 wird "9/185" angezeigt.

"MX": Umwandlung eines unechten Bruches (Zähler größer als Nenner) in seine Gestalt als 'ganze Zahl + echter Bruch'. Die Eingabe besteht aus Zähler und Nenner (nach Y und X). Die Zahlen können positiv und negativ sein. Die Ausgabe besteht aus dem ganzen Anteil (in X) und dem Zähler und Nenner des echten Bruches (in Z und Y); ist Flag 10 gesetzt, wird die 'gemischte Gestalt' in der Anzeige vorgewiesen. Beispiel: '-747/126' wird mit '747, CHS, ENTER, 126, XEQ "MX"' in '-5 13/14' verwandelt und bei gesetztem Flag 10 gerade so angezeigt; -5 liegt in X, 14 in Y und 13 in Z.

"GD": größter gemeinsamer Teiler (vgl. Routine 15-13).
"QR": Vielfaches des Nenners und Rest (vgl. Routine 15-12).
"VA": Alpha-Register betrachten (vgl. Routine 4-1).

Vorschlag für Tastenzuweisungen (A-E, H-J):

'A'	'B'	'C'	'D'	'E'
"F+"	"F-"	"F*"	"F/"	"RE"
		"QR"	"GD"	"MX"
		'H'	'I'	'J'

```
01♦LBL "F-"     28 CF 29        55 RTN          82 X=0?
02 CHS          29 CLA          56♦LBL "MX"     83 GTO 14
03♦LBL "F+"     30 ARCL Y       57 CF 11        84 "⊢ "
04 ST* T        31 "⊢/"         58 FS?C 10      85 ARCL X
05 X<> Z        32 ARCL X       59 SF 11        86 "⊢/"
06 *            33 FIX 2        60 XEQ 12       87 ARCL Y
07 ST+ Z        34 SF 29        61 X<>Y         88♦LBL 14
08 X<> L        35 XEQ "VA"     62 CF 14        89 X<> Z
09 *            36 RTN          63 X<0?         90 FIX 2
10 GTO 12       37♦LBL "GD"     64 SF 14        91 SF 29
11♦LBL "F/"     38♦LBL 13       65 ABS          92 XEQ "VA"
12 X<>Y         39 MOD          66 X<>Y         93 RTN
13♦LBL "F*"     40 LASTX        67 XEQ "QR"     94♦LBL "QR"
14 ST* Z        41 X<>Y         68 X<>Y         95 X<>Y
15 X<> T        42 X≠0?         69 FS?C 14      96 STO J
16 *            43 GTO 13       70 CHS          97 X<>Y
17 X<>Y         44 +            71 LASTX        98 MOD
18♦LBL "RE"     45 RTN          72 X<>Y         99 ST- J
19♦LBL 12       46♦LBL "VA"     73 FS?C 11      100 LASTX
20 RCL Y        47 SF 25        74 SF 10        101 ST/ J
21 RCL Y        48 PRA          75 FC? 10       102 CLX
22 XEQ 13       49 SF 25        76 RTN          103 X<> J
23 ST/ Z        50 FS?C 21      77 FIX 0        104 X<>Y
24 /            51 CF 25        78 CF 29        105 END
25 FC? 10       52 AVIEW        79 CLA
26 RTN          53 FC?C 25      80 ARCL X
27 FIX 0        54 SF 21        81 X<> Z
```

(219 Bytes)

- Quelle: John Kennedy (918) (bearbeitet nach PPC CJ, V7N8P8-11 & PPC ROM).

14-8 Umwandlung Dezimalzahlen ↔ 'englische Brüche' ("F-D" & "D-F")

Diese Routine wandelt Dezimalzahlen in 'englische Brüche' um und umgekehrt. 'englische Brüche' sind Brüche, deren Nenner aus Potenzen von 2 bestehen, also 1/8, 3/16, 5/32 usw. Die Routine ist nützlich, wenn man mit solchen Brüchen rechnen muß: man wandelt in Dezimalzahlen um, führt mit ihnen die Arithmetik durch und wandelt dann zurück. *Umwandlung Bruch→Dezimalzahl:* 'ganzer Anteil des Bruches (auch wenn er 0 ist), ENTER, Zähler, ENTER, Nenner, XEQ "F-D"'. Als Ergebnis erhält man die Dezimalzahl. *Umwandlung Dezimalzahl→Bruch:* 'Dezimalzahl, XEQ "D-F"'. Danach enthält der Stapel in X, Y, Z und T den Nenner, den Zähler, den ganzen Anteil des Bruches bzw. die ursprüngliche Dezimalzahl.

So wie die Routine geschrieben ist, approximiert "D-F" die Dezimalzahl einem 'englischen Bruch' auf das nächstliegende 32-tel, also mit einem Fehler, der kleiner als 1/64 ist. Will man z.B. nur auf das nächste 16-tel approximieren, muß Zeile 10 ('32') in '16' geändert werden; Sinngemäßes gilt für andere Näherungswerte. Beispiel: Wieviel Zoll betragen - als 'englischer Bruch' ausgedrückt - 60% von 5 1/2 Zoll? Lösung: '5, ENTER, 1, ENTER, 2, XEQ "F-D", 60, %, XEQ "D-F"'; Antwort also: "3 5/16" Zoll.

```
01♦LBL "F-D"    16 GTO 02    31 ST* Z     46 ARCL Y
02 /            17 X=Y?      32 *         47♦LBL 03
03 +            18 GTO 04    33♦LBL 02    48 X<> Z
04 RTN          19 ENTER↑    34 CLA       49 RDN
05♦LBL "D-F"    20♦LBL 01    35 R↑        50 FIX 2
06 ENTER↑       21 CLX       36 INT       51 SF 29
07 FRC          22 2         37 X=0?      52 PROMPT
08 FIX 0        23 ST/ Z     38 X=Y?      53 RTN
09 CF 29        24 /         39 ARCL X    54♦LBL 04
10 32           25 ENTER↑    40 "├ "      55 CLX
11 ENTER↑       26 FRC       41 RDN       56 1
12 X<> Z        27 X=0?      42 X=0?      57 ST+ T
13 *            28 GTO 01    43 GTO 03    58 CLX
14 RND          29 CLX       44 ARCL X    59 GTO 02
15 X=0?         30 2         45 "├/"      60 END
```

(96 Bytes)

- Quelle: Richard Kimmel (6003) (PPC CJ, V7N10P24).

14-9 Gerade Rundung ("ER")

Die Systemfunktion 'RND' rundet auf, wenn die erste 'wegzurundende' Ziffer '5' lautet und ihr nur Nullen folgen. Die untenstehende kleine Routine rundet in solchen Fällen so auf bzw. ab, daß die letzte 'übrigbleibende' Ziffer stets gerade ist.

'LBL "ER", 2, /, RND, 2, *, RTN' (12 Bytes)

Beispiele:

Zahl	"ER"	"RND"
0,545	0,54	0,55
0,555	0,56	0,56
-0,565	-0,56	-0,57
-0,575	-0,58	-0,58

Kapitel 15

ARITHMETIK UND ALGEBRA

15-1 Summe ganzer Zahlen ("ΣI")

Diese Routine berechnet die Summe $x = 1 + 2 + 3 + \ldots + n$ nach der Formel $x = [n(n+1)]/2 = (n^2 + n)/2$. Beispiel: die in Zinsproblemen auftauchende 78 ergibt sich aus den 12 Monaten eines Jahres: $1 + 2 + \ldots + 12 = 78$.

```
'LBL "ΣI", X↑2, LASTX, +, 2, /, RTN'
```
(12 Bytes)

- Quelle: Richard Nelson (1) (65 Notes, V1N4P3).

15-2 Summe von Quadratzahlen ("ΣS")

Die Routine berechnet die Summe der ersten n Quadratzahlen $x = 1^2 + 2^2 + \ldots + n^2$ ohne Schleife nach der Formel $x = [n(n+1)(2n+1)]/6$. Beispiel: für $n = 10$ ergibt sich $x = 385$. - Quelle: Dom Tocci (189) (65 Notes, V2N8P3).

```
'LBL "ΣS", X↑2, LASTX, +, LASTX, 2, *, 1, +, *, 6, /, RTN'
```
(18 Bytes)

Eine andere Version benutzt die Registerarithmetik und Register L:

```
'LBL "ΣS", X↑2, ST+ X, LASTX, +, ST* L, LASTX, +, 6, /, RTN'
```
(18 Bytes)

- Quelle: Bill Kolb (265).

15-3 Summe von Kubikzahlen ("Σ3")

Diese Routine berechnet die Summe der ersten n Kubikzahlen $x = 1^3 + 2^3 + \ldots + n^3$ aus der Formel $x = [n(n+1)/2]^2$. Beispiel: für $n = 5$ ergibt sich $x = 225$.

```
'LBL "Σ3", X↑2, LASTX, +, 2, /, X↑2, RTN'
```
(13 Bytes)

- Quelle: Bill Kolb (265) (BP 67/97).

15-4 Summe der Ziffern einer ganzen Zahl ("ΣD")

Diese Routine ermittelt die Summe der Ziffern einer in X liegenden ganzen Zahl. Beispiel: die ganze Zahl 1.234.556.789 hat die Ziffernsumme 50.

```
01♦LBL "ΣD"     05 10          09 ST- 01      13 ST* 01
02 STO 01       06 /           10 X≠0?        14 RCL 01
03 ST- 01       07 ST+ 01      11 GTO 01      15 END
04♦LBL 01       08 INT         12 10
```
(29.Bytes)

- Quelle: John Kennedy (918) (PPC J, V5N7P4).

15-5 Beliebige Zahl in rein gebrochene oder ganze Zahl verwandeln ("-DEC" & "-INT")

Mit "-DEC" und "-INT" kann man die in X liegende Dezimalzahl in eine rein gebrochene bzw. ganze Dezimalzahl mit denselben Ziffern verwandeln. Beispiel: '123.45, XEQ "-DEC"' liefert im 'FIX 5'-Modus '0.12345'; mit XEQ '"-INT"' gelangt man zu '12345.00000'.

```
01♦LBL "-DEC"     06 RTN          11♦LBL 02     16 10
02♦LBL 01         07 10           12 INT        17 *
03 FRC            08 /            13 LASTX      18 GTO 02
04 LASTX          09 GTO 01       14 X=Y?       19 END
05 X=Y?           10♦LBL "-INT"   15 RTN
```

(39 Bytes)

- Quelle: James Davidson (547) (65 Notes, V3N1P2).

15-6 Umkehrung der Ziffernfolge ("IV")

Diese Routine kehrt die Reihenfolge der Ziffern einer in X liegenden ganzen Zahl um.

```
01♦LBL "IV"       05 FRC          09 10         13 GTO 01
02 STO 01         06 ST+ 01       10 ST* 01     14 RCL 01
03 ST- 01         07 LASTX        11 /          15 END
04♦LBL 01         08 INT          12 X≠0?
```

(27 Bytes)

- Quelle: James Davidson (547) (65 Notes, V2N10P10).

15-7 Fibonacci-Zahlen ("FB" & "FIB")

'XEQ "FB"' liefert die Folge der sog. Fibonacci-Zahlen: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,..., deren Bildungsgesetz darin besteht, daß jede Zahl in dieser Folge die Summe ihrer beiden Vorgänger ist.

```
01♦LBL "FB"       04 ENTER↑       07 PSE        10 LASTX
02 FIX 0          05 0            08 X<>Y       11 GTO 01
03 1              06♦LBL 01       09 +          12 END
```

(21 Bytes)

- Quelle: James Davidson (547) (65 Notes, V3N9P14).

Mit 'n, XEQ "FIB"' erhält man direkt (ohne Schleife) die n-te Zahl in der Folge der Fibonacci-Zahlen. Beispiel: '11, XEQ "FIB"' liefert '55' (= 21 + 34). Die absolute Genauigkeit der Routine ist beschränkt auf den Bereich Nr. 1 (= 0) bis Nr. 41 (= 102.334.155). Es wird nur der Stapel benutzt. - Quelle: Heinz Dalkowski.

```
01♦LBL "FIB"      08 /            15 ,5         22 SQRT
02 1              09 STO Z        16 R↑         23 /
03 -              10 ,5           17 -          24 INT
04 ENTER↑         11 +            18 X<>Y       25 END
05 5              12 X<>Y         19 Y↑X
06 SQRT           13 Y↑X          20 -
07 2              14 LASTX        21 5
```

(36 Bytes)

15-8 Den Inhalt von X mit kleinem Fehler behaften

(I) 'EEX, CHS, 8, + (oder -)'. (II) 'SQRT, X↑2'. Diese zweite Methode versagt häufig, insbesondere dann, wenn der Inhalt von X eine Quadratzahl ist. - Quelle: Bill Kolb (265) (BP 67/97). Vgl. auch 22-24.

15-9 Kettenrechnungen

Kettensubtraktion: Will man eine Konstante 'k' wiederholt von der Zahl 'n' abziehen, wird dies mit 'k, CHS, ENTER, ENTER, ENTER, n' vorbereitet. Dann führt man '+' so oft wie gewünscht aus.

Kettenaddition: Man verfährt wie im vorangegangenen Beispiel, jedoch ohne den Befehl 'CHS'.

Kettendivision: Will man eine Zahl 'n' wiederholt durch eine Konstante 'k' dividieren, wird dies mit 'k, 1/X, ENTER, ENTER, ENTER, n' vorbereitet. Dann führt man '*' so oft wie gewünscht aus.

Kettenmultiplikation: Man verfährt wie im vorangegangenen Beispiel, jedoch ohne den Befehl '1/X'.

- Quelle: HP Key Notes, V4N3P11.

15-10 Wiederholte Multiplikation mit oder Division durch 10

Will man 'n' wiederholt mit 10 multiplizieren, kann man so verfahren: 'n, ENTER, EEX, 3, X<>Y, %, %, %,...'. Für eine wiederholte Division durch 10 geht man ähnlich vor: 'n, ENTER, EEX, 1, X<>Y, %, %, %,...'. Ist der mit 10 zu verändernde Wert im Register R_{mn}, wird X<>Y durch 'RCL mn' ersetzt. - Quelle: Curt Rostenback (382) (PPC J, V5N3P15).

15-11 Kenngrößen einer Division ("I/")

Ist p/q ein Bruch, und liegen p in Y und q in X, so liefert 'XEQ "I/"' den (eindeutig bestimmten) ganzzahligen Anteil der Division von p durch q in X ab, den (eindeutig bestimmten) Rest, der bei dieser Division verbleibt, in Y; der Inhalt von Z bleibt erhalten, und q bleibt in L zurück. p und q müssen positiv sein, aber nicht ganzzahlig.

'LBL "I/", RCL Y, X<>Y, MOD, ST- Y, X<>Y, LASTX, /, RTN' (16 Bytes)

- Quelle: David Motto (2339) (PPC CJ, V7N7P14). - Vgl. auch 15-12.

15-12 Vielfaches des Nenners und Rest ("QR")

Ist p/q ein Bruch, und liegen p in Y und q in X, so liefert 'XEQ "QR"' den ganzzahligen Anteil der Division von p durch q in Y ab, den Rest in X; die Inhalte von T und Z bleiben erhalten, und q bleibt in L zurück. Ein Teil des Alpha-Registers, Register O, wird für Notizzwecke benutzt und anschließend gelöscht. Diese Tatsache hat auf

Texte im Alpha-Register solange keinen Einfluß, wie diese aus nicht mehr als 14 Zeichen bestehen. Will man synthetische Funktionen vermeiden, muß in den Zeilen 03, 06, 08 und 10 ein gewöhnliches Register, etwa R_{00}, angesprochen werden. - Vgl. auch 15-11.

```
01♦LBL "QR"     04 X<>Y         07 LASTX        10 X<> J
02 X<>Y         05 MOD          08 ST/ J        11 X<>Y
03 STO J        06 ST- J        09 CLX          12 END
```

(23 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM).

15-13 Größter gemeinsamer Teiler ("GCD" & "GD")

Diese Routinen berechnen den größten gemeinsamen Teiler der in X und Y liegenden Werte. "GCD" läßt den Inhalt von T im Stapel zurück; "GD" läßt die Inhalte von Z und T im Stapel zurück.

```
'LBL "GCD", LBL 01, STO Z, MOD, X≠0?, GTO 01,  +,  RTN'
```

(16 Bytes)

```
'LBL "GD", LBL 01, MOD, LASTX, X<>Y, X≠0?, GTO 01, +, RTN'
```

(15 Bytes)

- Quelle: John Kennedy (918) (PPC J, V6N5P31; PPC CJ, V7N8P8).

15-14 Primteiler, Primzahltest, Primzahltabelle, Primfaktorzerlegung ("PD", "TP", "GT" & "PF")

"PD" (Primteiler): Legt man eine ganze Zahl 'n' ($\geq$ 2) nach Y und eine Zahl 't' (= 2 oder ungerade und $\geq$ 3) nach X, so liefert 'XEQ "PD"' den ersten Primteiler von 'n', der $\geq$ t ist (mit der Einschränkung: sofern ein solcher existiert), nach X; 'n' bleibt in Y erhalten. Drückt man anschließend 'R/S', gibt die Routine den nächsten Primteiler von 'n'. *"TP" (Primzahltest):* 'XEQ "TP"' prüft die in X liegende Zahl auf Primzahleigenschaft. Die geprüfte Zahl bleibt in X liegen, so daß man bei Bedarf mit 'XEQ "PF"' fortfahren kann, wenn durch "NICHT PRIM" gemeldet wird, daß eine Zerlegung möglich ist. *"GT" (Primzahltabelle):* 'XEQ "GT"' erzeugt eine (mit 1 beginnende) Primzahltabelle. *"PF" (Primfaktoren):* 'XEQ "PF"' zerlegt die in X liegende Zahl in ihre Primfaktoren.

```
01♦LBL "PD"   12 2          23 ST/ Y          34 RTN          45 ST+ Y        56 PSE
02♦LBL 01     13 X=Y?       24 GTO 01         35♦LBL "GT"     46 XEQ 01       57 X=Y?
03 RCL Y      14 DSE X      25♦LBL "TP"       36 1            47 X=Y?         58 RTN
04 RCL Y      15 ABS        26 VIEW X         37 VIEW X       48 VIEW X       59 ST/ Y
05 /          16 +          27 2              38 PSE          49 RDN          60 GTO 05
06 X<Y?       17 GTO 01     28 XEQ 01         39 2            50 GTO 04       61 END
07 GTO 02     18♦LBL 02     29 " NICHT PRIM"  40 VIEW X       51♦LBL "PF"
08 FRC        19 RDN        30 X=Y?           41 PSE          52 2
09 X=0?       20♦LBL 03     31 ASHF           42 1            53♦LBL 05
10 GTO 03     21 RDN        32 AVIEW          43♦LBL 04       54 XEQ 01
11 CLX        22 RTN        33 X<>Y           44 2            55 VIEW X
```

(117 Bytes)

- Quelle: John Kennedy (918) (PPC CJ, V7N3P6; V7N9P11).

15-15 Nächste Primzahl ("NP")

Legt man eine ganze Zahl 'n' ($\geq$ 2) nach Y und eine Zahl 'd' (= 2 oder ungerade und $\geq$ 3) nach X, so liefert 'XEQ "NP"' den ersten Primteiler von 'n', der $\geq$ t ist (mit der Einschränkung: sofern ein solcher existiert), nach X; drückt man anschließend 'R/S', erscheint der nächste Primteiler; stets vorausgesetzt, 'n' enthält keine kleineren Primfaktoren als 'd'. "NP" arbeitet einwandfrei bis hinauf zu 10-ziffrigen positiven ganzen Zahlen. Will man eine vollständige Primfaktorzerlegung durchführen, tastet man 'n, ENTER, 2, XEQ "NP"'. Ist die Zahl 'n' prim, wird sie daraufhin nach X zurückgegegeben; ist sie nicht prim, werden jeweils auf 'R/S' hin die Primfaktoren der Reihe nach abgeliefert; mehrfache Faktoren erscheinen so oft, wie sie enthalten sind; sobald '1' auftaucht, ist die Zerlegung beendet. Alle Berechnungen finden allein im Stapel statt; numerische Datenregister werden nicht benutzt.

Beispiel: von der Zahl 40.013.933 sei bekannt, daß sie zwei Primfaktoren enthalte, deren einer > 5.000 ist. Dann muß auch der zweite Faktor > 5.000 sein, und die Zerlegung kann mit '40.013.933, ENTER, 5.001, XEQ "NP"' gestartet werden. Nach etwa 55 Sekunden hält das Programm mit '5.309' in X an. Zweimal 'R/S' gibt nacheinander '7.537' und '1'. Daher gilt: 40.013.933 = 5.309 × 7.537.

```
01♦LBL "NP"     09 R↑          17 X=Y?        25 ENTER↑
02♦LBL 11       10 R↑          18 SIGN        26 RTN
03 RCL Y        11 X<>Y        19 +           27 ST/ Y
04 SQRT         12 MOD         20 GTO 09      28 GTO 11
05 LASTX        13 X=0?        21♦LBL 10      29 END
06 X<> Z        14 GTO 10      22 R↑
07♦LBL 09       15 X<> L       23 LASTX
08 X>Y?         16 2           24 X>Y?
```

(43 Bytes)

- Quelle: Phi Trinh (6171) (PPC ROM).

15-16 Wert in X durch Exponenten oder Mantisse ersetzen ("XPN" & "MAN")

Mit 'XEQ "XPN"' ersetzt man den in X liegenden Wert 'w' durch seinen Exponenten zur Basis 10; mit 'XEQ "MAN"' ersetzt man ihn durch seine Mantisse. 'w' ist beschränkt auf den Bereich: $10^{-99} \leq w \leq 9{,}999.999.884 \times 10^{99}$.

```
01♦LBL "XPN"    08 RTN         15 RTN         22 GTO 01
02 LOG          09 X>0?        16♦LBL "MAN"   23 10
03 ENTER↑       10 RTN         17 ENTER↑      24 /
04 ENTER↑       11 X=0?        18 LOG         25♦LBL 01
05 FRC          12 RTN         19 INT         26 /
06 -            13 1           20 10↑X        27 END
07 X=Y?         14 -           21 X<=Y?
```

(43 Bytes)

- Quelle: Rob Jung (2455) (PPC J, V5N7P6) & John Martellaro (1896) (PPJ, V5N8P9).

15-17 Mantisse betrachten ("VMAN")

Will man alle 10 Mantissenziffern der in X liegenden Zahl sehen, setze man 'SCI 9' oder 'FIX 9' fest und führe 'XEQ "VMAN"' aus.

'LBL "VMAN", CLA, ARCL X, AVIEW, RTN' (13 Bytes)

- Quelle: HP Key Notes, V4N3P12.

15-18 Synthetisch Mantisse oder Exponenten ermitteln ("MANT" & "EXP")

'XEQ "MANT"' ersetzt den Wert 'w' in X durch seine Mantisse; die Inhalte von Y, Z, T und L bleiben unversehrt. 'XEQ "EXP"' ersetzt 'w' durch seinen Exponenten zur Basis 10; die Inhalte von Y, Z und T bleiben unversehrt. Der Trick dieser Routine beruht darauf, das letzte Byte der Zahl 'w' durch 50 (d.i. der Hexadezimalwert des Buchstabens P) zu ersetzen und dann - in Abhängigkeit vom Vorzeichen des Exponenten - durch E50 zu dividieren bzw. mit E50 zu multiplizieren.

```
01♦LBL "MANT"     07 STO \        13 ST/ [        19 X<> \
02 CLA            08 CLX          14 X<> [        20 LOG
03 CF 24          09  E50         15 RTN          21 END
04 STO [          10 SF 25        16♦LBL "EXP"
05 ASTO [         11 ST* [        17 XEQ "MANT"
06 "├P"           12 FC?C 25      18 ST/ \
```

(56 Bytes)

- Quelle: Roger Hill (4940) (PPC CJ, V7N8P2).

15-19 Synthetisch Mantisse betrachten, Mantisse ermitteln, Exponenten ermitteln ("VM", "EX" & "MT")

Mit 'XEQ "VM"' kann man die volle Mantisse der in X liegenden Zahl betrachten; der Stapel bleibt unberührt; Flag 21 wird gelöscht. 'XEQ "EX"' und 'XEQ "MT"' ersetzen die in X befindliche Zahl durch ihren Exponenten bzw. ihre Mantisse. Auch hierbei bleibt der Stapel (bis auf X) unberührt. Zeile 37 ("├-") lautet dezimal 242, 127, 160.

```
01♦LBL "VM"       12 LASTX        23 X<> [        34 X≠0?
02 CF 21          13 RTN          24 ASHF         35 "├♦"
03 XEQ "MT"       14♦LBL "EX"     25 "├♦Δ"        36 X=0?
04 X<> [          15 CLA          26 ST- [        37 "├"
05 RDN            16 X≠0?         27 X<> [        38 CLX
06 VIEW ]         17 "ßß"         28 RTN          39 ST+ [
07 RCL d          18 INT          29♦LBL "MT"     40 X<> [
08 FIX 9          19 X≠0?         30 CLA          41 END
09 VIEW [         20 CLA          31 STO [
10 STO d          21 RDN          32 ASTO [
11 RDN ↓          22 LASTX        33 INT
```

(84 Bytes)

- Quelle: Roger Hill (4940) & Dave Kaplan (3678) (PPC ROM).

15-20 Schnelle Lösung von Bruch-Gleichungen ("R1", "R2" & "R3")

Diese Routinen lösen Bruch-Gleichungen der Form A/B = C/D für jede der 4 Größen. "R1" benutzt nur den Stapel; "R2" benutzt nur die Register R_{01}-R_{04}; "R3" benutzt wahlweise die Register R_{01}-R_{04} ('A' im USER-Modus) oder den Stapel ('B' im USER-Modus). Die 4 beteiligten Größen A, B, C und D werden in dieser Reihenfolge in den Stapel oder nach R_{01}-R_{04} gebracht; für die *unbekannte Größe* wird dabei *'0'* eingegeben.

```
01♦LBL "R1"   07 X≠0?
02 X<>Y       08 RDN
03 X≠0?       09 RDN
04 RDN        10 /
05 X≠0?       11 /
06 RDN        12 END
```

(19 Bytes)

```
01♦LBL "R2"   09 RDN
02 RCL 01     10 X≠0?
03 RCL 02     11 RDN
04 RCL 04     12 RDN
05 RCL 03     13 /
06 X≠0?       14 /
07 RDN        15 END
08 X≠0?
```

(22 Bytes)

```
01♦LBL "R3"   10 RDN
02♦LBL A      11 X≠0?
03 RCL 01     12 RDN
04 RCL 02     13 X≠0?
05 RCL 03     14 RDN
06 RCL 04     15 RDN
07♦LBL B      16 /
08 X<>Y       17 /
09 X≠0?       18 END
```

(27 Bytes)

- Quelle: Chris Stevens (3005) (PPC J, V5N7P4).

15-21 Kubikwurzel aus beliebiger reellen Zahl ("CURT")

Die folgende Methode erspart Bytes gegenüber Versionen, die einen Flagtest benutzen, um zu entscheiden, welches Vorzeichen das Ergebnis hat. Ändert man '3' in eine beliebige andere ungerade Zahl 'n', erhält man die n-te Wurzel.

```
'LBL "CURT", SIGN, LASTX, ABS, 3, 1/X, Y↑X, *, RTN'
```

(16 Bytes)

- Quelle: Valentin Albillo (4747).

15-22 Quadratwurzel aus verschiedenen Quadratsummen

Man benutze 'R-P', um $\sqrt{x^2 + y^2}$ zu berechnen; x in X und y in Y.
'R-P, LASTX, R-P' berechnet $\sqrt{2x^2 + y^2}$; setzt man noch 'X<>Y' davor, wird $\sqrt{x^2 + 2y^2}$ berechnet.
Mit 'a, ENTER, b, R-P, c, R-P, d, R-P,...' berechnet man $\sqrt{a^2 + b^2 + c^2 + d^2 + \ldots}$.
- Quelle: Bill Kolb (265) (BP 67/97).

15-23 Schnelle Faktorzerlegung von Fakultäten ("FFFF")

'XEQ "FFFF"' zerlegt für eine ganze Zahl 'n' ($3 \leq n \leq 9999$) die Fakultät n! in ihre Primzahlpotenzen. Es werden in jedem 'Arbeitsgang' zwei Primzahlpotenzen gedruckt; ist kein Drucker angeschlossen, erhält man auf 'R/S' hin jeweils zwei Primzahlpoten-

zen angezeigt, bis der 'BEEP' den Abschluß bekannt gibt. Beispiel: 12! = 2↑10 · 3↑5 · ·5↑2 · 7↑1 · 11↑1.

```
01♦LBL "FFFF"      22 GTO 05      43 X=0?        64 "⊢ , "
02 CF 00           23 XEQ 02      44 RTN         65 ARCL 01
03 SF 21           24 GTO 01      45 -           66 "⊢↑"
04 FIX 0           25♦LBL 02      46 1           67 ARCL 03
05 CF 29           26 STO 01      47 +           68 FS? 00
06 ADV             27 ENTER↑      48 /           69 AVIEW
07 "TEILER VON "   28 ENTER↑      49 GTO 00      70 FC?C 00
08 ARCL X          29 SQRT        50♦LBL 03      71 SF 00
09 "⊢ FAKLT.:"     30♦LBL 00      51 CLX         72 RTN
10 AVIEW           31 ENTER↑      52 STO 03      73♦LBL 05
11 STO 02          32 FRC         53 RCL 02      74 FS?C 00
12 2               33 X=0?        54♦LBL 04      75 AVIEW
13 XEQ 02          34 RTN         55 RCL 01      76 SF 29
14 3               35 -           56 /           77 FIX 2
15 XEQ 02          36 1           57 INT         78 RDN
16♦LBL 01          37 X=Y?        58 ST+ 03      79 CLD
17 RCL 02          38 GTO 03      59 X≠0?        80 BEEP
18 RCL 01          39 *           60 GTO 04      81 END
19 2               40 /           61 FC? 00
20 +               41 ENTER↑      62 CLA
21 X>Y?            42 FRC         63 FS? 00
```

(143 Bytes)

- Quelle: Joel Lichtenwalner (2957) (PPC J, V5N8P46).

15-24 Komplexe Zahlen im Stapel multiplizieren ("MC")

Diese Routine multipliziert zwei komplexe Zahlen $a_k + ib_k$ ($k = 1,2$), wenn sie wie folgt im Stapel abgelegt sind: a_1 in X, b_1 in Y, a_2 in Z und b_2 in T. Nach Ausführung liegt der Realteil des Ergebnisses in X und der Imaginärteil in Y. Die Routine benutzt keine numerischen Datenregister und verwendet weder trigonometrische Funktionen noch 'P-R' oder 'R-P', wodurch sie besonders schnell ist: 0,6 Sekunden Ausführungszeit.

```
01♦LBL "MC"        05 X<> Z       09 ST* L       13 RDN
02 STO L           06 ST* Z       10 X<> L       14 +
03 R↑              07 R↑          11 R↑          15 R↑
04 ST* Y           08 ST* Y       12 -           16 END
```

(30 Bytes)

- Quelle: Valentin Albillo (4747).

15-25 Reelle Wurzeln einer quadratischen Gleichung ("QEQ")

Diese Routine berechnet die beiden reellen Wurzeln $x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})/2a$ einer quadratischen Gleichung der Form $ax^2 + bx + c = 0$ (wenn sie existieren). Komplexe Wurzeln führen auf die Fehlermeldung "DATA ERROR". Es werden weder Datenregister noch Flags benutzt. Die Routine wird mit 'a, ENTER, b, ENTER, c, XEQ "QEQ"' aufgerufen. Die Wurzeln liegen anschließend in X und Y.

```
01♦LBL "QEQ"    05 /        09 X↑2      13 ST- Z
02 X<> Z        06 CHS      10 R↑       14 +
03 ST/ Z        07 ENTER↑   11 -         15 END
04 ST+ X        08 ENTER↑   12 SQRT
```

(27 Bytes)

- Quelle: Robert Groom (5127).

15-26 Wurzeln einer quadratischen Gleichung mit Anzeige ("QE" & "PRQE")

Mit 'a, ENTER, b, ENTER, c, XEQ "QE"' ermittelt man die Wurzeln einer quadratischen Gleichung $ax^2 + bx + c = 0$. Sind die Wurzeln reell, liegen sie nach Ausführung in X und Y (Flag 04 ist gelöscht); sind sie komplex, liegt der Realteil in X und der Imaginärteil in Y (Flag 04 ist gesetzt). Die Routine "QE" benutzt lediglich den Stapel, keine Datenregister; sie kann als Unterprogramm eingesetzt werden. - Quelle: Robert Groom (5127).

Fährt man mit 'R/S' fort, wird "PRQE" ausgeführt. Diese Routine formatiert, zeigt an und druckt die Ergebnisse von "QE". Sie setzt Flag 21 und löscht Flag 04. Soll dieser Programmteil automatisch ohne 'R/S' ausgeführt werden, müssen die Zeilen 21 und 22 getilgt werden. - Quelle: John Herzfeld (5428).

```
01♦LBL "QE"     11 R↑       21 RTN          31 AVIEW
02 CF 04        12 -        22♦LBL "PRQE"   32 RTN
03 X<> Z        13 X<0?     23 SF 21        33♦LBL 14
04 ST/ Z        14 SF 04    24 FS?C 04      34 CLA
05 ST+ X        15 ABS      25 GTO 14       35 ARCL X
06 /            16 SQRT     26 "WURZ. 1= "  36 "⊢ +- "
07 CHS          17 ST- Z    27 ARCL X       37 ARCL Y
08 ENTER↑       18 X<>Y     28 AVIEW        38 "⊢ I"
09 ENTER↑       19 FC? 04   29 "WURZ. 2= "  39 AVIEW
10 X↑2          20 +        30 ARCL Y       40 END
```

(94 Bytes)

15-27 Große Fakultäten ("BF")

Diese Routine approximiert n! ($n > 69$) durch Berechnung der ersten drei Glieder der Moivreschen Reihenentwicklung der Stirlingschen Näherung für n! Es werden keine numerischen Datenregister benutzt, und weil keine Schleife verwendet wird, ist die Ausführung schnell. 'n, XEQ "BF"'; anschließend liegen die Mantisse des Ergebnisses in X und der Exponent (zur Basis 10) in Y. Die Anzahl der signifikanten Mantissenziffern in X beträgt mindestens 10, vermindert um die Anzahl der Exponentenziffern in Y.

```
01♦LBL "BF"     10 PI       19 ENTER↑   28 LN
02 ENTER↑       11 *        20 R↑       29 /
03 LN           12 SQRT     21 X↑2      30 INT
04 1            13 LN       22 /        31 LASTX
05 -            14 +        23 30       32 FRC
06 RCL Y        15 R↑       24 /        33 10↑X
07 *            16 12       25 -        34 END
08 LASTX        17 *        26 +
09 ST+ X        18 1/X      27 10
```

(46 Bytes)

- Quelle: Larry Trammell (6824). - Hinweis: "BF" arbeitet auch mit beliebigen (nichtganzen) Werten in X und berechnet daher die Gammafunktion $\Gamma(x)$ für große x: 'x, ENTER, 1, -, XEQ "BF"' ergibt $\Gamma(x)$.

15-28 Y↑X für große Werte in X und Y ("BYX")

Sind die Inhalte x und y von X und Y so beschaffen, daß 'Y↑X' die Meldung "OUT OF RANGE" ergäbe, führt man 'XEQ "BYX" aus. Anschließend liegt die Mantisse von y^x in X und der Exponent (zur Basis 10) in Y. Die Genauigkeitsgrenzen werden durch die Genauigkeit der verwendeten Funktionen 'LOG' und '10↑X' festgelegt. Beispiel: 25^{75} wird mit '25, ENTER, 75, ENTER, XEQ "BYX"' bestimmt. Im 'FIX 9'-Modus erhält man für die Mantisse '7.006493122' und für den Exponenten (mit 'X<>Y') '104.0000000'; also ist $25^{75} = 7.006493122 \times 10^{104}$.

'LBL "BYX", X<>Y, LOG, *, INT, LASTX, FRC, 10↑X, RTN' (15 Bytes)

- Quelle: Bill Derrick (1393) (PPC J, V5N7P4).

15-29 Polynome ("POLY")

Diese Routine berechnet den Wert des Polynoms $ax^3 \pm bx^2 \pm cx \pm d$ gemäß dem sog. Horner-Schema. Die Vorzeichen '+' und '-' müssen dem jeweiligen Polynom entsprechend in die Routine eingesetzt werden. Will man Polynome höherer Ordnung berechnen, sind weitere Schrittfolgen 'RCL mn, + oder -, *' einzufügen.

```
01♦LBL "POLY"    05 ENTER↑    09 +/-       13 *
02 RCL 00        06 RCL 01    10 *         14 RCL 04
03 ENTER↑        07 *         11 RCL 03    15 +/-
04 ENTER↑        08 RCL 02    12 +/-       16 END
```

(25 Bytes)

R00: x
R01: a
R02: b
R03: c
R04: d

- Quelle: Bill Kolb (265) (BP 67/97).

15-30 Polynom-Berechnung ("PE")

Diese Routine berechnet ein Polynom beliebiger Ordnung k: $a_k x^k + a_{k-1} x^{k-1} + \ldots + a_2 x^2 + a_1 x + a_0$. Eine Kontrollzahl 'lmn.pqr' legt den Block fest, dessen Register die Koeffizienten $a_0, a_1, \ldots, a_k$ enthalten; der Koeffizient a_k des Terms höchster Ordnung liegt in R_{lmn}, das konstante Glied a_0 in R_{pqr}. Die Routine wird mit 'lmn.pqr, ENTER, x, XEQ "PE"' gestartet. Anschließend liegt der Wert des Polynoms in X und das verwendete Argument in Y. Beispiel: man berechne $3x^4 - 2x^3 - 5x^2 + 6x + 12$ für $x = 7$. Lösung: '3, STO 01, 2, CHS, STO 02, 5, CHS, STO 03, 6, STO 04, 12, STO 05, 1.005, ENTER, 7, XEQ "PE"'. Ergebnis: '6.326,00'.

```
01♦LBL "PE"     05 R↑          09 ISG Y       13 END
02 STO Z        06 *           10 GTO 01
03 CLX          07 RCL IND Y   11 R↑
04♦LBL 01       08 +           12 X<>Y
```

(24 Bytes)

- Quelle: Larry Trammell (6824).

15-31 Polynom-Multiplikation ("P*")

Diese Routine füllt die Register eines durch eine Registeradresse festgelegten Blokkes A mit den Koeffizienten eines Polynoms, welches durch Multiplikation zweier anderen Polynome entsteht. Die den Block A bestimmende Kontrollzahl liegt am Schluß in X und R_{03}. Das Produktpolynom kann daher anschließend für spezielle Argumente mit "PE" (15-30) berechnet werden. Die Koeffizienten der beiden Ausgangspolynome müssen vorher in zwei Blöcken B und C abgelegt werden, und zwar so, daß jeweils das niedrigste Register von B und C den Koeffizienten des Terms höchster Ordnung enthält. Ferner sind R_{01}-R_{03} wir folgt zu füllen:

R_{01}: Kontrollzahl '$l_1 m_1 n_1 . p_1 q_1 r_1$' für Block B (Polynom Nr. 1)
R_{02}: Kontrollzahl '$l_2 m_2 n_2 . p_2 q_2 r_2$' für Block C (Polynom Nr. 2)
R_{03}: Adresse des ersten Registers von Block A (Produktpolynom).

Die Routine verändert den Inhalt von R_{01}; sie benutzt R_{00} und Flag 10. Ab R_{04} sind die Register frei verfügbar für die Ein- und Ausgabeblöcke. Die Anzahl der für das Produktpolynom benötigten Register ist um 1 größer als der Grad dieses Polynoms.

Beispiel: Man berechne die Koeffizienten des aus dem Produkt $(3x^3 + 2x^2 - 5)(4x^2 + 6)$ hervorgehenden Polynoms. Die Koeffizienten des ersten Polynoms lauten 3, 2, 0, -5, die des zweiten Polynoms 4, 0, 6. Sie seien in R_{04}-R_{07} bzw. R_{08}-R_{10} abgelegt (vgl. die rechterhand aufgelisteten Register). Das Produktpolynom hat den Grad 5 ($3x^3 \cdot 4x^2 = 12x^5$); seine 6 Koeffizienten sollen nach R_{11}-R_{16} gelangen (also muß noch '11' in R_{03} gelegt werden). Die Mindestanzahl der benötigten Register beträgt somit 17.

R01: 4.007
R02: 8.010
R03: 11
R04: 3
R05: 2
R06: 0
R07: -5
R08: 4
R09: 0
R10: 6

Man führe nun 'XEQ "P*"' aus. Nach dem Programmlauf liegt '11.016' (die Kontrollzahl für den Ausgabeblock A) in X. Eine Durchsicht der Register R_{11}-R_{16} (u.U. mit "PRREGX") ergibt

R11= 12, R12= 8, R13= 18, R14= -8, R15= 0, R16= -30.

Das Produktpolynom lautet also

$$12x^5 + 8x^4 + 18x^3 - 8x^2 - 30.$$

Liegt "PE" im Speicher, läßt sich dieses Polynom sofort für spezielle Werte berechnen: 'RCL 03, 2, XEQ "PE"' ergibt '594,00'; 'RCL 03, 3, XEQ "PE"' ergibt '3.948,00'.

```
01♦LBL "P*"     12 ISG 03     23 RTN          34 FC? 10
02 RCL 03       13♦LBL 00     24♦LBL 02       35 ST+ IND Y
03 INT          14 ISG 01     25 CLX          36 X<> L
04 STO 00       15 GTO 02     26 STO IND Y    37 ISG Y
05 STO 03       16 RDN        27 GTO 01       38♦LBL 00
06 SF 10        17 DSE X      28♦LBL 12       39 ISG Z
07♦LBL 01       18 ↓ E3       29 RCL IND Z    40 GTO 12
08 RCL 02       19 /          30 X<>Y         41 CF 10
09 RCL 03       20 RCL 00     31 *            42 END
10 RCL IND 01   21 +          32 FS? 10
11 XEQ 12       22 STO 03     33 STO IND Y
```

(71 Bytes)

- Quelle: Larry Trammell (6824).

15-32 Addition und Subtraktion von Dezibel-Werten ("dB+" & "dB-")

'dB_1, ENTER, dB_2, XEQ "dB+"' addiert zwei Dezibel-Werte dB_1 und dB_2; für eine Subtraktion ist 'XEQ "dB-"' auszuführen. Die Routine benutzt keine Datenregister.

```
01♦LBL "dB+"    08 ABS          15 "⊢ dB"       22 X<>Y
02 XEQ 00       09♦LBL 01       16 AVIEW        23 10↑X
03 +            10 LOG          17 RTN          24 X<>Y
04 GTO 01       11 10           18♦LBL 00       25 10↑X
05♦LBL "dB-"    12 *            19 10           26 END
06 XEQ 00       13 CLA          20 ST/ Z
07 -            14 ARCL X       21 /
```

(53 Bytes)

- Quelle: HP-41C Users' Library Solutions, Heating, Ventilating & Air Conditioning, S. 65-68.

15-33 Gebrochene Gammafunktion ("FG")

Die Gammafunktion ist eine Verallgemeinerung der Fakultätsfunktion auf nichtganze Argumente. Die im Lösungsbuch 'Höhere Mathematik', S. 32-36, aufgelistete Routine gilt nur für Argumente ≥ 1 und gibt lediglich approximative Ergebnisse. In vielen Anwendungen hat man jedoch nur Argumente, die ganze Vielfache von 0.5 sind. Die Routine "FG" liefert exakte Ergebnisse für 0.5, 1, 1.5, 2, 2.5, ... , 70.5. Man tastet 'x, XEQ "FG"' und bekommt das Ergebnis; für Folgeberechnungen reicht 'x, R/S'. Zur Berechnung wird nur der Stapel benutzt. Auf unzulässige Argumente wird man durch die Meldung "UNZUL. X" hingewiesen.

```
01♦LBL "FG"   08 X=Y?      15 X≠Y?     22 GTO 04   29 SQRT        36 GTO 05
02♦LBL 05     09 GTO 02    16 GTO 01   23 ST* Z    30 R↑          37♦LBL 02
03 ENTER↑     10 RDN       17 1        24 1        31 *           38 1
04 ENTER↑     11 2         18 R↑       25 +        32 GTO 00      39 -
05 X<=0?      12 *         19 ,5       26 GTO 03   33♦LBL 01      40 FACT
06 GTO 01     13 ENTER↑    20♦LBL 03   27♦LBL 04   34 "UNZUL. X"  41♦LBL 00
07 INT        14 INT       21 X=Y?     28 PI       35 PROMPT      42 END
```

(66 Bytes)

- Quelle: HP Key Notes, V5N2P2.

15-34 x kubieren ("X↑3")

Diese Routine legt die dritte Potenz von x nach X, und zwar schneller als 'x, ENTER, 3, Y↑X', läßt die Inhalte von Y, Z und T unversehrt und bewahrt x in L auf.

'LBL "X↑3", X↑2, X<>L, ST* L, X<>L, RTN' (15 Bytes)

- Quelle: HP Key Notes, V5N2P13.

15-35 Exponenten ermitteln ("XP")

Diese Routine legt den Exponenten von x nach X, bewahrt x in Y auf und läßt den Inhalt von Y in Z und T zurück. Man erhält exakte Ergebnisse bis 9.999 998 733 E99.

'LBL "XP", ENTER, ABS, LOG, 100, +, INT, 100, -, RTN' (19 Bytes)

- Quelle: HP Key Notes V5N2P10. Vgl. auch 15-16 bis 15-19.

Kapitel 16

GEOMETRIE, TRIGONOMETRIE UND ANALYSIS

16-1 Vektorlänge positiv machen

Will man einem Vektor, dessen Länge 'r' (Betrag) im Verlauf irgendeiner Rechnung (unzulässigerweise) als negative Zahl auftritt, wieder positive Länge geben, hilft diese Befehlsfolge: 'RCL φ, RCL r, P-R, R-P'. - Quelle: Dave Wilder (452) (BP 67/97).

16-2 Singularitäten der Funktion 'TAN'

Man beachte, daß im 'DEG'- bzw. im 'GRAD'-Modus tan 90° (bzw. 100°) zum Singularitätenwert '9.9999999 99' führt, und 'ATAN' diesen Wert korrekt in '90' bzw. '100' umkehrt. Im 'RAD'-Modus dagegen wird die Singularität $\pi/2$ *wegen der Präzision des HP-41* notwendigerweise verfehlt.

16-3 Winkel in Schranken einschließen

Hat man einen Winkel α, und sucht man einen anderen Winkel α' mit $\sin \alpha = \sin \alpha'$, aber im Bereich $-90^{\circ} \leq \alpha' \leq 90^{\circ}$, verwende man 'SIN, ASIN'. Mit 'COS, ACOS' erhält man α' im Bereich $0 \leq \alpha' \leq 180^{\circ}$, wobei $\cos \alpha = \cos \alpha'$ gilt. - Quelle: Dave Wilder (452) (BP 67/97).

16-4 Einen Winkel mit 'MOD' eingrenzen

Will man einen negativen Winkel oder einen Winkel größer als 360° in einen entsprechenden Winkel φ aus dem Bereich $0 \leq \varphi < 360^{\circ}$ abändern, benutzt man - im 'DEG'-Modus - einfach '360, MOD'; im 'GRAD'-Modus wählt man '400, MOD', im 'RAD'-Modus 'PI, ST+ X, MOD'. Will man sich von der Beschränkung durch die trigonometrischen Modi befreien, wird '1, ASIN, 4, *, MOD' verwendet; will man dabei den Stapel nicht beanspruchen, wählt man '1, ASIN, ST+ X, ST+ X, MOD'. - Quelle: HP Key Notes, V5N2P6. - Will man einen Winkel zwischen -180° und 180° eingrenzen, leistet '180, +, 360, MOD, 180, -' das Gewünschte. Eine noch allgemeinere Lösung für das Eingrenzen von Ergebnissen findet man in 16-8.

16-5 Eine Anzeige von > 60 Minuten oder Sekunden ausschließen

'HR, HMS' vermeidet 'Überlaufanzeigen' bei Minuten und Sekunden. Beispiel: '2.6563, HR, HMS' führt zu korrekter Anzeige '3.0603'. - Quelle: John Martellaro (1896) (65 Notes, V4N9P1).

16-6 cos und sin zugleich berechnen

Die beiden Schritte '1, P-R' berechnen cos x und sin x in einem Arbeitsgang (x ursprünglich in X) und legen die Ergebnisse ins X- bzw. Y-Register. - Quelle: Joachim Bolz (401) (65 Notes, V2N9P25).

16-7 arctan y/x

Will man arctan y/x berechnen (y in Y, x in X) benutzt man statt '/, ATAN' besser 'R-P, RDN'. Auf diese Weise vermeidet man Arithmetikfehler bei x = 0 und kann überdies alle vier Fälle y/x, -y/-x, -y/x und y/-x unterscheiden. - Quelle: Dave Wilder (452) (BP 67/97).

16-8 Ein Ergebnis eingrenzen

Will man ein Ergebnis 'α' nur bezüglich eines festen Intervalles (a,b) angeben (z.B. einen Winkel α: $-180^\circ \leq \alpha < 180^\circ$), lege man α nach Z, b nach Y und a nach X und benutze

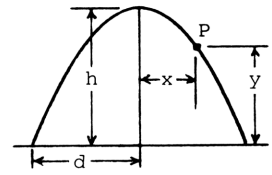
```
01♦LBL C        04 LASTX        07 RDN          10 R↑
02 -            05 -            08 X<>Y         11 +
03 X<>Y         06 LASTX        09 MOD          12 RTN
```

(13 Bytes)

- Quelle: Heinz Dalkowski.

16-9 Höhe eines Punktes auf einer Parabel ("PN")

Will man auf einer Parabel die Höhe 'y' eines Punktes 'P', der von der Mittelachse den Abstand 'x' hat, finden, führe man 'XEQ "PN"' aus. Die Routine fordert zur Eingabe von 'd', 'h' und 'x' auf und berechnet die Höhe $y = h \cdot [1 - (x/d)^2]$. Es wird nur der Stapel benutzt. Für Anschlußrechnungen mit neuen Werten 'x' reicht 'x, R/S'.




```
01♦LBL "PN"     08 PROMPT       15 -            22 LASTX
02 "D?"         09♦LBL 14       16 X<>Y         23 1
03 PROMPT       10 R↑           17 *            24 R↑
04 "H?"         11 /            18 "Y= "        25 GTO 14
05 PROMPT       12 LASTX        19 ARCL X       26 END
06 1            13 RDN          20 PROMPT
07 "X?"         14 X↑2          21 R↑
```

(44 Bytes)

- Quelle: John Dearing (2791).

16-10 Der cos-Satz

Der cos-Satz $c = \sqrt{a^2 + b^2 - 2ab \cos \gamma}$ wird besonders elegant durch 'RCL γ, RCL a, P-R, RCL b, -, R-P' ausgewertet. - Quelle: Bill Kolb (265).

16-11 Flächeninhalt und Länge eines Parabelsegmentes ("AP" & "SP")

'd, ENTER, h, XEQ "AP"' berechnet den Flächeninhalt des Parabelsegmentes; 'd, ENTER, h, XEQ "SP"' berechnet die Länge des zugehörigen Bogens.

```
01♦LBL "AP"     10 X<>Y         19 ENTER↑       28 X↑2
02 *            11 STO 02       20 ENTER↑       29 RCL 01
03 4            12 X↑2          21 RCL 01       30 ST+ X
04 *            13 X<>Y         22 ST+ X        31 /
05 3            14 X↑2          23 +            32 *
06 /            15 4            24 RCL 02       33 +
07 RTN          16 *            25 /            34 END
08♦LBL "SP"     17 +            26 LN
09 STO 01       18 SQRT         27 RCL 02
```

(48 Bytes)

16-12 Flächeninhalt eines regulären Polygons ("N↑S" & "N↑R")

Bezeichnet man mit 'n' die Anzahl der Seiten eines regelmäßigen Vielecks, mit 's' die Länge einer Seite und mit 'r' den Umkreisradius, so berechnen 'n, ENTER, s, XEQ "N↑S"' bzw. 'n, ENTER, r, XEQ "N↑R"' den Flächeninhalt aus den jeweils gegebenen Größen. Der Rechner muß im DEG-Modus sein.

```
01♦LBL "N↑S"    09 /            17 X↑2          25 2
02 X↑2          10 4            18 X<>Y         26 /
03 X<>Y         11 /            19 *            27 "A="
04 *            12 "A="         20 360          28 ARCL X
05 180          13 ARCL X       21 LASTX        29 AVIEW
06 LASTX        14 AVIEW        22 /            30 END
07 /            15 RTN          23 SIN
08 TAN          16♦LBL "N↑R"    24 *
```

(54 Bytes)

- Quelle: Hugh Kenner (103) (PPC CJ, V7N5P7).

16-13 Flächeninhalt eines regulären Polygons bei beliebigem trigonometrischem Modus ("AR")

Bezeichnet man mit 'n' die Anzahl der Seiten eines regelmäßigen Vielecks und mit 'r' den Umkreisradius, so berechnet 'n, ENTER, r, XEQ "AR"' den Flächeninhalt F nach der Formel $F = \frac{1}{2}\, nr^2 \sin(360^\circ/n)$. Die Schrittfolge '1, ASIN, 4, *' befreit von Beschränkungen bezüglich des trigonometrischen Modus (vgl. 13-4).

```
01♦LBL "AR"     05 *            09 X<>Y         13 2
02 1            06 RCL Z        10 X↑2          14 /
03 ASIN         07 /            11 *            15 END
04 4            08 SIN          12 *
```

(23 Bytes)

16-14 Koordinatenwechsel sphärisch ↔ rechtwinklig; Euler-Transformation ("R-S", "S-R" & "ET")

Die folgenden ersten beiden Routinen dienen zum Wechsel zwischen sphärischen und rechtwinkligen 3-dimensionalen Koordinaten. 'z, ENTER, y, ENTER, x, XEQ "R-S"' liefert r, θ und φ in den Registern X, Y und Z ab. 'φ, ENTER, θ, ENTER, r, XEQ "S-R"' liefert umgekehrt x, y und z in X, Y und Z ab. Beide Routinen lassen den Inhalt von T unzerstört.

```
01♦LBL "R-S"    04 X<> T    07♦LBL "S-R"    10 RDN
02 R-P          05 R-P      08 P-R          11 P-R
03 R↑           06 RTN      09 X<> T        12 END
```

(28 Bytes)

Die Wirkung einer beliebigen Anzahl von Drehungen des Achsenkreuzes im zweidimensionalen Raum auf einen in Polarkoordinaten (r,φ) gegebenen Punkt kann dadurch beschrieben werden, daß man der φ-Koordinate einfach einen bestimmten Winkel α zuschlägt. Im dreidimensionalen Raum benötigt man zur Beschreibung einer beliebigen Anzahl von Drehungen der drei Achsenkreuze drei Winkel, die sog. Eulerschen Winkel. Solche Euler-Transformationen werden häufig benutzt: z.B. für Programme, die perspektivisch zeichnen; bei Positionsangaben astronomischer Objektive am Nachthimmel; für die Orientierung von Kristallgittern bei der Untersuchung ihrer Brechungseigenschaften; usw. Die von Hewlett-Packard im HP-41 zur Verfügung gestellten Befehle sind hervorragend dazu geeignet, solche Berechnungen, die von Hand auszuführen einen das Fürchten lehren kann, umstandslos durchzuführen. Die Eulerschen Winkel α, β und γ (die hier, sphärische Koordinaten vorausgesetzt, Rotationen der z-, der y'- bzw. der z"-Achse entsprechen sollen) werden von der folgenden Routine "ET" in den Registern R_{01}-R_{03} erwartet. Mit 'φ, ENTER, θ, ENTER, r, XEQ "ET"' werden die Kugelkoordinaten (r, θ, φ) dann den durch die Winkel α, β und γ festgelegten Drehungen entsprechend transformiert, wobei nur der Stapel benutzt wird. Weil der Wert der Koordinate r unter den Drehungen unverändert bleibt, kann man dafür eine beliebige Konstante (≠ 0) wählen.

```
01♦LBL "ET"     07 P-R      13 X<>Y     19 RCL 03
02 P-R          08 R↑       14 P-R      20 ST- T
03 X<> Z        09 R-P      15 RDN      21 RDN
04 RCL 01       10 X<>Y     16 R-P      22 END
05 -            11 RCL 02   17 R↑
06 X<>Y         12 -        18 R-P
```

(31 Bytes)

- Quelle: Phil Fraundorf (1025) (PPC CJ, V7N8P26).

16-15 Rechtwinklige Dreiecke ganzzahliger Seitenlänge ("IT" & "PT")

Die Routine "IT" ermittelt nacheinander alle rechtwinkligen Dreiecke, deren Seitenlängen ganzzahlig sind und deren eine Seite eine vorgegebene Länge hat. Ist Flag 00 ge-

löscht, erklingt 'TONE 0', und die Ausführung wird beendet, sobald alle Dreiecke gefunden sind. Ist Flag 00 gesetzt, wird die eingegebene Zahl inkrementiert und die Ausführung fortgesetzt. Ohne Drucker gelangt man mit 'R/S' zur Anzeige des jeweils nächsten Dreiecks. Ist die Seite, deren Länge eingegeben wurde, eine Hypothenuse, erscheint ein Stern hinter der Zahl. Weil schiefwinklige Dreiecke als aus rechtwinkligen Dreiecken gemeinsamer Höhe zusammengesetzt betrachtet werden können, lassen sich mit der Routine auch Probleme mit schiefwinkligen Dreiecken ganzzahliger Seitenlänge behandeln.

Gebrauchsanweisung: 1. 'CF 00', wenn nur für eine bestimmte Seitenlänge alle Dreiecke gesucht sind; andernfalls 'SF 00'. 2. 'XEQ "IT"'. 3. 'n, R/S'. Alle rechtwinkligen Dreiecke, deren eine Seite die Länge 'n' hat, werden angezeigt/gedruckt. Ohne Drucker 'R/S' nach jeder Anzeige. 4. Ist Flag 00 gelöscht, erklingt beim Abschluß 'TONE 0', und in der Anzeige erscheint 0.00. Bei gesetztem Flag 00 fährt die Routine mit 'n+1', 'n+2',... fort. Will man abbrechen, führt man 'R/S, CF 00, XEQ 14' aus. Beispiel: 'XEQ "IT", 5, R/S' führt zur Anzeige von "5 12 13" und "5* 3 4". - Quelle: Richard Smith (4856) (PPC CJ, V7N4P30).

```
01♦LBL "IT"     30 ENTER↑       59♦LBL 10       88 /            117 "⊢ "        146 RCL 00
02 SF 21        31 STO 01       60 X↑2          89 STO 04       118 ARCL 03     147 2
03 FIX 0        32 2            61 STO 00       90 RCL 05       119 AVIEW       148 /
04 CF 29        33 /            62 FS? 01       91 X<>Y         120 RTN         149 RCL 03
05 "SL.?"       34 FRC          63 GTO 08       92 X<=Y?        121♦LBL 06      150 X<Y?
06 PROMPT       35 ,5           64♦LBL 09       93 GTO 05       122 2           151 GTO 04
07 1            36 X=Y?         65 RCL 00       94 RCL 04       123 ST/ 02      152 GTO 14
08 -            37 GTO 12       66 RCL 05       95 FRC          124 ST/ 03      153♦LBL 03
09 STO 06       38 RDN          67 RCL 01       96 X=0?         125 RTN         154 2
10 GTO 00       39 X=0?         68 X=Y?         97 XEQ 07       126♦LBL 05      155 ST/ 00
11♦LBL 14       40 GTO 11       69 GTO 05       98 4            127 RCL 01      156 RCL 00
12 FS? 00       41♦LBL 13       70 RDN          99 ST+ 05       128 2           157 RCL 03
13 GTO 00       42 "NEIN"       71 /           100 GTO 08       129 *           158 -
14 CLX          43 AVIEW        72 STO 04      101♦LBL 07       130 STO 00      159 STO 02
15 CLD          44 GTO 14       73 2           102 RCL 04       131 0           160 RCL 01
16 CF 01        45♦LBL 12       74 /           103 RCL 05       132 STO 03      161 X>Y?
17 SF 29        46 3            75 1           104 -            133♦LBL 04      162 GTO 14
18 FIX 2        47 RCL 01       76 X=Y?        105 2            134 1           163 CLA
19 TONE 0       48 X<Y?         77 GTO 24      106 /            135 ST+ 03      164 ARCL 00
20 RTN          49 GTO 13       78 RDN         107 STO 02       136 RCL 00      165 "⊢* "
21 GTO "IT"     50 2            79 FRC         108 RCL 05       137 RCL 03      166 ARCL 01
22♦LBL 00       51 *            80 X=0?        109 +            138 -           167 "⊢ "
23 CF 01        52 SF 01        81 XEQ 07      110 STO 03       139 LASTX       168 ARCL 02
24 2            53 GTO 10       82 2           111 FS? 01       140 *           169 AVIEW
25 STO 05       54♦LBL 11       83 ST+ 05      112 XEQ 06       141 SQRT        170 2
26 ADV          55 4            84 GTO 09      113 CLA          142 STO 01      171 ST* 00
27 1            56 RCL 01       85♦LBL 08      114 ARCL 01      143 FRC         172 GTO 04
28 ST+ 06       57 X<Y?         86 RCL 00      115 "⊢ "         144 X=0?        173 END
29 RCL 06       58 GTO 13       87 RCL 05      116 ARCL 02      145 GTO 03
```

(253 Bytes)

Die folgende kürzere Routine "PT" ermittelt dieselben Dreieckszahlen - auch pythagoräische Tripel genannt -, jedoch aufsteigend geordnet und ohne Wiederholung. Rechterhand sind die ersten 11 Tripel aufgelistet. Die Laufzeiten zwischen dem Auffinden

zweier Tripel sind sehr unterschiedlich. Vor Aufruf sollte man 'FIX 0, CF 29' festsetzen.

```
01♦LBL "PT"     15 CLA          29 GTO 00       43 GTO 03
02 3            16 ARCL 00      30♦LBL 02       44 -
03 STO 00       17 "⊦/"         31 1            45 GTO 04
04 4            18 ARCL 01      32 ST+ 00       46♦LBL 03
05 STO 01       19 "⊦/"         33 RCL 00       47 RDN
06 STO 03       20 ARCL 02      34 +            48 2
07♦LBL 00       21 AVIEW        35 STO 01       49 /
08 RCL 00       22♦LBL 01       36 RCL 00       50 1
09 RCL 01       23 RCL 01       37 X↑2          51 -
10 R-P          24 RCL 03       38 2            52♦LBL 04
11 STO 02       25 X<=Y?        39 /            53 STO 03
12 FRC          26 GTO 02       40 ENTER↑       54 GTO 00
13 X≠0?         27 1            41 FRC          55 END
14 GTO 01       28 ST+ 01       42 X=0?
```

(77 Bytes)

```
3/4/5
5/12/13
6/8/10
7/24/25
8/15/17
9/12/15
9/40/41
10/24/26
11/60/61
12/16/20
12/35/37
```

- Quelle: Heinz Dalkowski.

16-16 Funktionen von x und $\sqrt{1 \pm x^2}$

Im Bereich $-1 < x < 1$ berechnet man

$$\left.\begin{array}{l} \sqrt{1-x^2} \\ x/\sqrt{1-x^2} \\ \sqrt{1-x^2}/x \\ 1/\sqrt{1+x^2} \\ x/\sqrt{1+x^2} \end{array}\right\} \text{ am besten mit } \left\{\begin{array}{l} \text{'ACOS, SIN'} \\ \text{'ASIN, TAN'} \\ \text{'ACOS, TAN'} \\ \text{'ATAN, COS'} \\ \text{'ATAN, SIN'.} \end{array}\right.$$

- Quelle: Bill Kolb (265) (BP 67/97).

16-17 Hyperbolische Funktionen ("SINH", "COSH", "TANH", "ASINH", "ACOSH" & "ATANH")

Man tastet das Argument ein und führt die gewünschte Funktion aus. Beispiel: artanh x wird mit 'x, XEQ "ATANH"' berechnet. Datenregister werden nicht benutzt; ebensowenig lokale Marken oder interne Unterprogramme. Der Inhalt von Y bleibt in jedem Fall erhalten; bei "SINH" und "COSH" gilt das auch für den Inhalt von Z.

```
01♦LBL "SINH"   10 E↑X          19 ENTER↑       28 X↑2          37 X↑2          46 X<>Y
02 E↑X          11 ENTER↑       20 ENTER↑       29 1            38 1            47 +
03 ENTER↑       12 1/X          21 1/X          30 +            39 -            48 1
04 1/X          13 +            22 ST- Z        31 SQRT         40 SQRT         49 LASTX
05 -            14 2            23 +            32 +            41 +            50 -
06 2            15 /            24 /            33 LN           42 LN           51 /
07 /            16 RTN          25 RTN          34 RTN          43 RTN          52 SQRT
08 RTN          17♦LBL "TANH"   26♦LBL "ASINH"  35♦LBL "ACOSH"  44♦LBL "ATANH"  53 LN
09♦LBL "COSH"   18 E↑X          27 ENTER↑       36 ENTER↑       45 1            54 END
```

(102 Bytes)

- Quelle: John Kennedy (918) (PPC CJ, V7N8P11). Vgl. auch 16-21, 16-22.

16-18 Nullstellen-Bestimmung ("SV")

Diese Routine approximiert die Nullstelle einer Gleichung der Form $f(x) = 0$ (in Abhängigkeit vom Anzeige-Modus) mit Hilfe einer Newton'schen (Sekanten-)Methode. Man legt den Namen der Funktion f ins Register R_{06} und einen Schätzwert für die Nullstelle x_0 ins X-Register; dann 'XEQ "SV"'. Die Ausgabe in X besteht aus dem Wert x, für den der absolute Betrag $|f(x)|$ am kleinsten ist. Bei gesetztem Flag 10 kann man die aufeinanderfolgenden Approximationen von x in der Anzeige verfolgen. Die Güte der Approximation, bei der die Routine abbricht, hängt von der Anzahl der festgesetzten Nachkommastellen ab. Die Routine benutzt die Datenregister R_{06}-R_{09}.

```
01♦LBL "SV"     09 STO 08       17 RCL 08       25 RND
02 STO 07       10 RCL 07       18 X≠0?         26 X≠Y?
03 1            11 FS? 10       19 /            27 GTO 04
04 %            12 VIEW X       20 STO 09       28 RCL 07
05 STO 09       13 XEQ IND 06   21 X<> 07       29 END
06 CLST         14 ST* 09       22 ST+ 07
07♦LBL 04       15 ST- 08       23 RND
08 RCL Z        16 RCL 09       24 RCL 07
```

(45 Bytes)

- Quelle: John Kennedy (918), Richard Schwartz (2289) & Graeme Dennes (1757) (PPC ROM).

16-19 Integration ("IG")

Diese Routine entspricht der im HP-34C fest eingebauten Integrationsfunktion. Der Name des Integranden wird ins Register R_{10} gelegt, die untere Integrationsgrenze ins Y-Register und obere ins X-Register. Die Genauigkeit des Ergebnisses hängt von der Anzahl der festgesetzten Nachkommastellen ab. Die Routine approximiert sehr langsam! Sie benutzt die Register R_{10}-R_{18}. Bei gesetztem Flag 10 kann man die aufeinanderfolgenden Approximationen in der Anzeige verfolgen.

```
01♦LBL "IG"     19 STO 14       37 RCL 17       55 RCL 15       73 X<> Z
02 STO 17       20 RCL 11       38 +            56 RCL 16       74 /
03 X<>Y         21 CHS          39 XEQ IND 10   57 1,5          75 RCL IND 12
04 -            22 Y↑X          40 RCL 13       58 *            76 +
05 4            23 ST* 14       41 *            59 *            77 ISG 12
06 /            24 1            42 ST+ 15       60 RCL 14       78 GTOP
07 STO 16       25              43 1            61 *            79 DSE 13
08 ST- 17       26♦LBL 02       44 RCL 12       62♦LBL 03       80 GTO 03
09 ST- 17       27 STO 12       45 RCL 14       63 R↑           81 STO IND 12
10 0            28 X↑2          46 +            64 4            82 FS? 10
11 STO 15       29 -            47 X<Y?         65 *            83 VIEW X
12 STO 11       30 STO 13       48 GTO 02       66 ENTER↑       84 FS?C 09
13 STO 18       31 2            49 RCL 11       67 DSE Y        85 GTO 01
14 SF 09        32 +            50 STO 13       68 X<> Z        86 RND
15♦LBL "        33 RCL 12       51 18           69 ENTER↑       87 X≠Y?
16 1            34 *            52 STO 12       70 X<> IND 12   88 GTO 01
17 ENTER↑       35 RCL 16       53 1            71 ST- Y        89 LASTX
18 2            36 *            54 ST+ 11       72 RND          90 END
```

(129 Bytes)

- Quelle: Read Predmore (5184) (PPC ROM).

16-20 Erste Ableitung ("FD")

Diese Routine approximiert die erste Ableitung einer Funktion in einem Punkt x. Der Name der Funktion wird ins Register R_{11} gelegt, die Schrittweite (0.01 ist i.a. ein geeigneter Wert) ins Register R_{13} und Wert x ins X-Register; dann 'XEQ "FD"'.

```
01♦LBL "FD"     10 ST+ X        19 *            28 RCL 14
02 STO 12       11 +            20 ST- 14       29 +
03 RCL 13       12 XEQ IND 11   21 RCL 12       30 RCL 13
04 +            13 ST- 14       22 RCL 13       31 6
05 XEQ IND 11   14 9            23 3            32 *
06 ST+ X        15 ST* 14       24 *            33 /
07 STO 14       16 RCL 12       25 +            34 END
08 RCL 12       17 XEQ IND 11   26 XEQ IND 11
09 RCL 13       18 11           27 ST+ X
```

(52 Bytes)

- Quelle: Richard Schwartz (2289) (PPC CJ, V7N9P11-13, V7N10P10).

16-21 Verkürzte Berechnung von tanh x und artanh x ("TH" & "ATH")

Diese Routinen sind kürzer doch langsamer als "TANH" und "ATANH" aus 16-17.

'LBL "TH", E↑X, ATAN, ST+ X, COS, CHS, RTN' (13 Bytes)

'LBL "ATH", CHS, ACOS, 2, /, TAN, LN, RTN' (14 Bytes)

- Quelle: Kiyoshi Akima (3456).

16-22 sinh x für kleine x ("SINH")

Diese Routine ist länger doch für kleine Werte von x präziser als die Version in 16-17. Dies beruht auf dem Umstand, daß für kleine x die Funktion 'E↑X' gegen 1, 'E↑X-1' jedoch gegen x strebt. Dadurch werden Fehlberechnungen der Form SINH(x) = (1-1)/2 = 0 vermieden.

'LBL "SINH", E↑X-1, LASTX, CHS, E↑X-1, -, 2, /, RTN' (16 Bytes)

- Quelle: Kiyoshi Akima (3456).

Kapitel 17

ZAHLENSYSTEME

17-1 Synthetischer Wechsel der Basis ("BD" & "TB")

Diese beiden Routinen gestatten den Wechsel aus einer Basis b ($2 \leq b \leq 19$) in die Basis 10 und umgekehrt. Die jeweilige Zahl wird zur Basis b *im Alpha-Register* dargestellt und zur Basis 10 im X-Register. Die Basis b, aus der oder in die gewechselt werden soll, wird von beiden Routinen im Register R_{06} erwartet. Die für Basen $b > 10$ benötigten weiteren Ziffern werden durch die Buchstaben A-I dargestellt: A ≙ 10, B ≙ 11,..., H ≙ 17, I ≙ 18. Beispiele: (1) 5379_{10} soll zur Basis 13 dargestellt werden. '13, STO 06, 5379, XEQ "TB"' ergibt $25AA_{13} = 2 \cdot 13^3 + 5 \cdot 13^2 + 10 \cdot 13 + 10$. (2) $G5GB2_{17}$ soll zur Basis 10 dargestellt werden. '17, STO 06, ALPHA "G5GB2" ALPHA, XEQ "BD"' ergibt $1.365.714_{10} = 16 \cdot 17^4 + 5 \cdot 17^3 + 16 \cdot 17^2 + 11 \cdot 17 + 2$. Beachten Sie, daß auch für Basen $b < 10$ (also Zahlen ohne die Ziffern A-I) die Eingabe ins Alpha-Register erfolgt, wenn in die Basis 10 gewechselt werden soll: '7, STO 06, ALPHA "66" ALPHA, XEQ "BD"' liefert $48_{10} = 6 \cdot 7 + 6$. - Quelle: George Eldridge (5575) (PPC CJ, V7N9P12; PPC ROM). Vgl. auch 17-7.

"BD" (Basis b in Basis 10):

Eingabe:	Ausgabe:
Alpha-R.: n_b	X-Reg.: n_{10}
R_{06}: b	Alpha-R.: gelöscht

"TB" (Basis 10 in Basis b)

Eingabe:	Ausgabe:
X-Reg.: n_{10}	Alpha-Reg.: n_b
R_{06}: b	X-Reg.: 0

```
01♦LBL "BD"    20 DSE X       39♦LBL 03      58 RDn
02 CLST        21 9           40 ENTER↑      59 RCL 06
03♦LBL 01      22 +           41 INT         60 /
04 "⊢ "        23 X<0?        42 RCL 06      61 INT
05 X<> ]       24 GTO 02      43 MOD         62 X≠0?
06 X=0?        25 X<>Y        44 9           63 GTO 03
07 GTO 01      26 RCL 06      45 -           64♦LBL 05
08 X<> [       27 *           46 X>0?        65 "⊢ "
09 R↑          28 +           47 ISG X       66 CLX
10 X<> \       29 ,           48♦LBL 04      67 RCL [
11 "⊢♦▲"       30 GTO 01      49 39          68 X≠Y?
12 X<> \       31♦LBL 02      50 +           69 GTO 05
13 RDN         32 RDN         51 10↑X        70 CLX
14 X<> [       33 CLA         52 STO ]       71 X<> ]
15  E          34 RTN         53 "⊢      "   72 X<> \
16 *           35♦LBL "TB"    54 CLX         73 STO [
17 39          36 "'          "   55 X<> ]   74 CLST
18 -           37 RCL [       56 X<> \       75 AVIEW
19 X>0?        38 X<>Y        57 STO [       76 END
```

(139 Bytes)

Hinweis: Zeile 11 lautet dezimal 243, 127, 0, 8; Zeile 36 lautet dezimal 254, 39, 32,...,32 (13 mal Leerzeichen); Zeile 53 fügt 6 Leerzeichen (dezimal 32) an.

17-2 Stapelbedienung für Wechsel in die Basis 10

Will man eine Zahl, die zur Basis b ($2 \leq b \leq 10$) dargestellt ist, zur Basis 10 darstellen, gibt es eine (auf dem Horner-Schema (vgl. 15-29/30) beruhende) Methode. Man lädt den gesamten Stapel mit der Basis b und arbeitet sich dann von links nach rechts durch die gegebene Zahl auf die folgende Weise hindurch: erste Ziffer eintasten, *, zweite Ziffer eintasten, +, *,..., vorletzte Ziffer eintasten, +, *, letzte Ziffer eintasten, +. Das Ergebnis ist die gegebene Zahl zur Basis 10. Beispiel: 72.305_8 wird so umgerechnet: '8, ENTER, ENTER, ENTER, 7, *, 2, +, *, 3, +, *, 0, +, *, 5, +'; Ergebnis: $29.893_{10} = 7 \cdot 8^4 + 2 \cdot 8^3 + 3 \cdot 8^2 + 5$. - Quelle: Paul Fields (3114) (PPC J, V6N7P23).

17-3 Darstellungswechsel oktal ↔ dezimal für reelle Zahlen ("ROCT" & "RDEC")

Diese Routinen erlauben den Wechsel von Dezimaldarstellung in Oktaldarstellung und umgekehrt für beliebige reelle Zahlen. In zehnziffrigen Zahlen können bei der Umwandlung Fehler in den letzten Stellen auftreten, insbesondere wenn die Maschinengenauigkeit während der Berechnung überschritten wird. Der Inhalt des Stapels wird zerstört, aber Datenregister werden nicht benutzt. Beispiel: π soll oktal dargestellt werden. Lösung: 'π, XEQ "ROCT"'; Ergebnis: $3{,}110375524_8 = 3 + \frac{1}{8} + \frac{1}{8^2} + 3 \cdot \frac{1}{8^4} + 7 \cdot \frac{1}{8^5} + 5 \cdot \cdot \frac{1}{8^6} + 5 \cdot \frac{1}{8^7} + 2\ \frac{1}{8^8} + 4 \cdot \frac{1}{8^9} = 3{,}141592652_{10}$.

```
01♦LBL "ROCT"     08 *          15♦LBL "RDEC"     22 *
02 ENTER↑         09 INT        16 ENTER↑         23 INT
03 INT            10 OCT        17 INT            24 DEC
04 OCT            11 1 E10      18 DEC            25 1073741824
05 RCL Y          12 /          19 RCL Y          26 /
06 FRC            13 +          20 FRC            27 +
07 1073741824     14 RTN        21 1 E10          28 END
```

(70 Bytes)

- Quelle: HP Key Notes, V4N1P7.

17-4 Schnelle Umwandlung dezimal → hexadezimal ("DX")

Diese Routine läßt sich nur auf ganze Zahlen 'n' zur Basis 10 im Bereich $0 \leq n \leq 65.535$ anwenden. Sie soll insbesondere dazu dienen, Hexadezimaladressen eines Speichers von bis zu 64K Bytes zu ermitteln. Man startet mit 'XEQ "DX", n, R/S'. Weitere Werte 'n' werden einfach mit 'n, R/S' ermittelt.

```
01♦LBL "DX"    18 "8"         35 "EING. D, R/S"   52 16
02 "0"         19 ASTO 08     36 PROMPT           53 *
03 ASTO 00     20 "9"         37♦LBL 01           54 ARCL IND X
04 "1"         21 ASTO 09     38 65535            55 FRC
05 ASTO 01     22 "A"         39 X<>Y             56 16
06 "2"         23 ASTO 10     40 X>Y?             57 *
07 ASTO 02     24 "B"         41 GTO 00           58 ARCL IND X
08 "3"         25 ASTO 11     42 FIX 0            59 FRC
09 ASTO 03     26 "C"         43 " "              60 16
10 "4"         27 ASTO 12     44 ARCL X           61 *
11 ASTO 04     28 "D"         45 "⊢ DEC"          62 RND
12 "5"         29 ASTO 13     46 AVIEW            63 ARCL IND X
13 ASTO 05     30 "E"         47 4096             64 "⊢ HEX"
14 "6"         31 ASTO 14     48 /                65 AVIEW
15 ASTO 06     32 "F"         49 CLA              66 STOP
16 "7"         33 ASTO 15     50 ARCL IND X       67 GTO 01
17 ASTO 07     34♦LBL 00      51 FRC              68 END
```

(148 Bytes)

- Quelle: John Kennedy (918) (PPC CJ, V7N4P22).

17-5 Synthetische Umwandlung hexadezimal → dezimal ("XD")

Die Anwendung dieser Routine ist auf zweiziffrige im Alpha-Register stehende Hexadezimalzahlen beschränkt. Sie liefert das Ergebnis in X ab und dient insbesondere zur Umwandlung der Hexadezimalzahlen der HP-41-Byte-Tabelle in die entsprechenden Dezimalzahlen. Die Eingabe ins Alpha-Register wird nicht auf Richtigkeit geprüft.

```
01♦LBL "XD"    09 ,9          17 +                25 LASTX
02 "⊢♦&"       10 ST* Z       18 RTN              26 ST/ I
03 RCL [       11 *           19♦LBL "0R"         27 CLX
04  E2         12 INT         20 X<>Y             28 X<> I
05 XEQ "0R"    13 X<>Y        21 STO I            29 X<>Y
06 29          14 INT         22 X<>Y             30 END
07 ST- Z       15 16          23 MOD
08 -           16 *           24 ST- I
```

(59 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM).

17-6 Schnelle synthetische Umwandlung dezimal → binär ("db")

Diese Routine wandelt ganze Dezimalzahlen bis zu 16.777.215 in Binärzahlen, dargestellt im Alpha-Register, um. Weil das Rücksprungadressen-Register a benutzt wird, darf "DB" auf höchstens zweiter Unterprogrammebene aufgerufen werden, andernfalls kann sich der Adreßzeiger 'verirren'. Es wird nur der Stapel benutzt; Datenregister werden nicht benötigt.

```
01♦LBL "db"     09 CLA          17 FC? IND a    25 *
02 OCT          10 ISG Y        18 "⊢0"         26 DSE Y
03 ENTER↑       11♦LBL 00       19 FS? IND a    27 GTO 00
04 LOG          12 FIX IND X    20 "⊢1"         28 FIX 2
05 INT          13 37,039       21 ISG a        29 AVIEW
06 STO Z        14 STO a        22 GTO 01       30 .END.
07 10↑X         15 RDN          23 FRC
08 /            16♦LBL 01       24 10
```

(58 Bytes)

- Quelle: HP Key Notes, V5N2P10.

17-7 Synthetischer Wechsel der Basis (">T" & ">b")

Für diese beiden Routinen gelten genau dieselben Beschreibungen wie für die Routinen aus 17-1 (">T" entspricht "BD", ">b" entspricht "TB"); nur sind die Versionen ">T" und ">b" kürzer (sie passen auf eine Kartenspur) und schneller. Zeile 11 lautet dezimal 243, 127, 0, 8.

```
01♦LBL ">T"     18 DSE X        35 FIX 0        52 10↑X
02♦LBL 00       19 -            36 CF 29        53 ASTO a
03 ,            20 9            37 ARCL 06      54 ASHF
04 "⊢ "         21 +            38 STO d        55 ASTO T
05 X<> ]        22 X<0?         39 RDN          56 ASHF
06 X=0?         23 GTO 01       40♦LBL 02       57 ASTO L
07 GTO 00       24 X<>Y         41 ENTER↑       58 STO [
08 X<> [        25 RCL 06       42 INT          59 ASHF
09 ,            26 *            43 RCL 06       60 ARCL a
10 X<> \        27 +            44 ST/ Z        61 ARCL T
11 "⊢♦A"        28 GTO 00       45 MOD          62 ARCL L
12 X<> \        29♦LBL 01       46 9            63 RDN
13 RDN          30 RDN          47 X<Y?         64 INT
14 X<> [        31 RTN          48 DSE X        65 X≠0?
15 HMS          32♦LBL ">b"     49 -            66 GTO 02
16 40           33 "'"          50 39           67 AVIEW
17 X>Y?         34 RCL d        51 +            68 END
```

(112 Bytes)

- Quelle: Gerard Westen (4780).

Kapitel 18

UMRECHNUNG VON MAßEINHEITEN UND ABKÜRZUNGSVORSCHLÄGE

18-1 Neugrad in und aus Altgrad und Bogenmaß wandeln

Es gilt: $\pi \triangleq 180^{\circ}$ Altgrad $\triangleq 200^{\circ}$ Neugrad. *G-D:* '.9, *'; *D-G:* '.9, /'; *G-R:* '.9, *, D-R'; *R-G:* 'R-D, .9, /'.

18-2 Fuß,Inch in Fuß (oder Yard) umrechnen ("FIN")

Eingabe von Fuß und/oder Inch in der Form fg,ih (z.B. 12,09 für 12 Fuß, 9 Inch), dann 'XEQ "FIN"'. Die zwei Schritte '3, /' vor 'RTN' rechnen Fuß in Yard um und können entfallen.

```
'LBL "FIN", ENTER, FRC, .12, /, X<>Y, INT, +, (3, /,) RTN'
```
(19 Bytes)

18-3 Näherungsfaktor für Meilen → Kilometer Umrechnung

Englische Meilen (oder Meilen/Stunde) werden mit dem Faktor 1.609344 multipliziert, um Kilometer (oder km/h) zu erhalten. Stattdessen kann man auch (Bytes sparend) mit dem Faktor 1,609437912 = ln 5 rechnen; das verursacht einen Fehler von nur 0,0058%. -- Quelle: Neil Murphy (6) (65 Notes, V1N2P3).

18-4 Einheiten englisch oder metrisch eingeben aber metrisch speichern

Die folgende Methode ist nur anwendbar, wenn alle Eingabewerte positiv sind (z.B. Gewicht oder Höhe). Man tastet die Werte negativ ein, wenn sie in englischen Einheiten gegeben sind, doch positiv, wenn sie metrisch gegeben sind. Im Programm steht der jeweilige Umrechnungsfaktor englisch → metrisch vor dem 'PROMPT', welchem 'X > 0?, 1, *' folgt. Das erste Beispiel unten fordert zur Eingabe von Gewicht (in Pound oder Kilogramm) auf, rechnet, falls nötig, in Kilogramm um und speichert dann den Wert in R_{01}; das zweite Beispiel arbeitet entsprechend für Inch bzw. Zentimeter.

```
'..., -.4536, "GEW?", PROMPT, X > 0?, 1, *, STO 01,...'
'..., -2.54, "H.?", PROMPT, X > 0?, 1, *, STO 02,...'
```

18-5 Schnelle Null

Soll eine Programmzeile den Zahleneintrag '0' bewirken, benutzt man besser das Dezimalkomma ',' statt '0', weil dies schneller ausgeführt wird. Dem Komma braucht keine weitere Ziffer zu folgen. ',' bewirkt nur den Eintrag einer '0', keine Löschung des X-Registers. - Quelle: Bill Kolb (265) (BP 67/97).

18-6 Synthetische schnelle Eins

Soll eine Programmzeile eine '1' ins Register X eintragen, benutzt man besser 'E' (Byte 1B, Dezimalwert 27) statt '1', weil dies schneller ausgeführt wird.

18-7 Multiplikation mit kleiner Zahl

Beispiel: statt 'EEX, 6, CHS, *' besser 'EEX, 6, /'. Dies spart 'Tastenarbeit', wenn von Hand ausgeführt, und ein Byte, falls im Programm angewendet. - Quelle: Bill Kolb (265) (BP 67/97).

18-8 Division durch 100

Statt '100, /' besser '1, % '. Dies spart zwei Bytes, und der Dividend bleibt im Y-Register zurück. - Quelle: John Martellaro (1896) (PPC J, V5N1P16).

18-9 Multiplikation mit $\sqrt{2}$

Statt '2, SQRT, *' besser 'ENTER, R-P'. Will man x durch $x \cdot \sqrt{2}$ ersetzen, ohne den Stapel anzuheben, verwende man 'X↑2, ST+ X, SQRT'. - Quelle: Bill Kolb (265) (BP).

18-10 $\frac{4}{3}\pi$

Statt '4, PI, *, 3, /' besser '240, D-R'. - Quelle: Bill Kolb (265) (BP).

18-11 $\pi/180$

Statt 'PI, 180, /' besser '1, D-R'; dies spart 3 Bytes ein. - Quelle: John Martellaro (1896) (PPC J, V5N1P16).

18-12 $1/y^x$

Statt 'Y↑X, 1/X' besser 'CHS, Y↑X'; dies wird schneller ausgeführt. - Quelle: Bill Kolb (265) (BP 67/97).

18-13 $y^{x/2}$

Statt '2, /, Y↑X' besser 'Y↑X, SQRT'; ein Byte gespart. - Quelle: Bill Kolb (265) (BP).

18-14 Umwandlung Grad + Dezimalminuten ↔ Dezimalgrad ("DM-D" & "D-DM")

Diese Routine unterstützt die Verwendung des Nautischen Jahrbuches, dessen Angaben in Grad und Dezimalminuten gemacht werden. Sie fordert sinnfällig zur Eingabe auf und versieht die Ausgabe mit Text, so daß stets Klarheit darüber besteht, ob Dezimalgrad oder Grad + Dezimalminuten einzugeben sind bzw. ausgegeben wurden. Will man Grad +

Dezimalminuten in Dezimalgrad umrechnen, ruft man "DM-D" auf (die Eingabe erfolgt im Format DDDMM,MM); mit 'XEQ "D-DM"' erfolgt die Umwandlung in umgekehrter Richtung. Wird die Routine nicht als Unterprogramm, sondern unabhängig verwendet, kann man mit 'R/S, R/S' jeweils zurückwandeln. Beispiel: 'XEQ "DM-D", 12345.67, R/S' wandelt $123^{\circ}45.67'$ in 123.7612° um; 'R/S, R/S' ergibt wieder "123 D. 45,67 M".

```
01♦LBL "DM-D"      12 INT          23 PROMPT      34 60
02 CF 29           13 +            24 ENTER↑      35 *
03 "DDDMM.MM ?"    14 FIX 4        25 INT         36 FIX 2
04 PROMPT          15 CLA          26 FIX 0       37 ARCL X
05 1 E2            16 ARCL X       27 CLA         38 "┣ M"
06 /               17 "┣ DEG"      28 ARCL X      39 AVIEW
07 ENTER↑          18 AVIEW        29 "┣ D. "     40 +
08 FRC             19 RTN          30 1 E2        41 END
09 .6              20♦LBL "D-DM"   31 *
10 /               21 CF 29        32 X<>Y
11 X<>Y            22 "DEC DEG ?"  33 FRC
```

(103 Bytes)

- Quelle: Hugh Kenner (103) (PPC CJ, V7N5P7, V7N6P35). Vgl. auch 18-16.

18-15 Zahlen in Abhängigkeit von ihrer Größe durch 1 oder 0 ersetzen

Sollen (von Hand oder Programm getätigte) Zahleneinträge durch '1' ersetzt werden, sobald sie größer oder gleich einer Konstanten k sind, und alle Einträge < k durch '0', kann man das durch diese Befehlsfolge erreichen: 'k, -, SIGN, X < 0?, 0'. Sie kann für jede Konstante k und jeden Bereich von Einträgen verwendet werden. Sollen die Einträge $\geq$ k durch '0' und die < k durch '1' ersetzt werden, muß man 'k, -, SIGN, X > 0?, 0, ABS' verwenden.

- Quelle: Heinz Dalkowski.

18-16 DDD.MM,MM ↔ DDD.DDDD

Die Routine 01 wandelt Grad + Dezimalminuten (Format DDD,MMMM) in Dezimalgrad (Format DDD,DDDD) um; die Routine 02 wandelt in umgekehrter Richtung um. Flag 02 muß gelöscht sein, wenn die Routine 01 aufgerufen wird. Beispiel: '123.4567, XEQ 01' liefert 123.7612° (= 123° 45,67').

'LBL 02, SF 02, LBL 01, INT, LASTX, FRC, .6, FS?C 02, 1/X, /, +, RTN'

(15 Bytes)

- Quelle: Bill Boulton (700) (PPC CJ, V8N1P22). Vgl. auch 18-14.

18-17 Umrechnung von Temperatureinheiten: Celsius ↔ Fahrenheit ("TEMP")

Bei gesetztem Flag 00 werden $^{\circ}C$ in $^{\circ}F$ umgerechnet, bei gelöschtem Flag 00 umgekehrt $^{\circ}F$ in $^{\circ}C$. Beispiel: 'SF 00, 50, XEQ "TEMP"' ergibt '122' ($^{\circ}F$); 'CF 00, 50, XEQ "TEMP"' ergibt '10' ($^{\circ}C$).

```
01♦LBL "TEMP"     05 1.8          09 ST- 01       13 END
02 40             06 ST* 01       10 -
03 +              07 /            11 FS? 00
04 STO 01         08 40           12 RCL 01
```

(29 Bytes)

- Quelle: James Davidson (547) (65 Notes, V3N9P14).

18-18 Umrechnung Celsius ↔ Fahrenheit im Stapel ("TMP")

Bei gesetztem Flag 00 werden ^{o}C in ^{o}F umgerechnet, bei gelöschtem Flag 00 umgekehrt ^{o}F in ^{o}C. Datenregister werden nicht benutzt.

'LBL "TMP", 40, +, 1.8, FC? 00, 1/X, *, 40, -, RTN' (21 Bytes)

- Quelle: Ron Ryen (205) (65 Notes, V2N5P9).

18-19 Umrechnung Celsius ↔ Fahrenheit in Abhängigkeit von der Interpretation der Eingabe ("TEM")

Diese Routine wandelt sowohl in ^{o}C als auch in ^{o}F um. Ist die Eingabe als Wert in ^{o}F zu verstehen, steht das Ergebnis (in ^{o}C) in X; ist die Eingabe als Wert in ^{o}C zu verstehen, steht das Ergebnis (in ^{o}F) in Y. Es werden weder Flags noch Datenregister benutzt.

```
01♦LBL "TEM"      04 32           07 1.8          10 +
02 ENTER↑         05 ST- Z        08 ST/ T        11 X<>Y
03 ENTER↑         06 X<>Y         09 *            12 END
```

(25 Bytes)

- Quelle: Bill Derrick (1393) (PPC J, V5N7P4).

18-20 Eingabe in ^{o}C oder ^{o}F (Spezialanwendung)

Im Bereich der Photographie oder klinischen Medizin kommen keine Temperaturen oberhalb 50°C vor. Jeder Wert darüber muß daher ein Wert in ^{o}F sein, jeder Wert darunter ein Wert in ^{o}C. (Die Schranke kann für andere Zwecke verändert werden.)

Fall I (Umrechnung in ^{o}C): '...,50, "TEMP?", PROMPT, X≤Y?, GTO 00, 32, -, 1.8, /, LBL 00,...'.

Fall II (Umrechnung in ^{o}F): '...,50, "TEMP?", PROMPT, X>Y?, GTO 00, 1.8, *, 32, +, LBL 00,...'. - Quelle: Cary Reinstein (2046) & Henry Casson (5047).

18-21 Umrechnung von Temperatureinheiten ("TC")

Die Routine benutzt keine Datenregister, Flags oder numerische Marken. Sie wandelt Werte *aus jeder in jede* der folgenden Maßeinheiten um: Celsius, Kelvin, Fahrenheit und Rankine (^{o}C, ^{o}K, ^{o}F bzw. ^{o}R). Gebrauchsanweisung: Der umzurechnende Wert wird eingegeben, dann führt man 'XEQ "TC"' (am besten der USER-Taste 'E' zugewiesen) aus;

anschließend *erst* die dem Eingabewert entsprechende USER-Taste ('A'-'D'), *danach* die dem gewünschten Ausgabewert entsprechende USER-Taste ('A'-'D') drücken. Für die nächste Umrechnung neuen Wert eingeben, dann 'R/S'.

°F	°C	°R	°K	"TC"
'A'	'B'	'C'	'D'	'E'

```
01♦LBL "TC"      10 GTO D      19 PROMPT     28 -
02 SF 27         11♦LBL B      20♦LBL A      29 GTO D
03 "F C R K"     12 273,15     21 1,8        30♦LBL C
04 PROMPT        13 +          22 *          31 1,8
05♦LBL A         14 GTO D      23 459,67     32 *
06 459,67        15♦LBL C      24 -          33♦LBL D
07 +             16 1,8        25 GTO D      34 END
08 1,8           17 /          26♦LBL B
09 /             18♦LBL D      27 273,15
```

(93 Bytes)

- Quelle: Nach HP-41C Users' Library Solutions 'Heating, Ventilating & Air Conditioning', S. 69-72.

18-22 %-Anteil ("%T")

Diese Routine berechnet den Prozentanteil des Wertes in X am Wert in Y: '50, ENTER, 35, XEQ "%T"' ergibt '70%' (70% von 50 sind 35). Die Inhalte von Y und Z bleiben erhalten, die von T und L gehen verloren. Mit einem zusätzlichen 'STO Z' hinter 'LBL "%T"' geht der Inhalt von Z verloren, aber der Wert aus X bleibt in Z und T zurück.

'LBL "%T", RCL Y, /, 100, *, RTN' (14 Bytes)

- Quelle: HP Key Notes V5N2P10.

Kapitel 19

STATISTIK UND KOMBINATORIK

19-1 Statistische Summation mit Häufigkeitsfaktor ("ΣF")

Diese Routine gestattet die statistische Summation (Σ+ und Σ-) unter Berücksichtigung der Häufigkeit der Meßwertpaare, so daß mehrfach auftretende Wertepaare x,y nur einmal eingegeben zu werden brauchen. "ΣF" setzt 'ΣREG 04' fest, benutzt die Flags 00, 21 und 27 und erfordert 10 Datenregister. Gebrauchsanweisung: 1. 'XEQ "ΣF"'. 2. Die (nur einmal am Anfang erscheinende) Aufforderung "X,↑,Y,↑,F" wird sinngemäß mit 'x_i, ENTER, y_i, ENTER, f_i' (f_i = Häufigkeit) beantwortet, und zwar für jedes Tripel x_i, y_i, f_i (i = 1,2,...,n). Jeder Eintrag wird mit der USER-Taste 'A' abgeschickt. Tastet man versehentlich 'R/S', mahnt die Meldung "A OD. C TAST." daran, 'A' zu benutzen. 3. Korrekturen werden mit 'C' vorgenommen: falsches Tripel x_i, y_i, f_i noch einmal eingeben. 'C' tasten. 4. Mit 'E' kann man Zwischen- oder Endwerte ausgeben. Ist der Drucker nicht angeschlossen, muß man jede weitere Ausgabe mit 'R/S' anfordern. 5. Will man Tripel dazurechnen, fährt man mit Schritt 2 fort. 6. Will man eine neue Berechnung beginnen, geht man zu Schritt 1 zurück.

Beispiel:

x	1	2	4	6
y	1	3	5	7
f	15	2	2	1

Ergebnisse:

ΣX= 33.00	ΣY↑2=132.00
ΣX↑2=91.00	ΣXY=109.00
ΣY=38.00	N=20.00

Bemerkung: Hinter 'LBL E' können weitere Berechnungen und Ausgabe-Befehle eingefügt werden, z.B. 'MEAN, "MW.X", XEQ 02, X<>Y, "MW.Y", XEQ 02', um die Mittelwerte von x und y zu erhalten.

Eingabe		Korrektur		Ergebnisse
'A'	'B'	'C'	'D'	'E'

```
01♦LBL "ΣF"          24 CHS        47 *          70 XEQ 02
02 ADV               25 ST+ 09     48 FS? 00     71 RCL 05
03 9                 26 RDN        49 CHS        72 "ΣX↑2"
04 STO 00            27 STO 02     50 ST+ 06     73 XEQ 02
05 CLX               28 RDN        51 RCL 01     74 RCL 06
06♦LBL 01            29 STO 01     52 X↑2        75 "ΣY"
07 STO IND 00        30 R↑         53 RCL 03     76 XEQ 02
08 DSE 00            31 R↑         54 *          77 RCL 07
09 GTO 01            32 ABS        55 FS? 00     78 "ΣY↑2"
10 CF 00             33 *          56 CHS        79 XEQ 02
11 SF 21             34 *          57 ST+ 05     80 RCL 08
12 SF 27             35 FS? 00     58 RCL 01     81 "ΣXY"
13 ΣREG 04           36 CHS        59 RCL 03     82 XEQ 02
14 "X,↑,Y,↑,F: A"    37 ST+ 08     60 *          83 RCL 09
15 PROMPT            38 RCL 02     61 FS?C 00    84 "N"
16♦LBL 00            39 X↑2        62 CHS        85♦LBL 02
17 "A OD. C TAST."   40 RCL 03     63 ST+ 04     86 "⊢="
18 PROMPT            41 *          64 RCL 09     87 ARCL X
19♦LBL C             42 FS? 00     65 STOP       88 AVIEW
20 SF 00             43 CHS        66 GTO 00     89 END
21♦LBL A             44 ST+ 07     67♦LBL E
22 STO 03            45 RCL 02     68 RCL 04
23 FS? 00            46 RCL 03     69 "ΣX"
```

(174 Bytes)

Register:

R00: Zähler
R01: x_i
R02: y_i
R03: f_i
R04: Σx
R05: Σx^2
R06: Σy
R07: Σy^2
R08: Σxy
R09: n

- Quelle: Nach 'Einfache Statistiken mit zwei Variablen', HP-41C Stat. Pac., S. 10-13.

19-2 Kehrwert von Summen über Kehrwerte ("ΣRECIP")

Die Daten können in beliebiger Reihenfolge eingegeben werden; Zwischenergebnisse lassen sich jederzeit ausgeben. Die untenstehende Auflistung gilt für die Gleichung

$$R = \frac{1}{\Sigma S + \Sigma\frac{1}{T} + \Sigma\frac{U}{2} + \Sigma\frac{V}{W}}$$

Für andere Gleichungen muß die Routine umgeschrieben werden. Sie benutzt nur den Stapel, keine Datenregister. Gebrauchsanweisung: 1. 'XEQ "ΣRECIP"' (der USER-Taste 'E' zugewiesen). Die Belegung der USER-Tasten wird angezeigt. 2. Will man 'S' (oder '1/T', 'U/2', 'V/W') hinzurechnen, gibt man 'S' (bzw. 'T', 'U', 'V, ENTER, W') ein und drückt 'A' (bzw. 'B', 'C', 'D'). 3. Schritt 2 wird solange wiederholt, bis man ein Ergebnis sehen will. 4. Zwischen- oder Endergebnisse erhält man mit 'R/S'. 5. Sollen weitere Werte hinzugerechnet werden, fährt man mit Schritt 2 fort. 6. Will man eine neue Berechnung beginnen, kehrt man zu Schritt 1 zurück. 7. Hat man nach Eingabe eines Wertes die Bedeutung der USER-Tasten vergessen, kann man kurz in den ALPHA-Modus schalten, um die Belegung noch einmal zu betrachten.

S	T	U	V,↑,W,	"ΣRECIP"
'A'	'B'	'C'	'D'	'E'

```
01♦LBL "ΣRECIP"     14♦LBL A        27 /            40 1/X
02♦LBL E            15 "S"          28 GTO 02       41 +
03 ADV              16 XEQ 00       29♦LBL D        42 1/X
04 SF 21            17 GTO 02       30 X<>Y         43 GTO 01
05 SF 27            18♦LBL B        31 "V"          44♦LBL 00
06 0                19 "T"          32 XEQ 00       45 "⊢="
07♦LBL 01           20 XEQ 00       33 X<>Y         46 ARCL X
08 " S T U V↑W"     21 1/X          34 "W"          47 SF 25
09 PROMPT           22 GTO 02       35 XEQ 00       48 PRA
10 " R="            23♦LBL C        36 /            49 CF 25
11 ARCL X           24 "U"          37♦LBL 02       50 END
12 AVIEW            25 XEQ 00       38 X<>Y
13 GTO 01           26 2            39 X≠0?
```

(108 Bytes)

19-3 'LASTX' kann bei Summation verlorengehen

Frühe HP-41C-Modelle (bis etwa zur Ser.-Nr. 1938A2000) hatten einen 'LASTX-Fehler': 'LASTX' ging verloren, wenn 'Σ+' oder 'Σ-' aufgerufen wurde. Falls Ihr Rechner diesen Fehler aufweist, müssen Sie im Anschluß an 'Σ+'- und 'Σ-'-Befehle noch 'STO L' drücken. Der Fehler wird beseitigt, wenn der Rechner zum Kundendienst gelangt. - Quelle: Bill Kolb (265).

19-4 Synthetischer Sucher für den Statistik-Block ("Σ?")

Mit 'XEQ "Σ?"' erhält man die Adresse des ersten Registers des Statistik-Blockes. Vgl. auch Routine 1-17.

```
01♦LBL "Σ?"     11 RCL c        21 FS?C 10      31 FS?C 15
02 CLA          12♦LBL 14       22 SF 07        32 SF 14
03 XEQ "C?"     13 STO [        23 FS?C 11      33 FS?C 16
04 RCL \        14 "⊢♦♦A"       24 SF 09        34 SF 15
05 XEQ 14       15 X<> [        25 FS?C 12      35 X<> d
06 CLA          16 X<> d        26 SF 10        36  E38
07 X<>Y         17 CF 01        27 FS?C 13      37 /
08 -            18 CF 02        28 SF 11        38 INT
09 RTN          19 CF 04        29 FS?C 14      39 DEC
10♦LBL "C?"     20 CF 07        30 SF 13        40 END
```

(87 Bytes)

- Quelle: Keith Jarett (4360) & Roger Hill (4940) (PPC ROM).

19-5 Synthetische Ermittlung von Σx und Σy ("RΣ")

Diese Routine simuliert die Funktion 'RCLΣ' des HP-67/97, indem sie Σx nach X und Σy nach Y bringt. Sie ist nützlich bei Anwendungen der Vektor-Analysis. Die vorstehende Routine "Σ?" wird als Unterprogramm aufgerufen.

```
'LBL "RΣ", XEQ "Σ?", 2, X<>Y, +, RCL IND X, RCL IND L, RTN'
```

(18 Bytes)

- Quelle: John Dearing (2791).

19-6 Sucher für den Statistik-Block ("ΣR")

Diese Routine ersetzt die Inhalte von Y und X durch die Werte Σy bzw. Σx aus dem Statistik-Block. Sie simuliert auf diese Weise die Funktion 'RCLΣ' anderer Rechner. Die Inhalte von X und Y bleiben in Z und T zurück; 'n' gelangt ins Register L. Der Statistik-Block selbst bleibt unversehrt. Die Routine arbeitet unabhängig von der Lage des Blockes, jedoch darf 'n' nicht '0' sein. Ein zusätzlicher Sucher für die Lage des ersten Registers des Blockes wird nicht benötigt, wodurch die Ausführungszeit nur etwa 1 Sekunde beträgt.

```
01♦LBL "ΣR"     04 Σ+       07 MEAN      10 ST* Y
02 0            05 0        08 X<> L     11 X<> L
03 ENTER↑       06 Σ-       09 ST* L     12 END
```

(23 Bytes)

- Quelle: Jürgen K. Cappel (6015) (PPC CJ, V8N2P16).

19-7 Summation von x und y 'per Hand' ("XYΣ")

Wollen Sie laufend die Werte x und y von Hand summieren und dazu nur den Stapel benutzen, gehen Sie folgendermaßen vor: 1. 'XEQ "XYΣ"', das löscht den Stapel. 2. 'y, ENTER, x, R/S'. 3. Schritt 2 sooft wie nötig wiederholen. 4. Σx steht in X; Σy kann mit 'X<>Y' betrachtet werden. 5. Will man weitere Paare addieren, 'X<>Y', dann zurück zu Schritt 2.

'LBL "XYΣ", CLST, LBL 14, ST+ Z, RDN, ST+ Z, RDN, STOP, GTO 14' (18 Bytes)

- Quelle: John Dearing (2791).

19-8 Block-Addition ("BP")

Diese Routine berechnet die Summe der Werte eines Registerblockes B, welcher durch eine Kontrollzahl 'abc.lmn' in X festgelegt ist. Bei gesetztem Flag 10 entsteht diese Summe im ersten Register des durch 'ΣREG' bestimmten Statistik-Blockes S; ein in diesem Register vorhandener Inhalt ist also im Ergebnis mit enthalten, es sei denn, man führt vorher 'CLΣ' aus. Die Summe der Quadrate der Werte des Blockes B wird dem Inhalt des zweiten Statistik-Registers zugeschlagen. Die Anzahl der Register des Blokkes B gelangt in das letzte (sechste) Register des Blockes S. X wird gelöscht. Wenn Flag 10 gelöscht ist, gelangt die Summe der Werte des Blockes B nach X und nach Register R_{abc+1}. In beiden Fällen bleiben die Inhalte des Blockes B selbst unversehrt.

```
01♦LBL "BP"     05 RCL IND 00     09 CLX        13 GTO 06
02 STO 00       06 FS? 10         10 FC? 10     14 FC? 10
03 0            07 Σ+             11 +          15 STO IND 00
04♦LBL 06       08 FS? 10         12 ISG 00     16 END
```

(31 Bytes)

- Quelle: Richard Nelson (1) (PPC CJ, V7N10P7).

19-9 Einfache Block-Addition ("B+")

Mit einer Kontrollzahl 'abc.lmn' in X zur Festlegung eines Registerblockes liefert 'XEQ "B+"' die Summe der Inhalte des Blockes. Es werden keine Flags benutzt und keine Datenregister verändert.

'LBL "B+", 0, LBL 06, RCL IND Y, +, ISG Y, GTO 06, RTN' (16 Bytes)

19-10 Synthetisch betriebener Statistik-Block ("ΣB")

Mit einer Kontrollzahl 'abc.lmn' in X zur Festlegung eines Registerblockes liefert 'XEQ "ΣB"' die folgenden Ergebnisse: Σx in X, Σx^2 in Y, n in Z und $\bar{x}$ in T. Das Alpha-Register wird benutzt.

```
01♦LBL "ΣB"     06 RCL IND X    11 ISG X        16 LASTX
02 CLA          07 ST+ [        12 GTO 08       17 RCL ]
03♦LBL 08       08 X↑2          13 RCL [        18 RCL [
04 ISG \        09 ST+ ]        14 RCL \        19 END
05 ""           10 RDN          15 /
```

(36 Bytes)

- Quelle: John Dearing (2791).

19-11 Block-Statistik ("BΣ")

'abc.lmnij' in Y sei eine Kontrollzahl, die einen Block I von y-Werten festlegt und 'k' in X sei die Adresse des ersten Registers eines zweiten Blockes II von x-Werten. Dann legt 'XEQ "BΣ"' die gewohnten Werte Σx, Σx^2, Σy, Σy^2, Σxy und n in den durch 'ΣREG' festgelegten Statistik-Block. Bei sorgfältiger Wahl der Kontrollzahlen können die x- und y-Werte in benachbarten Registern liegen; z.B. behandelt '2.00802, ENTER, 1.00002, XEQ "BΣ"' die Inhalte der Register R_{01}, R_{03}, R_{05} und R_{07} als x-Werte und die Inhalte der Register R_{02}, R_{04}, R_{06} und R_{08} als y-Werte. Die Routine "BΣ" kann auch als Matrixprogramm angesehen werden, weil sie das innere Produkt zweier Vektoren zu bilden vermag und somit zur Berechnung von Matrixprodukten benutzt werden kann: durch passende Wahl der Eingabe-Parameter lassen sich die Zeilen einer Matrix mit den Spalten einer zweiten Matrix multiplizieren. - Vgl. auch 10-28.

```
01♦LBL "BΣ"     05 RCL IND Y    09 ISG X        13 END
02 CLΣ          06 Σ+           10 STO X
03♦LBL 12       07 R↑           11 ISG Y
04 RCL IND Y    08 R↑           12 GTO 12
```

(26 Bytes)

- Quelle: John Kennedy (918) & Richard Schwartz (2289) (PPC ROM).

19-12 Permutationen und Kombinationen ("PERM" & "COMB")

"PERM" berechnet die Anzahl der Permutationen von 'n' Elementen, wobei nur jeweils 'k' Elemente zugelassen sind (Variationen von n Elementen zur k-ten Klasse): P(n,k) = n!/(n - k)!. Die Berechnung erfolgt mit 'n, ENTER, k, XEQ "PERM"'. Beispiel: P(73,4) = 26.122.320. "COMB" berechnet die Anzahl der Kombinationen von 'n' Elementen, wobei nur jeweils 'k' ausgewählt werden (Kombinationen von n Elementen zur k-ten Klasse): C(n,k) = n!/[k!(n-k)!]. Die Berechnung erfolgt mit 'n, ENTER, k, XEQ "COMB"'. Beispiel: C(73,4) = 1.088.430.

```
01♦LBL "PERM"     09 -              17 STO 01      25 ST- 01
02 STO 00         10 *              18 1           26 *
03 X<>Y           11 DSE 00         19♦LBL 13      27 DSE 00
04 STO 01         12 GTO 14         20 RCL 01      28 GTO 13
05 DSE 00         13 RTN            21 RCL 00      29 END
06♦LBL 14         14♦LBL "COMB"     22 /
07 RCL 01         15 STO 00         23 *
08 RCL 00         16 X<>Y           24 1
```

(51 Bytes)

- Quelle: Jim Davidson (547) & Bill Derrick (1393) (PPC J, V5N7P3).

19-13 Permutationen und Kombinationen bündig berechnen ("PRM", "COM" & "P+C")

Diese Routinen sind kürzer und schneller als die voranstehenden; sie benutzen auch keine Datenregister, doch müssen 'n' und 'k' ≤ 69 sein. Die Eingabe lautet für jede der Routinen 'n, ENTER, k'. "PRM" legt das Ergebnis, die Anzahl der Permutationen, nach X; "COM" legt die Anzahl der Kombinationen nach X; "P+C" legt die Anzahl der Permutationen nach X und die Anzahl der Kombinationen nach Y. Beispiele: P(9,4)=3024; C(9,4)=126.

```
01♦LBL "PRM"   09 /
02♦LBL 12      10 RTN
03 X<>Y        11♦LBL "COM"
04 FACT        12 XEQ 12
05 LASTX       13 X<>Y
06 RCL Z       14 FACT
07 -           15 /
08 FACT        16 END
```

(33 Bytes)

```
01♦LBL "P+C"   09 RCL Y
02 X<>Y        10 FACT
03 FACT        11 1/X
04 RCL Y       12 X<>Y
05 ST- L       13 ST* Y
06 X<> L       14 END
07 FACT
08 /
```

(27 Bytes)

- Quelle: Chris Stevens (3005) (PPC CJ, V7N5P44).

19-14 Permutationen und Kombinationen im Stapel berechnen ("PM" & "CM")

P(n,k) ist die Anzahl der Variationen von 'n' Elementen 'k'-ter Ordnung, also die Anzahl der Permutationen aller Teilmengen k-ter Ordnung einer Menge von n Elementen. C(n,k) ist die Anzahl der Kombinationen von 'n' Elementen 'k'-ter Ordnung, also die Anzahl aller Teilmengen k-ter Ordnung (die Anordnung der Elemente bleibt im Gegensatz zu P(n,k) unberücksichtigt) einer Menge von n Elementen. Die Eingabe lautet für

beide Routinen 'n, ENTER, k'. "PM" rettet die Inhalte von Z und T nach Y bzw. Z; "CM" läßt den Inhalt von Y unversehrt. Datenregister werden nicht benutzt.

```
01♦LBL "PM"     10 ST* L        19 RCL Y        28 X<> T
02 CHS          11 DSE X        20 X*Y?         29 LASTX
03 X<>Y         12 GTO 06       21 -            30 ST- Y
04 SIGN         13♦LBL 07       22 X>Y?         31 /
05 X<> L        14 RDN          23 X<>Y         32 ST* Y
06 ST+ Y        15 X<> L        24 ST+ T        33 DSE L
07♦LBL 06       16 RTN          25 SIGN         34 GTO 08
08 X=Y?         17♦LBL "CM"     26 X<>Y         35 RDN
09 GTO 07       18 RCL Y        27♦LBL 08       36 END
```

(63 Bytes)

- Quelle: John Kennedy (918) & Keith Jarett (4360) (PPC ROM).

19-15 Schnelle Ermittlung des Populationsparameters σ

Will man nach Eingabe statistischer Größen die *Standardabweichungen der Grundgesamtheiten* von x und y ermitteln, führe man 'MEAN, Σ+, SDEV' aus, dann liegen $\sigma_x = \sqrt{(n-1)/n} \cdot$ SDEV(x) in X und $\sigma_y = \sqrt{(n-1)/n} \cdot$ SDEV(y) in Y. Mit 'MEAN, Σ-' werden die Statistik-Register wieder in den alten Zustand versetzt. - Quelle: Joseph Horn (1537) (PPC CJ, V7N4P13).

19-16 Statistik mit 3 Variablen ("CLΣZ", "Σ+Z" & "Σ-Z")

Die Routine benutzt die Register R_{01}-R_{13}; diese Register werden zu Beginn mit 'XEQ "CLΣZ"' gelöscht. Dann verarbeitet man Zahlentripel mit 'z, ENTER, y, ENTER, x, XEQ "Σ+Z"'. Jeder Aufruf von "Σ+Z" akkumuliert die eingegebenen Werte gemäß der rechts angegebenen Tabelle und führt dann die Werte x, y und z in ihre ursprünglichen Register X, Y und Z zurück. Das Produkt x · y gelangt ins Register T und liegt dort für ein u.U. notwendiges korrigierendes "Σ-Z" bereit. 'n' wird nicht gezeigt, kann jedoch in R_{06} oder R_{12} betrachtet werden. Ein falscher Eintrag wird mit sofortigem 'XEQ "Σ-Z"' gelöscht: die falschen Daten dürfen nicht erneut eingegeben werden, weil sonst x · y aus dem Stapel geschoben wird! Nach der Löschung fährt man mit dem richtigen Tripel fort. Sind die letzten Werte verarbeitet, können die Statistiksummen den Registern R_{01}-R_{13} entnommen werden.

$R_{01} = \Sigma x$
$R_{02} = \Sigma x^2$
$R_{03} = \Sigma y$
$R_{04} = \Sigma y^2$
$R_{05} = \Sigma x \cdot y$
$R_{06} = n$
$R_{07} = \Sigma z$
$R_{08} = \Sigma z^2$
$R_{09} = \Sigma x$
$R_{10} = \Sigma x^2$
$R_{11} = \Sigma x \cdot z$
$R_{12} = n$
$R_{13} = \Sigma y \cdot z$

```
01♦LBL "CLΣZ"   08 RDN          15 ΣREG 07      22 RTN          29 LASTX
02 1.013        09 RTN          16 Σ+           23♦LBL "Σ-Z"    30 RCL Z
03♦LBL 14       10♦LBL "Σ+Z"    17 RCL Z        24 R↑           31 ΣREG 07
04 STO IND X    11 ΣREG 01      18 ST* L        25 ST- 13       32 Σ-
05 ST- IND X    12 Σ+           19 X<> L        26 RDN          33 RDN
06 ISG X        13 LASTX        20 ST+ 13       27 ΣREG 01      34 END
07 GTO 14       14 RCL Z        21 RDN          28 Σ-
```

(74 Bytes)

- Quelle: Bill Kolb (265).

19-17 Synthetische Statistik-Blöcke

Mit Hilfe eines synthetisch festgelegten Statistik-Blockes B lassen sich die Summen einiger ausgefallenen Funktionen bilden.

(a) Setzt man 'ΣREG T' fest, bilden die Register T, Z, Y, X, L und M in dieser Reihenfolge einen Statistik-Block B. Weil X und Y jedoch zugleich zur Eingabe der Werte und Ausgabe der Ergebnisse von 'Σ+' verwendet werden, ergeben sich in B andere Summen und Werte als gewöhnlich, und B ist zudem nur beschränkt auszubeuten: in T entsteht die Summe $\Sigma(x + y^2)$, in Z entsteht die Summe $\Sigma(x + y^2)^2$. Vor Eingabe von Wertepaaren x,y muß außerdem 'R↓, R↓' oder 'R↑, R↑' ausgeführt werden.

(b) Setzt man 'ΣREG Z' fest, wird B durch die Register Z, Y, X, L, M und N gebildet. In M entsteht die Summe $\Sigma((x + y)^3 + (x + y)y)$.

(c) Setzt man 'ΣREG Y' fest, wird B durch die Register Y, X, L, M, N und O gebildet. In M entsteht Σy^2, in N entsteht $\Sigma(x^2 + x) \cdot (x + y)$.

In keinem der drei Fälle läßt sich 'Σ-' zur Korrektur falsch eingegebener Werte benutzen, jedoch arbeitet 'CLΣ' normal, so daß

(d) nach einem 'ΣREG X' das X-Register, das L-Register und das Alpha-Register auf einen Schlag mit 'CLΣ' bytesparend gelöscht werden können.

- Quelle: Heinz Dalkowski.

Kapitel 20

UHR, KALENDER UND FINANZEN

20-1 Julianischer Tag ("JD")

"JD" ist ein Kalender-Programm, das den Julianischen Tag eines Gregorianischen Datums berechnet. Es läßt sich für Datumsangaben vom 14. Sept. 1752 an benutzen. Nur der Stapel wird verwendet. Die Eingabe erfolgt im Format MM,TTJJJJ; dann 'XEQ "JD"'. Das Ergebnis wird nach X gelegt.

```
01♦LBL "JD"     12 ST- T        23 ENTER↑       34 ST+ Z
02 FRC          13 RDN          24 INT          35 X<> L
03 3            14 9            25 ST+ Z        36 4
04 LASTX        15 +            26 -            37 /
05 INT          16♦LBL 07       27 1 E2         38 INT
06 1            17 30,6         28 *            39 1720997
07 +            18 *            29 INT          40 +
08 X>Y?         19 INT          30 CHS          41 +
09 GTO 07       20 1 E2         31 36525        42 +
10 +            21 R↑           32 LASTX        43 INT
11 1 E-6        22 *            33 *            44 END
```

(76 Bytes)

- Quelle: Fred Wheeler (1150) (PPC CJ, V7N8P11).

20-2 Umrechnungen Kalenderdatum ↔ Julianischer Tag ("CJ" & "JC")

"CJ" (Kalenderdatum → Julianischer Tag): Diese Routine berechnet den Julianischen Tag aus einem gegebenen Datum. Sie läßt sich für Datumsangaben vom 1. März des Jahres 0 an benutzen. Ist Flag 10 gelöscht, wird die Eingabe als Gregorianisches Kalenderdatum interpretiert; ist Flag 10 gesetzt, wird die Eingabe als Julianisches Kalenderdatum interpretiert. Die Eingabe erfolgt so: 'Jahr, ENTER, Monat, ENTER, Tag'; dann 'XEQ "CJ"'. - *"JC" (Julianischer Tag → Kalenderdatum):* Diese Routine kehrt "CJ" um: sie berechnet das Kalenderdatum aus dem Julianischen Tag. Die Eingabe erfolgt in X; die Ausgabe liegt so: Jahr in Z, Monat in Y, Tag in X. Ist Flag 10 gelöscht, erfolgt die Ausgabe als Gregorianisches Datum; ist Flag 10 gesetzt, erfolgt die Ausgabe als Julianisches Datum. - Beide Routinen benutzen keine Datenregister.

```
01♦LBL "CJ"     23 FS? 10       45 -2           67 -
02 INT          24 ISG X        46 FS? 10       68 ,3
03 X<>Y         25 %            47 GTO 09       69 -
04 INT          26 INT          48 36524,25     70 STO Y
05 2,85         27 ,75          49 /            71 30,6
06 -            28 ST* Z        50 INT          72 ST/ Y
07 12           29 *            51 ST+ Y        73 X<>Y
08 /            30 RDN          52 4            74 INT
09 R↑           31 -            53 /            75 *
10 INT          32 INT          54 INT          76 ST- Y
11 +            33 R↑           55♦LBL 09       77 ISG Y
12 X<0?         34 -            56 -            78 X<> L
13 SQRT         35 INT          57 X<0?         79 -3
14 ENTER↑       36 1721115      58 SQRT         80 X↑2
15 INT          37 +            59 STO Y        81 X<Y?
16 ST- Z        38 RTN          60 365,25       82 ISG T
17 X<>Y         39♦LBL "JC"     61 ST/ Y        83 X<> L
18 367          40 INT          62 X<>Y         84 -
19 *            41 1721119,2    63 INT          85 X<>Y
20 INT          42 -            64 ST* Y        86 INT
21 ST+ Z        43 ENTER↑       65 RDN          87 END
22 SIGN         44 FS? 10       66 INT
```

(158 Bytes)

- Quelle: John Kennedy (918), Fred Wheeler (1150) & Roger Hill (4940) (PPC ROM).

20-3 Stoppuhr ("TM")

Auf die Eingabe-Aufforderung hin wird die Zeit im Format HH,MMSS eingegeben; dann 'R/S'. Läuft die 'Uhr' zu schnell oder zu langsam, kann man Zeile 07 entsprechend abändern. Achtung: Der HP-41 enthält einen Oszillator, keinen Kristall. Daher läßt sich diese Routine nur unter festen unverändert bleibenden Bedingungen feinabstimmen, und auch dann kann keine Genauigkeit garantiert werden. Die 'Uhr' kann mit 'R/S' gestoppt werden. Die Zeit liegt dann in X *oder* in Y.

```
01♦LBL "TM"          04 FIX 4      07 ,000057     10 GTO 00
02 "ZEIT, HH.MMSS?"  05 CF 21      08 HMS+        11 END
03 PROMPT            06♦LBL 00     09 VIEW X
```

(42 Bytes)

- Quelle: HP Key Notes, V4N3P11.

20-4 Kalender-Druck ("PC")

1. Man setzt 'SIZE 004' (oder größer) fest. 2. 'XEQ "PC"'. 3. Den Aufforderungen entsprechend wird das Anfangsjahr, der Anfangsmonat (0-12) und die Anzahl der auszudruckenden Monate eingegeben. Gibt man als Anfangsmonat '0' oder gar nichts ein, wird die nächste Aufforderung übersprungen und das ganze Jahr gedruckt. Die Routine gilt für jedes Jahr.

```
01♦LBL "PC"          50 ST* 00         :99 RTN              148 ACA
02 "ANFANGSJAHR?"    51 *              100♦LBL 09           149 9 E-3
03 PROMPT            52 +              101 "SEP"            150 SF 10
04 INT               53 -              102 SIGN             151 GTO IND L
05 X<=0?             54 INT            103 RTN              152♦LBL 14
06 LN                55 RCL 00         104♦LBL 11           153 GTO IND Z
07 STO 01            56 -              105 "NOV"            154♦LBL 00
08 ,23               57 INT            106 SIGN             155 ISG X
09 -                 58 3              107 RTN              156 ACA
10 STO 00            59 +              108♦LBL 02           157 ACX
11 CLX               60 7              109 "FEB"            158♦LBL 01
12 "MONAT?(0-12)"    61 MOD            110 SIGN             159 ISG X
13 PROMPT            62 7 E-5          111 RCL 01           160 ACA
14 ABS               63 +              112 %                161 ACX
15 INT               64 STO 00         113 ENTER↑           162♦LBL 02
16 ENTER↑            65♦LBL 20         114 INT              163 ISG X
17 X=0?              66 SF 12          115 X≠Y?             164 ACA
18 SIGN              67 CLX            116 RCL 01           165 ACX
19 STO 02            68 ADV            117 4                166♦LBL 03
20 12                69 XEQ IND 02     118 MOD              167 ISG X
21 STO 03            70 GTO 13         119 X≠0?             168 ACA
22 X<Y?              71♦LBL 01         120 SIGN             169 ACX
23 ASIN              72 "JAN"          121 2                170♦LBL 04
24 /                 73 RTN            122 +                171 ISG X
25 ST+ 00            74♦LBL 03         123 RTN              172 ACA
26 X>Y?              75 "MRZ"          124♦LBL 13           173 ACX
27 GTO 10            76 RTN            125 ACA              174♦LBL 05
28 SIGN              77♦LBL 05         126 RCL 01           175 ISG X
29 "MONATSANZ.?"     78 "MAI"          127 ACX              176 ACA
30 PROMPT            79 RTN            128 2                177 ACX
31 INT               80♦LBL 07         129 SKPCHR           178♦LBL 06
32 STO 03            81 "JUL"          130 ADV              179 ISG X
33♦LBL 10            82 RTN            131 CF 12            180 ACA
34 PRBUF             83♦LBL 08         132 24               181 ACX
35 FIX 0             84 "AUG"          133 R↑               182 PRBUF
36 CF 29             85 RTN            134 -                183 ACA
37 ,012              86♦LBL 10         135 RCL 00           184 DSE Z
38 ST+ 02            87 "OKT"          136 7                185 GTO 00
39 31                88 RTN            137 MOD              186 FS?C 10
40 RCL 00            89♦LBL 12         138 +                187 GTO 14
41 *                 90 "DEZ"          139 STO 00           188 ISG 02
42 INT               91 RTN            140 "  SO MO DI MI"  189 GTO 10
43 RCL 00            92♦LBL 04         141 "├ DO FR SA"     190 12
44 INT               93 "APR"          142 PRA              191 ST- 02
45 STO 00            94 SIGN           143 3                192 ISG 01
46 1                 95 RTN            144 LASTX            193♦LBL 10
47 %                 96♦LBL 06         145 *                194 DSE 03
48 INT               97 "JUN"          146 SKPCHR           195 GTO 20
49 ,75               98 SIGN           147 " "              196 END
```

(370 Bytes)

```
     JUN 1982
SO MO DI MI DO FR SA
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

```
     JUL 1982
SO MO DI MI DO FR SA
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

```
     AUG 1982
SO MO DI MI DO FR SA
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

```
     JAN 2000
SO MO DI MI DO FR SA
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

- Quelle: Roger Hill (4940) (PPC CJ, V7N6P14).

20-5 Einkommensteuer ("EKST")

Das folgende Programm berechnet die in der Bundesrepublik Deutschland abzuführende Einkommensteuer nach den seit 1981 geltenden Sätzen. Eingabe: Zu versteuerndes Einkommen, Ausgabe: Einkommensteuer. Bei gesetztem Flag 1 wird der sog. Splittingtarif angewendet. Beispiele: Auf ein zu versteuerndes Jahreseinkommen von 58.317,- DM sind 19.146,- DM Einkommensteuer zu zahlen. Erhöht sich das Einkommen um 10.000,- DM, bleiben davon nur 4.882,- DM übrig! Bei Anwendung des Splittingtarifs blieben 6.274,- DM übrig.

```
01♦LBL "EKST"     26 *          51 2200       76 XEQ 07
02 FC? 01         27 14837      52 +          77 XEQ 08
03 GTO 05         28 -          53 *          78 ,09
04 2              29 GTO 00     54 3034       79 *
05 /              30♦LBL 01     55 +          80 5,45
06 INT            31 0          56 GTO 00     81 -
07♦LBL 05         32 GTO 00     57♦LBL 07     82 *
08 4213           33♦LBL 02     58 54         83 88,13
09 X<>Y           34 XEQ 07     59 /          84 +
10 X<Y?           35 ,22        60 INT        85 *
11 GTO 01         36 *          61 54         86 5040
12 18001          37 926        62 *          87 +
13 X<>Y           38 -          63 RTN        88 *
14 X<Y?           39 GTO 00     64♦LBL 08     89 20018
15 GTO 02         40♦LBL 03     65 X<>Y       90 +
16 6 E4           41 XEQ 07     66 RDN        91♦LBL 00
17 X<>Y           42 XEQ 08     67 X<>Y       92 INT
18 X<Y?           43 3,05       68 -          93 FS? 01
19 GTO 03         44 *          69 1 E4       94 2
20 13 E4          45 73,76      70 /          95 FS? 01
21 X<>Y           46 -          71 ENTER↑     96 *
22 X<Y?           47 *          72 ENTER↑     97 "ST: "
23 GTO 04         48 695        73 ENTER↑     98 ARCL X
24 XEQ 07         49 +          74 RTN        99 AVIEW
25 ,56            50 *          75♦LBL 04    100 END
```

(195 Bytes)

- Quelle: Heinz Dalkowski.

Kapitel 21

KARTENLESER UND LESESTIFT

21-1 Stromverbrauch und Abnutzung des Kartenlesers

Beschränken Sie das Anschließen und Entfernen des Kartenlesers 82104A auf das unbedingt notwendige Maß. Lassen Sie ihn auch dann angeschlossen, wenn Sie ihn nicht verwenden. Schalten Sie den Rechner nach Anschluß des Kartenlesers einmal an und aus. Wenn der Kartenleser das erste Mal angeschlossen wird, kann es sein, daß er ungewöhnlich viel Strom verbraucht. Schalten Sie den Rechner in diesem Fall an und aus, um zum Normalverhalten zu gelangen. Wenn Sie den Kartenleser soviel wie möglich angeschlossen lassen, vermeiden Sie Fehlverhalten durch elektrostatische Aufladung, Abnutzung der metallischen Plastik-Kontaktflächen und das Eindringen von Staub und Fremdkörpern in die Anschlußbuchsen. - Quelle: Philipp Karras (3480) (PPC CJ, V7N2P56).

21-2 Leseoperation abschließen

Man schont die Batterien, wenn man Kartenleseoperationen zügig durchführt und nicht etwa eine Kaffeepause einlegt, während der HP-41 mit 'RDY nm OF 1k" in der Anzeige wartet. - Quelle: Richard Nelson (1) (PPC J, V6N7P4).

21-3 Karten-Stop wegen schwacher Batterien

Gewöhnlich warnt Sie der Kartenleser, wenn die Batterien zur Neige gehen und Sie eine Karte zu lesen oder zu beschreiben beabsichtigen, und zieht dann die Karte durch, ohne zu lesen oder zu schreiben. Wenn Sie jedoch vergessen, daß die Batterien schon schwach sind (weil der Normalbetrieb des Rechners noch möglich ist), und dennoch den Kartenleser benutzen, kann sich folgendes ereignen: Die Warnung "LOW BAT" erscheint, die Karte wird eingezogen, sie bleibt jedoch auf halbem Wege knirschend stehen und wird in dieser Stellung beharrlich festgehalten. Wenn Sie dann eine gewaltsame Beförderung (hindurch oder zurück) von Hand versuchen, können Sie mechanische Zerstörungen im Kartenleser anrichten. Überdies bleibt der Rechner - mit der Batterie-Meldung in der Anzeige - betriebsunfähig und läßt sich nicht einmal ausschalten. Folgendes Vorgehen hilft: 1. Kartenleser entfernen. 2. Rechner ausschalten. 3. Neue Batterien einlegen. 4. Kartenleser wieder anschließen. Die Karte vollendet jetzt ihren Lauf. 5. Rechner anschalten und eine Karte zum Lesen oder Beschreiben einführen. - Quelle: Fred Wheeler (1150) (PPC CJ, V7N3P27).

21-4 Wirkungsweise von 'RSUB' und 'MRG'

Die Kartenleser-Funktionen 'RSUB' und 'MRG' arbeiten etwas anders als im Handbuch angegeben: 'RSUB' ersetzt das letzte im Programmspeicher befindliche Programm L nur dann durch das Programm P auf der Karte, wenn L nicht durch ein 'END' abgeschlossen ist, also mit dem '.END.' endet; andernfalls wird P hinter L gesetzt. 'MRG' zeigt "MRG ERR" an, wenn der Rechner nicht auf das letzte Programm positioniert ist, doch wird die Karte auch dann gelesen und der Inhalt hinter das letzte 'END' gesetzt. - Quelle: Dennis Green (4213) (PPC CJ, V7N3P27). - Vgl. auch 21-10.

21-5 Eine Karte läßt sich nicht lesen

Zunächst hauche man die magnetisierte (dunkle) Seite an. - Quelle: John Burkhart (4382). - Man versuche, sie am Hemd abzuwischen. - Quelle: Henry Casson (5047). - Man kann sie gründlicher reinigen, indem man einen weichen Lappen mit Alkohol anfeuchtet und sie damit abreibt. Beachten Sie auch, daß Magnetkarteninhalte zerstört werden, wenn die Karte magnetisierte Gegenstände berührt oder in deren Nähe gelangt. - Quelle: Bill Kolb (265).

21-6 Dateneingabe als Programmteil ausführen

In vielen Anwendungen kann es zu Irrtümern und Verwechselungen kommen, wenn zugleich Programm- und Daten-Karten eingesetzt werden. Häufig kann man die Benutzung von Karten einschränken und den Programmablauf vereinfachen, wenn die 'Daten' als Teil eines Programms eingegeben werden. Flag 11 (automatischer Programmlauf) kann bei Programmaufzeichnung zu diesem Zweck gesetzt werden, und der erste Programmteil läßt sich dann dazu benutzen, die Daten automatisch in die Register zu bringen. Diese Methode, Dateneingabe als Programmschritte auszuführen, vereinfacht die Handhabung von Programmen erheblich. - Quelle: HP Key Notes, V7N2P7.

21-7 Status-Karte als erste Karte eines Satzes von Programmkarten einlesen

Eine Status-Karte läßt sich dazu verwenden, den Rechner vor Programmstart in den erforderlichen Zustand zu versetzen. 'SIZE' wird festgelegt (Meldung "SIZE ERR" erfolgt, falls nicht genügend Speicherplatz vorhanden ist), die Flags 00-43 werden eingestellt, 'ΣREG' wird bestimmt, und der Stapel und das Alpha-Register erhalten den gewünschten Inhalt; all das mit Spur 1 der Karte. (Weitere Spuren - u.U. auf mehreren Status-Karten - können Tastenzuweisungen enthalten.) Eine passende Alpha-Meldung (von höchstens 6 Zeichen) läßt sich ins X-Register legen, um anzuzeigen, daß oder wie es weitergehen kann. Das Alpha-Register selbst kann eine Meldung von bis zu 24 Zeichen enthalten, die zu betrachten der Text "ALPHA" im X-Register auffordern kann. - Quelle: HP Key Notes, V4N2P7.

21-8 Karte für das Löschen von Tastenzuweisungen

Herstellung: Zunächst Totallöschung; anschließend eine Zuweisung tätigen, diese wieder löschen; 'XEQ "WSTS"' ausführen und dabei dieselbe Spur der Karte zweimal benutzen, so daß nur der zweite Teil der Information aufgezeichnet wird. *Benutzung:* Die (einzige) Spur der Karte einlesen; Korrekturtaste drücken, um die Aufforderung "RDY 01 OF 02" zu löschen. Danach den Rechner einmal aus- und anschalten. - Quelle: Roger Hill (4940) (PPC CJ, V7N8P22). - Vgl. auch 1-26.

21-9 'Hängenbleiben' mit 'SIZE', 'DEL' oder 'GTO.'

Wenn ein Kartenleser angeschlossen ist, kann es vorkommen, daß der HP-41 'hängenbleibt', sobald Sie 'SIZE', 'DEL' oder 'GTO.' ausführen, 2 Ziffern eingeben und dann (2 mal) die Korrekturtaste drücken. Mit jeder außer der Korrekturtaste können Sie diese Blockade aufheben; die eingetastete Funktion geht dabei verloren. Der Fehler liegt im Kartenleser, nicht im HP-41, wovon Sie sich leicht durch Entfernen des Kartenlesers überzeugen können. - Quelle: Bill Kolb (265).

21-10 'Letztes Programm' im Sinne des Kartenlesers

Der Kartenleser faßt als 'letztes Programm' den Teil des Programmspeichers auf, der dem letzten 'END' folgt, also den durch '00 REG mn,..., .END. REG mn' festgelegten Abschnitt. Wenn das an letzter Stelle im Programmspeicher liegende Programm ein 'END' enthält, ist es also *nicht* das 'letzte Programm' im Sinne des Kartenlesers. Daher kann man z.B. kein 'MRG' in dem an letzter Stelle liegenden Programm ausführen, sobald dieses ein 'END' enthält. Beachten Sie auch, daß der Kartenleser keine 'END'-Befehle aufzeichnet, was sich im Hinblick auf die voranstehenden Erläuterungen als sinnvoll erweist. - Quelle: Bill Wickes (3735). - Vgl. auch 21-4.

21-11 Teil eines Satzes von 'WALL'-Karten lesen

Im Gegensatz zu den Angaben im Handbuch muß ein 'WALL'-Satz von Karten nicht vollständig eingelesen werden. Man kann den Eingabevorgang jederzeit dadurch unterbrechen, daß man die Batterien kurzzeitig entfernt und anschließend die Korrekturtaste drückt. Verwenden Sie diese Technik zur partiellen Wiedergewinnung von Tastenzuweisungen und Programmen, vorausgesetzt es steht genügend Programmspeicherplatz zur Verfügung, um das Einlesen des Kartensatzes überhaupt zu starten. - Quelle: Bill Kolb (265).

21-12 Wiederaufnahme der Programmausführung nach einem 'WALL'

'WALL' zeichnet auch die Lage des Adreßzeigers und die Rücksprungadressen auf, so daß die Unterbrechung und Wiederaufnahme laufender Programme ermöglicht wird. - Quelle: Bill Wickes (3735).

21-13 Aufzeichnung von Tastenzuweisungen

Zeichnet man sich für besondere Zwecke eine Reihe von Tastenzuweisungen auf, ist es sinnvoll, der Karte einen entsprechenden Namen zu geben und diesen gleichzeitig ins Alpha-Register zu legen, um ihn zusammen mit den Tastenzuweisungen auf die Karte zu lesen. Ist der Alpha-Modus aktiv, wenn die Karte eingelesen wird, erscheint der Name zur Bestätigung korrekten Einlesens in der Anzeige. Ebenso kann eine kurze Alpha-Gedächtnisstütze mit 'ASTO' ins X-Register gelegt und aufgezeichnet werden. Statt eines Namens kann man auch mnemotechnische Abkürzungen der Tastenzuweisungen verwenden. - Quelle: Bill Kolb (265).

21-14 Eintasten von Funktionen aus nichtangeschlossenen Peripheriegeräten

Sind Drucker, Lesestift oder Kartenleser nicht angeschlossen, kann man deren Funktionen dennoch in Programme eintasten, indem man einfach ihren Namen 'buchstabiert': um z.B. 'PRX' ohne Drucker zu programmieren, tastet man 'XEQ, ALPHA, P, R, X, ALPHA'. Wenn das Programm dann ausgeführt wird, sucht es zunächst nach einem Benutzerprogramm mit diesem Namen und führt, wenn es ein solches nicht findet und der Drucker angeschlossen ist, 'PRX' aus dem Drucker aus. - Quelle: Bill Kolb (265). - Vgl. auch 22-9.

21-15 'VER' oder 'WPRV' in ein Programm einfügen

1. Bei angeschlossenem Kartenleser 'VER' bzw. 'WPRV' einer Taste zuweisen. 2. HP-41 ausschalten; Kartenleser entfernen; HP-41 anschalten; im PRGM-Modus die die Zuweisung tragende Taste drücken; die Programmzeilen erscheinen als 'XROM 30,05' bzw. 'XROM 30,09'. 3. HP-41 ausschalten; Kartenleser wieder anschließen; HP-41 anschalten; PRGM-Modus einschalten; die Zeilen lauten jetzt 'VER' bzw. 'WPRV' und werden auch als solche ausgeführt. - *Synthetische Methode:* Dasselbe Ergebnis erreicht man ohne Entfernen des Kartenlesers, indem man diese Funktionen synthetisch erzeugt: 'VER' ist dezimal 167, 133 (hexadezimal A7 85); 'WPRV' ist dezimal 167, 137 (hexadezimal A7 89).

Es ist günstig, einem 'WDTAX' im Programm ein 'VER' folgen zu lassen, um zu sehen, ob das Beschreiben erfolgreich war. Im untenstehenden Beispiel werden die Register R_{00}-R_{16} mit ihrer eigenen Adresse geladen; dann wird der Befehl 'WDTAX' ausgeführt, der zum Beschreiben zweier Spuren mit den Daten auffordert. Anschließend erscheint die Aufforderung 'CARD' (von 'VER' verursacht), so daß man die beiden beschriebenen Spuren überprüfen kann. Man sieht "TYPE D TR 01" (bzw. "...02"). Will man die Programmausführung fortsetzen, drückt man zweimal 'R/S' oder '←, R/S'. Ergibt die Prüfung einen Schreibfehler (man sieht in diesem Fall "CARD ERR"), löscht man die Fehlermeldung mit der Korrekturtaste, geht mit 'BST' zum Befehl 'WDTAX' zurück und startet mit 'R/S' erneut den Schreibvorgang.

```
01♦LBL "X"     06 STO IND Y    11 LASTX              16 PSE
02 ,016        07 ISG X        12 WDTAX              17 CLD
03 ENTER↑      08♦LBL 00       13 VER                18 END
04 INT         09 ISG Y        14 "WEIT. PR.-SCHR."
05♦LBL 14      10 GTO 14       15 AVIEW
```

(48 Bytes)

- Quelle: Cary Reinstein (2046) & John Herzfeld (5428).

21-16 Interferenz-Erscheinungen beim Lesestift

Haben Sie Probleme mit dem Lesestift und stellen Sie insbesondere fest, daß er an manchen Tagen besser arbeitet als an anderen, so kann es sich um eine Art Interferenz, verursacht durch einen Wechselstrom, handeln. Wenn Sie z.B. den Lesestift unterhalb des Schalters halten, kann zwischen Ihrem Körper und dem Lesestift ein 'menschlicher' 60 Hz Wechselstrom fließen. Dies ist möglich, weil die elektronischen Bausteine, die sehr schwache optische Signale aus der Papierreflexion erkennen müssen, hochverstärkt arbeiten. Sobald stärkere Wechselstromsignale in den Lesestift gelangen, entsteht eine Leseblockade, und Sie erhalten keine Antwort mehr aus dem Rechner. Ein solcher Effekt kann auch auftreten, wenn der HP-41 in die Nähe eines Transformators, eines stromdurchflossenen Leiters, einer fluoreszierenden Lichtquelle o.ä. gelangt. Der Drucker wirkt, sobald er angeschlossen ist, wie eine Antenne und kann zusammen mit dem Anschlußkabel zu Interferenzen beitragen und sie verstärken. Die folgenden Tips können helfen, wenn Sie vermuten, daß Interferenzen die Funktion des Lesestiftes beeinträchtigen. (a) Halten Sie den Stift am oberen Endes des Schalters. Die meisten Benutzer tun dies ohnehin. (b) Wenn Sie Interferenzen vermuten, entfernen Sie die anderen Peripheriegeräte, insbesondere, wenn diese ans Stromnetz angeschlossen sind; oder verbringen Sie das ganze System an einen andereren Ort. Bemerkungen: 1. Wenn Sie Probleme mit dem Stift haben, testen Sie die Ausführung von Funktionen, indem Sie diese auf dem Barcode-Tastenfeld überstreichen. Testen Sie ein und dieselbe Funktion mehrfach: bei etwa 25 Versuchen sollten wenigstens 23 einwandfreie Antworten gegeben werden. Schlechte Barcode-Qualität erhöht natürlich die Empfindlichkeit gegenüber Interferenzen. 2. Wechseln Sie den Ort, wenn Sie Interferenz vermuten. Falls Sie an einem sehr hellen Tisch oder Arbeitsplatz tätig sind, können Interferenzen durch Fluoreszenz auftreten. Schon eine Verschiebung des Rechners um wenige Zentimeter vermag die Lage entscheidend zu verbessern, zumal wenn Sie sehr dicht an der Störquelle sind. 3. Die Funktionsfähigkeit des Stiftes läßt sich schnell und bequem mit dem Befehl '+' testen: man lädt Y, Z und T mit '1' und X mit '0'; dann überstreicht man 25 mal '+'. Anschließend zeigt die Anzeige die Anzahl der fehlerfreien Lesevorgänge. Interferenz kann diese Anzahl von 24 oder 25 auf 12-16 herunterdrücken. Interferenz sollte aber selten ein ernstes Problem darstellen, da man weiß, wie sie zu erkennen und auszuschließen ist. - Quelle: Richard Nelson (1) (PPC CJ members letter, June 1980).

21-17 Tips für den optischen Lesestift

Kopien von Kopien von Barcodes lassen sich häufig nicht mehr optisch abtasten. Man erzielt bessere Ergebnisse, wenn man die erste Kopie um 90° oder 180° dreht, bevor man sie kopiert. - Man decke Barcodes stets mit einer *Schutzfolie* (für die nichtblendendes Material vorzuziehen ist) ab, bevor man sie abtastet, so daß Kopien nicht verwischt oder beschmutzt werden können. - Man vermeide das *Verdrehen* des Anschlußkabels unter Spannung, damit keine Kabelbrüche entstehen. - Barcodes, die auf *dünnem Papier* gedruckt sind, können vom Druckbild der Rückseite 'durchblutet' werden; dagegen hilft eine *schwarze Unterlage*. - Wenn Sie den Stift *interaktiv* benutzen, ist es sinnvoll, den Rechner nach beendetem Programm auszuschalten; dies schont die Batterien, und weil der Stift den HP-41 anzuschalten vermag, hat man keine Unbequemlichkeiten. - Bauen Sie eine *Datenprüfung* in Ihr Programm ein, wenn Sie 'WNDSCN' benutzen, um einfache 'positionierte' Daten einzulesen. Dies ist zu empfehlen, weil der Stift Striche, die er mit 'WNDSCN' liest, nicht prüft. Selbst eine einfache Bereichsprüfung ist besser als gar keine. - Benutzen Sie *akustische Signale* für die Aufforderung, Barcodes abzutasten; falls Sie für diese Programmhilfe nur ein ungeduldiger Zuhörer sind, verwenden Sie zweimal den synthetischen Ton 89 (9F 89). - Lediglich die Funktion 'WNDTST' ist aus HP-41-Befehlen zusammengesetzt und läßt sich daher als Programm auflisten; die anderen fünf Funktionen 'WNDDTA, WNDDTX, WNDLNK, WNDSUB, WNDSCN' sind Mikrokode-Funktionen. - Quelle: Richard Nelson (1) (PPC CJ, V7N5P22).

Kapitel 22

DER DRUCKER

22-1 Eingabe-Aufforderung und Eingabe drucken

'VIEW' und 'AVIEW' bewirken, daß die durch diese Befehle ausgelösten Anzeigen auch gedruckt werden, wenn der Drucker angeschlossen und eingeschaltet ist (Flags 21 und 55 gesetzt). Vorschlag: statt das Programm mit 'PROMPT' anzuhalten, um mit einer Alpha-Meldung zur Eingabe aufzufordern, kann man 'AVIEW, STOP' benutzen; dann wird die Meldung auch gedruckt. Schließt man ein 'VIEW X' an, wird die Eingabe ebenfalls gedruckt, und zwar rechtsbündig; schließt man stattdessen mit 'CLA, ARCL X, AVIEW' an, wird die Eingabe linksbündig gedruckt. - Quelle: HP Key Notes, V3N4P4. - Vgl. auch 4-18, 6-14.

22-2 Papiervorschub ("PO")

Will man dem Drucker eine Ausgabe entnehmen, 'XEQ "PO"'; dies schiebt den Papierstreifen entsprechend vor.

```
'LBL "PO", ADV, ADV, ADV, ADV, ADV, RTN'          (12 Bytes)
```

- Quelle: PPC ROM.

22-3 Tips für den Drucker

Register R_{06}: Soll der Graph einer Funktion f mit 'PRPLOT' geplottet werden, kann man auf R_{06} zugreifen, wenn der Wert x mehrfach zur Berechnung von f(x) benötigt wird.

NORMal-Modus: Zahlen und Alpha-Ketten werden gedruckt, sobald ihre Eingabe (mit 'ENTER' bzw. 'ALPHA aus') abgeschlossen wird; Funktionsnamen werden gedruckt, sobald sie vom Tastenfeld aus ausgeführt werden. Alle Druckfunktionen arbeiten. (Musteranwendung: Dokumentation von Einnahmen und Ausgaben, Kontoführung.)

'PROMPT': Druckt den Inhalt des Alpha-Registers im NORMal- und TRACE-Modus.

Batteriesatz: WARNUNG: Der Batteriesatz muß sich im Drucker befinden, wenn das Ladegerät angeschlossen ist. Andernfalls kann der Drucker beschädigt werden.

Klassifikation der Druckerfunktionen:

Stapel & Alpha:	Daten:	Programme:	Zustand:	Plotten:	Puffer:	Grafik:
PRX & VIEW	PRREG	PRP	PRFLAGS	"PRPLOT"	ACA	ACCOL
PRA & AVIEW	PRREGX	LIST	PRKEYS	"PRPLOTP"	ACX	SKPCOL
PRSTK	PRΣ		CAT 1+	"PRAXIS"	ACCHR	BLDSPEC
				REGPLOT	SKPCHR	ACSPEC
		(+ im TRACE-Modus)		STKPLOT	PRBUF	
					ADV	

Programmieren mit dem Drucker: Im TRACE- und NORMal-Modus wird jede Zeile, die in ein Programm gelangt, gedruckt. Eine Programmausführung im TRACE-Modus protokolliert jeden Programmschritt und jedes Zwischenergebnis, verbraucht dafür jedoch eine Unmenge Papier.

Programmausführung ohne Druckfunktionen: Wenn man den Drucker während einer Programmausführung ausschaltet, kann man die Ausführungsgeschwindigkeit leicht erhöhen, weil der HP-41 während einer Programmausführung mit dem Drucker verkehrt, sobald er eingeschaltet ist, was die Ausführung verlangsamt.

Flag 55: ist stets gesetzt, wenn der Drucker angeschlossen ist, unabhängig davon, ob er ein- oder ausgeschaltet ist.

Flag 21: unterdrückt Druckbefehle, wenn es gelöscht ist, auch dann, wenn der Drucker eingeschaltet ist.

Druckbreite: Die maximale Druckbreite beträgt 24 Zeichen = 168 Spalten. 'ACCOL' gestattet den Druck von Sonderzeichen bis zur Breite von 43 Spalten (beim HP-IL-Drukker 98 Spalten); 'SKPCOL' überspringt bis zu 167 Spalten.

Drucken und Papiervorschub: Ein Druck auf die Taste PRINT bewirkt im PRGM-Modus das Einfügen des Befehls 'PRX' in das Programm; falls sich der HP-41 im ALPHA-Modus befindet, das Einfügen des Befehls 'PRA'. Ein Druck auf die Taste PAPER ADVANCE bewirkt im PRGM-Modus das Einfügen des Befehls 'ADV' ins Programm (es sei denn, die Taste wird länger als eine Sekunde gedrückt; in diesem Fall wird nur das Papier vorgeschoben, nicht aber 'ADV' ins Programm eingefügt). Im TRACE- und NORMal-Modus wird 'ADV' auf jeden Fall ins Programm eingefügt *und* das Papier vorgeschoben, wenn die Taste länger gedrückt wird.

Kataloge drucken: Kataloge werden nur im TRACE-Modus ausgedruckt. 'CAT 1' druckt die Anzahl der von jedem Programm im Programmspeicher belegten Bytes mit aus.

Druck-Puffer löschen ohne zu drucken: Der Druck-Puffer wird gelöscht, wenn man den Drucker ausschaltet. Nach dem Wiedereinschalten ist er leer.

Akkumulationsfunktionen in Programmen: Sobald Programme Akkumulationsfunktionen enthalten, sollten sie nicht im TRACE- oder NORMal-Modus ausgeführt werden, weil diese Modi den Puffer benutzen und deshalb vorzeitig ausdrucken.

Plotten mit dem Drucker: 1. Der Name des Programms, zu dessen Eingabe mit "NAME?" aufgefordert wird, darf höchstens 6 Zeichen enthalten. 2. Wird man mit "X INC?" zur Eingabe des Inkrements aufgefordert, bildet eine positive Eingabe das x-Inkrement, eine negative Eingabe hingegen die *Anzahl* der x-Inkremente. 3. Wenn die Aufforderung "AXIS?" mit einer Alpha-Eingabe beantwortet wird, wird keine x-Achse gedruckt. 4. Enthält R_{03} ein mit 'BLDSPEC' erzeugtes Zeichen, wird dieses als Plot-Zeichen verwendet; andernfalls bildet das kleine hochgestellte x (Byte 01) das Plot-Zeichen.

Zahlen ohne Stern-Tripel drucken: Man verwende 'VIEW X' statt 'PRX'.

Rechner-Zustand mit 'PRFLAGS' ermitteln: Die ersten 5 Druckzeilen zeigen 'SIZE' und 'ΣREG' sowie den trigonometrischen und den Anzeige-Modus an. An dieser Stelle stoppt man mit 'R/S', um das Drucken der Flag-Zustände zu verhindern.

- Quelle: 'Bedienungshandbuch für den Drucker 82143A', © Copyright (Nov. 1979) Hewlett-Packard Company. Abgedruckt mit Erlaubnis.

22-4 Register L drucken ("PRL")

Diese Routine druckt den Inhalt des Registers L. Man kann sie zur Ermittlung des Stapelinhalts in Verbindung mit der Druckerfunktion 'PRSTK' benutzen.

'LBL "PRL", "L= ", ARCL L, PRA, RTN' (17 Bytes)

22-5 Fotokopieren mit Gelbfilter

Fotokopien des Druckstreifens können verbessert werden, wenn man einen Gelbfilter benutzt, z.B. einen gelben Transparent-Umschlag. - Quelle: Frits Kuyt (236) (PPC CJ, V7N3P5).

22-6 Papiersparend Byte-Anzahl drucken

Will man die Byte-Anzahl nur eines Programms mit 'CAT 1' finden, kann man Papier sparen, indem man im MANual-Modus (u.U. mit 'SST') bis zu dem *vor* dem betreffenden Programm stehenden 'END' vordringt, mit 'R/S' stoppt (falls nötig mit 'BST' auf das 'END' zurückgeht), den Drucker in den TRACE-Modus schaltet und dann mit 'SST' die Byte-Anzahl ermittelt.

22-7 Der Druck-Puffer

Der Druck-Puffer ist ein Speicher im Drucker, der *Zeichen und* (Druckmatrix-)*Spalten* akkumuliert, bis ein Druckbefehl auftritt. Er enthält eine Anzahl von Speicherzellen; sobald sie alle gefüllt sind, wird ihr Inhalt automatisch - also auch ohne Druckbefehl - ausgedruckt (oder die erste Zeile wird automatisch gedruckt, falls der Puffer 'Druckmaterial' für mehr als eine Zeile enthält). Jedes in den Puffer gelangte Zeichen besetzt eine Zelle, unabhängig davon, ob es mit 'ACX', 'ACA' oder 'ACCHR' dorthin gelangt ist; ebenso besetzt jede Spalte eines mit 'ACCOL' oder 'ACSPEC' erzeugten Sonderzeichens eine Speicherzelle. Der Befehl 'SKPCHR' belegt eine Zelle, unabhängig von der Anzahl der zu überspringenden Zeichen. 'SKPCOL' belegt eine Zelle, wenn die Anzahl der zu überspringenden Spalten ein Vielfaches von 7 oder kleiner als 7 ist, andernfalls zwei Zellen. Die Höchstanzahl von Zeichen oder Spalten, die auf irgendeine Weise akkumuliert werden können, beträgt 43 (beim HP-IL-Drucker 101), doch ist der Puffer häufig gefüllt, bevor dieses Höchstzahl erreicht wird. Warum?

Jede Änderung des 'Druck-Modus' belegt ebenfalls eine Speicherzelle, wobei folgende drei Arten von 'Druck-Modi' zu unterscheiden sind: (a) Druck in einfacher oder doppelter Zeichenbreite (festgelegt durch den Zustand von Flag 12 zu dem Zeitpunkt, in dem Zeichen, Spalten oder ein 'SKP...'-Befehl akkumuliert werden), (b) Druck eines Zeichens oder einer Spalte, (c) Druck in Groß- oder Kleinbuchstaben (festgelegt durch den Zustand von Flag 13 zu dem Zeitpunkt, zu dem die Zeichen in den Puffer gelangen). Daher gibt es 2^3=8 Druck-Modi (nicht zu verwechseln mit den Drucker-Modi MAN/TRACE/NORM).

Sobald irgendein Befehl auftritt, der eine Druck-Puffer-Eingabe auslöst, prüft der Drucker, ob der Druck-Modus (der durch diesen Befehl in Zusammenhang mit dem Zustand der Flags 12 und 13 festgelegt ist) mit dem alten Modus (festgelegt durch die vorangegangene Eingabe in den Puffer) übereinstimmt. Wenn das der Fall ist, werden Zeichen bzw. Spalten akkumuliert, ohne daß eine zusätzliche Zelle benötigt wird, um den Druck-Modus zu notieren. Ändert sich dagegen der Druck-Modus, wird zunächst eine Speicherzelle dazu verwendet, den neuen Modus festzuhalten, und dann erst werden die Zeichen oder Spalten akkumuliert. Die Befehle 'ACA', 'ACX', 'ACCHR' und 'SKPCHR' legen den Druck-Modus 'ZEICHEN' fest, während 'ACCOL', 'ACSPEC' und 'SKPCOL' den Druck-Modus 'SPALTE' bestimmen. Andere Drucker-Befehle wie 'PRX', 'PRA', 'PRP', 'PRFLAGS' usw. lassen den Drucker im 'Zeichen-Modus' zurück (sie veranlassen auch, daß der Pufferinhalt ausgedruckt wird, falls sich ein solcher nach dem letzten Druckbefehl angesammelt hat oder aus einem Puffer-Überlauf zurückgeblieben ist). Die Veränderung der Zustände von Flag 12 und Flag 13 haben keine Wirkung, solange keine Druck-Puffer-Eingabe erfolgt; erst zu diesem Zeitpunkt werden die Druck-Modi abgefragt, und, falls erforderlich, die Änderung in einer Zelle notiert.

Beachten Sie, daß lediglich *eine* Zelle für den Übergang von einem Druck-Modus zum anderen benötigt wird (z.B. vom Druck schmaler Großbuchstaben zum Druck breiter Kleinbuchstaben). Andererseits kann eine Speicherzelle auch dann belegt werden, wenn die Modus-Änderung ohne Wirkung bleibt, z.B. wenn man 'ACA' ausführt, obwohl das Alpha-Register leer ist, oder wenn man in den Modus 'KLEIN' wechselt, obwohl sich keine Buchstaben unter den zu akkumulierenden Zeichen befinden. Befehle, die lediglich den Puffer-Inhalt ausdrucken, also 'PRBUF' und 'ADV', benutzen die Flags 12 und 13 überhaupt nicht, weder für das Drucken selbst noch zur Neufestsetzung des Druck-Modus.

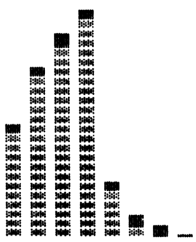
Machen Sie sich die Vorgänge beim Druck-Modus-Wechsel am folgenden Beispiel klar: zwischen zwei mit 'BLDSPEC' erzeugten Sonderzeichen soll eine Leerstelle eingefügt werden. Benutzt man '7, SKPCOL', wird nur eine Speicherzelle belegt, weil der Drukker nach dem 'ACSPEC' für das erste Zeichen schon in den Modus 'SPALTE' gelangte. Doch '1, SKPCHR' oder '32, ACCHR' oder 'ACA' mit 'SPACE' im Alpha-Register belegen jeweils 3 Zellen, weil jedesmal zusätzlich in den Zeichen-Modus und dann zurück in den Spalten-Modus gewechselt wird.

Ein weiteres Beispiel für den sinnvollen Umgang mit dem Druck-Puffer ist der Druck von Kleinbuchstaben. Man kann sehr viele Programm-Bytes (nicht zu reden von der Ausführungszeit) einsparen, wenn man Kleinbuchstaben direkt in Textzeilen bringt (z.B. synthetisch oder mit dem optischen Lesestift) und ständig mit gelöschtem Flag 13 akkumuliert, statt gewöhnliche Großbuchstaben zu benutzen und sie unter Verwendung von Flag 13 zu akkumulieren. Dieses Verfahren spart nicht nur Programm-Bytes, sondern auch Speicherzellen im Druck-Puffer. Denn jedesmal, wenn Flag 13 seinen Zustand wechselt und anschließend Zeichen in den Puffer gelangen, wird eine weitere Zelle mit der Information über die Modus-Änderung belegt, während solche Änderungen überflüssig sind, wenn die Zeichen ohne Flag-Änderung akkumuliert werden. Verwendet man 'ACCHR' mit den ASCII-Kodes für die Kleinbuchstaben (97-122), kann man ebenfalls Platz im Druck-Puffer sparen, gewöhnlich jedoch keine Programm-Bytes.

- Quelle: Roger Hill (4940) (PPC CJ, V7N6P19-20).

22-8 Hochauflösendes Histogramm ("HG")

Will man zur Darstellung von Zahlen eine 'Säulenstatistik' oder 'Balkengrafik' anfertigen, kann man sich der folgenden Routine bedienen. Sie erwartet y_{min} in R_{06}, $y_{max}-y_{min}$ in R_{07} und verarbeitet den in X liegenden Wert w ($y_{min} \leq w \leq y_{max}$) zu einer Säule, deren Höhe durch y_{min}, y_{max} und w bestimmt ist. In dem untenstehenden Beispiel ist y_{min} = 0, $y_{max}-y_{min}$ = 100 und die Werte w sind der Reihe nach 50, 75, 90, 100, 25, 10, 5 und 1.



```
01♦LBL "HG"     13 GTO 05     25 ACCHR      37 7
02 RCL 06       14 RDN        26 DSE Y      38 +
03 -            15 GTO 06     27 GTO 01     39♦LBL 06
04 RCL 07       16♦LBL 05     28♦LBL 02     40 127
05 /            17 -          29 LASTX      41♦LBL 03
06 167          18 7          30 7          42 ACCOL
07 *            19 /          31 *          43 DSE Y
08 1            20 INT        32 ,5         44 GTO 03
09 +            21 X<=0?      33 +          45 PRBUF
10 INT          22 GTO 02     34 7          46 END
11 7            23 31         35 MOD
12 X<=Y?        24♦LBL 01     36 INT
```

(69 Bytes)

- Quelle: Ronald Gordon (3449) (PPC CJ, V7N9P17).

22-9 Synthetische Funktion 'eGØBEEP' für Druckerfunktionen

'eGØBEEP' ist eine besondere synthetische Tastenzuweisung, die dazu benutzt werden kann, mit ganz wenigen Tasten sämtliche Druckerfunktionen aufzurufen oder in Programme einzufügen, das letztere unabhängig davon, ob der Drucker angeschlossen ist oder nicht.

Man weist 'eGØBEEP' einer Taste zu, wofür man sich am besten eines Tastenzuweisungsprogramms bedient (z.B. "KA" aus 'Synthetische Programmierung auf dem HP-41C/CV' von W.C. Wickes, S. 69-71, dort abgedruckt als Barcode-Programm im Anhang 2). Die Eingabe für dieses Programm lautet '4, ENTER, 167, ENTER, mn' ('mn' der Zeilen-Spalten-Kode der ausgewählten Taste).

eGØBEEP	Funktion	XROM
65	ACA	29,01
66	ACCHR	29,02
67	ACCOL	29,03
68	ACSPEC	29,04
69	ACX	29,05
70	BLDSPEC	29,06
71	LIST	29,07
72	PRA	29,08
73	XROM "PRAXIS"	29,09
74	PRBUF	29,10
75	PRFLAGS	29,11
76	PRKEYS	29,12
77	PRP	29,13
78	XROM "PRPLOT"	29,14
79	XROM "PRPLOTP"	29,15
80	PRREG	29,16
81	PRREGX	29,17
82	PRΣ	29,18
83	PRSTK	29,19
84	PRX	29,20
85	REGPLOT	29,21
86	SKPCHR	29,22
87	SKPCOL	29,23
88	STKPLOT	29,24

Wenn man (*mit oder ohne Drucker*) programmiert und eine Druckerfunktion ins Programm bringen will, führt man einfach 'eGØBEEP' aus, indem man (im USER-Modus) die Taste drückt, der diese Funktion zugewiesen ist, und dann die Aufforderung zur Zifferneingabe (angezeigt durch zwei Unterstriche wie z.B. im Anschluß an 'RCL') sinngemäß beantwortet. Die nebenstehende Tabelle zeigt den Zusammenhang zwischen Eingaben und Druckerfunktionen. [*Warnung:* die Eingabe von '89' verursacht im RUN-Modus einen 'GAU' (vgl. 1-23), und im PRGM-Modus verlöscht die Anzeige für etwa 20 Sekunden. Beim HP-IL-Drucker ist '89' erlaubt: '89' entspricht dort 'FMT'.] Im PRGM-Modus gelangt der entsprechende Befehl ins Programm, und zwar mit dem Funktionsnamen, wenn der Drucker angeschlossen ist, und mit dem entsprechenden XROM-Kode, falls man ohne Drucker programmiert. Im RUN-Modus wird die der Zifferneingabe entsprechende Druckerfunktion ausgeführt. Es gibt aber *zwei Ausnahmen:* 77 (für 'PRP') schlägt fehl, und 71 (für 'LIST') arbeitet anders als vorgesehen: der Druckvorgang beginnt auf der Zeile, auf der der Adreßzeiger gerade steht, und nach 71 Zeilen (oder beim Erreichen eines 'END') wird der Druck abgebrochen.

- Quelle: Robert Edelen (339) (PPC CJ, V7N3P16). - Vgl. auch 21-14.

Nachtrag: 'eGØBEEP' läßt sich auf sämtliche Funktionen des HP-IL-Moduls anwenden. Die obige Tabelle ist dazu wie folgt zu ergänzen:

eGØBEEP	Funktion	XROM	eGØBEEP	Funktion	XROM	eGØBEEP	Funktion	XROM	eGØBEEP	Funktion	XROM
01	CREATE	28,01	11	READSUB	28,11	21	WRTR	28,21	33	LOCAL	28,33
02	DIR	28,02	12	RENAME	28,12	22	WRTRX	28,22	34	MANIO	28,34
03	NEWM	28,03	13	SEC	28,13	23	WRTS	28,23	35	OUTA	28,35
04	PURGE	28,04	14	SEEKR	28,14	24	ZERO	28,24	36	PWRDN	28,36
05	READA	28,05	15	UNSEC	28,15	27	AUTOIO	28,27	37	PWRUP	28,37
06	READK	28,06	16	VERIFY	28,16	28	FINDID	28,28	38	REMOTE	28,38
07	READP	28,07	17	WRTA	28,17	29	INA	28,29	39	SELECT	28,39
08	READR	28,08	18	WRTK	28,18	30	IND	28,30	40	STOPIO	28,40
09	READRX	28,09	19	WRTP	28,19	31	INSTAT	28,31	41	TRIGGER	28,41
10	READS	28,10	20	WRTPV	28,20	32	LISTEN	28,32	89	FMT	29,25

Jedoch sollte man nicht 'eGØBEEP 03' ≙ 'NEWM' benutzen, weil man nicht zur Eingabe der Datei-Anzahl aufgefordert wird!

22-10 Standard-Zeichen-Satz ("CE")

Diese Routine druckt den Standard-Zeichen-Satz des Druckers in Form einer kompakten Matrix. Die Indizes der Matrixelemente (Zeilen- und Spalten-Nr.) bilden die 'ACCHR'-Nummer des jeweiligen Zeichens. 'XEQ "CE"' liefert den unten rechts stehenden Ausdruck.

```
01♦LBL "CE"          14 ADV          27 STO Z     40 FC? 25
02 ADV               15 "0 1 2 3 4 "  28 10        41 GTO 02
03 FIX 0             16 "├5 6 7 8 9"  29 /         42 ISG X
04 CF 29             17 ACA          30 INT       43 ISG Y
05 SF 25             18 3            31 ARCL X    44 GTO 01
06 SF 12             19 SKPCOL       32 ACA       45 GTO 00
07 CF 13             20 ,127         33 RDN       46♦LBL 02
08 " *HP-41C/V*"     21♦LBL 00       34 X<>Y      47 PRBUF
09 PRA               22 CLA          35 SF 12     48 FIX 2
10 CF 12             23 ADV          36 " "       49 SF 29
11 " STANDARD-ZEICH" 24 CF 12        37 ACA       50 CF 12
12 "├EN-SATZ"        25 ,009         38♦LBL 01    51 CLX
13 PRA               26 X<>Y         39 ACCHR     52 END
```

(148 Bytes)

```
 *HP-41C/V*
STANDARD-ZEICHEN-SATZ

   0 1 2 3 4 5 6 7 8 9
0  ♦×⌧←αβΓ↓Δσ
1  ♦λμ∡τΦΘΩδÅ
2  ȧÄäÖöÜÜÆœ≠
3  £▒ !"#$%&'
4  ()*+,-./01
5  23456789:;
6  <=>?@ABCDE
7  FGHIJKLMNO
8  PQRSTUVWXY
9  Z[\]↑_ᵀabc
10 defghijklm
11 nopqrstuvw
12 xyzπ∡→Σ├
```

- Quelle: Ronald Gordon (3449) (PPC CJ, V7N1P23) & HP Key Notes, V4N2P11.

22-11 Benennung von Daten ("DN")

Wenn man Programme dokumentiert, ist es nützlich, *die Art* der in den einzelnen Registern verarbeiteten Daten in eine Liste aufzunehmen: eine Bezeichnung wie 'Zeiger', 'Rabatt' o.ä. ist klarer als ein Zahlenbeispiel. Man kann den Drucker sinnvoll dafür einsetzen, insbesondere deswegen, weil er mehr Zeichen zur Verfügung stellt als eine gewöhnliche Schreibmaschine. Die folgende Routine ist ein solches 'Dienstprogramm'; sie fordert zunächst zur Eingabe der Registernummern auf, schaltet anschließend in den Alpha-Modus und erwartet dann den Text für die Daten-Bezeichnung. Alle Eingaben werden mit 'R/S' abgeschlossen.

```
01♦LBL "DN"          10 GTO 00          19 10        28 PRBUF
02 FIX 0             11 "LETZTES REG.?"  20 X>Y?      29 1
03 CF 29             12 PROMPT          21 "├0"      30 +
04 ADV               13 X<Y?            22 RDN       31 X<=Y?
05♦LBL 00            14 GTO 00          23 ARCL X    32 GTO 01
06 "ERSTES REG.?"    15 X<>Y            24 "├= "     33 FIX 2
07 PROMPT            16 AON             25 ACA       34 SF 29
08 ENTER↑            17♦LBL 01          26 PROMPT    35 AOFF
09 X<0?              18 "R"             27 ACA       36 END
```

(86 Bytes)

Beispiele:

```
R00= ZAEHLER

R09= % RABATT
R10= 6,5 % MWST
R11= <NOTIZEN>

R99= a
R100= b
```

- Quelle: John Dearing (2791) (PPC CJ, V8N2P17).

22-12 Text drucken ("TX")

'XEQ "TX"', wenn man Text drucken will. Sobald "TEXT?" erscheint, kann man bis zu 24 Zeichen eintasten. Beim 24. Zeichen erklingt der Warnton: man drückt '←', um das (die) letze(n) Zeichen zu löschen, falls man inmitten einer Silbe war, und tastet (falls erwünscht oder nötig) einen Bindestrich ein; dann 'R/S'. Will man den Textdruck abschließen, gibt man ein Leerzeichen ('SPACE') ein und drückt 'R/S'. Hinweis: Man kann Texte auch ganz ohne Hilfsprogramm drucken: dazu schaltet man einfach in den Alpha-Modus, tastet Text wie beschrieben ein und gibt ihn dann mit einem Druck auf die Taste PRINT (='PRA') auf dem Drucker aus; der anschließende Textteil wird genauso behandelt.

```
01♦LBL "TX"     05♦LBL 00       09 ASTO Y       13 AOFF
02 " "          06 "TEXT?"      10 PRBUF        14 CLX
03 ASTO X       07 PROMPT       11 X≠Y?         15 END
04 AON          08 ACA          12 GTO 00
```

(33 Bytes)

- Quelle: HP Key Notes, V4N3P10.

22-13 Trennlinie ("LINE")

Diese Routine benutzt einen Bindestrich (Minus-Zeichen), um - unabhängig vom Zustand von Flag 12 - eine zusammengesetzte Linie zu drucken. Es können auch andere Symbole verwendet werden.

```
'LBL "LINE", "------------", FC? 12, "⊢------------", PRA, RTN'
```

(40 Bytes)

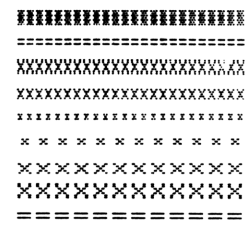
22-14 Trennlinien für Ausdrucke ("DIV" & "DV")

Diese Routinen benutzen keine numerischen Datenregister. Sobald die Frage "ACCHR-NR.?" erscheint, gibt man die gewünschte 'ACCHR'-Nr. (1-127, vgl. das Bedienungshandbuch des Druckers, S. 38/39) ein, dann 'R/S'. Unabhängig vom Zustand von Flag 12 wird eine vollständige Zeile, bestehend aus dem entsprechenden Zeichen, gedruckt. Drückt man 'R/S', ohne eine Zahl einzutasten, wird die 'Voreinstellung 45' wirksam, und es entsteht eine Zeile von Gedankenstrichen. "DIV" benutzt eine numerische Marke und umfaßt 40 Bytes; "DV" benutzt das Alpha- aber keine Daten-Register; "DV" ist 12 Bytes länger aber wesentlich schneller als "DIV". Will man eine der beiden Routinen zum aufruffähigen Unterprogramm machen, muß man "ACCHR-NR.?", '45' und 'PROMPT' löschen. Der Wert in X vor Aufruf der Routine als Unterprogramm bestimmt dann die Gestalt der Trennlinie. - Quelle: John Dearing (2791) (PPC CJ, V8N2P16).

```
01♦LBL "DIV"      10 ACCHR          04 ENTER↑    13 ASTO X
02 "ACCHR-NR.?"   11 DSE Y          05 PROMPT    14 ARCL X
03 45             12 GTO 14         06 BLDSPEC   15 FC? 12
04 PROMPT         13 PRBUF          07 CLA       16 ARCL X
05 12             14 END            08 ARCL X    17 FC? 12
06 FC? 12                           09 ARCL X    18 ARCL X
07 ST+ X          01♦LBL "DV"       10 ARCL X    19 PRA
08 X<>Y           02 "ACCHR-NR.?"   11 ASTO X    20 CLX
09♦LBL 14         03 45             12 ARCL X    21 END
```

(40 Bytes)

(52 Bytes)



22-15 Ausrichten von Druck-Spalten ("AN" & "P2")

Diese Routine druckt eine oder zwei Spalten von Zahlen mit 'ausgerichteten' Dezimalkommata, indem sie die zu überspringenden Druckpositionen vor dem Druck berechnet. Wenn Sie sehen wollen, wie die Routine arbeitet, weisen Sie "AN" der Taste 'Σ+' (11) und "P2" der Taste 'LN' (15) zu.

Gebrauchsanweisung:

1. Drücken Sie 'A', um die in X liegende Zahl im 'FIX 2'-Format und ohne Zifferngruppierung (Flag 29 gelöscht) in den Druck-Puffer zu bringen; die Anwendung der aufgelisteten Routine ist beschränkt auf Zahlen mit höchstens 7 Ziffern vor dem Dezimalkomma. Wenn der Puffer vorher leer war, wird die Zahl (einschließlich führender Leerstellen für fehlende Ziffern) linksbündig ausgedruckt, sobald man die Funktion 'PRBUF' aufruft; hinter der Zahl wird außerdem eine einzelne Leerstelle akkumuliert.

2. Man hat die Möglichkeit, eine Alpha-Kette von bis zu 12 Zeichen im Puffer anzuschließen, so daß die (in Schritt 1 akkumulierte) Zahl und die Alpha-Kette in einer Zeile gedruckt werden. Dazu schaltet man in den Alpha-Modus, gibt die Zeichen ein, schaltet Alpha wieder aus und drückt 'B'.

3. Mit 'C' wird der Pufferinhalt ausgedruckt. Vgl. Beispiele 1 & 2.

4. Die Schritte 1-3 lassen sich beliebig oft wiederholen; folgend eingegebene Zahlen werden mit untereinanderstehenden Dezimalkommata ausgedruckt.

5. Mit 'D' kann man jederzeit das Alpha-Register löschen.

6. Will man zwei Zahlen in einer Zeile drucken, geschieht dies mit 'erste Zahl, ENTER, zweite Zahl, XEQ "P2"' ('XEQ "P2"'='XEQ E').

7. Soll der Puffer ohne Ausdruck gelöscht werden, muß man den Drucker aus- und wiedereinschalten (vgl. auch 22-3).

8. Die Schritte 04 ('FIX 2') und 14 ('6') können nach Bedarf geändert werden. Im Beispiel 3 wurde '3' in Schritt 14 verwendet, was Alpha-Text bis zu 15 Zeichen erlaubt. Allgemein muß die in Schritt 14 stehende Zahl um 1 kleiner sein als die Anzahl der höchstens benötigten Vorkommastellen.

```
Beisp. 1:       123,45                    Beisp. 3: -9999,99 ABCDEFGHIJKLMNO
               -1234567,12                             5,00 SONDERAUSGABEN
                                                     400,00 MIETE
Beisp. 2:       198,00 KILOMETER
```

'A'	'B'	'C'	'D'	'E'
Zahl akkumulieren	Text akkumulieren	Puffer ausdrucken	Alpha-Reg. löschen	zwei Zahlen drucken

```
01♦LBL "AN"     12 LOG          23 RTN          34♦LBL E
02♦LBL A        13 INT          24♦LBL B        35 CF 12
03 SF 27        14 6            25 ACA          36 X<>Y
04 FIX 2        15 -            26 RTN          37 XEQ A
05 CF 29        16 SKPCHR       27♦LBL C        38 RDN
06 CLA          17 X<>Y         28 PRBUF        39 RDN
07 ENTER↑       18 ACX          29 RTN          40 XEQ A
08 INT          19 " "          30♦LBL D        41 PRBUF
09 X=0?         20 ACA          31 CLA          42 END
10 ISG X        21 SF 29        32 RTN
11 ABS          22 CLX          33♦LBL "P2"
```

(76 Bytes)

- Quelle: Richard Nelson (1) (PPC J, V6N5P31).

22-16 Text links, Zahlen rechts drucken ("AX")

Will man den Inhalt des Alpha-Registers linksbündig gedruckt sehen und den des X-Registers in derselben Zeile rechtsbündig, kann dies mit der Routine "AX" bewerkstelligt werden. Vor Aufruf von "AX" muß man eine Kontrollzahl S eintasten. S ist hierbei die Anzahl der Zeichen im Alpha-Register, vermehrt um die Anzahl der (vom gegenwärtigen 'FIX'-Modus abhängigen) Nachkommastellen. Liegt beispielsweise "NR. 01" (6 Zeichen) im Alpha-Register und 256,98 ('FIX 2'-Modus) in X, dann ist S=8. (Tip: beachten Sie, daß man 'letzte Zeichen' eines Alpha-Textes am besten mit Hilfe von 'APPEND' und '←' ändert.)

```
Nr. 01          256,98  (S = 8)     AEPFEL          99  (S = 5)
Nr. 02       58.966,01  (  "  )     ORANGEN        234  (S = 6)
Nr. 03       -5.987,63  (  "  )     APFELSINEN       5  (S = 9)
```

Für den Fall 'FIX 0, CF 29' gelten besondere Regeln: soll der X-Inhalt rechtsbündig gedruckt werden, vermindert man S um 1. Soll der X-Inhalt um k Stellen nach links verschoben gedruckt werden, muß man S-1+k (im Falle 'FIX 0, SF 29' aber S+k) als Kontrollzahl benutzen.

```
GROSS      13.          (S = 15)
KLEIN          27.      (S = 10)
TOTAL              42.  (S =  5)
```

Will man (bei festem Anzeige-Modus) stets mit derselben Kontrollzahl arbeiten, müssen die Alpha-Ketten durch Anfügen von Leerstellen auf die Länge der längsten Alpha-Kette verlängert werden. Hinweis: die Funktion 'LOG' rundet die Funktionswerte einiger großen Zahlen ab, so daß in diesem ungewöhnlichen Fällen unverwertbare Zahlen verarbeitet werden, die Fehldrucke verursachen.

```
01♦LBL "AX"     07 X≠0?         13 /            19 RDN
02 21           08 LOG          14 INT          20 ACX
03 X<>Y         09 INT          15♦LBL 00       21 PRBUF
04 -            10 FC? 29       16 -            22 END
05 RCL Y        11 GTO 00       17 ACA
06 ABS          12 ,75          18 SKPCHR
```

(39 Bytes)

- Quelle: William Cheeseman (4381) (PPC CJ, V7N5P8, V7N9P17).

22-17 Tabelle von Funktionswerten drucken ("FN")

Diese Routine druckt eine Tabelle von x- und y-Werten für jede Funktion f, die f(x) für x nach X legt. Anschließend kann die Funktion f mit "PRPLOT" aufgezeichnet werden.

Gebrauchsanweisung: 1. Man programmiert f und gibt ihr einen Namen von höchstens 6 Zeichen. Beachten Sie, daß, wenn "FN" oder "PRPLOT" in die Berechnung Ihrer Funktion f eintreten, der x-Wert in X und in R_{06} liegt. 2. 'XEQ "FN"'; Eingabe-Aufforderungen beantworten: Name von f, x_{min}, x_{max} und x_{inc} (Schrittweite von x, nur positive Werte zulässig). 3. Es wird eine Tabelle der x- und y-Werte gedruckt, und zwar mit ausgerichteten Dezimalkommata. 4. Bei Bedarf kann anschließend 'XEQ "PRPLOT"' ausgeführt werden; die Eingaben dafür können bequem der gerade entstandenen Tabelle entnommen werden.

Die Routine druckt die Werte in zwei parallelen Spalten, wobei die x- und die y-Werte je bis zu 9 Ziffern enthalten dürfen. So wie die Routine untenstehend aufgelistet ist, werden 2 Nachkommastellen ('FIX 2') angezeigt, während 7 Vorkommastellen erlaubt sind (vgl. Beispiele 1 & 2). Sind Werte negativ, wird das Minuszeichen gedruckt; positive Vorzeichen erscheinen nicht. Natürlich läßt sich das Format leicht ändern: Zeile 58 enthält eine Zahl, die um 1 kleiner als die maximal erforderliche Anzahl von Vorkommastellen ist; Schritt 02 kann entsprechend angepaßt werden: für Beispiel 3 ist Schritt 02 in 'FIX 4' und Schritt 58 in '4' abgeändert worden.

```
01♦LBL "MUSTER"
02 STO 06
03 SF 25
04 X↑2
05 LASTX
06 X↑2
07 2
08 /
09 -
10 RCL 06
11 1/X
12 -
13 END
```

Beispiel 1

WERTE D. FKT. MUSTER X	Y
-2,00	2,50
-1,00	1,50
0,00	0,00
1,00	-0,50
2,00	1,50
3,00	4,17
4,00	7,75

Beispiel 2

WERTE D. FKT. MUSTER X	Y
3237,00	5239084,50
3238,00	5242322,00
3239,00	5245560,50
3240,00	5248800,00
3241,00	5252040,50
3242,00	5255282,00
3243,00	5258524,50

Beispiel 3

WERTE D. FKT. MUSTER X	Y
0,0001	-10000,0000
0,0003	-3333,3333
0,0005	-2000,0000
0,0007	-1428,5714
0,0009	-1111,1111
0,0011	-909,0909
0,0013	-769,2308

$$x = x^2 - \frac{x^2}{2} - \frac{1}{x}$$

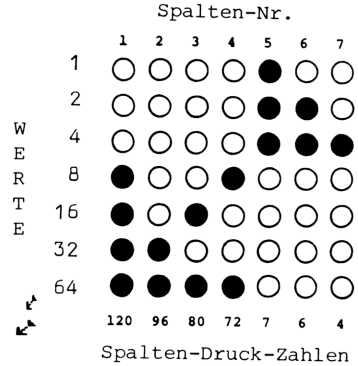
```
01♦LBL "FN"      18 PROMPT          35 STO 00     52 INT
02 FIX 2         19 STO 10          36 RCL 06     53 X=0?
03 AON           20 ADV             37 XEQ 12     54 ISG X
04 "NAME ?"      21 "  WERTE D. FKT." 38 RCL 00     55 ABS
05 PROMPT        22 "┣ "            39 XEQ 12     56 LOG
06 AOFF          23 ARCL 11         40 PRBUF      57 INT
07 ASTO 11       24 CF 12           41 RCL 10     58 6
08♦LBL 14        25 PRA             42 ST+ 06     59 -
09 "X MIN ?"     26 "   X     Y"    43 RCL 09     60 SKPCHR
10 PROMPT        27 SF 12           44 RCL 06     61 X<>Y
11 STO 06        28 PRA             45 X<=Y?      62 ACX
12 "X MAX ?"     29 "------------"  46 GTO 13     63 " "
13 PROMPT        30 PRA             47 RTN        64 ACA
14 STO 09        31 CF 12           48♦LBL 12     65 SF 29
15 X<Y?          32♦LBL 13          49 CF 29      66 END
16 GTO 14        33 RCL 06          50 CLA
17 "X INC ?"     34 XEQ IND 11      51 ENTER↑
```

(164 Bytes)

- Quelle: John Dearing (2791).

22-18 Synthetisches 'BLDSPEC'

Man stellt eine 56 Bit lange Binärzahl her, deren erste 7 Bits stets 0001000 lauten; die verbleibenden 49 Bits formen ein 7 × 7 Gitter, welches das Sonderzeichen darstellt, wobei Einsen den dunklen Punkten und Nullen den Kreisen entsprechen. Beginnen Sie in der linken unteren Ecke des Gitters und arbeiten Sie sich in der ersten Spalte nach oben bis zur ersten Zeile durch; verarbeiten Sie die zweite und jede weitere Spalte genauso (enthält das Gitter Spalten, die nur aus Nullen bestehen, müssen diese nichtsdestoweniger folgerichtig verarbeitet werden: es sind dafür 7 Nullen aufzuführen). Gruppieren Sie die entstandenen 56 Bits in Abschnitte zu je 4 Stück, und bilden Sie daraus eine 7-Zeichen-Textzeile,



deren Hexadezimalziffern den 4-er Abschnitten entsprechen. Für das obenstehende Beispiel erhalten wir:

0001	0001	1110	0011	0000	0101	0000	1001	0000	0001	1100	0011	0000	0100
1	1	E	3	0	5	0	9	0	1	C	3	0	4

Die daraus resultierende Textzeile, der ein 'TEXT 7'-Byte vorangehen muß, lautet also 'F7 11 E3 05 09 01 C3 04', das entspricht dezimal 247, 17, 227, 5, 9, 1, 195, 4. Diese Zeile fügt man dem Programm, welches das Sonderzeichen drucken soll, ein und schließt 'RCL M, ACSPEC' an (sorgen Sie dafür, daß Flag 21 gesetzt ist, bevor die Druckbefehle auftreten).

Byte-Vergleich: das normale Verfahren benötigt 30 Bytes:

'0, ENTER, 120, BLDSPEC, 96, BLDSPEC, 80, BLDSPEC, 72, BLDSPEC, 7, BLDSPEC, 6, BLDSPEC, 4, BLDSPEC, ACSPEC';

das synthetische Verfahren verwendet dagegen nur 12 Bytes:

'"▒▒⊼ ▒⊼ ▒⊼", RCL M, ACSPEC'.

Beide Verfahren erzeugen das Sonderzeichen und setzen es in den Druck-Puffer; es kann mit 'PRBUF' oder 'ADV' ausgedruckt werden. Druckt man die obige synthetische Zeile mit 'PRP' oder 'LIST' aus, erscheint sie als "Ω β σ ˣ α". Es treten also nur 5 Zeichen auf, was daran liegt, daß der Drucker in Textzeilen nur Bytes aus der oberen Hälfte der Byte-Tabelle wiedergibt. Zeichen, die Bytes aus der unteren Tabellenhälfte entsprechen, sind unsichtbar. Überdies verwendet der Druck-Puffer Bytes aus den Zeilen A, B, D und E für eigene Zwecke, die mit dem Druck von Sonderzeichen, der Zeichenbreite usw. zusammenhängen. Darum können Textzeilen, die Zeichen aus diesen 4 Zeilen enthalten, ein sehr merkwürdiges Druckbild aufweisen. Enthielte beispielsweise eine Textzeile ein dem Byte-Kode 'D5' entsprechendes Zeichen, würde eine diese Textzeile umfassende Programm-Auflistung ein Druckbild hervorrufen, in welchem die dem Byte 'D5' folgenden Zeichen in doppelter Breite und Kleinschreibung aufträten. - Quelle: William Wickes (3735), 'Synthetische Programmierung auf dem HP-41 C/CV', S. 103-105. - Vgl. auch 22-28, 25-5.

22-19 Synthetisches PPC-Siegel ("LG")

Diese Routine legt das 'PPC-Siegel' in den Druck-Puffer; es kann mit 'PRBUF' oder 'ADV' ausgedruckt werden:

PPC PPC PPC PPC PPC PPC PPC PPC PPC PPC PPC PPC PPC

Stellen Sie sicher, daß Flag 21 vor einem Ausdruck gesetzt ist.

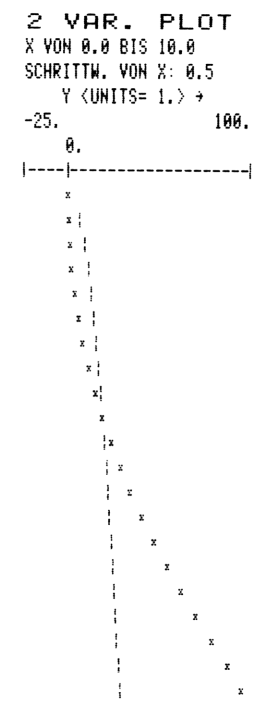
```
01♦LBL "LG"          04 X<> ]        07 ACSPEC       10 X<> ]
02 "Ω∡<zΩf>£=x"      05 ACSPEC       08 X<> [        11 END
03 "⊢Ω≠ N"           06 X<> \        09 ACSPEC
```

(47 Bytes)

Die Zeilen 02 und 03 sind synthetische Textzeilen und lauten dezimal 254, 17, 194, 228, 124, 60, 122, 241, 17, 102, 62, 30, 61, 120, 249 bzw. 248, 127, 17, 158, 29, 155, 191, 78, 135. - Quelle: Richard Nelson (1) (PPC ROM).

22-20 Plotten mit zwei Variablen ("2V")

Mit den Plot-Programmen des Druckers lassen sich nur Funktionen einer Variablen behandeln. Allerdings kann ein die x-Achse bildendes Zeichen gedruckt werden, und zwar in einer vom Benutzer festgelegten Spalte. Die Routine "2V" berechnet und plottet dieses Achsenbestimmungszeichen als Variable einer zweiten Funktion, indem sie sich die Tatsache zunutze macht, daß 'REGPLOT' (ebenso wie 'STKPLOT') das Achsenbestimmungszeichen in jede der 168 Druckspalten zu legen vermag. Die Routine fordert mit "F-1?" und "F-2?" zur Eingabe der beiden Funktionsnamen auf, mit "YMIN?" und "YMAX?" zur Eingabe des kleinsten bzw. größten y-Wertes (gültig für beide Funktionen) und mit "XMIN?", "XMAX?" und "XINC?" zur Eingabe der entsprechenden x-Werte bzw. der Schrittweite von x. *Hinweise:* Sobald die y-Werte der zweiten Funktion unter den Wert von y_{min} sinken, wird der Graph der Funktion verfälscht: die y-Werte werden dann solange in der Spalte y_{max} geplottet, bis sie sich wieder über y_{min} erheben. Beim Eintritt in die Berechnung einer der beiden Funktionen befindet sich x in X und in R_{06}. Der Achsenursprung (x = y = 0) wird dann und nur dann auf der y-Achse markiert, wenn er sich zwischen y_{min} und y_{max} (beide Werte eingeschlossen) befindet; andernfalls wird nur y_{min} markiert.



Gebrauchsanweisung: Man programmiert die beiden zu plottenden Funktionen, jede mit einem Namen von höchstens 6 Zeichen; dann 'XEQ "2V"'; Eingabe-Aufforderungen sinngemäß beantworten und 'R/S' drücken. Daraufhin wird eine Überschrift, die y-Achse und der Doppel-Plot gedruckt. Es werden mindestens 12 Datenregister benötigt. *Beispiel:* Geben Sie diese beiden Funktionen ein: 'LBL "AA", X↑2, RTN, LBL "BB", SQRT, 10, *, RTN'; rufen Sie "2V" auf und beantworten die Aufforderungen der Reihe nach mit "AA", "BB", '-25', '100', '0', '10' und '.5'; auf 'R/S' hin entsteht (bei gesetztem Flag 28) der oben gezeigte Doppel-Plot.

```
01♦LBL "2V"  23 "XINC?"       45 FIX 1          67 *
02 AON       24 PROMPT        46 "X VON "       68 R↑
03 "F-1?"    25 STO 09        47 ARCL 05        69 CLX
04 PROMPT    26 RCL 00        48 "⊢ BIS "       70 X=Y?
05 ASTO 07   27 X>0?          49 ARCL 10        71 1
06 "F-2?"    28 GTO 00        50 PRA            72 +
07 PROMPT    29 RCL 01        51 "SCHRITTW. VON" 73 INT
08 AOFF      30 X<0?          52 "⊢ X: "        74 1 E3
09 ASTO 08   31 GTO 00        53 ARCL 09        75 /
10 "YMIN?"   32 0             54 PRA            76 168
11 PROMPT    33 GTO 02        55 XROM "PRAXIS"  77 +
12 STO 00    34♦LBL 00        56 RCL 11         78 STO 02
13 "YMAX?"   35 RCL 00        57 STO 06         79 RCL 06
14 PROMPT    36♦LBL 02        58♦LBL 01         80 XEQ IND 07
15 STO 01    37 STO 04        59 XEQ IND 08     81 REGPLOT
16 "XMIN?"   38 168           60 RCL 00         82 RCL 09
17 PROMPT    39 STO 02        61 -              83 ST+ 06
18 STO 05    40 ADV           62 RCL 01         84 RCL 10
19 STO 11    41 SF 12         63 RCL 00         85 RCL 06
20 "XMAX?"   42 "2 VAR. PLOT" 64 -              86 X<=Y?
21 PROMPT    43 PRA           65 /              87 GTO 01
22 STO 10    44 CF 12         66 168            88 END
```

(197 Bytes)

- Quelle: Jake Schwartz (1820) (PPC CJ, V7N1P24).

22-21 Quergestellte Druckzeichen ("PRSW")

Wollen Sie Druckzeichen längsseitig des Papierstreifens ausgeben, benutzen Sie die unten aufgelisteten Zahlen, um daraus mit 'BLDSPEC' die entsprechenden Sonderzeichen herzustellen. Sie können diese Sonderzeichen - dem Vorschlag gemäß - in den Datenregistern R_{00}-R_{63} unterbringen und dann von dort aus holen, um sie ganz nach Bedarf in den Druck-Puffer zu legen. Vor Akkumulation müssen Sie mit Flag 12 die Druckbreite bestimmen. Wegen der Beschränkung des Puffers auf 43 Zeilen können höchstens 5 'Querzeilen' auf einmal auf dem Papierstreifen untergebracht werden (beim HP-IL-Drucker entfällt diese Beschränkung). Es ist zu empfehlen, die Querzeilen - 'SKPCOL' verwendend - durch 1-7 Druckspalten voneinander zu trennen.

Das folgende Beispiel liefert diesen Ausdruck: DMOUM

```
RCL 01, ACSPEC, 3, SKPCOL  (8 Puffer-Zellen)
RCL 02, ACSPEC, 3, SKPCOL  (       "       )
RCL 03, ACSPEC, 3, SKPCOL  (       "       )
RCL 04, ACSPEC, 3, SKPCOL  (       "       )
RCL 05, ACSPEC, PRBUF      (7 Puffer-Zellen)
```

Die folgende Routine druckt den oben rechts abgebildeten vollständigen 64-Zeichen-Satz aus.

```
01♦LBL "PRSW"     08 RCL Z      15 -          22 ACSPEC
02 ADV            09 32         16 ISG X      23 1
03 ,031           10 +          17 GTO 00     24 SKPCHR
04♦LBL 00         11 XEQ 01     18 RTN        25 RDN
05 XEQ 01         12 PRBUF      19♦LBL 01     26 CF 12
06 6              13 X<>Y       20 RCL IND X  27 ACSPEC
07 SKPCHR         14 32         21 SF 12      28 END
```

(57 Bytes)

Registeradressen, Zeichen und 'BLDSPEC'-Zahlen zur Erzeugung der Zeichen:

R00	SPACE	0	0	0	0	0	0	0
R01	A	65	65	127	65	65	34	28
R02	B	63	65	65	63	65	65	63
R03	C	62	65	1	1	1	65	62
R04	D	31	34	66	66	66	34	31
R05	E	127	1	1	31	1	1	127
R06	F	1	1	1	31	1	1	127
R07	G	94	97	113	1	1	65	62
R08	H	65	65	65	127	65	65	65
R09	I	62	8	8	8	8	8	62
R10	J	14	17	17	16	16	16	124
R11	K	65	33	17	15	17	33	65
R12	L	127	1	1	1	1	1	1
R13	M	65	65	65	73	85	99	65
R14	N	65	97	81	73	69	67	65
R15	O	28	34	65	65	65	34	28
R16	P	1	1	1	63	65	65	63
R17	Q	92	34	81	65	65	34	28
R18	R	65	33	17	63	65	65	63
R19	S	63	64	64	62	1	1	126
R20	T	8	8	8	8	8	8	127
R21	U	62	65	65	65	65	65	65
R22	V	8	20	20	34	34	65	65
R23	W	34	85	73	73	65	65	65
R24	X	65	34	20	8	20	34	65
R25	Y	8	8	8	8	20	34	65
R26	Z	127	2	4	8	16	32	127
R27	$	63	84	84	62	21	21	126
R28	%	97	98	4	8	16	35	67
R29	#	20	20	127	20	127	20	20
R30	0	29	34	69	73	81	34	92
R31	1	62	8	8	8	8	12	8
R32	2	127	1	2	60	64	65	62
R33	3	62	65	64	56	16	32	127
R34	4	16	16	127	18	20	24	16
R35	5	62	65	64	64	63	1	127
R36	6	60	66	65	63	1	2	124
R37	7	4	4	4	8	16	33	127
R38	8	62	65	65	62	65	65	62
R39	9	12	16	32	126	65	65	62
R40	.	28	28	28	0	0	0	0
R41	,	4	8	28	28	0	0	0
R42	?	8	0	8	48	64	65	62
R43	!	12	0	12	12	12	12	12
R44	:	12	12	0	0	0	12	12
R45	;	4	8	12	0	0	12	12
R46	'	0	0	0	0	12	12	12
R47	_	127	0	0	0	0	0	0
R48	"	0	0	0	0	54	54	54
R49	&	110	17	41	70	6	9	6
R50	[	62	2	2	2	2	2	62
R51	]	62	32	32	32	32	32	62
R52	+	8	8	8	127	8	8	8
R53	-	0	0	0	127	0	0	0
R54	*	8	73	42	28	42	73	8
R55	/	1	2	4	8	16	32	64
R56	=	0	127	0	0	127	0	0
R57	≠	1	2	127	8	127	32	64
R58	<	96	24	6	1	6	24	96
R59	>	3	12	48	64	48	12	3
R60	↑	8	8	8	73	42	28	8
R61	↓	8	28	42	73	8	8	8
R62	→	8	16	32	127	32	16	8
R63	←	8	4	2	127	2	4	8

- Quelle: Richard Nelson (1) (PPC CJ, V7N1P23).

22-22 Akkumulation vertikaler zweiziffriger Zahlen ("V2")

Diese Routine akkumuliert im Druck-Puffer eine kleine zweiziffrige Zahl, deren Druckbild um 90° gedreht ist. Sie kann sinnvoll eingesetzt werden, wenn man Druckstreifenabschnitte kennzeichnen oder Histogrammsäulen (vgl. 22-8) mit Zahlen versehen will.

Die Zeichen entstehen aus einer 3 × 5 Matrix; daher besetzen 2 Ziffern nur 5 Druckspalten. Die ganz kleinen Ziffern entstehen bei gelöschtem Flag 12, die größeren bei gesetztem Flag 12. Die Routine selbst akkumuliert nur die Zeichen. Für das Drukken und das Akkumulieren evtl. nötiger Abstandsspalten mit 'SKPCOL' muß der Benutzer selbst sorgen. Datenregister werden nicht verwendet, nur der Stapel. Benutzung: zweiziffrige Zahl eintasten, 'XEQ "V2"'.

```
01♦LBL "V2"      16 16          31♦LBL 01      46♦LBL 06
02 10            17 *           32 ,22232      47 ,25316
03 /             18 RCL Z       33 RTN         48 RTN
04 ENTER↑        19 10          34♦LBL 02      49♦LBL 07
05 FRC           20 *           35 ,72452      50 ,22247
06 10            21 +           36 RTN         51 RTN
07 *             22 ACCOL       37♦LBL 03      52♦LBL 08
08 XEQ IND Y     23 FRC         38 ,34243      53 ,25252
09 XEQ IND Y     24 X<>Y        39 RTN         54 RTN
10♦LBL 14        25 X≠0?        40♦LBL 04      55♦LBL 09
11 10            26 GTO 14      41 ,47564      56 ,34652
12 *             27 RTN         42 RTN         57 END
13 FRC           28♦LBL 00      43♦LBL 05
14 LASTX         29 ,25552      44 ,34317
15 INT           30 RTN         45 RTN
```

(124 Bytes)

- Quelle: Cliff Carrie (834) (PPC CJ, V8N1P15).

22-23 Wahlmöglichkeiten mit und ohne Drucker

Die folgende Zusammenstellung zeigt die verschiedenen Möglichkeiten zu drucken oder nicht zu drucken, zu unterbrechen, zu pausieren usw. in Abhängigkeit davon, ob der Drucker angeschlossen (und eingeschaltet) oder abgekoppelt ist.

Wahlmöglichkeit:		Drucker ist:	Verhalten des Systems:	Befehlsfolge:
drucken oder unterbrechen	1	angeschl.:	Druck ohne Unterbrechung	'...,SF 21, "MELDUNG", AVIEW,...'
		abgek.:	Unterbrechung zur Anzeige	
drucken oder pausieren	2	angeschl.:	Druck ohne Pause	'...,CF 21, FS? 55, SF 21, "MELDUNG", AVIEW, FC? 55, PSE, ...'
		abgek.:	Pause zur Anzeige	
drucken und unterbrechen	3	angeschl.:	Druck und Unterbrechung	'..., SF 21, "MELDUNG", AVIEW, FS? 55, STOP,...'
		abgek.:	Unterbrechung zur Anzeige	

Wahlmöglichkeit:		Drucker ist:	Verhalten des Systems:	Befehlsfolge:
drucken und pausieren	4	angeschl.:	Druck und Pause	'..., CF 21, FS? 55, SF 21, "MELDUNG", AVIEW, PSE, ...'
		abgek.:	Pause zur Anzeige	
nicht drucken aber unterbrechen	5	angeschl.:	Unterbrechung aber kein Druck	'..., "MEL DUNG", PROMPT,...'
		abgek.:	Unterbrechung zur Anzeige	
nicht drucken aber pausieren	6	angeschl.:	Pause aber kein Druck	'...,CF 21, "MELDUNG", AVIEW, PSE, ...'
		abgek.:	Pause zur Anzeige	
drucken aber nicht unterbrechen	7	angeschl.:	Druck ohne Unterbrechung	'..., CF 21, FS? 55, SF 21, "MELDUNG", FS? 55, AVIEW, ...'
		angek.:	keine Unterbrechung	

Beispiel: Am Ende einer Initialisierung soll ein Programm nach der ersten Meldung pausieren, nach der zweiten Meldung jedoch unterbrechen. Beide Meldungen sollen gedruckt werden, wenn der Drucker angeschlossen ist; anschließende Ausgaben sollen ohne Unterbrechung gedruckt werden, es sei denn, der Drucker ist abgekoppelt; in diesem Fall soll auch bei den Folgeausgaben unterbrochen werden. *Lösung:* Kombination von 4 und 3:

```
'..., CF 21, FS? 55, SF 21, "MELDUNG1", AVIEW, PSE,
SF 21, "MELDUNG2", AVIEW, FS? 55, STOP,..., "AUSGABE", AVIEW,...'
```

Vgl. auch 2-31, 3-12, 4-1, 4-18, 6-4, 6-6, 6-14.

22-24 Automatischer Ausdruck mehrerer Ausgaben in einer Zeile

Will man Ausgaben, die in einer Schleife entstehen, nicht Stück für Stück in je einer Zeile ausdrucken, hilft eine einfache in die Schleife eingebaute Befehlsfolge:

```
'..., LBL 01,..., " ", ARCL X, ACA,..., DSE 00, GTO 01, PRBUF,...'
```

Der Druck-Puffer druckt nach Akkumulation von mehr als 42 Zeichen (beim HP-IL-Drucker mehr als 98 Zeichen) automatisch eine volle Zeile, bevor er weitere Zeichen annimmt. Will man Mehrdeutigkeiten ausschließen, die auftreten können, wenn bei der akkumulierten Ausgabe Zahlen von einer Zeile in die andere überlaufen, kann " " durch "X"

ersetzt werden; oder man wählt, sofern die Ausgabe aus ganzen Zahlen besteht, den Anzeige-Modus 'FIX 0, SF 29', dann dient das Dezimalkomma als Trennzeichen.

Beispiele: 1.) Ausdruck der Teilsummen einer Reihe. 2.) Ausdruck aller Primzahlen aus einem vorgegebenen Intervall. 3.) Die folgende Musterroutine zeigt, wie man die Ausgabe in leicht lesbare Kolonnen bringen kann. "=0" druckt alle in einem bestimmten Intervall befindlichen ganzen Zahlen, für die die Hintereinanderausführung der Funktionen 'SQRT' und 'X↑2' präzis die Ausgangszahl zurückgibt. Die Routine erwartet die untere und obere Intervallgrenze in Y bzw. X. Das Beispiel rechterhand ergibt sich für '100, ENTER, 136, XEQ "=0"'. Hinter Zeile 10 ('ADV') kann man eine Textzeile von einer oder mehreren Leerstellen ('SPACE') einfügen, gefolgt von 'ACA', um die erste Zeile einzurücken oder Ausgleich für Anfangsausgaben, deren Zeilenlänge sich von den folgenden Ausgaben unterscheidet, zu schaffen.

```
100 101 102 106 109 110
114 118 120 121 126 127
128 129 130 135 136
```

```
01♦LBL "=0"     08 LASTX        15 ARCL Y       22 GTO 01
02 FIX 0        09 ENTER↑       16 X=Y?         23 PRBUF
03 CF 29        10 ADV          17 ACA          24 FIX 2
04 X<>Y         11♦LBL 01       18 SIGN         25 SF 29
05 -            12 SQRT         19 +            26 BEEP
06 ISG X        13 X↑2          20 ENTER↑       27 END
07 STO X        14 " "          21 DSE Z
```

(46 Bytes)

- Quelle: Robert Swanson (5993). Vgl. auch 15-8.

22-25 Synthetische Erzeugung von Sonderzeichen ("SC", "SCT", "SCL" & "SCX")

Die Routine "SC" erlaubt dem Benutzer, Sonderzeichen in 3er-Gruppen in einen zusammenhängenden Block von Datenregistern zu bringen. Sie können dann von dort aus abgerufen, mit 'ACSPEC' in den Druck-Puffer gebracht und mit Druckbefehlen ausgegeben werden. Sämtliche durch "SC" verfügbaren Zeichen werden weiter unten in einer Tabelle aufgelistet und in einem Register-Inhaltsverzeichnis aufgeführt.

Gebrauchsanweisung: Man lade die Register R_{06}-R_{08} wie folgt:

R_{06}: Nr. der ersten gewünschten 3er-Gruppe, l

R_{07}: Nr. der letzten gewünschten 3er-Gruppe, m

R_{08}: Adresse des ersten Registers des Blockes von Registern, welcher die Zeichen aufnehmen soll (09 oder größer), n.

Dann wird 'XEQ "SC"' ausgeführt; die Zeichen der 3er-Gruppen l bis m werden in einen zusammenhängenden Block von Registern, beginnend mit R_n, geladen. Die Routine benutzt vornehmlich die Register M, N und O zur Ablage und zum Rückruf von Kodes. Die eigentlichen Sonderzeichen werden als Textzeilen von 7 oder 14 Alpha-Zeichen ins Alpha-Register gebracht. Weil sich unter diesen Zeichen auch solche befinden, die gleichzeitig Steuerzeichen für den Druck-Puffer bilden (vgl. 22-18), ergeben sich ei-

nige Seltsamkeiten bei der Programmauflistung von "SC" (so z.B. Zeile 43). Die letzte Gruppe von Zeichen bleibt im Stapel (in X, Y und Z) zurück. Achten Sie darauf, daß Flag 21 gesetzt ist, bevor Sie die Sonderzeichen benutzen.

Beispiel 1: Man lege die Gruppe Nr. 9 (∫, √ und ©) in die Register R_{09}-R_{11} (Mindestanzahl von Datenregistern also 12). Lösung: '9, STO 06, STO 07, STO 08, XEQ "SC"'. Die Zeichen liegen anschließend so:

R_{09} und Z: ∫
R_{10} und Y: √
R_{11} und X: ©

Will man z.B. "√ " ausdrucken, gelingt das mit 'RDN' (oder 'RCL 10'), 'ACSPEC' und 'PRBUF' oder 'ADV'.

Beispiel 2: Will man die folgend abgedruckte Tabelle der verfügbaren Sonderzeichen ausgeben, tastet man "SCT" ein und führt dann (Mindestanzahl von Datenregistern 66) '1, STO 06, 19, STO 07, 9, STO 08, XEQ "SC", XEQ "SCT"' aus.

```
01♦LBL "SCT"          12 ENTER↑   23 RCL IND Z   34 GTO 14
02 ADV                13 10       24 ACSPEC      35 FIX 2
03 CF 12              14 ENTER↑   25 XEQ 13      36 SF 29
04 "  ›› SONDERZEIC"  15 1,019     26 RDN         37 CF 12
05 "HHEN ‹‹"          16♦LBL 14   27 ISG Z       38 RTN
06 PRA                17 X<Y?     28 GTO 12      39♦LBL 13
07 ADV                18 XEQ 13   29 ,003        40 " "
08 SF 12              19 ACX      30 ST+ T       41 ACA
09 FIX 0              20 XEQ 13   31 RDN         42 END
10 CF 29              21 XEQ 13   32 PRBUF
11 9,011              22♦LBL 12   33 ISG X
```

(111 Bytes)

›› SONDERZEICHEN ‹‹

1	0	1	2
2	3	4	5
3	6	7	8
4	9	0	1
5	2	3	4
6	5	6	7
7	8	9	"
8	X	Y	Z
9	∫	√	©
10	dx	dy	dt
11	±	~	≈
12	≤	≥	▽
13	-	+	Π
14	∈	Ψ	ω
15	⊡	⊡	⊡
16	⊞	⊠	⊞
17	♦	♥	♠
18	♣	◀	▶
19	⊂	⊃	₹

Beispiel 3: Will man die verfügbaren Zeichen als Inhaltsverzeichnis der Register R_{09}-R_{65} wie nachstehend abgedruckt ausgeben, tastet man "SCL" ein. Falls man schon das vorausgehende Beispiel 2 ausgeführt hat, reicht 'XEQ "SCL"'; andernfalls 'SIZE 066, 1, STO 06, 19, STO 07, 9, STO 08, XEQ "SC", XEQ "SCL"'.

```
01♦LBL "SCL"    16 ARCL X
02 ADV          17 "⊢= "
03 SF 12        18 ACA
04 FIX 0        19 RCL IND X
05 CF 29        20 ACSPEC
06 9,065        21 ADV
07 "R09= "      22 RDN
08 ACA          23 ISG X
09 RCL IND X    24 GTO 11
10 ACSPEC       25 CF 12
11 ADV          26 FIX 2
12 RDN          27 SF 29
13 ISG X        28 END
14♦LBL 11
15 "R"
```

(65 Bytes)

```
R09= 0    R24= 5    R39= ±    R54= ⊡
R10= 1    R25= 6    R40= ~    R55= ⊠
R11= 2    R26= 7    R41= ≈    R56= ⊞
R12= 3    R27= 8    R42= ≤    R57= ♦
R13= 4    R28= 9    R43= ≥    R58= ♥
R14= 5    R29= ▫    R44= ∇    R59= ♠
R15= 6    R30= x    R45= -    R60= ♣
R16= 7    R31= y    R46= •    R61= ◀
R17= 8    R32= z    R47= Π    R62= ▶
R18= 9    R33= ∫    R48= ∈    R63= ⊂
R19= 0    R34= √    R49= Ψ    R64= ⊃
R20= 1    R35= ©    R50= ω    R65= ™
R21= 2    R36= dx   R51= ⊡
R22= 3    R37= dy   R52= ⊡
R23= 4    R38= dt   R53= ⊠
```

Beispiel 4: Die Formel $e^{2x+5} \approx 4.3$ läßt sich mit der folgenden Routine "SCX" auf dem Drucker ausgeben. Voraussetzung ist, daß sich die Kodes der Sonderzeichen schon in den entsprechenden Registern befinden.

$e^{2x+5} \approx 4,3$

```
01♦LBL "SCX"    07 ACSPEC    13 ACSPEC    19 CF 12
02 FIX 1        08 RCL 30    14 RCL 41    20 FIX 2
03 SF 12        09 ACSPEC    15 ACSPEC    21 END
04 101          10 RCL 46    16 4,3
05 ACCHR        11 ACSPEC    17 ACX
06 RCL 11       12 RCL 14    18 PRBUF
```

(48 Bytes)

Die Zeichen der letzten beiden Gruppen Nr. 18 und Nr. 19 können dazu benutzt werden, die Symbole der Mondphasen zu drucken; akkumuliert man die ersten beiden Zeichen der Gruppe Nr. 19 hintereinander, hat man das 'Vollmond'-Symbol im Puffer. Weitere Beispiele findet man im PPC CJ, V7N10P14.

```
01♦LBL "SC"                 25♦LBL 05                   47 "⊢ΩGx̄>H"             72 GTO 00
02 RCL 07,                  26 "θ←      K♦♦θx̄     [♦   48 GTO 00                73♦LBL 17
03 E3                       ♦"                          49♦LBL 11                74 "θ ΔΘ8"
04 /                        27 "⊢θ♦ǂ♦♦"                 50 "θx̄$K"♦θ  AαΔΔ         75 "⊢θθǂμ"
05 ST+ 06                   28 GTO 00                   51 "⊢θäΩ"""              76 GTO 00
06 ,9                       29♦LBL 06                   52 GTO 00                77♦LBL 18
07 ST+ 08                   30 "θx̄N♦♦θ←N♦♦"             53♦LBL 12                78 "θρ@θ♦x "
08 GTO IND 06               31 "⊢θ♦ 0♦♦"                54 "θx̄          θx̄*(♦"   79 "⊢Ω♦♦"
09♦LBL 01                   32 GTO 00                   55 "⊢θμkö,F"             80 GTO 00
10 "θ♦qΩ      ♦♦θ♦♦♦♦"      33♦LBL 07                   56 GTO 00                81♦LBL 19
11 "⊢θ♦R♦♦"                 34 "θ←0♦♦θ♦0♦♦"             57♦LBL 13                82 "θ♦xHΩΓ$G♦♦"
12 GTO 00                   35 "⊢θ♦8P♦♦"                58 "θ♦ @♦♦θ♦ ♦♦"         83 "⊢θα⊢μαx"
13♦LBL 02                   36 GTO 00                   59 "⊢θx̄ǂ?"               84♦LBL 00
14 "θ♦          S♦♦θ♦8C♦    37♦LBL 08                   60 GTO 00                85 RCL ]
♦"                          38 "θ♦x                     61♦LBL 14                86 STO IND 08
15 "⊢θ♦S♦♦"                 ♦θ♦♦3               ♦"      62 "θ♦xJ♦θμaǂΓ←"         87 ISG 08
16 GTO 00                   39 "⊢θ♦x                ♦"   63 "⊢θ$↓θ"8"             88 RCL \
17♦LBL 03                   40 GTO 00                   64 GTO 00                89 STO IND 08
18 "9♦S♦♦θ♦ΔA♦♦"            41♦LBL 09                   65♦LBL 15                90 ISG 08
19 "⊢θ♦S♦♦"                 42 "θx̄α↓          @θ@ǂ@"   66 "ΩμO0'ΩμX4'"          91 RCL [
20 GTO 00                   43 "⊢θqä                    67 "⊢ΩμY4'"              92 STO IND 08
21♦LBL 04                   [illegible]                 68 GTO 00                93 ISG 08
22 "θ♦8S♦♦θxG♦♦♦"           44 GTO 00                   69♦LBL 16                94 ISG 06
23 "⊢θx̄'♦♦♦"                45♦LBL 10                   70 "Ω∡X5'Ω∡Y5'"          95 GTO IND 06
24 GTO 00                   46 "ΩGööHΩGäθö"             71 "⊢Ω∡7'"               96 END
```

(560 Bytes)

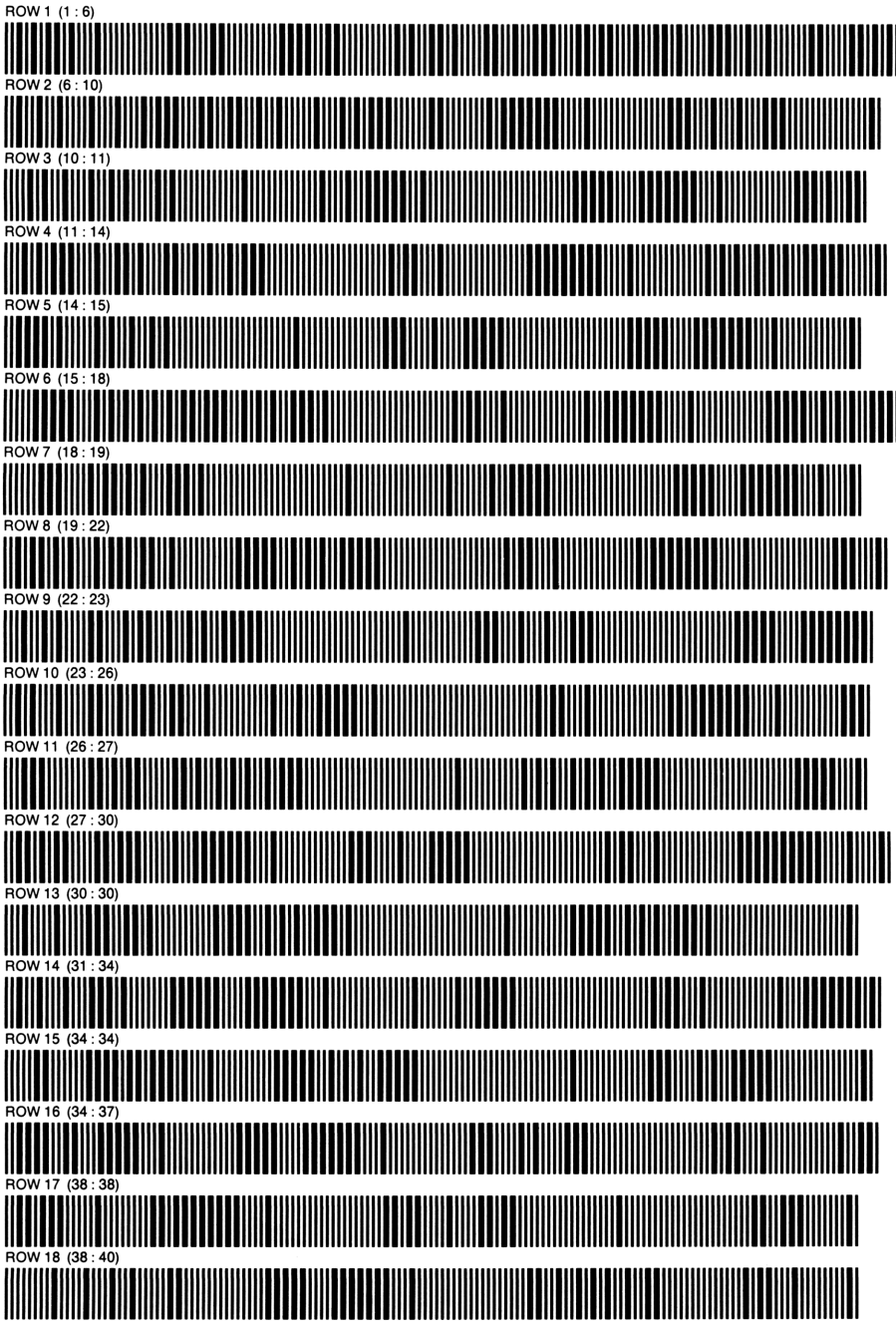
- Quelle: Jake Schwartz (PPC CJ, V7N10P11-15).

Nachstehend finden Sie eine Tabelle, mit deren Hilfe Sie die synthetischen Zeilen von "SC" unter Verwendung von "LB" (Kapitel 25) herstellen können. Vergleichen Sie sie mit der vorausstehenden Programmauflistung von "SC". Die Spalte "LB"-PROMPT gibt die Byte-Nr. an, mit der Sie von "LB" zur Eingabe des jeweils ersten Bytes der entsprechenden Zeile aus "SC" aufgefordert werden; Sie können anhand dieser Nr. laufend prüfen, ob Sie bezüglich der Byte-Nr. auch fehlerfrei eintasten. Die Anzahl der für die Verwendung von "LB" notwendigen Strohmann-Bytes beträgt 476 = 7 · INT(464/7) + 14. Lesen Sie Kapitel 25, die Beispiele eingeschlossen, bevor Sie damit beginnen, die folgenden Bytes für das Programm "SC" zu erzeugen.

"SC" Zeilen-Nr.	"LB"-PROMPT	Bytes (Dezimalwerte)
03	[1]	27, 19
10	[3]	254, 16, 0, 113, 17, 192, 0, 0, 16, 0, 137, 242, 0, 0, 0
11	[18]	248, 127, 16, 0, 233, 82, 224, 0, 0
14	[27]	254, 16, 0, 169, 83, 224, 0, 0, 16, 0, 56, 67, 224, 0, 0
15	[42]	248, 127, 16, 0, 185, 83, 160, 0, 0
18	[51]	254, 16, 0, 249, 83, 160, 0, 0, 16, 0, 8, 19, 224, 0, 0
19	[66]	248, 127, 16, 0, 249, 83, 224, 0, 0
22	[75]	254, 16, 0, 56, 83, 224, 0, 0, 16, 1, 196, 71, 0, 0, 0
23	[90]	248, 127, 16, 2, 39, 200, 0, 0, 0
26	[99]	254, 16, 3, 165, 75, 128, 0, 0, 16, 2, 165, 79, 128, 0, 0
27	[114]	248, 127, 16, 0, 225, 15, 128, 0, 0
30	[123]	254, 16, 2, 229, 78, 128, 0, 0, 16, 3, 229, 78, 128, 0, 0
31	[138]	248, 127, 16, 0, 32, 79, 128, 0, 0
34	[147]	254, 16, 3, 229, 79, 128, 0, 0, 16, 0, 225, 79, 128, 0, 0
35	[162]	248, 127, 16, 0, 56, 80, 224, 0, 0
38	[171]	254, 16, 0, 1, 176, 134, 192, 0, 16, 0, 0, 51, 128, 192, 0
39	[186]	248, 127, 16, 0, 1, 146, 164, 192, 0
42	[195]	254, 16, 2, 4, 7, 192, 64, 128, 16, 64, 130, 15, 224, 64, 129
43	[210]	248, 127, 16, 113, 21, 218, 181, 81, 28
46	[219]	254, 17, 226, 71, 240, 18, 24, 72, 17, 226, 71, 240, 22, 16, 24
47	[234]	248, 127, 17, 226, 71, 240, 2, 62, 72
50	[243]	254, 16, 2, 36, 75, 241, 34, 0, 16, 32, 32, 65, 4, 8, 8
51	[258]	248, 127, 16, 136, 137, 20, 81, 34, 34
54	[267]	254, 16, 2, 133, 138, 148, 168, 128, 16, 2, 141, 42, 150, 40, 0
55	[282]	248, 127, 16, 12, 107, 24, 44, 70, 131
58	[291]	254, 16, 0, 32, 64, 128, 0, 0, 16, 0, 32, 224, 128, 0, 0
59	[306]	248, 127, 16, 2, 15, 248, 63, 224, 128
62	[315]	254, 16, 0, 1, 197, 74, 128, 0, 16, 12, 97, 15, 228, 6, 3
63	[330]	248, 127, 16, 226, 36, 7, 16, 34, 56
66	[339]	254, 17, 254, 12, 25, 48, 96, 255, 17, 254, 12, 88, 52, 96, 255
67	[354]	248, 127, 17, 254, 12, 89, 52, 96, 255
70	[363]	254, 17, 254, 13, 88, 53, 96, 255, 17, 254, 13, 89, 53, 96, 255
71	[378]	248, 127, 17, 254, 13, 216, 55, 96, 255
74	[387]	254, 16, 32, 227, 239, 239, 142, 8, 16, 56, 251, 239, 143, 143, 142
75	[402]	248, 127, 16, 48, 243, 207, 239, 15, 12
78	[411]	254, 16, 112, 229, 255, 247, 206, 28, 16, 0, 1, 199, 207, 191, 255
79	[426]	248, 127, 17, 255, 251, 231, 199, 0, 0
82	[435]	254, 16, 0, 1, 196, 72, 160, 193, 17, 6, 10, 36, 71, 0, 0
83	[450]	248, 127, 16, 4, 127, 145, 12, 4, 120
85	[459]	144, 119
88	[461]	144, 118
91	[463]	144, 117

Sonderzeichen "SC"

Speicherplatzbedarf: 80 Register



Sonderzeichen "SC"

ROW 19 (41 : 42)

ROW 20 (42 : 44)

ROW 21 (44 : 46)

ROW 22 (46 : 47)

ROW 23 (48 : 50)

ROW 24 (50 : 51)

ROW 25 (51 : 54)

ROW 26 (54 : 55)

ROW 27 (55 : 58)

ROW 28 (58 : 59)

ROW 29 (59 : 62)

ROW 30 (62 : 63)

ROW 31 (63 : 66)

ROW 32 (66 : 67)

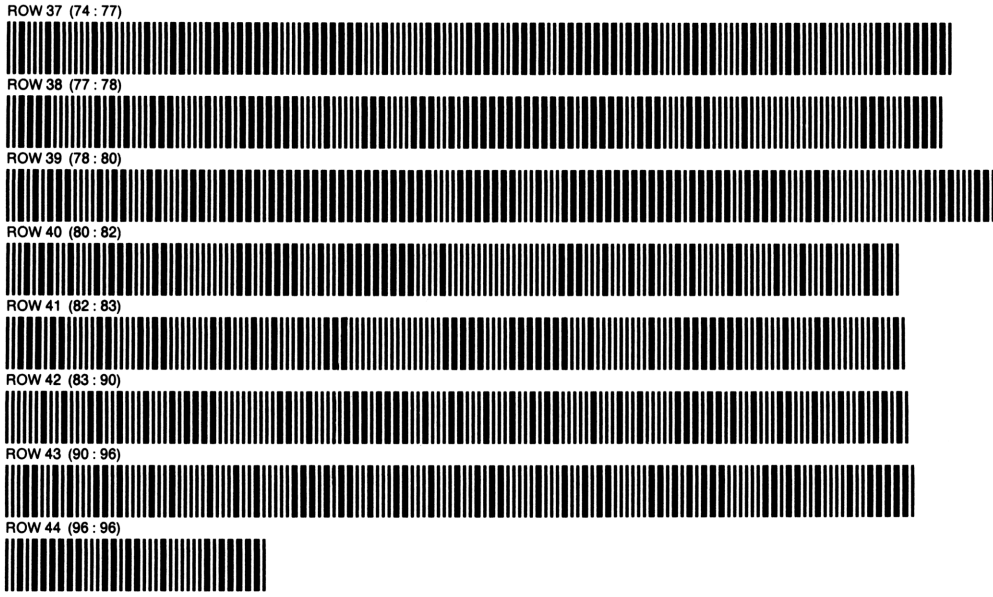
ROW 33 (67 : 70)

ROW 34 (70 : 71)

ROW 35 (71 : 74)

ROW 36 (74 : 74)

Sonderzeichen "SC"



22-26 Synthetische Erzeugung der Tierkreiszeichen ("ZODIAC")

Die folgende Tabelle gibt die Dezimalwerte an, welche als Eingaben für "LB" (25-1) dienen, um synthetische Textzeilen der Länge 14 zu erzeugen. Schließt man einer solchen Textzeile die Befehle 'RCL N, ACSPEC, RCL M, ACSPEC' an, hat man das entsprechende Tierkreissymbol im Druck-Puffer akkumuliert. Die Technik dieser synthetischen Erzeugung von Sonderzeichen ist unter Punkt 22-18 ausführlich erläutert. Hier werden jeweils *zwei* Sonderzeichen zu einem Druck-Symbol vereinigt. - Vgl. auch 22-27, 22-28.

Zeichen

1.	Widder	♈	254,16,0,16,16,32,65,124,16,8,8,16,32,128,0
2.	Stier	♉	254,16,0,3,41,177,34,68,17,18,36,214,64,0,0
3.	Zwilling	♊	254,16,0,4,20,72,159,34,16,137,242,36,80,64,0
4.	Krebs	♋	254,16,1,52,153,50,99,193,17,7,140,153,50,89,0
5.	Löwe	♌	254,16,1,5,42,174,64,129,16,5,140,152,208,16,0
6.	Jungfrau	♍	254,16,8,8,23,192,64,190,16,4,15,226,34,67,0
7.	Waage	♎	254,16,2,134,12,218,120,193,17,7,142,156,216,40,0
8.	Skorpion	♏	254,16,8,8,23,192,64,190,16,4,11,232,20,24,112,
9.	Schütze	♐	254,16,0,0,9,12,24,201,16,20,24,240,0,0,0
10.	Steinbock	♑	254,16,0,0,2,2,66,181,17,54,44,182,32,0,0
11.	Wassermann	♒	254,16,144,145,36,146,36,36,16,72,146,73,18,18,18
12.	Fische	♓	254,16,0,0,8,42,142,8,16,32,226,168,32,0,0

- Quelle: Heinz Dalkowski.

Mit der Routine "ZODIAC", welche die voranstehenden Textzeilen der Reihe nach enthält, kann man die Symbole des Tierkreises nach Bedarf im Druck-Puffer akkumulieren: man setzt die Druckbreite mit Flag 12 fest, legt die zum Symbol gehörige Zahl ('1' für 'Widder', '2' für 'Stier',...) ins X-Register und führt 'XEQ "ZODIAC"' aus. Mit 'XEQ "LISTZ"' lassen sich alle 12 Zeichen zugleich in einfacher und doppelter Druckbreite auflisten. Mit 'XEQ "WAVES"' erzeugt man eine aus dem Symbol 'Wassermann' zusammengesetzte Wellenlinie, diese allerdings - wegen der Anzahl der erforderlichen Pufferzellen - nur auf dem HP-IL-Drucker. Bezüglich des merkwürdigen Druckbildes einiger der Textzeilen 4, 7, 10, ... , 37 sei auf 22-18 verwiesen.

```
01♦LBL "ZODIAC"
02 GTO IND X
03♦LBL 01
04 "θ♦θθ A∠θΔΔθ ♦"
05 GTO 13
06♦LBL 02
07 "θ♦←)
"DQδ$.*
08 GTO 13
09♦LBL 03
10 "θ♦αάH"θ$P@♦"
11 GTO 13
12♦LBL 04
13 "θ×42cQ↓2Y♦"
14 GTO 13
15♦LBL 05
16 "θ×β*                @
θβθ♦"
17 GTO 13
18♦LBL 06
19 "θΔΔŏ        @ θαΨ"C♦"
20 GTO 13
21♦LBL 07
22 "θx̄ɯ'(♦"
23 GTO 13
24♦LBL 08
25 "θΔΔŏ        @ θαλάöρ"
26 GTO 13
27♦LBL 09
28 "θ♦♦σμöθάö♦♦♦"
29 GTO 13
30♦LBL 10
31 "θ♦♦x̄x̄B
Q6,
♦♦"
32 GTO 13
33♦LBL 11
34 "θ$$$θHIδδδ"
35 GTO 13
36♦LBL 12
37 "θ♦♦Δ*Δθ          ♦♦
"
38♦LBL 13
39 RCL \
40 ACSPEC
41 RCL [
42 ACSPEC
43 R↑
44 R↑
45 RTN
46♦LBL "LISTZ"
47 ADV
48 CF 12
49 1,012
50♦LBL 00
51 XEQ "ZODIAC"
52 "  "
53 ACA
54 FC?C 12
55 SF 12
56 FS? 12
57 GTO 00
58 PRBUF
59 ISG X
60 GTO 00
61 ADV
62 RTN
63♦LBL "WAVES"
64 SF 12
65 6
66 ENTER↑
67 11
68♦LBL 14
69 XEQ "ZODIAC"
70 DSE Y
71 GTO 14
72 PRBUF
73 CF 12
74 END
```

(320 Bytes)

- Quelle: John Dearing (2791).

22-27 Synthetische Erzeugung der Planetensymbole

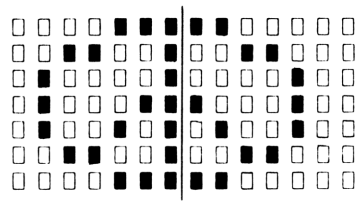
Die folgende Tabelle gibt die Dezimalwerte an, welche als Eingaben für "LB" (25-1) dienen, um synthetische Textzeilen der Länge 14 zu erzeugen. Schließt man einer solchen Textzeile die Befehle 'RCL N, ACSPEC, RCL M, ACSPEC' an, hat man das entsprechende Planetensymbol im Druck-Puffer akkumuliert. - Vgl. auch 22-18, 22-26, 22-28.

SONNE	☉	254,16,0,226,36,80,96,201,17,6,10,36,71,0,0
MOND	☽	254,16,0,0,8,48,96,227,17,117,18,35,128,0,0
MERKUR	☿	254,16,0,0,16,170,149,122,16,169,80,80,32,0,0
VENUS	♀	254,16,0,0,0,202,84,249,16,165,72,96,0,0,0
ERDE	⊕	254,16,0,226,165,82,100,255,17,38,74,165,71,0,0
MARS	♂	254,16,0,3,9,18,36,73,17,37,168,48,224,0,0
JUPITER	♃	254,16,9,11,21,41,81,32,16,131,250,4,8,16,0
SATURN	♄	254,16,0,8,23,228,68,132,16,8,20,36,68,134,0
URANUS	♅	254,16,0,168,224,129,50,127,17,144,32,65,197,64,0
NEPTUN	♆	254,16,12,32,64,137,17,127,16,137,32,64,129,1,128
PLUTO	♇	254,16,0,24,65,10,21,251,16,173,64,128,128,192,0

- Quelle: Heinz Dalkowski.

22-28 Dezimalwerte synthetisch erzeugter Sonderzeichen ("DSS")

In 22-18 wird beschrieben, wie man eine 7 × 7 Matrix auswertet, um die Bytes zur synthetischen Herstellung von Sonderzeichen zu ermitteln. Insbesondere kann man die entsprechenden Dezimalzahlen als Eingabewerte für "LB" (25-1) benutzen. Das folgende Programm enthebt den Benutzer der Mühe, die Dezimalwerte selbst festzustellen. Es erwartet die gemäß 22-18 ausgezählten Einsen und Nullen in den Registern R_{01}-R_{07} und druckt dann die Dezimalwerte aus. Die Inhalte von R_{01}-R_{07} werden durch das Programm abgeändert. Beispiel: Das Friedenssymbol sei mit Hilfe zweier 7 × 7 Matrizen so dargestellt:



Dann liefert - führende Nullen außer acht lassend (man zählt von unten aus!) -

'0, STO 01, 11100, STO 02, 100010, STO 03, STO 04, 1010001, STO 05, 1001001, STO 06, 1111111, STO 07, XEQ "DSS"'

den Ausdruck

"16 0 226 36 84 100 255",

und anschließendes

'1001001, STO 01, 1010001, STO 02, 100010, STO 03, STO 04, 11100, STO 05, 0, STO 06, STO 07, R/S'

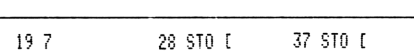
gibt den Ausdruck

"17 38 138 36 71 0 0".

Mithin lautet die Textzeile, die sich mit "LB" erzeugen läßt (nicht das vorangehende 'TEXT 14'-Byte vergessen),

254, 16, 0, 226, 36, 84, 100, 255, 17, 38, 138, 36, 71, 0, 0.

Folgt ihr 'RCL N, ACSPEC, RCL M, ACSPEC, PRBUF', wird das Friedenszeichen ausgedruckt:



```
01♦LBL "DSS"  10 ISG 08     19 7          28 STO [      37 STO [      46 2        55 RCL IND 08
02♦LBL 12     11 GTO 13     20 RCL 08     29 "⊢♦♦♦♦♦♦"  38 "⊢↑"       47 *        56 FRC
03 FIX 0      12 ADV        21 INT        30 RCL [      39 RCL \      48 DSE 00   57  E8
04 CF 29      13 SF 29      22 -          31 4          40 X<> d      49 GTO 03   58 *
05 1,007      14 FIX 2      23 10↑X       32 *          41 0          50 RDN      59 STO IND 08
06 STO 08     15 STOP       24 /          33 7,00304    42 1          51 X<>Y     60 END
07  E4        16 GTO 12     25 STO IND 08 34 +          43♦LBL 03     52 X<> d
08♦LBL 13     17♦LBL 14     26 INT        35 STO 00     44 FS? IND 00 53 X<>Y
09 XEQ 14     18 RCL IND 08 27 +          36 0          45 ST+ Y      54 ACX
```

- Quelle: Heinz Dalkowski. - Vgl. auch 25-5. (114 Bytes)

Kapitel 23

SCHRIFT-FAHNEN

23-1 Druck von Fahnen ("BANR", "CHAR" & "CODE")

Diese Routine kann dazu benutzt werden, 'Fahnen' zu drucken. Für insgesamt 95 Fahnen-Zeichen werden 'Steuerkodes' angegeben. Die Routine selbst paßt auf eine einzelne Magnetkartenspur; für die von der Routine benötigten Daten sind zusammen 7 Kartenspuren erforderlich. Die ersten 10 Register enthalten 'Blockbausteine' für die Fahnen-Zeichen; R_{10} ist ein Notizregister für die Routine; die Register R_{11}-R_{105} enthalten die 95 Steuerkodes. Um Fahnen-Zeichen zu erhalten, die die gesamte Streifenbreite ausfüllen, muß Flag 12 gesetzt sein; der Zustand von Flag 12 kann zwischen dem Druck einzelner Zeichen verändert werden, so daß Ausdrucke, in denen Zeichen voller mit solchen halber Höhe wechseln, möglich sind.

Gebrauchsanweisung: Es gibt 3 Möglichkeiten, die Routine zu verwenden. (1) 'XEQ "BANR"' führt zur Aufforderung "ZEICH. NR.?"; darauf antwortet man mit der Kode-Nr. (0 für eine Leerstelle, 1 für A, 2 für B usw.), dann 'R/S'; das der Kode-Nr. entsprechende Fahnen-Zeichen wird gedruckt, und die Eingabe-Aufforderung erscheint erneut, so daß man mit dem nächsten Zeichen fortfahren kann. (2) Man kann auch die Kode-Nr. eintasten und 'XEQ "CHAR"' ausführen; die Marke "CHAR" ist für die Benutzung der Routine als Unterprogramm, das von anderen Programmen aufgerufen wird, gedacht. (3) Schließlich kann man auch einen der Steuerkodes (z.B. '.1802020218' für 'A') eintasten und 'XEQ "CODE"' ausführen; dann wird ebenfalls das entsprechende Fahnen-Zeichen gedruckt. Statt der untenstehenden Steuerkodes lassen sich auch selbständig gebildete Steuerkodes verwenden, mit denen man Zeichen eigener Bauart drucken kann. Falls wenig Platz im Speicher zur Verfügung steht, legt man nur die Routine, die Blockbausteine und das Notizregister ('SIZE 011') in den Speicher und benutzt "CODE".

Herstellung der Blockbausteine: Zunächst '0, ENTER, 31, XEQ "BLDSPEC"' (vgl. 1-1); dann liegt das Alpha-Zeichen 31 - angezeigt als Vollzeichen ▩ - in X (natürlich kann man statt des Zeichens 31 auch ein beliebiges anderes Zeichen wählen). Anschließend benutzt man - im Alpha-Modus - 'ARCL X', 'SPACE' und 'ASTO mn', um die Blockbausteine herzustellen und in den Registern abzulegen: 'CLA, SPACE (6mal), ASTO 00' für R_{00}, 'CLA, ARCL X (6mal), ASTO 01' für R_{01}, 'CLA, ARCL X, SPACE (4mal), ARCL X, ASTO 02' für R_{02} usw.

```
01♦LBL "BANR"     10 X=0?          19 ARCL IND X   28 ADV
02 "ZEICH. NR.?"  11 GTO 00        20 FRC          29 ADV
03 PROMPT         12 10            21 10           30 RTN
04 XEQ "CHAR"     13 +             22 *            31♦LBL "CODE"
05 GTO "BANR"     14 RCL IND X     23 ARCL IND X   32 ,004
06♦LBL "CHAR"     15♦LBL 00        24 FRC          33 STO 10
07 ,004           16 CLA           25 PRA          34 RDN
08 STO 10         17 10            26 ISG 10       35 GTO 00
09 RDN            18 *             27 GTO 00       36 END
```

(97 Bytes)

Beispiel: Die Routine 'LBL "X", 1.095, LBL 01, ENTER, XEQ "CHAR", R↑, ISG X, GTO 01, RTN' druckt sämtliche 95 Fahnenzeichen, vorausgesetzt, die Register sind mit den Steuerkodes gefüllt und die Blockbausteine liegen bereit.

Datenregister, Blockbausteine bzw. Steuerkodes, Kode-Nr. und Fahnen-Zeichen

R00 = " "
R01 = "▮▮▮▮▮▮"
R02 = "▮ ▮"
R03 = " ▮"
R04 = "▮ "
R05 = " ▮▮▮▮ "
R06 = " ▮▮ "
R07 = " ▮▮ "
R08 = "▮▮▮▮▮ "
R09 = " ▮▮▮▮▮"
R10 = 0 0 = SPACE
R11 = .1802020218 1 = A
R12 = .1142424295 2 = B
R13 = .9843434367 3 = C
R14 = .1143434398 4 = D
R15 = .1142424243 5 = E
R16 = .1102020203 6 = F
R17 = .9843432353 7 = G
R18 = .1104040411 8 = H
R19 = .4343114343 9 = I
R20 = .5040439103 10 = J
R21 = .1134766743 11 = K
R22 = .1140404040 12 = L
R23 = .1107060711 13 = M
R24 = .1105345011 14 = N
R25 = .9843434398 15 = O
R26 = .1102020205 16 = P
R27 = .9843436398 17 = Q
R28 = .1132726245 18 = R
R29 = .0542424293 19 = S
R30 = .0303110303 20 = T
R31 = .9140404091 21 = U
R32 = .0190409001 22 = V
R33 = .1160706011 23 = W
R34 = .8976347689 24 = X
R35 = .0934803409 25 = Y
R36 = .4383224943 26 = Z

R37 = .1134343434 27 = ├
R38 = .0003090300 28 = ┬
R39 = .3074967430 29 = ◆

R40 = .9823224298 30 = 0
R41 = .4047114040 31 = 1
R42 = .8723424245 32 = 2
R43 = .6743424295 33 = 3
R44 = .3130301130 34 = 4
R45 = .6142424293 35 = 5
R46 = .9823232357 36 = 6
R47 = .0383330209 37 = 7
R48 = .9542424295 38 = 8
R49 = .0502020218 39 = 9

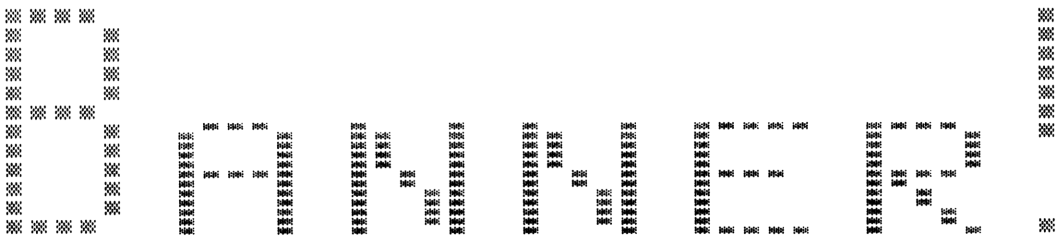
R50 = .6 40 =.
R51 = .406 41 = ,
R52 = .0703130205 42 = ?
R53 = .21 43 = !
R54 = .67 44 = :
R55 = .4067 45 = ;
R56 = .000007 46 = '
R57 = .00006 47 = .
R58 = .00070007 48 = "
R59 = .5522526580 49 = &

R60 = .0000114343 50 = [
R61 = .434311 51 =]
R62 = .0034554343 52 = {
R63 = .43435534 53 = }
R64 = .0000986743 54 = (
R65 = .436798 55 =)
R66 = .0607110706 56 = ↑
R67 = .7060116070 57 = ↓
R68 = .2267987634 58 = →
R69 = .3476986722 59 = ←
R70 = .3434983434 60 = +

R71 = .3434343434 61 = -
R72 = .00763476 62 = x
R73 = .6776117667 63 = *
R74 = .3434223434 64 = ÷
R75 = .4060980703 65 = /
R76 = .0307986040 66 = \
R77 = .7676767676 67 = =
R78 = .7676117676 68 = ≠
R79 = .0606060031 69 = $^{-1}$
R80 = .3476556743 70 = <
R81 = .4367557634 71 = >
R82 = .8024854743 72 = Δ
R83 = .4542114293 73 = $
R84 = .0418424347 74 = £
R85 = .9843114367 75 = ¢
R86 = .4760980763 76 = %
R87 = .6711671167 77 = #
R88 = .9843464265 78 = @
R89 = .05020205 79 = °

R90 = .9044449044 80 = α
R91 = .1862426295 81 = β
R92 = .1103030303 82 = Γ
R93 = .8028432880 83 = Δ
R94 = .9822222298 84 = θ
R95 = .1060406090 85 = μ
R96 = .3014041406 86 = π
R97 = .4389892222 87 = Σ
R98 = .5020209030 88 = σ
R99 = .3030103004 89 = τ

R100 = .1111111111 90 = ▮
R101 = .0303030303 91 = $\bar{}$
R102 = .4040404040 92 = _
R103 = .05020202 93 = ^{c}
R104 = .5020202050 94 = o
R105 = .1802184242 95 = Æ



- Quelle: Dean Lampman (41) (PPC J, V6N6P16).

23-2 Text-Fahnen ("LET", "DIG" & "SYM")

Diese Fahnendruck-Routine druckt die Buchstaben, die Ziffern und 45 weitere Symbole. Dazu werden geeignete 'Steuerkodes' in die Register R_{01}-R_{94} geladen. Jedes Fahnendruckzeichen wird aus 'Bausteinen' zusammengesetzt, die ihrerseits gerade aus dem Alpha-Zeichen gebildet werden, das mit dem jeweiligen Fahnendruckzeichen übereinstimmt. Bei gesetztem Flag 12 haben die Fahnendruckzeichen volle Höhe und bestehen aus dem zugehörigen Großbuchstaben; bei gelöschtem Flag 12 haben sie halbe Höhe und bestehen aus dem jeweiligen Kleinbuchstaben. Bemerkung: die Routine ist sehr langsam; will man eine eigene Schrift-Fahne drucken, sollte man daher am besten dafür eine Routine schreiben; z.B. druckt die folgende Routine "X" das Wort 'Banner!'.

```
01♦LBL "X"      07 XEQ "LET"     13 XEQ "LET"     19 BEEP
02 SF 12        08 14            14 18            20 OFF
03 2            09 XEQ "LET"     15 XEQ "LET"     21 END
04 XEQ "LET"    10 14            16 SF 12
05 CF 12        11 XEQ "LET"     17 33
06 1            12 5             18 XEQ "SYM"
```

(62 Bytes)

Gebrauchsanweisung: Routine und Daten laden. Flag 12 einstellen. Buchstaben drucken: Kode-Nr. eingeben (s. nachstehende Tabelle), 'XEQ "LET"'. Ziffer drucken: Ziffer eingeben, 'XEQ "DIG"'. Symbol drucken: Kode-Nr. eingeben, 'XEQ "SYM"'. Eine Leerstelle wird mit '32, XEQ "SYM"' 'gedruckt'. Die nachstehend aufgelistete Routine paßt auf 3 Magnetkartenspuren; die Daten dafür benötigen genau 6 Spuren. Will man allein Buchstaben drucken, setzt man 'SIZE 039' fest, bevor man die Datenkarten einliest.

Kode-Nummern

Buchstaben			Ziffern	Symbole				
1 = A	11 = K	21 = U	0 = 0	3 = ←	29 = ≠	39 = '	58 = :	93 =]
2 = B	12 = L	22 = V	1 = 1	4 = α	30 = £	40 = (	59 = ;	94 = ↑
3 = C	13 = M	23 = W	2 = 2	5 = β	31 = ▮	41 =)	60 = <	95 = _
4 = D	14 = N	24 = X	3 = 3	6 = Γ	32 =	42 = *	61 = =	96 = $\overline{\text{T}}$
5 = E	15 = O	25 = Y	4 = 4	7 = ↓	33 = !	43 = +	62 = >	
6 = F	16 = P	26 = Z	5 = 5	8 = Δ	34 = "	44 = ,	63 = ?	123 = π
7 = G	17 = Q		6 = 6	9 = σ	35 = #	45 = -	64 = @	124 = |
8 = H	18 = R		7 = 7	10 = ◆	36 = $	46 = .		125 = →
9 = I	19 = S		8 = 8		37 = %	47 = /	91 = [	126 = Σ
10 = J	20 = T		9 = 9		38 = &		92 = \	127 = ⊢

```
01♦LBL "LET"    39 STO 38     77 X<=Y?       115 XEQ 00
02 1            40 3          78 GTO 08      116 FRC
03 X>Y?         41 X>Y?       79 CLX         117 10
04 GTO 08       42 GTO 08     80 33          118 *
05 CLX          43 CLX        81 -           119 RCL IND X
06 27           44 11         82 GTO 09      120 XEQ 00
07 X<=Y?        45 X>Y?       83♦LBL 04      121 FRC
08 GTO 08       46 GTO 04     84 CLX         122 PRBUF
09 X<>Y         47 CLX        85 47          123 ISG 10
10 STO 38       48 29         86 +           124 GTO 10
11 10           49 X>Y?       87 GTO 09      125 ADV
12 +            50 GTO 08     88♦LBL 05      126 ADV
13 96           51 CLX        89 CLX         127 RTN
14 ST+ 38       52 48         90 29          128♦LBL 00
15 32           53 X>Y?       91 +           129 ,005
16 FS? 12       54 GTO 05     92 GTO 09      130 STO 37
17 ST- 38       55 CLX        93♦LBL 06      131 RDN
18 ,004         56 58         94 CLX         132♦LBL 03
19 STO 10       57 X>Y?       95 19          133 10
20 RCL IND T    58 GTO 08     96 +           134 *
21 GTO 10       59 CLX        97 GTO 09      135 ENTER↑
22♦LBL "DIG"    60 65         98♦LBL 07      136 INT
23 X<0?         61 X>Y?       99 CLX         137 X=0?
24 GTO 08       62 GTO 06    100 7           138 GTO 01
25 10           63 CLX       101 -           139 RCL 38
26 X<=Y?        64 91        102 GTO 09      140 GTO 02
27 GTO 08       65 X>Y?      103♦LBL 08      141♦LBL 01
28 X<>Y         66 GTO 08    104 "KODE-FEHLER" 142 32
29 48           67 CLX       105 PROMPT      143♦LBL 02
30 +            68 97        106 RTN         144 ACCHR
31 STO 38       69 X>Y?      107♦LBL 09      145 RDN
32 8            70 GTO 07    108 ,004        146 RDN
33 -            71 CLX       109 STO 10      147 FRC
34 ,004         72 123       110 RCL IND Y   148 ISG 37
35 STO 10       73 X>Y?      111♦LBL 10      149 GTO 03
36 RCL IND Y    74 GTO 08    112 10          150 RDN
37 GTO 10       75 CLX       113 *           151 END
38♦LBL "SYM"    76 128       114 RCL IND X
```

(265 Bytes)

Daten für "LET": Nachstehend werden die Daten aufgelistet, auf die die Routine "LET" zugreift. Die Zahlen in R_{00}-R_{09} dienen zur Herstellung der Bausteine; R_{10} und R_{37} sind Zählregister für Schleifen; R_{38} enthält die Kode-Nr.; R_{39} ist unbenutzt; in den übrigen Registern liegen die Steuerkodes.

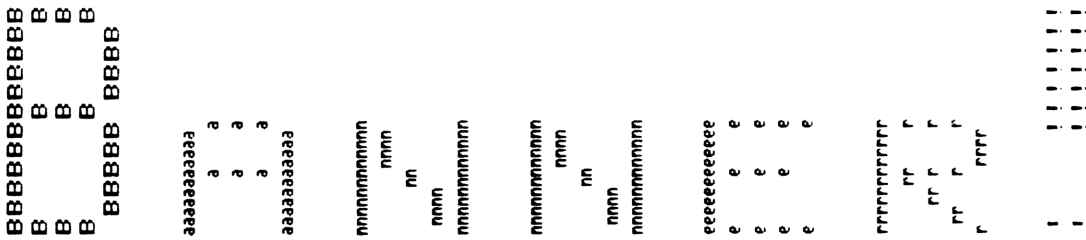
```
R00= 0
R01= 0.111111
R02= 0.100001
R03= 0.000001
R04= 0.100000
R05= 0.011110
R06= 0.011000
R07= 0.000110
R08= 0.111110
R09= 0.011111
R10= (Notizen)
R11= 1.802020218-01   A
R12= 1.142424295-01   B
R13= 9.843434367-01   C
R14= 1.143434398-01   D
R15= 1.142424243-01   E
R16= 1.102020203-01   F
R17= 9.843432353-01   G
R18= 1.104040411-01   H
R19= 4.343114343-01   I
R20= 5.040439103-01   J
R21= 1.134766743-01   K
R22= 1.140404040-01   L
R23= 1.107060711-01   M
R24= 1.105345011-01   N
R25= 9.843434398-01   O
R26= 1.102020205-01   P
R27= 9.843436398-01   Q
R28= 1.132726245-01   R
R29= 5.424242930-02   S
R30= 3.031103030-02   T
R31= 9.140404091-01   U
R32= 1.904090010-02   V
R33= 1.160706011-01   W
R34= 8.976347689-01   X
R35= 9.348034090-02   Y
R36= 4.383224943-01   Z
R37= (Notizen)
R38= (Notizen)
R39= (unbenutzt)
R40= 9.823224298-01   0
R41= 4.047114040-01   1
R42= 8.723424245-01   2
R43= 6.743424295-01   3
R44= 3.130301130-01   4
```

R45= 6.142424293-01 5
R46= 9.823232357-01 6
R47= 3.833302090-02 7
R48= 9.542424295-01 8
R49= 5.020202180-02 9
R50= 3.476986722-01 ←
R51= 9.044449044-01 α
R52= 1.862426295-01 β
R53= 1.103030303-01 Γ
R54= 7.060116070-01 ↓
R55= 8.028432880-01 Δ
R56= 5.020209030-01 σ
R57= 3.074967430-01 ◆
R58= 7.676117676-01 ≠
R59= 4.184243470-02 £
R60= 1.111111111-01 ■
R61= 0.000000000

R62= 2.121000000-01 !
R63= 7.070007070-02 "
R64= 6.711671167-01 #
R65= 4.542114293-01 $
R66= 4.760980763-01 %
R67= 5.522526580-01 &
R68= 7.070000000-06 '
R69= 9.867430000-05 (
R70= 4.367980000-01)
R71= 6.776117667-01 *
R72= 3.434983434-01 +
R73= 4.060600000-01 ,
R74= 3.434343434-01 -
R75= 6.060000000-03 .
R76= 4.060980703-01 /
R77= 6.767000000-01 :
R78= 4.067670000-01 ;

R79= 3.476556743-01 <
R80= 7.676767676-01 =
R81= 4.367557634-01 >
R82= 7.031302050-02 ?
R83= 9.843464265-01 @
R84= 1.143430000-05 [
R85= 3.079860400-02 \
R86= 4.343110000-01]
R87= 6.071107060-02 ↑
R88= 4.040404040-01 _
R89= 3.090300000-04 ▀
R90= 3.014041406-01 ▚
R91= 1.100000000-05 ▕
R92= 2.267987634-01 →
R93= 4.389892222-01 Σ
R94= 1.134343434-01 ▙

- Quelle: Bruce Murdock (2916) (PPC CJ, V7N1P27).



Kapitel 24

Auflösung von Gleichungen nach verschiedenen Variablen

24-1 Austausch der Lösungsvariablen I ("IS1")

Dies ist eine Programm-*Skizze*, kein Programm. Sie muß im Einzelfall Ihren speziellen Bedingungen angepaßt werden; so wie sie geschrieben ist, kann sie dazu dienen, eine Gleichung, die 5 Variable A, B, C, D und E miteinander verknüpft, nach jeder Variablen aufzulösen, wenn die Werte der jeweils vier anderen Variablen bekannt sind. Liegen mehr als 5 Variable vor, können die Marken 'F'-'J' und die Register R_{06}-R_{10} zur sinngemäßen Erweiterung benutzt werden. Es erfolgt kein Druck. Benutzung: 1. 'XEQ "IS1"'. 2. Eingabe-Aufforderungen beantworten, wenn der Wert der Variablen gegeben ist, andernfalls mit 'R/S' überspringen. 3. Um nach der Unbekannten aufzulösen, entsprechende Taste drücken. 4. Soll der Wert nur einer Variablen geändert werden, neuen Wert eintasten und 'STO' mit der Taste dieser Variablen ausführen [z.B. 'STO D' (= 'STO 04'), wenn der Wert der Variablen D geändert werden soll]; dann Schritt 3 wiederholen. - Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

'A'	'B'	'C'	'D'	'E'
LBL "IS1"	STO 03	(Ber. v. A)	(Ber. v. C)	(Ber. v. E)
SF 27	"D?"	STO 01	STO 03	STO 05
"A?"	PROMPT	"A"	"C"	"E"
PROMPT	STO 04	GTO 88	GTO 88	LBL 88
STO 01	"E?"	LBL B	LBL D	"⊢="
"B?"	PROMPT	(Ber. v. B)	(Ber. v. D)	ARCL X
PROMPT	STO 05	STO 02	STO 04	PROMPT
STO 02	CLX	"B"	"D"	END
"C?"	STOP	GTO 88	GTO 88	
PROMPT	LBL A	LBL C	LBL E	

R00 : (unbenutzt)
R01: A
R02: B
R03: C
R04: D
R05: E

Beispiel: Die Gleichung laute $A^B + C^D = E^2$, dann wird unter

'LBL A': $A = (E^2 - C^D)^{1/B}$

'LBL B': $B = \ln(E^2 - C^D)/\ln A$

'LBL C': $C = (E^2 - A^B)^{1/D}$

'LBL D': $D = \ln(E^2 - A^B)/\ln C$

'LBL E': $E = \sqrt{A^B + C^D}$

berechnet.

24-2 Austausch der Lösungsvariablen II ("IS2")

Diese Version druckt alle Ein- und Ausgaben. Benutzung: 1. 'XEQ "IS2"'. 2. Eingabe-Aufforderungen beantworten bzw. mit 'R/S' überspringen. 3. Unbekannte durch Druck auf die zugehörige Taste berechnen. 4. Zur Abänderung eines Einzelwertes Schritte 1 und 2 erneut ausführen und nur die Aufforderung zur Eingabe des zu ändernden Variablen-Wertes beantworten; alle anderen Aufforderungen mit 'R/S' überspringen; dann Schritt 3 wiederholen. - Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

'A'	'B'	'C'	'D'	'E'
LBL "IS2"	"E"	"B"	LBL E	STO IND 00
SF 27	XEQ 99	GTO 88	(Ber. v. E)	FS? 22
0	CLX	LBL C	STO 05	FC? 55
STO 00	STOP	(Ber. v. C)	"E"	RTN
"A"	LBL A	STO 03	GTO 88	ARCL X
XEQ 99	(Ber. v. A)	"C"	LBL 99	AVIEW
"B"	STO 01	GTO 88	CF 22	RTN
XEQ 99	"A"	LBL D	1	LBL 88
"C"	GTO 88	(Ber. v. D)	ST+ 00	"⊢="
XEQ 99	LBL B	STO 04	RCL IND 00	ARCL X
"D"	(Ber. v. B)	"D"	"⊢="	AVIEW
XEQ 99	STO 02	GTO 88	PROMPT	END

R00: Zeiger
R01: A
R02: B
R03: C
R04: D
R05: E

Soll dieses Programm so abgeändert werden, daß ein neuer Wert durch Eingabe ins X-Register und anschließenden Druck auf die zugehörige Taste in sein Register gelangt statt durch die Wiederholung der Schritte 1 und 2, müssen zusätzliche Befehle eingefügt werden, die (beispielsweise für A) so lauten:

'LBL A, STO 01, "A", FS?C 22, GTO 88, (Ber. von A), STO 01, GTO 88'.

24-3 Austausch der Lösungsvariablen III ("IS3")

Es erfolgt kein Druck. Benutzung: 1. 'XEQ "IS3"'. 2. Eingaben tätigen bzw. Aufforderungen mit 'R/S' überspringen. 3. Die Unbekannte wird *automatisch* berechnet und angezeigt. Alle Werte müssen neu eingetastet werden, wenn der Programmlauf wiederholt werden soll. - Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

LBL "IS3"	XEQ 99	"B"	STO 04	"⊢ ?"	
1.1	"E"	STO 02	GTO 88	PROMPT	
STO 00	XEQ 99	GTO 88	LBL 05	STO IND 00	
CF 22	GTO IND 06	LBL 03	(Ber. v. E)	RCL 00	
"A"	LBL 01	(Ber. v. C)	"E"	FC?C 22	
XEQ 99	(Ber. v. A)	"C"	STO 05	STO 06	
"B"	"A"	STO 03	LBL 88	ISG 00	
XEQ 99	STO 01	GTO 88	"⊢="	END	
"C"	GTO 88	LBL 04	ARCL X		
XEQ 99	LBL 02	(Ber. v. D)	PROMPT		
"D"	(Ber. v. B)	"D"	LBL 99		

R00: Register-Zeiger
R01: A
R02: B
R03: C
R04: D
R05: E
R06: Unterprogramm-Zeiger

24-4 Austausch der Lösungsvariablen IV ("IS4")

Alle Ein- und Ausgaben werden gedruckt; Eingabe-Aufforderungen erfolgen nicht! Benutzung: 1. 'XEQ "IS4"'. 2. Die gegebenen Variablen-Werte werden eingetastet und jeweils die zugehörigen Tasten gedrückt. 3. Die Unbekannte wird mit Druck auf ihre Taste berechnet. 4. Zur Abänderung eines Wertes diesen eintasten und die zugehörige Taste drücken. 5. Will man einen Wert rückrufen, kann man einfach seine Taste drücken (Neuberechnung) oder 'RCL' und dann seine Taste.

'A'	'B'	'C'	'D'	'E'
LBL "IS4"	(Ber. v. A)	GTO 88	STO 04	FS?C 22
SF 27	STO 01	LBL C	"D"	GTO 88
"READY"	GTO 88	STO 03	FS?C 22	(Ber. v. E)
PROMPT	LBL B	"C"	GTO 88	STO 05
LBL A	STO 02	FS?C 22	(Ber. v. D)	LBL 88
STO 01	"B"	GTO 88	STO 04	"⊢="
+)	FS?C 22	(Ber. v. C)	GTO 88	ARCL X
"A"	GTO 88	STO 03	LBL E	AVIEW
FS?C 22	(Ber. v. B)	GTO 88	STO 05	STOP
GTO 88	STO 02	LBL D	"E"	END

R00: (unbenutzt)
R01: A
R02: B
R03: C
R04: D
R05: E

Die voranstehende Skizze arbeitet mit 5 Variablen. Bei Verwendung der Marken 'F'-'J' und der Register R_{06}-R_{10} kann dieses Verfahren auf Gleichungen mit bis zu 10 Variablen ausgedehnt werden. Die Verwendung eigener Ausgabe-Texte steht frei. - Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

+) An dieser Stelle können Eingaben umgeformt oder teilberechnet werden, bevor sie zur Verarbeitung gelangen. Soll eine Berechnung beispielsweise mit A/100 (oder log A) statt mit dem bloßen Eingabewert A erfolgen und ist z.B. R_{11} frei, dann setzt man hinter 'STO 01' die Befehlsfolge

'100, /, (bzw. LOG), STO 11, RCL 01'.

- Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

24-5 Austausch der Lösungsvariablen V ("IS5")

Es erfolgt kein Druck. Benutzung: 1. 'XEQ "IS5"'. 2. Eingaben den Aufforderungen entsprechend tätigen; Unbekannte mit 'R/S' überspringen. 3. Die Unbekannte wird *automatisch* berechnet und angezeigt. - Diese Version ist nicht geeignet für die Untersuchung von Gleichungen, bei denen man z.B. die Auswirkungen nur kleiner Änderungen in den einzelnen Variablen feststellen will, alte Werte also ständig neu eingeben müßte. Alpha-Texte der Ausgaben dürfen nicht mehr als 6 Zeichen betragen. Bemerkung: Die nachstehende Programm-Skizze ist für eine Gleichung der Form 1 = BC/ADE entworfen. Für andere Gleichungen muß sie umgeschrieben werden. Für Variable, die im Zähler stehen

(hier B und C), benutze man 'XEQ a', für Variable im Nenner (hier A, D und E) 'XEQ b' (Gedächtnisstütze dafür: a/b). - Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

```
LBL "IS5"
CF 00
CLX
"A"
PROMPT
X=0?
XEQ b
STO 01
CLX
"B"
PROMPT
X=0?
XEQ a
STO 02
CLX
"C"
PROMPT
X=0?
XEQ a
STO 03
CLX
"D"
PROMPT
X=0?
XEQ b
STO 04
CLX
"E"
PROMPT
X=0?
XEQ b
STO 05
RCL 02
RCL 03
*
RCL 01
/
RCL 04
/
RCL 05
/
FS?C 00
1/X
CLA
ARCL 00
"⊢="
ARCL X
PROMPT

LBL a
SF 00
LBL b
ASTO 00
1
END
```

R00: Ausgabe-Text
R01: A
R02: B
R03: C
R04: D
R05: E

24-6 Austausch der Lösungsvariablen VI ("IS6")

Es erfolgt kein Druck; es werden keine numerischen Datenregister benutzt. Diese Version ist geeignet für Fälle, in denen jeder Wert mit einem geeigneten Faktor oder einer passenden Routine in einen anderen der beteiligten Werte überführt werden kann (z.B. A in B ohne C oder D zu kennen; vgl. etwa Routine 18-21). Die Programm-Marke "IS6" ist in der nachstehenden Skizze der Taste 'E' zugewiesen. Benutzung: 1. Der umzurechnende Wert ist einzutasten. 2. 'XEQ "IS6"'. 3. Die zum Eingabewert gehörige Taste drücken. 4. Die zum Ausgabewert gehörige Taste drücken. Das Ergebnis wird geliefert. 5. Für den nächsten Fall umzurechnenden Wert eintasten, 'R/S', mit Schritt 3 fortfahren. - Quelle: John Dearing (2791) (PPC CJ, V7N8P22).

'A'	'B'	'C'	'D'	"IS6"

```
LBL "IS6"
SF 27
"A B C D"
PROMPT
LBL A
(A in D umr).

GTO D
LBL B
(B in D umr.)
GTO D
LBL C
(C in D umr.)

LBL D
PROMPT
LBL A
(D in A umr.)
GTO D
LBL B

(D in B umr.)
GTO D
LBL C
(D in C umr.)
LBL D
END
```

24-7 Austausch der Lösungsvariablen VII ("IS7")

Die nachstehend beschriebene Skizze dient dazu, ein *System* von Gleichungen mit mehr als einer Unbekannten zu lösen. Es seien z.B. die Werte von 3 von 5 Variablen, die durch 5 Gleichungen miteinander verknüpft sind, welche je 4 Variable enthalten, gegeben. Dann gibt es 10 mögliche Fälle für die Berechnung von 2 Unbekannten. Setzt man voraus, daß diese 10 Berechnungsmethoden bekannt sind, besteht das Problem darin, 10 verschiedene (lokale) Marken anzusteuern, um die jeweilige Berechnung der 2 Unbekannten durchzuführen, und zwar in Abhängigkeit von den jeweils 3 Variablen, deren Werte bekannt sind und eingegeben werden können. Man ordne den 5 Variablen die fünf

Flags 00-04 und die entsprechenden Register R_{00}-R_{04} zu, dann löst die folgende Programm-Skizze das Problem:

```
01♦LBL "IS7"
02 CF 00
03 CF 01
04 CF 02
05 CF 03
06 CF 04
07 CF 22
08♦LBL A
09 "A"
10 0
11 XEQ 99
12♦LBL B
13 "B"
14 1
15 XEQ 99
16♦LBL C
17 "C"
18 2
19 XEQ 99
20♦LBL D
21 "D"
22 3
23 XEQ 99
24♦LBL E
25 "E"
26 4
27 XEQ 99
28 CLST
29 FS? 00
30 1
31 +
32 FS? 01
33 2
34 +
35 FS? 02
36 4
37 +
38 FS? 03
39 8
40 +
41 FS? 04
42 16
43 +
44 GTO IND X
LBL 07
(D und E ber.)
LBL 11
(C und E ber.)
LBL 19
(C und D ber.)
LBL 13
(B und E ber.)
LBL 21
(B und D ber.)
LBL 25
(B und C ber.)
LBL 14
(A und E ber.)
LBL 22
(A und D ber.)
LBL 26
(A und C ber.)
LBL 28
(A und B ber.)
♦LBL 99
ENTER↑
"⊢ ?"
PROMPT
FS?C 22
SF IND Y
STO IND Y
RTN
R00: A
R01: B
R02: C
R03: D
R04: E
```

Man führt 'XEQ "IS7"' aus, beantwortet die Eingabe-Aufforderung, wenn der angeforderte Wert bekannt ist, und überspringt sie im gegenteiligen Fall mit 'R/S'. Nach dem letzten (fünften) 'R/S' verzweigt das Programm aufgrund der entstandenen Flag-Kombination zur richtigen numerischen Marke. Unter diesen Marken müssen vom Benutzer die dem speziellen Gleichungssystem entspringenden Berechnungen der Unbekannten durchgeführt und Verzweigungen zur gewünschten Ausgabe (Anzeige/Druck mit Text) eingebaut werden.

Unter	'LBL 07'	werden	D	und	E	berechnet,
"	'LBL 11'	"	C	"	E	" ,
"	'LBL 19'	"	C	"	D	" ,
"	'LBL 13'	"	B	"	E	" ,
"	'LBL 21'	"	B	"	D	" ,
"	'LBL 25'	"	B	"	C	" ,
"	'LBL 14'	"	A	"	E	" ,
"	'LBL 22'	"	A	"	D	" ,
"	'LBL 26'	"	A	"	C	" ,
"	'LBL 28'	"	A	"	B	" .

Hinweis: Die Zeilen 29 ('FS? 00') bis 43 ('+') können insgesamt durch die im X-Modul ansässige Funktion 'X < > F' ersetzt werden; in diesem Fall *müssen* auch die Zeilen 02-06 durch '0, X < > F' ersetzt werden.

Beispiel: Die voranstehende Skizze kann anhand des Systems $A \cdot B \cdot C \cdot D = 630$, $A + C \cdot D = E$, $A \cdot B - D - E = 1$, $A \cdot B + E = 8C$, $2(C + D) = E - B$, dessen Lösung $A = 5$, $B = 6$, $C = 7$, $D = 3$, $E = 26$ lautet, getestet werden.

- Quelle: Heinz Dalkowski.

Kapitel 25

SYNTHETISCHES LADEN VON BYTES

25-1 Programm zum synthetischen Laden von Bytes ("LB")

"LB" ist eine synthetische Routine, die synthetische Funktionen und synthetische Textzeilen zusammensetzt. Sie gestattet dem Benutzer, Programmen synthetische Funktionen und Zeilen einzufügen, indem er einfach die den Bytes entsprechenden Dezimalzahlen, welche man der Byte-Tabelle entnehmen kann, eintastet. Gewöhnliche Funktionen und Textzeilen werden in herkömmlicher Weise eingegeben (lassen sich allerdings auch mit "LB" im gleichen Arbeitsgang erzeugen).

"LB" wird in diesem Kapitel als Barcode-Programm abgedruckt*). Lesern, die keinen optischen Lesestift besitzen, wird empfohlen, sich einen solchen zum Einlesen des Programms auszuleihen. Man kann das ganze Programm auch mit den in dem Buch 'Synthetische Programmierung auf dem HP-41C/CV' von W. C. Wickes beschriebenen Techniken herstellen. Ein anschließendes Abspeichern auf Karten sollte man nicht versäumen. (Händler, die das Programm auf Karten bereithalten, leisten damit einen freundlichen Kundendienst.)

Gebrauchsanweisung für "LB":

1. Das Programm wird geladen und durch 'GTO..' mit einem 'END' versehen ('SIZE 000', falls der Speicherplatz knapp ist). Dann in den PRGM-Modus übergehen und die erste globale Marke Ihres zu schreibenden Programms eintasten.
2. Im PRGM-Modus tasten Sie die 3 Zeilen 'LBL "T", XEQ "LB", STOP' ein. Diesen Zeilen werden eine Anzahl 'Strohmannzeilen' angeschlossen, z.B. '+, +, +,.....'. Sie bilden einen Puffer, der die durch "LB" erzeugten synthetischen Zeilen aufnimmt. Die Anzahl A dieser Strohmannzeilen (oder Puffer-Bytes) bestimmt sich nach der Formel

$$A = 14 + 7 \cdot \mathrm{INT}(n/7),$$

worin 'n' die Anzahl der Programm-Bytes, die Sie mit "LB" zu erzeugen beabsichtigen, bedeutet. Zusätzliche Puffer-Bytes richten keinen Schaden an; man kann also auch abkürzend mit 'n + 14' rechnen, wenn man die Anzahl der Puffer-Bytes ermittelt.
3. Schalten Sie in den RUN-Modus, und führen Sie 'XEQ "T"' aus.
4. "LB" fordert Sie jetzt auf, eine Folge von dezimalen Byte-Kodes (0-255) einzugeben. Die Anzahl solcher Eingaben ist nur nach oben - und zwar durch die Anzahl der eingetasteten Strohmannzeilen - begrenzt. Jeder Eintrag wird mit 'R/S' abgeschlossen.

---

*) Anmerkung des Übersetzers: Die ersten 7 Bytes des Barcode-Programms "LB" ('LBL "BC", XEQ 01, X<>Y, XEQ 01, -, CHS, RTN') kann man nach dem Einlesen löschen; sie werden nicht benötigt.

Nach jedem siebenten Eintrag erfolgt die automatische Ablage der Bytes in Ihrem Programm.

5. Ein bereits abgeschickter falscher Eintrag kann - allerdings nur unmittelbar darauf - berichtigt werden: Sie drücken 'SST', um die Aufforderung zur Eingabe des k-ten Bytes zu löschen und erhalten daraufhin noch einmal Gelegenheit, das (k-1)-te Byte einzutasten [eine sehr geschickte Verwendung von Flag 51, die von Roger Hill (4940) vorgeschlagen wurde].

6. Der Eintrag der Bytes kann jederzeit dadurch abgeschlossen werden, daß Sie 'R/S' ohne Eingabe einer Zahl drücken. Die Ablage der Bytes in Ihrem Programm wird dann registergerecht beendet.

8. Die Meldung "NO MORE" zeigt an, daß der Eintrag weiterer Bytes das ständige '.END.' überschreiben würde. Wenn Sie diese Meldung erhalten (sie läßt sich durch anfängliches Einschieben hinreichend vieler Strohmannzeilen vermeiden), fahren Sie mit Schritt 11 fort.

8. Nach Beendigung der Eingabe ('R/S' ohne Eintrag, vgl. Schritt 6) schalten Sie in den PRGM-Modus (Sie befinden sich auf der dritten Zeile unterhalb von 'LBL "T"') und gehen mit mehreren 'SST' auf die erste mit "LB" erzeugte synthetische Zeile. Dann gehen Sie mit 'BST' auf das voranstehende '+' zurück und drücken anschließend so oft die Korrekturtaste, wie es nötig ist, um die '+', das 'STOP' sowie 'XEQ "LB"' und 'LBL "T"' zu löschen. Nun besteht Ihr Programm aus der globalen Marke, den gerade mit "LB" erzeugten Zeilen und einer unbekannten Anzahl von '+' dahinter. Diese '+' können bereits zu diesem Zeitpunkt oder später (vgl. Schritt 9) gelöscht werden.

9. Jetzt können Sie Ihre normalen Programmzeilen eintasten. Sobald eine (oder mehrere) synthetische Programmzeile(n) auftreten, springen Sie mit 'SST' darüber hinweg und nehmen dann das Edieren normaler Zeilen wieder auf. Wenn die letzte Zeile Ihres Programms (damit ist nicht das 'END' gemeint) eingetastet bzw. übersprungen wurde, drücken Sie 'SST'. Erblicken Sie daraufhin '+', erkennen Sie, daß aus dem Arbeitsgang mit "LB" noch Strohmannzeilen zurückgeblieben sind, die Sie nun mit 'DEL 999' (vgl. 2-1) sämtlichst löschen. Zum Schluß noch das gewohnte 'GTO..', um ein 'END' zu setzen, und Ihr Programm ist vollständig ediert, die synthetischen Zeilen eingeschlossen.

10. Das Verfahren mit "LB" kann sinngemäß an jeder Stelle Ihres Programms, an der Sie synthetische Zeilen haben möchten, eingesetzt werden. Möglicherweise möchten Sie mitten während normaler Ediertätigkeit eine synthetische Zeile einschieben. Sie können dann genau an dieser Stelle mit Schritt 2 dieses Verfahrens beginnen.

11. Die in Schritt 7 erwähnte Meldung "NO MORE" warnt davor, mit weiteren synthetischen Zeilen mehr als die anfangs eingetasteten Strohmannzeilen zu überschreiben. Sie müssen Ihre Einträge dann mit 'R/S' abschließen und den Anweisungen von Schritt 8 und Schritt 9 so weit folgen, als sie diesem Fall entsprechen. Die nächsten synthetischen Zeilen werden dann gemäß Schritt 10 erzeugt.

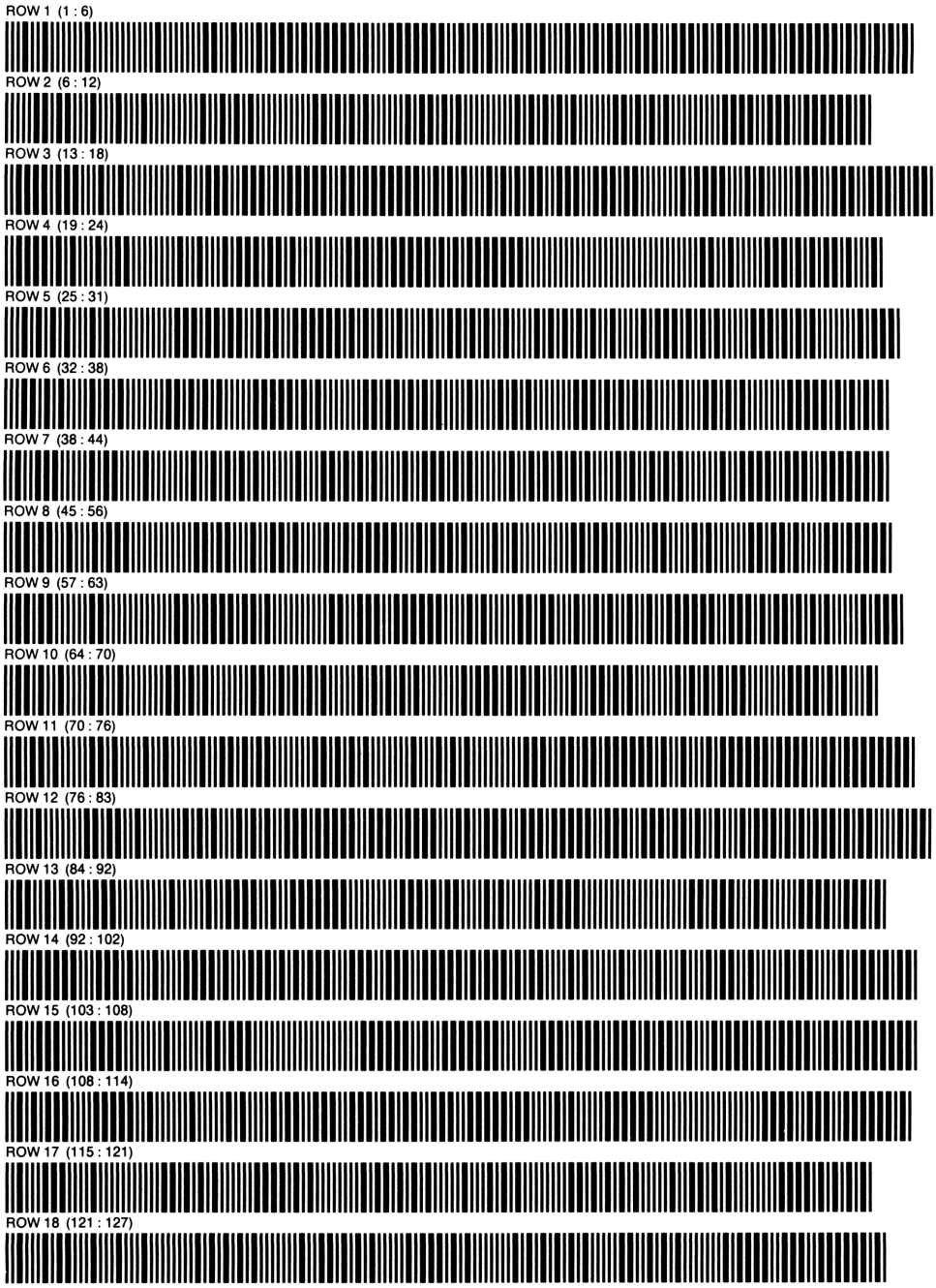
Programmauflistung von "LB":

```
01♦LBL 00       58 GTO 03       115 FS?C 07     172 RDN
02 STO [        59 X<> d        116 SF 05       173♦LBL 10
03 "⊢***"       60 4 E2         117 FS?C 08     174 FC?C 22
04 X<> [        61 ST+ d        118 SF 06       175 GTO 03
05 X<> d        62 RDN          119 FS?C 09     176 RCL [
06 FS?C 04      63 FS?C 11      120 SF 07       177 "DEC. BYTE "
07 SF 01        64 SF 12        121 FS?C 10     178 ARCL Y
08 FS?C 05      65 FS?C 10      122 SF 09       179 "⊢?"
09 SF 02        66 SF 11        123 FS?C 11     180 AVIEW
10 FS?C 06      67 FS?C 09      124 SF 10       181 CLA
11 SF 03        68 SF 10        125 FS?C 12     182 STO [
12 X<> d        69 FS? 07       126 SF 11       183 RDN
13 X<> [        70 SF 09        127 X<> d       184 TONE 7
14 "⊢**"        71 FS? 06       128 DEC         185 STOP
15 X<> \        72 SF 08        129 RTN         186 FS? 51
16♦LBL 01       73 SF 03        130♦LBL 03      187 GTO 11
17 "A"          74 ARCL d       131 RCL c       188♦LBL 03
18 X<> [        75 STO d        132 STO [       189 FC? 22
19 STO \        76 "⊢**"        133 "⊢♦♦♦♦"     190 ,
20 ASHF         77 CLX          134 X<> \       191 XEQ 02
21 RDN          78 X<> ↑        135 "⊢x̄"        192 ISG X
22 "⊢♦♦♦ₓ"      79 X<> ]        136 X<> [       193 GTO 10
23 RCL [        80 X<> \        137 X<> d       194 RCL [
24 INT          81 STO [        138 CF 00       195 RCL Z
25 LASTX        82 RDN          139 CF 01       196 XEQ 04
26 X<> d        83 RTN          140 CF 02       197 STO [
27 CF 09        84♦LBL 03       141 CF 03       198 X<> L
28 CF 10        85 "⊢♦"         142 X<> d       199 16
29 CF 11        86 RDN          143 STO \       200 +
30 FS?C 14      87 RTN          144 RTN         201 X<>Y
31 SF 11        88♦LBL 04       145♦LBL "LB"    202 FS? 22
32 FS?C 15      89 XEQ 03       146 CF 10       203 GTO 08
33 SF 13        90 X<>Y         147 CF 21       204 RTN
34 FS?C 16      91 ENTER↑       148 RCL b       205♦LBL 11
35 SF 14        92 X<> IND L    149 XEQ 00      206 RCL [
36 FS?C 17      93 RDN          150  E          207 CLA
37 SF 15        94 X<>Y         151 -           208 STO [
38 FS?C 18      95 X<> c        152 7           209 ASTO [
39 SF 17        96 RDN          153 /           210 RDN
40 FS?C 19      97 RTN          154 INT         211  E
41 SF 18        98♦LBL 03       155 XEQ 06      212 -
42 FS?C 20      99 16           156  E3         213 ENTER↑
43 SF 19        100 -           157 /           214 SF 22
44 X<> d        101 ABS         158 +           215 7
45  E3          102 RDN         159 FIX 0       216 MOD
46 *            103♦LBL 05      160  E          217 INT
47 DEC          104 XEQ 03      161♦LBL 08      218 X>0?
48 7            105 "⊢x"        162 7 E-3       219 GTO 09
49 *            106 X<> [       163 +           220 RDN
50 +            107 STO \       164 SF 22       221 7 E-3
51  E           108 "⊢AB"       165 DSE Y       222 -
52 -            109 X<> \       166 GTO 10      223 ISG Y
53 RTN          110 X<> c       167 "NO MORE"   224 ""
54♦LBL 02       111 RTN         168 AVIEW       225 GTO 10
55 INT          112♦LBL 06      169 TONE 6      226 END
56 OCT          113 XEQ 03      170 RTN
57 X=0?         114 X<> d       171♦LBL 09
```

(425 Bytes)

Bytes Laden "LB"

Speicherplatzbedarf: 61 Register



Bytes Laden "LB"

ROW 19 (128 : 133)

ROW 20 (134 : 139)

ROW 21 (140 : 145)

ROW 22 (145 : 151)

ROW 23 (152 : 160)

ROW 24 (161 : 167)

ROW 25 (167 : 172)

ROW 26 (173 : 177)

ROW 27 (177 : 182)

ROW 28 (182 : 190)

ROW 29 (191 : 196)

ROW 30 (196 : 203)

ROW 31 (204 : 213)

ROW 32 (214 : 221)

ROW 33 (222 : 226)

Byte-Formate der HP-41 Funktionen:

In den folgenden Beschreibungen wird auf die Byte-Tabelle (S. 180/181) Bezug genommen. Als Byte-Kode wird stets die dem Byte entsprechende Dezimalzahl (0-225) angegeben.

1. Ein-Byte-Funktionen:

Das Einzelbyte stammt - abgesehen vom Spezialfall 240 - aus den Zeilen 1-8 der Byte-Tabelle. Beispiele: 'MEAN' ≙ 124, 'PROMPT' ≙ 142, 'TEXT 0' ≙ 240 [synthetisches 'NOP' (""), vgl. 2-15].

2. Zwei-Byte-Funktionen:

Byte 1 stammt aus den Zeilen 9-B oder den letzten beiden Spalten der Zeile C; Byte 2 bildet die sog. Nachsilbe und wird für direkte Ausführung der oberen Tabellenhälfte, für indirekte Ausführung der unteren Tabellenhälfte entnommen. Beispiele:

```
144, 111 = RCL J (111)        155, 119 = ARCL O
144, 118 = RCL N              159,  96 = TONE 6 (96)
144, 126 = RCL d              159, 109 = TONE H (109)
145, 100 = STO 00 (100)       172,   0 = FS? 00
145, 117 = STO M              173, 246 = FC? IND N
145, 245 = STO IND M          174, 112 = GTO IND T
146, 119 = ST+ O              174, 240 = XEQ IND T
150, 118 = ISG N              206, 118 = X<> N
151, 117 = DSE M              206, 127 = X<> e
152, 117 = VIEW M             206, 245 = X<> IND M
154, 117 = ASTO M             207, 117 = LBL M (lokal!)
```

3. Alpha-Zeichen-Ketten:

Byte 1 stammt aus Zeile F; Folge-Bytes können aus jeder Zeile stammen; jedoch: nur, wenn diese Folge-Bytes aus der oberen Tabellenhälfte kommen, erscheinen sie beim Programm-Ausdruck; in Alpha-Ketten enthaltene Bytes aus der unteren Tabellenhälfte sind beim Programm-Ausdruck 'unsichtbar'. Byte 1 aus Zeile F bestimmt die Anzahl der Folge-Bytes, die als zur Alpha-Kette gehörig behandelt werden. 241 ('TEXT 1') hat das erste Folge-Byte als Text-Byte, 242 ('TEXT 2') hat die 2 folgenden Bytes als Text-Bytes usw. bis zu 255 ('TEXT 15') mit 15 Folge-Bytes. In 'Anhangstextzeilen' stammt das Byte 1 ebenfalls aus Zeile F, Byte 2 lautet 127 (Anhangsbyte). Dieses Anhangsbyte zählt als Folge-Byte mit: "AB" lautet also 242, 65, 66, "⊢AB" jedoch 243, 127, 65, 66. Nachstehend einige Beispiele für Alpha-Ketten, die in den Routinen dieses Buches auftreten:

```
"⊢◆"        = 242, 127, 0              "⊢×iλ◆"     = 245, 127, 1, 105, 11, 0
"⊢◆◆"       = 243, 127, 0, 0           "⊢◆↓****"   = 247, 127, 0, 7, 42, 42, 42, 42
"⊢◆◆◆"      = 244, 127, 0, 0, 0        "⊢◆◆*"      = 244, 127, 0, 0, 42
"⊢◆◆◆◆"     = 245, 127, 0, 0, 0, 0     "βP◆◆◆◆"    = 246, 5, 80, 0, 0, 0, 0
"⊢◆◆◆◆◆"    = 246, 127, 0, 0, 0, 0, 0  ",x̄◆"       = 243, 44, 2, 0
                                        "⊢◆◆A"      = 244, 127, 0, 0, 65
"⊢◆◆×"      = 244, 127, 0, 0, 1        "θΩ"        = 242, 16, 17
"⊢θ"        = 242, 127, 16             "⊢◆Δ"       = 243, 127, 0, 8
```

4. Kurzform-Exponenten:

Kurzform-Exponenten entbehren der überflüssigen führenden '1' und sind daher um 1 Byte kürzer. Wenn *die Zahl negativ* ist, lautet das erste Byte 28 (nicht etwa 84 ≙ 'CHS'!); das nächste Byte ist 27; wenn *der Exponent negativ* ist, folgt Byte 28; dann kommen ein (einziffrige Exponenten) oder zwei (zweiziffrige Exponenten) Bytes aus Zeile 1/Spalte 0-9 der Byte-Tabelle (Bytes 16-25 für die Ziffern 0-9; addiert man 16 auf die gewünschte Ziffer, hat man die richtige Dezimalzahl, beispielsweise: '5' ≙ 21 = 5 + 16). Ein 'E' (Byte 27) entspricht einer '1' (Byte 17), ist aber in der Ausführung schneller als diese (vgl. 18-6). Aufeinanderfolgende Zahleneinträge müssen durch Byte 0 ('NULL') oder Byte 131 ('ENTER') getrennt werden.

Beispiele:

```
 E   = 27                E-2  = 27, 28, 18          E38  = 27, 19, 24
- E   = 28, 27            - E2  = 28, 27, 18         - E45 = 28, 27, 20, 21
 E1  = 27, 17            - E-2 = 28, 27, 28, 18      E50  = 27, 21, 16
- E1  = 28, 27, 17        E3   = 27, 19              E99  = 27, 25, 25
 E2  = 27, 18            E-4  = 27, 28, 20          E-99 = 27, 28, 25, 25
```

5. Globale Marken:

Byte 1 lautet 192; das nächste Byte ist 0; Byte 3 kommt aus Zeile F (dabei muß ein Text-Byte gewählt werden, dessen Wert um 1 höher ist als die Anzahl der Zeichen der Marke); Byte 4 ist wieder 0; die Folge-Bytes, deren Anzahl durch Byte 3 bestimmt ist, kommen aus der oberen Tabellenhälfte. Beispiele: 'LBL "A#)"' ≙ 192, 0, 244, 0, 65, 35, 41; 'LBL "A"' ≙ 192, 0, 242, 0, 65 (nicht zu verwechseln mit der *lokalen* Marke 'LBL A' ≙ 207, 102!).

6. Globale 'GTO's und 'XEQ's:

Byte 1 lautet 29 bzw. 30; Byte 2 stammt aus Zeile F; Folge-Bytes, deren Anzahl durch Byte 2 bestimmt ist, kommen aus der oberen Tabellenhälfte. Beispiel: 'GTO "A#)"' ≙ 29, 243, 65, 35, 41.

7. Lokale 'GTO's und 'XEQ's:

In Kurzform-'GTO's stammt Byte 1 aus Zeile B, Byte 2 ist 0; in Langform-'GTO's und 'XEQ's stammt Byte 1 aus Zeile D bzw. E, Byte 2 ist 0, Byte 3 stammt aus der oberen Tabellenhälfte (nur direkte Ausführung); in 'GTO IND' und in 'XEQ IND' wird Byte 1 durch 174 gebildet, Byte 2 stammt für 'GTO IND' aus der oberen, für 'XEQ IND' aus der unteren Tabellenhälfte. Beispiele: 'GTO 01' ≙ 178; 'GTO 99' ≙ 208, 0, 99; 'XEQ IND 99' ≙ 174, 227; 'GTO M' ≙ 208, 0, 117 (nicht zu verwechseln mit einem *globalen* 'GTO "M"'!); 'GTO IND M' ≙ 174, 117.

8. Zahleneinträge:

Die Bytes stammen aus Zeile 1/Spalte 0-C. Aufeinanderfolgende Bytes bilden eine mehrziffrige Zahl, die in einer einzelnen Programmzeile untergebracht ist. Mit Byte 0 ('NULL') oder Byte 131 ('ENTER') werden die Einträge, die verschiedene Programmzei-

len bilden sollen, getrennt. Für negative Zahlen wird Byte 28 (nicht Byte 84) als Führungsbyte verwendet. Beispiel: '-1.75 E-10' ≙ 28, 17, 26, 23, 21, 27, 28, 17, 16.

9. 'XROM's: Vgl. die Routinen 25-3 und 25-4.

Beispiel für die Anwendung von "LB":

Edieren Sie als Beispiel für die Anwendung von "LB" die Routine "SK" aus 1-18:

1. "LB" laden; 'GTO..'.
2. Übergang in den PRGM-Modus.
3. Tasten Sie die ersten 3 Zeilen von "SK" ein: 'LBL "SK", SIGN, CLX'.
4. Weil die nächste Zeile den ersten synthetischen Befehl enthält, tasten Sie nun 'LBL "T", XEQ "LB", STOP' ein.
5. Um die Anzahl der Puffer-Bytes zu ermitteln, zählen Sie die synthetischen Zeilen aus "SK" aus:

Zeile	Bytes	dez. Byte-Zahl	Zeile	Bytes	dez. Byte-Zahl
04 X<> ⊢	2	206, 122	22 ""	1	240
07 ""	1	240	24 "⊢Φ"	4	243, 127, 15, 255
09 X<> e	2	206, 127	25 X<> N	2	206, 118
12 X<> M	2	206, 117	26 STO ⊢	2	145, 122
13 STO N	2	145, 118	27 X<> M	2	206, 117
20 "⊢◆"	3	242, 127, 0	28 STO e	2	145, 127

Es handelt sich um 25 Bytes; der Puffer aus '+'-Zeilen muß also 14 + 7 · INT(25/7) = 35 Bytes groß sein. Tasten Sie diese 35 '+'-Zeilen ein.

6. Verlassen Sie den PRGM-Modus, und führen Sie 'XEQ "T"' aus. Sie erblicken "DEC. BYTE 1.?".
7. Laden Sie nun die Bytes in einem Arbeitsgang mit 'R/S' nach jedem Eintrag: '206, R/S, 122, R/S, 240, R/S, ... , 117, R/S, 145, R/S, 127, R/S'.
8. Nach dem letzten 'R/S' (hinter '127') noch ein 'R/S' *ohne Eintrag*. Wenn die Programmausführung beendet ist, schalten Sie in den PRGM-Modus. Sie erblicken die Zeile '07 +' Ihrer jetzt wie folgt aussehenden Routine:

```
01◆LBL "SK"     08 +          15 STO \      22 STO e
02 SIGN         09 +          16 "⊢◆"       23 +
03 CLX          10 +          17 ""         24 +
04◆LBL "T"      11 X<> '      18 "⊢╫"       25 +
05 XEQ "LB"     12 ""         19 X<> \
06 STOP         13 X<> e      20 STO '
07 +            14 X<> [      21 X<> [
```

Die Aufteilung der 7 vor und hinter den synthetischen Zeilen stehenden '+'-Zeilen kann (in Abhängigkeit von den in Ihrem Rechner belegten Registern) von dem obigen Ausdruck abweichen. Beachten Sie auch, daß Anzeige und Ausdruck bei den meisten synthetischen Zeilen verschieden sind:

Anzeige:	M	N	O	P	Q	⊢	⊤	⁻
Ausdruck:	[	\	]	↑	_	⊤	""	◆

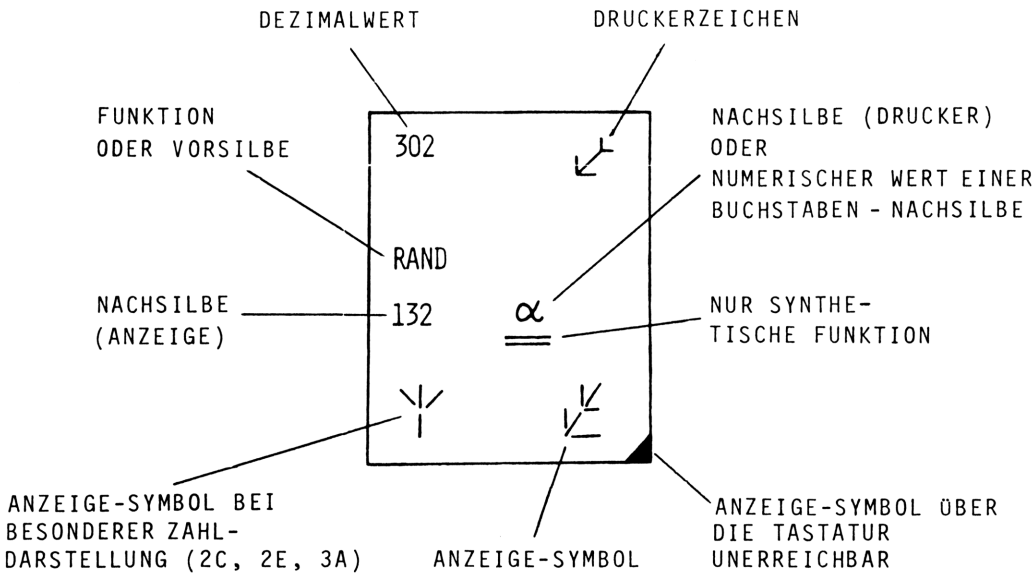
Zeile 18 erscheint in der Anzeige als "-▒▒".

9. Löschen Sie die Zeilen 04-10.

10. Überspringen Sie nun die synthetischen Zeilen mit 'SST', und setzen Sie die normalen Zeilen - beginnend mit 'XEQ 14' - dazwischen. Benutzen Sie (in Zeile 08) ',' statt '0' (vgl. 18-5).

11. Löschen Sie nach dem 'CLA' (Zeile 30) die aus dem Puffer zurückgebliebenen '+' mit 'DEL 999' (vgl. 2-1), und schließen Sie mit 'GTO..' ab. "SK" ist jetzt vollständig ediert und fertig zum Gebrauch.

- Quelle: Keith Jarett (4360) & William Cheeseman (4381) (PPC CJ, V7N10P21).



EIN MUSTERKÄSTCHEN DER BYTE-TABELLE

Eine ausführliche Beschreibung der Byte-Tabelle findet man in 'Synthetische Programmierung auf dem HP-41C/CV' von W.C. Wickes, Abschnitt 2B, 'Die Byte-Tabelle', S. 16-24.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	0 ◆ NULL 00 _	1 × LBL 00 01	2 x̄ LBL 01 02	3 ← LBL 02 03	4 α LBL 03 04	5 β LBL 04 05	6 Γ LBL 05 06	7 ↓ LBL 06 07	8 Δ LBL 07 08	9 σ LBL 08 09	10 ◆ LBL 09 10	11 √ LBL 10 11	12 μ LBL 11 12	13 ∠ LBL 12 13	14 π LBL 13 14	15 ▼ LBL 14 15	0
1	16 ▣ 0 16	17 Ω 1 17	18 δ 2 18	19 ä 3 19	20 ȧ 4 20	21 ä 5 21	22 ä 6 22	23 ö 7 23	24 ö 8 24	25 ü 9 25	26 ü . 26	27 fE EEX 27	28 œ NEG 28	29 ≠ GTO α 29	30 £ XEQ α 30	31 ▓ SPARE 31	1
2	32 RCL 00 32 (space)	33 ! RCL 01 33	34 " RCL 02 34	35 # RCL 03 35	36 $ RCL 04 36	37 % RCL 05 37	38 & RCL 06 38	39 ' RCL 07 39	40 (RCL 08 40	41) RCL 09 41	42 * RCL 10 42	43 + RCL 11 43	44 , RCL 12 44	45 − RCL 13 45	46 . RCL 14 46	47 / RCL 15 47	2
3	48 0 STO 00 48	49 1 STO 01 49	50 2 STO 02 50	51 3 STO 03 51	52 4 STO 04 52	53 5 STO 05 53	54 6 STO 06 54	55 7 STO 07 55	56 8 STO 08 56	57 9 STO 09 57	58 : STO 10 58	59 ; STO 11 59	60 < STO 12 60	61 = STO 13 61	62 > STO 14 62	63 ? STO 15 63	3
4	64 @ + 64	65 A - 65	66 B * 66	67 C / 67	68 D X<Y? 68	69 E X>Y? 69	70 F X≤Y? 70	71 G Σ+ 71	72 H Σ- 72	73 I HMS+ 73	74 J HMS- 74	75 K MOD 75	76 L % 76	77 M %CH 77	78 N P-R 78	79 O R-P 79	4
5	80 P LN 80	81 Q X^2 81	82 R SQRT 82	83 S Y^x 83	84 T CHS 84	85 U e^x 85	86 V LOG 86	87 W 10^x 87	88 X e^x-1 88	89 Y SIN 89	90 Z COS 90	91 [TAN 91	92 \ ASIN 92	93] ACOS 93	94 ↑ ATAN 94	95 _ DEC 95	5
6	96 ` 1/X 96	97 a ABS 97	98 b FACT 98	99 c X≠0? 99	100 d X>0? 00 100	101 e LN1+X 01 101	102 f X<0? A 102	103 g X=0? B 103	104 h INT C 104	105 i FRC D 105	106 j D-R E 106	107 k R-D F 107	108 l HMS G 108	109 m HR H 109	110 n RND I 110	111 o OCT J 111	6
7	112 p CL Σ T	113 q X<>Y Z	114 r PI Y	115 s CLST X	116 t R↑ L	117 u RDN M [	118 v LASTX N \	119 w CLX O]	120 x X=Y? P ↑	121 y X≠Y? Q _	122 z SIGN ⊢ ⊤	123 { X≤0? a	124 | MEAN b	125 → SDEV c	126 Σ AVIEW d	127 ⊢ CLD e	7
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
8	128 ◆ DEG 00	129 × RAD 01	130 ⊼ GRAD 02	131 ← ENTER 03	132 α STOP 04	133 ß RTN 05	134 Γ BEEP 06	135 ↓ CLA 07	136 Δ ASHF 08	137 σ PSE 09	138 ◆ CLRG 10	139 λ AOFF 11	140 μ AON 12	141 ∠ OFF 13	142 ⌐ PROMPT 14	143 ⇕ ADV 15	8
9	144 θ RCL 16	145 Ω STO 17	146 δ STO+ 18	147 á STO- 19	148 à STO* 20	149 ä STO/ 21	150 ä ISG 22	151 Ö DSE 23	152 ö VIEW 24	153 Ü ΣREG 25	154 ü ASTO 26	155 Æ ARCL 27	156 æ FIX 28	157 ≠ SCI 29	158 £ ENG 30	159 ▒ TONE 31	9
A	160 XROM 32	161 ! XROM 33	162 " XROM 34	163 # XROM 35	164 $ XROM 36	165 % XROM 37	166 & XROM 38	167 ' XROM 39	168 (SF 40	169) CF 41	170 * FS?C 42	171 + FC?C 43	172 , FS? 44	173 − FC? 45	174 . GTO IND XEQ IND 46	175 / SPARE 47	A
B	176 0 SPARE 48	177 1 GTO 00 49	178 2 GTO 01 50	179 3 GTO 02 51	180 4 GTO 03 52	181 5 GTO 04 53	182 6 GTO 05 54	183 7 GTO 06 55	184 8 GTO 07 56	185 9 GTO 08 57	186 : GTO 09 58	187 ; GTO 10 59	188 < GTO 11 60	189 = GTO 12 61	190 > GTO 13 62	191 ? GTO 14 63	B
C	192 @ GLOBAL 64	193 A GLOBAL 65	194 B GLOBAL 66	195 C GLOBAL 67	196 D GLOBAL 68	197 E GLOBAL 69	198 F GLOBAL 70	199 G GLOBAL 71	200 H GLOBAL 72	201 I GLOBAL 73	202 J GLOBAL 74	203 K GLOBAL 75	204 L GLOBAL 76	205 M GLOBAL 77	206 N X < > 78	207 O LBL 79	C
D	208 P GTO 80	209 Q GTO 81	210 R GTO 82	211 S GTO 83	212 T GTO 84	213 U GTO 85	214 V GTO 86	215 W GTO 87	216 X GTO 88	217 Y GTO 89	218 Z GTO 90	219 [GTO 91	220 \ GTO 92	221] GTO 93	222 ↑ GTO 94	223 _ GTO 95	D
E	224 ` XEQ 96	225 a XEQ 97	226 b XEQ 98	227 c XEQ 99	228 d XEQ 00 100	229 e XEQ 01 100	230 f XEQ A 102	231 g XEQ B 103	232 h XEQ C 104	233 i XEQ D 105	234 j XEQ E 106	235 k XEQ F 107	236 l XEQ G 108	237 m XEQ H 109	238 n XEQ I 110	239 o XEQ J 111	E
F	240 p TEXT 0 T	241 q TEXT 1 Z	242 r TEXT 2 Y	243 s TEXT 3 X	244 t TEXT 4 L	245 u TEXT 5 M [	246 v TEXT 6 N \	247 w TEXT 7 O]	248 x TEXT 8 P ↑	249 y TEXT 9 Q _	250 z TEXT 10 ⊢ ⊤	251 π TEXT 11 a	252 | TEXT 12 b	253 → TEXT 13 c	254 Σ TEXT 14 d	255 ⊢ TEXT 15 e	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

25-3 Dezimalwerte der XROM-Zahlen für "KA" und "LB" berechnen ("XR")

Diese Routine berechnet die Dezimalwerte, die man benötigt, um ROM-Funktionen aus einem Modul oder einem Peripheriegerät mit "KA" Tasten zuzuweisen oder mit "LB" in ein Programm einzufügen. Die Routine erwartet als Eingabe die jeweilige XROM-Zahl in der Form 'ab.cd' und gibt die beiden Dezimalwerte in der Form 'klm.npq' aus. Diese Dezimalwerte können dann von "KA" ('Synthetische Programmierung auf dem HP-41C/CV', S. 70, 71, 145, 146) oder "LB" verarbeitet werden. Die auf diese Weise hergestellten Tastenzuweisungen oder Programmschritte sind 'echte' XROM-Funktionen: wenn der entsprechende Modul bzw. das Peripheriegerät angeschlossen ist, arbeiten die Zuweisungen und Programmbefehle so, wie man es von normaler Zuweisung und Programmierung bei eingestecktem Modul bzw. Peripheriegerät her gewohnt ist. Ohne Modul oder Peripheriegerät erscheint natürlich die Meldung "NONEXISTENT", wenn eine so zugewiesene Funktion oder ein solchermaßen eingefügter Programmbefehl ausgeführt werden soll. Beispiel: Will man die Dezimalwerte für die Drucker-Funktion 'BLDSPEC' (XROM 29,06) ermitteln, tastet man '29.06, XEQ "XR"' und erhält '167.070'; also sind 167 und 70 die für "KA" oder "LB" zu verwendenden dezimalen Byte-Zahlen, um 'BLDSPEC' auf eine Taste zu legen oder in ein Programm einzufügen.

```
01♦LBL "XR"     08 LASTX      15 FRC        22 1 E3
02 FRC          09 STO T      16 256        23 /
03 LASTX        10 RDN        17 *          24 +
04 INT          11 160        18 X<>Y       25 FIX 3
05 4            12 +          19 1 E2       26 END
06 /            13 X<>Y       20 *
07 INT          14 RCL Z      21 +
```

(44 Bytes)

- Quelle: David Bartholomew (3666) (PPC CJ, V7N7P10).

25-4 Dezimalwerte für XROM-Zahlen berechnen ("XL")

Diese Routine leistet dasselbe wie "XR". Sie verlangt allerdings eine andere Eingabe und gibt auch die Ergebnisse anders aus: Will man beispielsweise die Dezimalwerte für die Drucker-Funktion 'ACSPEC' (XROM 29,04) ermitteln, tastet man '29, ENTER, 4, XEQ "XL"'; anschließend liegen '167' in X und '68' in Y. "XL" ruft die Routine "QR" (vgl. 15-12) als Unterprogramm auf.

```
01♦LBL "XL"     04 +          07 +          10 X<>Y
02 X<>Y         05 64         08 256        11 END
03 640          06 *          09 XEQ "QR"
```

(26 Bytes)

- Quelle: Roger Hill (4940) (PPC ROM).

25-5 "LB"-Eingaben für synthetisches 'BLDSPEC' ("BL")

Diese Routine verarbeitet 7 'BLDSPEC'-Zahlen (Spalten-Druck-Zahlen), um die für "LB" erforderlichen Dezimalwerte, mit denen man Sonderzeichen synthetisch erzeugen kann (vgl. 22-18), zu ermitteln. Nach dem Eintasten der ersten der 7 Spalten-Druck-Zahlen ruft man 'XEQ "BL"' auf; als Antwort erscheint die erste Dezimalzahl, dann werden die restlichen 6 Spalten-Druck-Zahlen *der Reihe nach* eingetastet und jeweils *mit 'R/S'* ermittelt. Eine neue Gruppe von 7 Spalten-Druck-Zahlen muß wieder *mit 'XEQ "BL"' beginnend* verarbeitet werden. Denken Sie daran, daß den 7 Text-Bytes ein 'TEXT 7'-Byte (dezimal 247) vorangehen muß und im Programm dieser Textzeile die Befehle 'RCL M, ACSPEC' (144, 117, 167, 68) folgen müssen (vgl. 22-18). "BL" ruft "QR" (vgl. 15-12) als Unterprogramm auf. Beispiel: Will man die zur Herstellung des Pfeil-Symbols (22-18) notwendigen Dezimalwerte aus den Spalten-Druck-Zahlen berechnen, tastet man: '120, XEQ "BL"' - Ergebnis '17' - '96, R/S' - Ergebnis '227' - ... - '4, R/S' - Ergebnis '4':

'BLDSPEC'-Zahlen:	120	96	80	72	7	6	4
Dezimalwerte für "LB":	17	227	5	9	1	195	4

Beachten Sie, daß es darauf ankommt, die 'BLDSPEC'-Zahlen in der richtigen Reihenfolge einzutasten, weil andernfalls falsche Dezimalwerte ausgegeben werden. Die vollständige Folge der von "LB" zu verarbeitenden Werte für das Pfeil-Symbol lautet somit:

'247, 17, 227, 5, 9, 1, 195, 4, 144, 117, 167, 68'.

```
01♦LBL "BL"     07♦LBL 02       13 RCL [        19 RDN
02 2            08 128          14 *            20 GTO 02
03 STO [        09 RCL [        15 X<> Z        21 END
04 X↑2          10 ST+ [        16 +
05 X↑2          11 /            17 STOP
06 X<>Y         12 XEQ "QR"     18 X<>Y
```

(39 Bytes)

- Quelle: Roger Hill (4940) & Keith Jarett (4360) (PPC ROM). - Vgl. auch 22-28.

25-6 "LB"-Eingaben für synthetisches Flag-Setzen ("FL")

Diese Routine berechnet die Dezimalwerte der Bytes, die als Eingaben für "LB" dienen, um die Gesamtheit aller 56 Flags in einem vorbestimmten Zustand zu versetzen (vgl. 6-8: Massensteuerung von Flags). Die Verwendung einer synthetischen Textzeile zur Gruppenkontrolle von Flags ist speicherplatzsparend, sobald mindestens 7 Flags zugleich gesetzt oder gelöscht werden sollen. Benutzung: 1. 'XEQ "FL"'; man erblickt '0'. 2. Erstes zu setzendes Flag eintasten, dann 'R/S'. Erklingt zur Antwort ein Ton, weist die Anzeige den ersten von 7 Dezimalwerten vor. Man fährt mit 'R/S' fort, um den nächsten Dezimalwert zu bekommen. Steht in der Anzeige jedoch ein negativer Wert (es handelt sich dabei um die zuletzt eingetastete Flag-Nummer; das akustische

Signal bleibt in diesem Falle aus), tastet man das nächste zu setzende Flag ein und drückt 'R/S'. 3. In dieser Weise verarbeitet man alle Flag-Nummern. 4. Sind alle Flag-Nummern eingegeben *und befindet sich Flag 55 nicht darunter*, muß die letzte Eingabe '56' lauten. 5. Die 7 zusammen mit 'TONE 7' erschienenen Dezimalwerte sind dann diejenigen, die von "LB" zu verarbeiten sind. Beachten Sie, daß diesen 7 Text-Bytes ein 'TEXT 7'-Byte (247) vorangehen muß und im Programm der Textzeile die Befehle 'RCL M, STO d' (144, 117, 145, 126) folgen müssen (vgl. 6-8). "FL" ruft "QR" (vgl. 15-12) als Unterprogramm auf. Beispiel: Zu Beginn eines Programms sollen genau die Flags 5, 25, 26, 28, 39, 40, 44 zugleich gesetzt werden. Lösung:

Eingabe	Anzeige	Bedeutung
XEQ "FL"	0.00	(Programm-Start)
5, R/S	-5.00	negativ - nächstes Flag eintasten
25, R/S	4.00	Ton-Signal: 1. Dezimalwert
R/S	0.00	Ton-Signal: 2. Dezimalwert
R/S	0.00	Ton-Signal: 3. Dezimalwert
R/S	-25.00	negativ - nächstes Flag eintasten
26, R/S	-26.00	negativ - nächstes Flag eintasten
28, R/S	-28.00	negativ - nächstes Flag eintasten
39, R/S	104.00	Ton-Signal: 4. Dezimalwert
R/S	1.00	Ton-Signal: 5. Dezimalwert
R/S	-39.00	negativ - nächstes Flag eintasten
40, R/S	-40.00	negativ - nächstes Flag eintasten
44, R/S	-44.00	negativ - nächstes Flag eintasten
56, R/S	136.00	Ton-Signal: 6. Dezimalwert
R/S	0.00	Ton-Signal: 7. Dezimalwert
R/S	-56.00	(Programm-Ende)

Mithin lauten die 7 Dezimalwerte 4, 0, 0, 104, 1, 136, 0 und die vollständige Folge der von "LB" zu verarbeitenden Werte

'247, 4, 0, 0, 104, 1, 136, 0, 144, 117, 145, 126'.

```
01♦LBL "FL"     11 X≠Y?         21  E           31 RDN
02 CLA          12 GTO 13       22 X=Y?         32 GTO 01
03 CLST         13 X<> [        23 XEQ 14       33♦LBL 14
04♦LBL 00       14 7            24 R↑           34 CLX
05 STOP         15 X<>Y         25 CHS          35 X<> \
06 RCL X        16 -            26 GTO 00       36 ISG [
07 8            17 2            27♦LBL 13       37 ""
08 XEQ "QR"     18 X<>Y         28 X<> [        38 TONE 7
09♦LBL 01       19 Y↑X          29 ENTER↑       39 STOP
10 X<> [        20 ST+ \        30 XEQ 14       40 END
```

(65 Bytes)

- Quelle: Keith Jarett (4360) (PPC ROM).

Kapitel 26

KRYPTOGRAPHIE

Hinweis: Die Programme dieses Kapitels verwenden einige Funktionen aus dem X-Modul.

26-1 Ver- und Entschlüsselung von Text ("KV" & "KE")

"KV" dient zur Verschlüsselung von Klartext, "KE" dient zur Entschlüsselung von Chiffretext, der durch "KV" entstanden ist. Die Routine gestattet die Verarbeitung der folgenden 59 Zeichen (Ø steht für das Leerzeichen 'SPACE'):

Ø	!	"	#	$	%	&	'	(	)
*	+	,	-	.	/	Ø	1	2	3
4	5	6	7	8	9	:	;	<	=
>	?	@	A	B	C	D	E	F	G
H	I	J	K	L	M	N	O	P	Q
R	S	T	U	V	W	X	Y	Z	

Diese Zeichen entsprechen den Dezimalwerten 32-90 der Byte-Tabelle. Unter diesen 59 Zeichen befinden sich 9 'nichttastbare' Zeichen:

! " # & ' ( ) ; @

Diese 9 Zeichen liegen nach dem ersten Aufruf von "KV" oder "KE" in der angegebenen Reihenfolge in den Registern R_{01}-R_{09} und können von dort mit 'ARCL' beim Eintasten von Text abgerufen werden. *Beispiel:* Der Text "HP-41CV" soll verschlüsselt werden. Als Antwort auf 'XEQ "KV"' erblickt man die Meldung "VERSCHL:", mit der man zur Eingabe von Klartext aufgefordert wird. Nachdem der Text ("HP-41CV") eingegeben worden ist, drückt man 'R/S'. Sobald der 'BEEP' ertönt, erscheint der Chiffretext "%=P@A.ME". Ist ein Drucker angeschlossen, wird der Chiffretext zusätzlich ausgedruckt. Man beachte, daß aus einem Klartext von 7 Zeichen ein Chiffretext von 8 Zeichen entstanden ist: Der Chiffretext enthält stets ein Zeichen mehr als der Klartext. Aus diesem Grunde dürfen Klartextteile aus höchstens 23 Zeichen bestehen. Ertönt also bei Eingabe von Klartext der Warnton wegen drohenden Überlaufs des Alpha-Registers, muß (mindestens) ein Zeichen gelöscht werden, so daß (höchstens) 23 Zeichen zum Verschlüsseln zurückbleiben. Zur Entschlüsselung führt man 'XEQ "KE"' aus. Jetzt fordert die Meldung "ENTSCHL:" zur Eingabe von Chiffretext auf. Gibt man "%=P@A.ME" ein ("@" erhält man mit 'ARCL 09') und drückt 'R/S', so erscheint der ursprüngliche Klartext "HP-41CV". Will man mehrere Textteile hintereinander ver- oder entschlüsseln, kann man nach der ersten Ver- bzw. Entschlüsselung einfach mit 'R/S' fortfahren. An der Meldung "VERSCHL:" bzw. "ENTSCHL:" erkennt man, daß man weiteren Klar- bzw.

Chiffretext eingeben kann. Von der erheblichen 'Verschleierungsfähigkeit' der Routine mag man sich an Hand von Klartexten, die 'eng benachbart' sind, überzeugen:

Klartext	Chiffretext
TILL	> Z Ø - I
TOLL	B + 4 O 7
SOLL	⍽ . L K >
SOLD	+ 1 6 Y I

Die entstandenen Chiffretexte geben keinen Hinweis mehr auf die Ähnlichkeit der Klartexte.

Sonderfall 'SPACE': Da das Leerzeichen "⍽" ('SPACE') gleichberechtigtes Zeichen ist, kann es gelegentlich als letztes Zeichen eines Chiffretextes auftreten (bei der Verschlüsselung ist ein Pseudo-Zufallsgenerator beteiligt). An letzter Stelle ist "⍽" aber unsichtbar und wäre nur durch Abzählen (der Chiffretext muß ein Zeichen mehr enthalten) feststellbar. Zur bequemen Erkennung weist daher der Chiffretext in solchen Fällen als letztes Zeichen die Unterstreichung "_" auf. "_" ist aber selbst nicht Zeichen des Chiffretextes, sondern zeigt nur an, daß zur Entschlüsselung an letzter Stelle ein Leerzeichen eingetastet werden muß. - (Man beachte in diesem Zusammenhang auch die in Punkt 5-3 angesprochenen Eigenschaften der Anzeige.) -

Beispiele:

Klartext	Chiffretext	Anzeige bzw. Ausdruck
W O D Z	< $ L 1 ⍽	< $ L 1 _
6 R E Z	Q I I U ⍽	Q I I U _
⍽ S L D	W ; U ' ⍽	W ; U ' _

Treten mehrere Leerzeichen am Schluß eines Chiffretextes auf, wird nur das letzte Leerzeichen durch "_" gekennzeichnet. Beispiel:

Klartext	Chiffretext	Anzeige bzw. Ausdruck
% I 7	. 1 ⍽ ⍽	. 1 _

Individualisierung des Programms: Das Programm kann an verschiedenen Stellen so abgeändert werden, daß seine Struktur erhalten bleibt, die Verschlüsselung aber wechselt. Z.B. können die Konstanten '100' (Zeile 91) oder 'PI' (Zeile 101) oder beide geändert werden, wodurch eine unerschöpfliche Quelle von individuellen Schlüsseln zur Verfügung steht. Statt '100' bzw. 'PI' kann jede Zahl zwischen 1 und 1000 gewählt werden. Ändert man beispielsweise '100' in '1,273' und 'PI' in '37,7' ab, so ergibt sich:

Klartext	Chiffretext
TILL	Ø < C = T
TOLL	# B - : #
SOLL	> X O % T
SOLD	6 B J H 6

Zusätzliche Möglichkeiten hat man durch 'schadloses' Einstreuen von Leerzeichen (Verschlüsselung jetzt wieder mit den ursprünglichen Konstanten '100' und 'PI'):

Klartext:	Chiffretext:
EINØDACH	)RXA5V$XL
EINØØDACH	#(YKS-?@6Ø
ØEINØDACH	YU2.?/XV8,
EINØDACHØ	>W=*IDGY*?
EINØDACHØØ	$F,!&NTL;!Z

Oder man verschlüsselt mehrfach (zu diesem Zweck ist es sinnvoll, hinter Zeile 48 ('AOFF') ein 'LBL 07' einzufügen und zur Verschlüsselung bereits verschlüsselten Textes einfach mit 'XEQ 07' ('XEQ, R↓') fortzufahren):

LAURENCE → TUIWBA12N → ,5*X"N!CZZ

SYLVIA → <%2YKA@ → CSØT*/AO → Ø;%4T:"P'

JOSEF → VU666I →)9&;@L# → 2WUBS7A- → OZPZK$L/,

```
01♦LBL "KV"      39 FC? 10        77 STO 00       115 GTO 01
02 CF 10         40 "VER"         78 RCL 10       116 FS? 55
03 GTO 00        41 FS? 10        79♦LBL 04       117 SF 21
04♦LBL "KE"      42 "ENT"         80 X↑2          118 FS? 10
05 SF 10         43 "FSCHL: "     81 *            119 GTO 05
06♦LBL 00        44 AVIEW         82♦LBL 01       120 RCL 00
07 SIZE?         45 CLA           83 ENTER↑       121 32
08 11            46 AON           84 ENTER↑       122 +
09 X>Y?          47 STOP          85 ATOX         123 XTOA
10 PSIZE         48 AOFF          86 32           124 XEQ 14
11 9             49 FC? 10        87 -            125 2
12 ENTER↑        50 GTO 03        88 X<>Y         126 -
13 64            51 XEQ 14        89 CLX          127 AROT
14 XEQ 02        52 CHS           90 RCL 10       128 32
15 59            53 AROT          91 1 E2         129 ATOX
16 XEQ 02        54 ATOX          92 *            130 X=Y?
17 41            55 32            93 R↑           131 95
18 XEQ 02        56 -             94 +            132 XTOA
19 40            57♦LBL 03        95 SF 25        133♦LBL 05
20 XEQ 02        58 ALENG         96 LN           134 AVIEW
21 39            59 STO 10        97 CF 25        135 BEEP
22 XEQ 02        60 FS? 10        98 FRC          136 RTN
23 38            61 GTO 04        99 1 E5         137 GTO 15
24 XEQ 02        62 CLST         100 *            138♦LBL 14
25 35            63 RCL 10       101 PI           139 ALENG
26 XEQ 02        64♦LBL 13       102 *            140 STO Y
27 34            65 ENTER↑       103 INT          141 0
28 XEQ 02        66 ENTER↑       104 FC? 10       142♦LBL 11
29 33            67 ATOX         105 CHS          143 ATOX
30♦LBL 02        68 XTOA         106 +            144 XTOA
31 CLA           69 *            107 59           145 +
32 XTOA          70 ST+ Z        108 MOD          146 DSE Y
33 ASTO IND Y    71 RDN          109 32           147 GTO 11
34 RDN           72 DSE X        110 +            148 R↑
35 DSE X         73 GTO 13       111 XTOA         149 MOD
36 RTN           74 X<>Y         112 R↑           150 END
37♦LBL 15        75 59           113 R↑
38 CF 21         76 MOD          114 DSE 10
```

(264 Bytes)

Bemerkung: Der in den Zeilen 11-36 enthaltene Programmteil kann durch Verwendung synthetischer Textzeilen verkürzt und beschleunigt werden:

```
11 "!"          16 ASTO 03      21 "("          26 ASTO 08
12 ASTO 01      17 "&"          22 ASTO 06      27 "@"
13 """          18 ASTO 04      23 ")"          28 ASTO 09
14 ASTO 02      19 "'"          24 ASTO 07
15 "#"          20 ASTO 05      25 ";"
```

- Quelle: Heinz Dalkowski.

26-2 Arabische ↔ römische Zahlen wandeln ("A↑R" & "R↑A")

"A↑R" wandelt eine im X-Register liegende ganze Zahl in ihre römische Form um. "R↑A" rechnet umgekehrt eine im Alpha-Register stehende römische Zahl in ihre arabische Form um. Bei beiden Routinen führen Eingabefehler zu falschen Ergebnissen. Zeile 69 in "A↑R" ist ein synthetisches 'NOP' (vgl. 2-15). Die Zeilen 62 ('PROMPT') in "A↑R" und 51 ('STOP') in "R↑A" dürfen nicht durch 'RTN's ersetzt werden.

```
01♦LBL "A↑R"    26 XEQ 00    51 "⊢IX"
02 3            27 "⊢C"      52♦LBL 12
03 STO 00       28 DSE 00    53 5
04 RDN          29 GTO 06    54 XEQ 00
05 CLA          30♦LBL 07    55 "⊢V"
06♦LBL 02       31 90        56♦LBL 13
07 DSE 00       32 XEQ 00    57 4
08  E3          33 "⊢XC"     58 XEQ 00
09 XEQ 00       34♦LBL 08    59 "⊢IV"
10 "⊢M"         35 50        60♦LBL 14
11 GTO 02       36 XEQ 00    61 X=0?
12♦LBL 03       37 "⊢L"      62 PROMPT
13 900          38♦LBL 09    63 1
14 XEQ 00       39 40        64 -
15 "⊢CM"        40 XEQ 00    65 "⊢I"
16♦LBL 04       41 "⊢XL"     66 GTO 14
17 500          42♦LBL 10    67♦LBL 00
18 XEQ 00       43 10        68 ISG 00
19 "⊢D"         44 XEQ 00    69 ""
20♦LBL 05       45 "⊢X"      70 X<>Y
21 400          46 DSE 00    71 X<Y?
22 XEQ 00       47 GTO 10    72 GTO IND 00
23 "⊢CD"        48♦LBL 11    73 -
24♦LBL 06       49 9         74 CHS
25  E2          50 XEQ 00    75 END
```

(160 Bytes)

```
01♦LBL "R↑A"    29 XEQ 14
02 CLRG         30 5
03 ALENG        31 X<>Y
04  E3          32 86
05 +            33 X=Y?
06 LASTX        34 GTO 00
07 /            35 1
08 STO 00       36 GTO 02
09♦LBL 01       37♦LBL 00
10 LASTX        38 R↑
11 ATOX         39 R↑
12 77           40♦LBL 02
13 XEQ 14       41 RCL 01
14 500          42 X<Y?
15 X<>Y         43 CHS
16 68           44 ST+ 02
17 XEQ 14       45 X<>Y
18  E2          46 STO 01
19 X<>Y         47 ISG 00
20 67           48 GTO 01
21 XEQ 14       49 ST+ 02
22 50           50 RCL 02
23 X<>Y         51 STOP
24 76           52♦LBL 14
25 XEQ 14       53 X=Y?
26 10           54 GTO 00
27 X<>Y         55 RDN
28 88           56 END
```

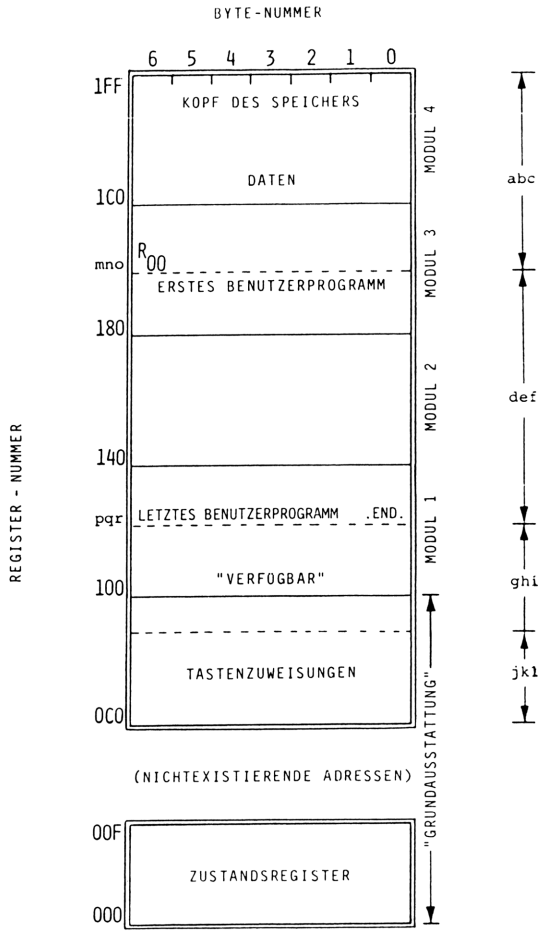
(95 Bytes)

- Quelle: Nach einer Idee von W. Meschede.

Kapitel 27

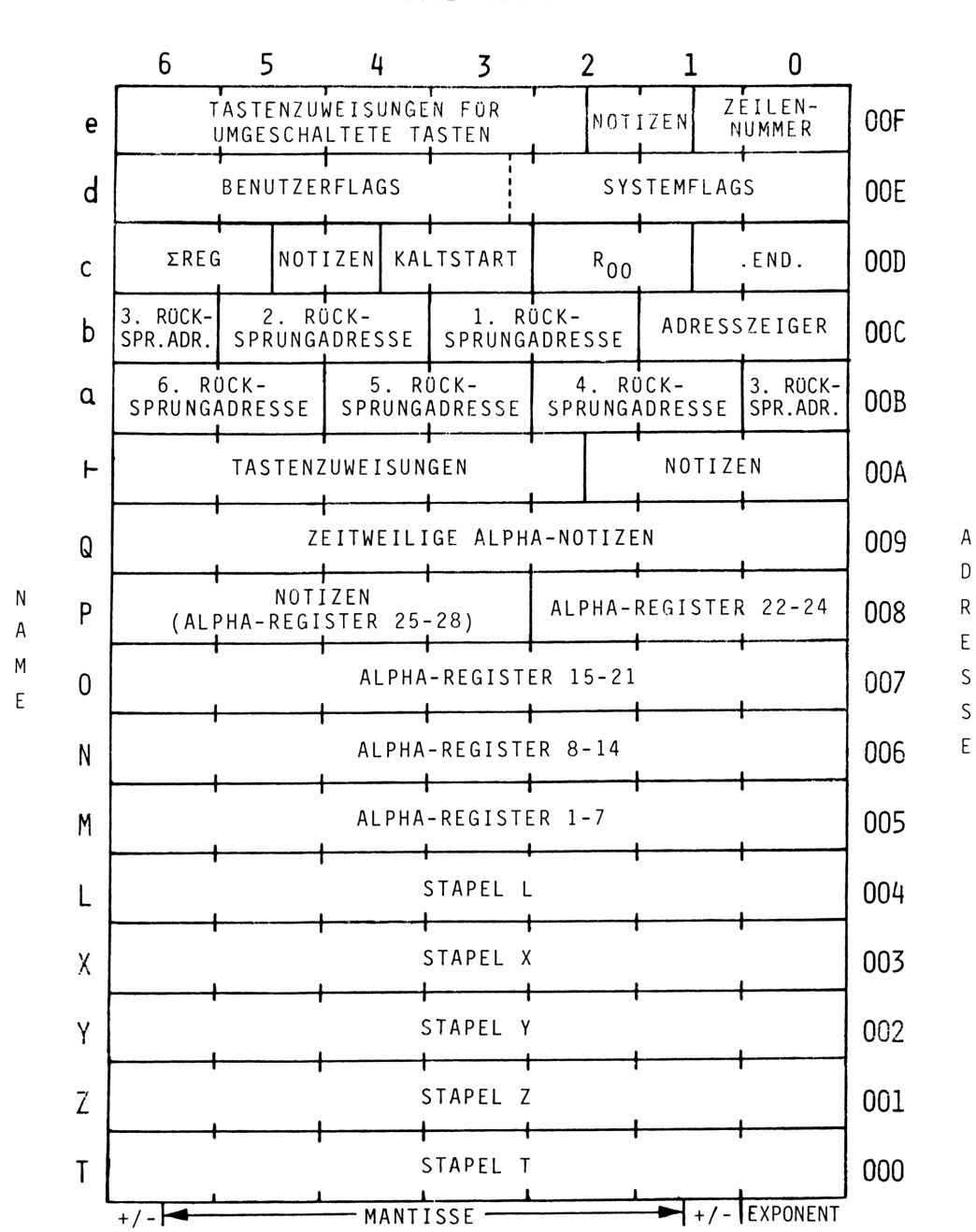
ANHANG

27-1 Die Aufteilung des HP-41C Arbeitsspeichers



- Quelle: W. C. Wickes, Synthetische Programmierung auf dem HP-41C/CV, S. 26.

27-2 Die Zustandsregister



- Quelle: W. C. Wickes, Synthetische Programmierung auf dem HP-41 C/CV, S. 48.

27-3 Umordnungen des Stapels

Die folgende Auflistung gibt alle 256 Möglichkeiten, den Stapel neu zu ordnen, wieder. Bis auf 22 Fälle sind dazu nur drei oder weniger Befehle nötig.

..... X .....

XTTT	R↑, STO Z, STO T, RDN
XTTX	R↑, RCL X, RCL Z
XTTY	R↑, ENTER, X<> Z
XTTZ	R↑, STO Z, X<>Y
XTXT	R↑, STO Z, RCL Y
XTXX	R↑, RCL Y, STO T
XTXY	R↑, RCL Y
XTXZ	R↑, X<>Y, STO Z
XTYT	R↑, STO T, X<>Y
XTYX	X<>Y, R↑, RCL Z
XTYY	R↑, RCL Z, X<> Z
XTYZ	R↑, X<>Y
XTZT	R↑, STO Z, RDN
XTZX	STO Y, X<> T, X<>Y
XTZY	RDN, RDN, X<> Z
XTZZ	X<> Z, STO Y, RDN, RDN

XXTT	R↑, STO Z, X<>Y, ENTER
oder	R↑, STO Z, X<>Y, RCL X
XXTX	STO Y, RDN, STO Y
XXTY	X<>Y, RDN, STO Y
XXTZ	STO Y, R↑, X<> Z
XXXT	STO Y, STO Z
XXXX	ENTER, ENTER, ENTER
oder	ENTER, ENTER, STO T
XXXY	RCL X, RCL X
XXXZ	ENTER, STO Z
XXYT	STO Z, X<>Y, X<> Z
XXYX	ENTER, STO T
XXYY	RCL Y, X<>Y, ENTER
oder	RCL Y, X<>Y, RCL X
XXYZ	RCL X
XXZT	STO Y
XXZX	STO Y, STO T
XXZY	RCL Z, X<>Y, ENTER
oder	RCL Z, X<>Y, RCL X
XXZZ	RCL Z, RDN, STO Y

XYTT	R↑, STO T, RDN
XYTX	STO Z, RDN, X<>Y
XYTY	X<>Y, STO Z, RDN
XYTZ	X<> Z, RDN, X<>Y
XYXT	STO Z
XYXX	STO T, STO Z
XYXY	RCL Y, RCL Y
XYXZ	X<>Y, RCL Y
XYYT	X<>Y, STO Z, X<>Y
XYYX	X<>Y, RCL X, RCL Z
XYYY	RCL Y, STO T, X<>Y
XYYZ	RCL Y, X<>Y
XYZT	(ursprüngliche Ordnung)
XYZX	STO T
XYZY	RCL Y, RDN
XYZZ	RCL Z, RDN

XZTT	X<>Y, RDN, RCL Z, RDN
XZTX	STO Y, RDN
XZTY	X<>Y, RDN
XZTZ	X<>Y, RDN, RCL Y, RDN
XZXT	STO Y, X<> Z, X<>Y
XZXX	STO Y, RDN, STO Z
XZXY	RCL Z, RCL Y
XZXZ	RCL Z, X<>Y, STO Z
XZYT	RDN, X<>Y, R↑
XZYX	X<> Z, RCL Z
XZYY	X<>Y, STO T, RDN
XZYZ	RCL Z, X<>Y
XZZT	X<> Z, STO Y, X<> Z
XZZX	RCL Z, RCL X, RCL Z
XZZY	RCL Z, ENTER, X<> Z
XZZZ	RCL Z, STO Z, X<>Y

..... Y .....

YTTT RDN, RCL Z, STO Z, RDN
YTTX R↑, RCL X, R↑
YTTY R↑, STO Y, RCL Z
YTTZ X<> T, ENTER, X<> Z
YTXT R↑, ENTER, X<> T
YTXX R↑, RCL Y, X<> T
YTXY R↑, RCL Z
YTXZ RDN, X<>Y, RDN
YTYT X<> T, STO Z, RCL Y
YTYX X<>Y, R↑, RCL Y
YTYY R↑, RCL Z, STO Z
YTYZ X<> T, RCL Y
YTZT RDN, RCL Z, X<>Y
YTZX X<> T, X<>Y
YTZY RDN, X<> Z, RCL Z
YTZZ X<> T, RCL Z, X<> Z

YXTT R↑, STO T, X<> Z
YXTX STO Z, RDN
YXTY STO Z, RDN, STO T
YXTZ X<> Z, RDN
YXXT STO Z, X<>Y
YXXX RCL X, RCL X, R↑
YXXY RCL X, RCL Z
YXXZ ENTER, X<> Z
YXYT X<>Y, STO Z
YXYX STO Z, RCL Y
YXYY RCL Y, STO T
YXYZ RCL Y
YXZT X<>Y
YXZX STO T, X<>Y
YXZY X<>Y, STO T
YXZZ RCL Z, X<> Z

YYTT RDN, STO Y, RCL Z, RDN
YYTX RDN, STO Y
YYTY RDN, STO Y, STO T
YYTZ X<> Z, RDN, STO Y
YYXT STO Z, RDN, ENTER
oder STO Z, RDN, RCL X
YYXX STO Z, X<>Y, ENTER
oder STO Z, X<>Y, RCL X
YYXY RCL Y, RCL Z
YYXZ X<>Y, RCL X
YYYT RDN, ENTER, STO Z
YYYX X<>Y, ENTER, ENTER
oder X<>Y, RCL X, RCL X
YYYY RCL Y, ENTER, STO Z
YYYZ RCL Y, STO Y
YYZT RDN, RCL X
YYZX RCL Y, X<>Y, RDN
YYZY RDN, ENTER, STO T
YYZZ RCL Z, X<> Z, STO Y

YZTT RDN, RCL Z, RDN
YZTX RDN
YZTY RDN, STO T
YZTZ RDN, RCL Y, RDN
YZXT X<> Z, X<>Y
YZXX STO T, RDN
YZXY RCL Z, RCL Z
YZXZ X<>Y, RCL Z, X<>Y
YZYT RDN, X<>Y, RCL Y
YZYX RDN, STO Z
YZYY RDN, STO Z, STO T
YZYZ RDN, RCL Y, RCL Y
YZZT RDN, RCL Y, X<>Y
YZZX RCL Z, RCL X, R↑
YZZY RCL Z, STO Y, RCL Z
YZZZ RCL Z, STO Y, X<> Z

..... Z .....

ZTTT R↑, STO Y, STO Z, R↑
ZTTX X<> T, STO Y, X<> Z
ZTTY R↑, STO Y, R↑
ZTTZ X<> T, STO Y, RCL Z
ZTXT R↑, STO Z, R↑
ZTXX STO Y, RDN, RDN
ZTXY RDN, RDN
ZTXZ RDN, RDN, STO T
ZTYT R↑, STO Y, X<> T
ZTYX X<>Y, RDN, RDN
ZTYY RDN, STO T, RDN
ZTYZ RDN, RCL Z, RCL Z
ZTZT RDN, RDN, RCL Y, RCL Y
ZTZX X<> Z, STO Y, X<> T, R↑
ZTZY RDN, RDN, STO Z
ZTZZ X<> T, RCL Z, STO Z

ZXTT R↑, STO Z, X<> T
ZXTX STO Y, RDN, X<>Y
ZXTY X<>Y, RDN, X<>Y
ZXTZ X<> Z, STO Y, RDN
ZXXT STO Y, X<> Z
ZXXX ENTER, STO Z, R↑
ZXXY RCL X, R↑
ZXXZ STO Y, RCL Z
ZXYT X<>Y, X<> Z
ZXYX ENTER, X<> T
ZXYY RCL Y, X<> T
ZXYZ RCL Z
ZXZT X<>Y, RDN, RCL Y
ZXZX ENTER, X<> T, STO Z
ZXZY RCL Z, X<>Y, RCL Y
ZXZZ RCL Z, STO Z

ZYTT RDN, RCL Z, X<> Z
ZYTX RDN, X<>Y
ZYTY RDN, STO T, X<>Y
ZYTZ RDN, X<>Y, STO T
ZYXT X<> Z
ZYXX STO T, X<> Z
ZYXY RCL Y, R↑
ZYXZ X<>Y, RCL Z
ZYYT RDN, ENTER, X<> Z
ZYYX RDN, STO Z, X<>Y
ZYYY RCL Y, STO Y, R↑
ZYYZ X<>Y, STO Y, RCL Z
ZYZT RDN, RCL Y
ZYZX RDN, X<>Y, STO Z
ZYZY RDN, STO Z, RCL Y
ZYZZ X<>Y, RCL Z, STO Z

ZZTT R↑, STO Y, RCL T, ENTER
oder R↑, STO Y, RCL T, RCL X
ZZTX RDN, RDN, ENTER
oder RDN, RDN, RCL X
ZZTY X<> T, RCL Z, ENTER
oder X<> T, RCL Z, RCL X
ZZTZ RDN, RDN, STO Z, ENTER
oder RDN, RDN, STO Z, RCL X
ZZXT X<> Z, STO Y
ZZXX STO Y, RCL Z, RCL X
ZZXY RCL Z, R↑
ZZXZ X<>Y, RCL Z, STO Y
ZZYT RDN, X<>Y, ENTER
oder RDN, X<>Y, RCL X
ZZYX X<> Z, RCL X
ZZYY RCL Y, X<> T, STO Y
ZZYZ RCL Z, STO Y
ZZZT RDN, RCL Y, STO Y
ZZZX RCL Z, ENTER, ENTER
oder RCL Z, RCL X, RCL X
ZZZY RCL Z, ENTER, STO Z
ZZZZ RCL Z, STO Y, STO Z

..... T .....

```
TTTT  R↑, ENTER, ENTER, ENTER
oder  R↑, ENTER, ENTER, STO T
TTTX  R↑, ENTER, ENTER
oder  R↑, RCL X, RCL X
TTTY  R↑, ENTER, STO Z
TTTZ  R↑, STO Y, STO Z
TTXT  R↑, ENTER, STO T
TTXX  STO Y, R↑, ENTER
oder  STO Y, R↑, RCL X
TTXY  R↑, RCL X
TTXZ  X<>Y, R↑, STO Y
TTYT  R↑, STO Y, STO T
TTYX  X<>Y, R↑, ENTER
oder  X<>Y, R↑, RCL X
TTYY  RDN, RCL X, R↑, ENTER
oder  RDN, RCL X, R↑, RCL X
TTYZ  R↑, STO Y
TTZT  RDN, RCL Z, STO Y
TTZX  X<> T, STO Y
TTZY  RDN, X<> Z, ENTER
oder  RDN, X<> Z, RCL X
TTZZ  X<> T, STO Y, RCL Z, RDN

TXTT  R↑, STO Z, STO T
TXTX  R↑, RCL Y, RCL Y
TXTY  X<>Y, X<> T, STO Z
TXTZ  R↑, STO Z
TXXT  STO Y, R↑, STO T
TXXX  STO Z, STO Y, R↑
TXXY  R↑, RCL Y, X<>Y
TXXZ  STO Y, R↑
TXYT  R↑, STO T
TXYX  STO Z, R↑
TXYY  R↑, RCL Z, RDN
TXYZ  R↑
TXZT  X<>Y, RDN, RCL Z
TXZX  STO Y, X<> T
TXZY  X<>Y, X<> T
TXZZ  RDN, RDN, STO T, RDN

TYTT  RDN, RCL Z, STO Z
TYTX  X<> T, STO Z
TYTY  X<> T, RCL Y, RCL Y
TYTZ  X<>Y, R↑, STO Z
TYXT  X<>Y, R↑, STO T
TYXX  STO Z, X<> T
TYXY  X<>Y, STO Z, R↑
TYXZ  X<>Y, R↑
TYYT  RDN, STO Y, RCL Z
TYYX  RDN, STO Y, X<> Z
TYYY  R↑, RCL Z, STO Z, RDN
TYYZ  X<>Y, STO Y, R↑
TYZT  RDN, RCL Z
TYZX  X<> T
TYZY  RDN, ENTER, X<> T
TYZZ  RDN, RCL Y, X<> T

TZTT  RDN, RCL Z, STO Y, RDN
TZTX  RDN, RDN, RCL Y
TZTY  X<> T, RCL Z, RCL Y
TZTZ  RDN, X<> Z, RCL Y, RCL Y
TZXT  X<> T, STO Y, RDN
TZXX  STO Y, RDN, X<> Z
TZXY  RDN, RDN, X<>Y
TZXZ  X<> Z, STO Y, X<> T
TZYT  RDN, X<>Y, RCL Z
TZYX  RDN, X<> Z
TZYY  RDN, STO T, X<> Z
TZYZ  RDN, RCL Y, R↑
TZZT  RDN, RDN, RCL X, RCL Z
TZZX  X<> Z, STO Y, R↑
TZZY  RDN, X<>Y, RCL X, R↑
TZZZ  X<> T, RCL Z, STO Z, RDN
```

- Quelle: John Dearing (2791) (PPC CJ V7N2P22). - Vgl. auch 2-20, 7-11.

27-4 Die Flag-Tabelle des HP-41C

In der nachstehenden Tabelle werden folgende Abkürzungen benutzt:

B Benutzerflag, dessen Zustand mit 'SF' oder 'CF' einstellbar ist.

S Systemflag, dessen Zustand i.a. nur mit 'FS?' bzw. 'FC?' abgefragt werden kann und lediglich in Einzelfällen *mit Systemfunktionen* (z.B. 'SCI 3', 'AON') einstellbar ist, es sei denn man wendet die Techniken synthetischer Programmierung an (vgl. 6-4).

D Dauerflag, dessen Zustand vom Langzeitspeicher des Rechners aufrechterhalten wird.

Z_0 Flag, das beim Einschalten grundsätzlich gelöscht ist: 'Zustand 0'.

Z_1 Flag, das beim Einschalten grundsätzlich gesetzt ist: 'Zustand 1'.

U Flag, das (außer in Fällen raffinierter oder exotischer synthetischen Programmierung) unverwertbar ist.

T Flag, das im Anschluß an eine Totallöschung gesetzt ist.

Flag-Nr.	Flag-Bedeutung	Flag-Typ
00-10	zur freien Verfügung des Benutzers	B D
11	automatischer Programmlauf	B Z_0
12	doppelte Druckbreite	B Z_0
13	Kleinbuchstaben	B Z_0
14	Beschreiben geschützter Magnetkarten	B Z_0
15	Drucker-Modus bei Druckern ohne entsprechenden Schalter; MAN ≙ 00, NORM ≙ 01, TRACE ≙ 10	B Z_0
16		B Z_0
17	unvollständiger Datensatz	B Z_0
18-20	zukünftiger Benutzung vorbehalten	B Z_0
21	Druckersteuerungsflag; beim Einschalten dem Zustand von Flag 55 angepaßt	B
22	numerische Eingabe ist erfolgt	B Z_0
23	ALPHA-Eingabe ist erfolgt	B Z_0
24	Bereichsfehler wird ignoriert	B Z_0
25	Fehler wird ignoriert, Operation wird nicht ausgeführt, und Flag wird gelöscht	B Z_0
26	Ton-Befehle werden ausgeführt	B Z_1 T
27	USER-Modus aktiv	B D
28	Dezimalpunkt wird angezeigt	B D T
29	Zifferngruppierung wird angezeigt	B D T
30	Katalog wird angezeigt (i.a. unverwertbar; vgl. jedoch 'Wickes', S. 120)	S Z_0 U

Flag-Nr.	Flag-Bedeutung	Flag-Typ				
31	verwendbar mit Time-Modul; 'MDY' ≙ 0, 'DMY' ≙ 1	S	D			
32	verwendbar mit HP-IL-Modul; 'AUTOIO' ≙ 0, 'MANIO' ≙ 1	S	D			
33	verwendbar mit HP-IL-Development-Modul	S	D			
34	verwendbar mit Modul für erweiterte Ein-/Ausgabe; 'ADRON' ≙ 0, 'ADROFF' ≙ 1	S	D			
35	verwendbar mit Modul 00041-15042 für automatischen Start und Kassetten-Duplikation	S	D			
36	der Hexadezimalwert dieser 4er-Gruppe bestimmt die Anzahl der angezeigten Nachkommastellen	S	D			
37		S	D			T
38		S	D			
39		S	D			
40	Anzeige-Modus; 'SCI' ≙ 00, 'ENG' ≙ 01, 'FIX' ≙ 10	S	D			T
41		S	D			
42	trigonometrischer Modus; 'DEG' ≙ 00, 'RAD' ≙ 01, 'GRAD' ≙ 10	S	D			
43		S	D			
44	Dauereinschaltung aktiv	S		Z_0		
45	Daten-Eingabe noch nicht abgeschlossen (i.a. unverwertbar, vgl. jedoch 'Wickes', S. 108)	S		Z_0	U	
46	Tasten-Folge noch nicht abgeschlossen	S		Z_0	U	
47	'SHIFT' gedrückt (ausnahmsweise verwertbar zur Abfrage 'FS?' oder 'FC?', wenn die Umschalttaste *während der Ausführung eines 'PSE'-Befehls* gedrückt wird)	S		Z_0	U	
48	ALPHA-Modus aktiv	S		Z_0		
49	Batterie-Kontrolle	S	D			
50	Meldung erfolgt (nur synthetisch verwertbar; vgl. 4-15 und 'Wickes', S. 120/121)	S		Z_0	U	
51	'SST' gedrückt (ausnahmsweise verwertbar zur Abfrage 'FS?' oder 'FC?'; vgl. z.B. Programm "LB", Zeilen 186, 187 und die Erläuterungen dazu auf S. 172)	S		Z_0	U	
52	PRGM-Modus aktiv	S		Z_0	U	
53	Ein-/Ausgabe erfolgt	S		Z_0	U	
54	'PSE' wird ausgeführt	S		Z_0	U	
55	Drucker angeschlossen	S				

27-5 Besetzte globale Marken

Zur Vermeidung von unbeabsichtigten Doppelbenennungen werden im folgenden drei Listen globaler Marken abgedruckt. Die Sortierung folgt den 'ACCHR'-Nummern.

A: Liste der in diesem Buch benutzten globalen Marken:

%T	BD	CURT	F/	IS?	N-A	QEQ	SC	T2	WRD?
-DEC	BE	D-DM	FB	IT	NC	QR	SCE	T3	X?Y
-INT	BF	D-F	FD	IV	NF	QS	SCL	T4	XB
2V	BI	DC	FFFF	JC	NH	QSS	SCR	T5	XD
55	BL	DF	FG	JD	NP	R-S	SCT	T6	XL
7E	BLR	DFD	FIB	KC	NUM?	R1	SCX	T7	XP
7F	BM	DIG	FIN	KE	N↑R	R2	SD	TANH	XPN
7S	BP	DIV	FL	KV	N↑S	R3	SE	TB	XR
=0	BR	DM-D	FN	LB	OE	RAN	SET	TC	XY
>T	BS	DN	FT	LD	OUT	RB	SI	TEM	XYΣ
>b	BV	DR	GCD	LET	P*	RD	SINH	TEMP	X↑3
?S	BX	DS	GD	LG	P+C	RDEC	SIZE?	TH	YN
ACOSH	BYX	DSP	GN	LINE	P-S	RE	SK	TITLE	ZODIAC
AD	BΣ	DSPR	GT	LO	P2	RED	SLC	TM	dB+
AL	C?	DSPS	HD	LR	PC	RF	SLD	TMP	dB-
AM	CD	DSS	HG	M1	PD	RI	SM	TO	db
AN	CE	DT	HN	M2	PE	RK	SORT	TON	Σ+Z
AP	CFA	DUP	I/	M3	PERM	RN	SP	TP	Σ-Z
AR	CFX	DV	IB	M4	PF	RNG	SR	TR	Σ3
ASINH	CHAR	DX	IF	M5	PHONE	ROOT	ST	TX	Σ?
ATANH	CJ	EB	IG	MA	PM	RRM	STACK	UD	ΣB
ATH	CLR	EKST	IN	MAN	PN	RS	STR	UR	ΣC
AV	CLRGX	EN	INB	MANT	PO	RV	STS	V2	ΣD
AVN	CLΣZ	ER	INBL	MARY	POLY	R↑A	STX	VA	ΣF
AX	CM	ET	IR	MC	PR	RΣ	SU	VB	ΣI
A↑R	CODE	EX	IS1	MIO	PRL	S-R	SV	VF	ΣR
B*	COM	EXP	IS2	MO	PRM	S1	SYM	VK	ΣRECIP
B+	COMB	F*	IS3	MOZ	PROE	S2	SZ	VM	ΣS
B?	COSH	F+	IS4	MS	PRSW	S3	SZE	VMAN	
BANR	CR	F-	IS5	MT	PT	S?	T+S	VP	
BC	CU	F-D	IS6	MX	QE	SA	T1	VS	

B: Liste globaler Marken der Länge 2, die von HP im HP-41, in den Peripheriegeräten und in Moduln verwendet werden:

*∠	*C	*a	C-	DH	GY	NE	RP	UW	X1
*0	*F	*b	C/	DI	HR	ON	RS	W0	X2
*1	*H	*c	C=	DL	HT	OR	R↑	W1	X3
*2	*I	*e	CF	DP	I*	PH	Se	W2	X4
*3	*J	10	CG	DR	ID	PI	SF	W3	X5
*4	*L	A0	CH	E0	IN	PL	SH	W4	X6
*5	*M	AZ	CL	ED	JD	PP	SL	W5	X7
*6	*N	BC	CO	EQ	L=	PV	ST	W6	X8
*7	*P	BG	CP	E↑	LN	Pb	SW	W7	X9
*8	*R	BO	CS	FA	LP	R2	TF	W8	X?
*9	*S	BT	CT	FF	LS	R=	TS	W9	Y?
*?	*T	BW	CW	FM	MI	RC	UG	WA	Σ+
*A	*W	C*	CZ	FV	MU	RL	UO	WB	Σ-
*B	*X	C+	DB	GC	NA	RM	UV	X0	

C: Liste globaler Marken der Länge 2, die im PPC ROM enthalten sind:

+K	Ab	BR	CJ	DF	F?	HD	JC	M3	MT	PD	RF	S3	SV	VA	Σ?
-B	BA	BV	CK	DP	FD	HN	L-	M4	NC	PK	RK	S?	SX	VF	ΣC
1K	BC	BX	CM	DR	FI	HP	LB	M5	NH	PM	RN	SD	Sb	VK	
2D	BD	BΣ	CP	DS	FL	HS	LF	MA	NP	PR	RT	SE	T1	VM	
A?	BE	C?	CU	DT	FR	IF	LG	MK	NR	PO	RX	SK	TB	VS	
AD	BI	CA	CV	E?	GE	IG	LR	ML	NS	PS	Rb	SM	TN	XD	
AL	BL	CB	CX	EP	GN	IP	M1	MP	OM	QR	S1	SR	UD	XE	
AM	BM	CD	DC	EX	HA	IR	M2	MS	PA	RD	S2	SU	UR	XL	

- Vgl. auch 2-12.

STICHWORTVERZEICHNIS

% 2-4;7-3;18-22
%CH 2-4;7-3
"%T" 18-22
"-DEC" 15-5
"-INT" 15-5
112-Byte-Zone 2-32
"2V" 22-20
"5S" 6-6
"7E" 4-9
"7F" 4-9
7PRTX 2-26
"7S" 4-9
"=0" 22-24
"›b" 17-7
"›T" 17-7
"?S" 1-16;3-3;3-4
Abfrage-Routine 3-8
Abkuerzungen 2-4
Ableitung 16-20
ABS 7-5
abstandsunabhaengiger Sprung 2-17
ACA 22-7
ACCHR 1-1;22-7;22-10
ACCOL 22-3;22-7
Achsenbestimmungszeichen 22-20
"ACOSH" 16-17
ACSPEC 22-7;22-25
ACX 22-7
"AD" 5-4
Adresse von R00 8-1;8-5
Adressverzeichnis 2-28
ADV 22-3;22-7
Akkumulationsfunktion 22-3
"AL" 12-5
alphabetisieren 12-5
Alpha-Kette 2-28;4-6
Alpha-Ketten speichern 5-1
Alpha-Ketten-Vergleich 5-2
Alpha-Register ausdrucken 2-31
Alpha-Texte 4-19
Alpha-Zeichen aussondern 5-5
Alpha-Zeichen ersetzen 5-5
Alpha-Zeichen-Ketten 25-1
Alpha-Zeichen loeschen 5-4
Alpha-Zeichen-Verlust 5-9
Altgrad 18-1
alter/neuer Wert 3-13;3-15
"AM" 5-1
"AN" 22-15
Anhangstextzeilen 25-1
Anzeige-Format 4-3;4-9
Anzeige-Format FIX/ENG 4-10
Anzeige-Modus 4-3;4-4;4-5;4-8
Anzeige ohne Unterbrechung 4-20
Anzeige von Registerinhalten 3-12
"AP" 16-11
"AR" 16-13
arabische in roemische Zahlen wandeln 26-2
Arbeitsspeicher des HP-41 27-1
ARCL 5-9
arctan 16-7
artanh 16-21
"ASINH" 16-17
ATAN 16-7
"ATANH" 16-17
"ATH" 16-21
Aufloesung von Gleichungen 24-1;24-2;24-3;24-4;24-5;24-6
Aufloesung von Gleichungssystemen 24-7
Aufzeichnung von Tastenzuweisungen 21-13
Ausgabe-Routine 3-7
Ausrichten von Druck-Spalten 22-15
Ausschalten im PRGM-Modus 2-24
Austausch der Loesungsvariablen 24-1;24-2;24-3;24-4;24-5;24-6;24-7
Auswahl ohne Wiederholungen 10-24
"AV" 4-18
AVIEW 2-6;4-2;6-14;22-1
AVIEW-Ersatz 4-18
AVIEW ohne Unterbrechung 4-1
"AVN" 4-18
"AX" 22-16
"A↑R" 26-2
"B≠" 10-28
"B+" 19-9
"B?" 10-26
Bach. J.S. 6-10
Balkengrafik 22-8
"BANR" 23-1
Barcodes 21-17
Basiswechsel 17-1;17-2;17-7
Batteriesatz 22-3
Batterie schonen 21-2;21-17
"BC" 10-23
"BD" 17-1
"BE" 10-14
bedingte Verzweigungen 1-6;6-17
Benutzerflag 27-4
Beschriftung von Tastaturschablonen 1-19
"BF" 15-27

"BI" 10-6
Binaerzahl 17-6
"BL" 25-5
BLDSPEC
1-1;22-3;22-7;22-18;22-21;25-5
Block-Addition 19-8;19-9
Block duplizieren 10-11
Block-Eingabe 10-3;10-4;10-5;10-6
Block identisch laden 10-1
Block inkrementierend laden 10-6
Blockinhalte aendern 10-26
Blockinhalte austauschen
10-13;10-14;10-15
Blockinhalte betrachten
10-7;10-8;10-9;10-26
Blockinhalte bewegen 10-12
Blockinhalte kopieren 10-11
Blockinhalte zyklisch vertauschen
10-16;10-17
Block invertieren 10-10
Block laden 10-2;10-27
Block loeschen
10-19;10-20;10-21;10-22;10-23
Block-Multiplikation 10-28
Block-Rotation 10-16;10-17
Block-Statistik 19-11
"BLR" 10-16
"BM" 10-12
Bogenmass 18-1
"BP" 19-8
"BR" 10-17
Bruch-Gleichungen 15-20
Bruchrechnung 14-7;14-8
Bruchteil 14-5
"BS" 1-27
BST 2-3
"BV" 10-9
"BX" 10-18
Byte-Anzahl ermitteln 22-6
Byte-Formate 25-1
Bytes 1-10;1-27
Bytes laden 25-1
Bytes sparen 1-27
Byte-Tabelle 25-1
Byte-Verbrauch 1-27
"BYX" 15-28
"BΣ" 19-11
"C?" 1-17;8-5
CAT 1 2-29;8-4
"CD" 5-6
"CE" 22-10
Celsius
18-17;18-18;18-19;18-20;18-21
"CFA" 6-3
"CFX" 6-3
"CHAR" 23-1
Chiffretext 26-1
"CJ" 20-2
CLD 2-6;4-2
CLP 8-2
"CLR" 6-15
"CLRGX" 10-19
CLΣ 10-21;19-17
"CLΣZ" 19-16
"CM" 19-14
"CODE" 23-1
"COM" 19-13
"COMB" 19-12
cos 16-6
"COSH" 16-17
cos-Satz 16-10
"CR" 10-22
"CU" 8-1
"CURT" 15-21
"D-DM" 18-14
"D-F" 14-8
D-R 2-4
Darstellungswechsel 17-3
Datei 11-2
Dateieingabe als Programmteil 21-6
Datenbasissystem 11-2
Daten-Karten 21-6
Datenpruefung 21-17
Datensatz 11-2
Datenregister-Anzahl
1-14.15.16.17;3-1.2.3.4.5
Datenregister laden 9-1
Datum 20-1;20-2
Dauerflag 27-4
Dauer-Tonprogramm 6-10
"db" 17-6
"dB+" 15-32
"dB-" 15-32
"DC" 5-6
DEL 2-1;8-2
Determinante einer Matrix 11-4
Dezibel-Werte 15-32
Dezimaldarstellung 17-3
Dezimalgrad 18-14;18-16
Dezimalzahl in Bruch wandeln
14-1;14-2;14-8
Dezimalkomma 18-5
Dezimalminuten 18-14;18-16
Dezimalzahl 17-4;17-5;17-6
Dezimalzahl in Zeichen wandeln 5-6
"DF" 14-1
"DFD" 14-2
Differentiation 16-20
"DIG" 23-2
"DIV" 22-14
Division durch 10 7-6;15-10
Division durch 100 18-8
"DM-D" 18-14
"DN" 22-11
Dokumentation von Programmen
1-22;22-11
Doppelbenennung von Marken 2-12;27-5
Doppel-Plot 22-20

Doppel-Speicherung 9-8
"DR" 11-2
Drei-Byte-GTOs 2-17;2-32
Dreieck 16-15
Dreieckszahlen 16-15
dritte Potenz 15-34
Druckbreite 22-3
drucken 22-23
Druckerflag 6-6
Druckerfunktionen 22-9
Drucker-Modus 22-7
Druckmatrix 22-7
Druck-Modus 22-7
Druck-Puffer 22-3;22-7
Druck-Spalten 22-15
"DS" 4-8
"DSP" 4-20
"DSPR" 4-3
"DSPS" 4-3
"DSS" 22-28
"DT" 4-17
"DUP" 10-11
"DV" 22-14
"DX" 17-4
"EB" 10-20
echter Bruch 14-7
Edieren 2-2;2-3
EEX 1-2;1-20;2-1
eGOBEEP 22-9
Eigen-Ladung 10-2;10-27
Ein-Byte-Funktionen 25-1
Eine kleine Nachtmusik 6-11
Einfuegen von Programmzeilen 2-27
Eingabe-Aufforderung 2-34;22-1
Eingabe ohne Programm-Stop 3-18
Eingabe-Routine 3-6
Einkommensteuer 20-5
Eins 18-6
"EKST" 20-5
elektrostatische Aufladung 21-1
"EN" 8-4
END 2-1;2-9;2-29;2-30
.END. 1-10;2-9;8-4;21-4;21-10
Englische Einheiten 18-4
Englische Brueche 14-8
Englische Meile 18-3
ENTER 1-12
Entleerung der Batterien 2-19
Entschluesselung von Chiffretext 26-1
"ER" 14-9
Ergebnis eingrenzen 16-8
Ersatzanzeige 2-6
Ersetzen der Graugans 4-14;4-15
erste Ableitung 16-20
"ET" 16-14
Eulerscher Winkel 16-14
Euler-Transformation 16-14
"EX" 15-19
"EXP" 15-18
Exponenten 1-20;15-16;15-18;15-19;15-35
Extremwerte eines Blockes 10-18
E↑X-1 16-22
"F*" 14-7
"F+" 14-7
"F-" 14-7
"F-D" 14-8
"F/" 14-7
Fahnendruck 23-1;23-2
Fahrenheit 18-17;18-18;18-19;18-20;18-21
Faktorzerlegung 15-23
Fakultaet 15-23;15-27
fast-stetige Laufschrift 4-11
faules T 1-3
"FB" 15-7
"FD" 16-20
fehlende globale Marke 2-29
fehlender Warnton 5-9
Fehler-Ignorierflag 3-1;4-14
Fehler im Kartenleser 21-9
Fehlverhalten des Rechners 2-24
FELIX 9-4
"FFFF" 15-23
"FG" 15-33
"FIB" 15-7
Fibonacci-Zahlen 15-7
"FIN" 18-2
FIX/ENG 4-10
"FL" 25-6
Flag 11 21-6
Flag 12 6-13;22-7
Flag 13 6-13;22-7
Flag 21 4-18;6-6;6-14;22-1;22-3
Flag 22 3-6;3-13;3-14;3-18
Flag 23 3-18
Flag 25 3-1;4-14
Flag 51 1-16;25-1
Flag 55 6-4;6-6;6-14;22-1;22-3
Flag loeschen 6-1;6-2;6-3;6-15
Flag setzen 6-1;6-2;6-3;6-15
Flags betrachten 6-5
Flagschaltungen 6-1;6-4
Flag-Tabelle des HP-41 27-4
Flag-Typ 27-4
Fluoreszens 21-16
"FN" 22-17
Fotokopien des Druckstreifens 22-5
freie Programmregister 2-30;8-3
Friedenssymbol 22-28
"FT" 6-1
Funktionen aus nichtang. Peripherieger. 21-14;22-9
Funktionswerte 22-17
Fuss 3-16;18-2
Gammafunktion 15-27;15-33
ganze Dezimalzahl 15-5

ganzzahliger Anteil 15-11;15-12
ganzzahlige Seitenlaenge 16-15
GAU 1-23
Gauss'sche Verteilung 13-4
"GCD" 15-13
"GD" 15-13
gebrochene Gammafunktion 15-33
Geheimschrift 26-1
Gelbfilter 22-5
gemischter Bruch 14-7
GGT 14-3;15-13
globale Alpha-Marken 1-10
globale Marken 2-11;2-17;2-29;25-1;27-5
Glockenkurve 13-4
"GN" 13-4
Grad 18-14;18-16
Graugans 1-3;2-6;4-2;4-14;4-15;6-4
Gregorianisches Datum 20-1;20-2
groesster gemeinsamer Teiler 14-3;15-13
groesste von 3 Zahlen 12-4
grosse Fakultaeten 15-27
grosse Potenzen 15-28
Grundausstattung 1-10
Grundgesamtheit 19-15
Gruppenkontrolle von Flags 6-8;25-6
"GT" 15-14
Haengenbleiben 21-9
Haeufigkeitsfaktor 19-1
"HD" 8-6
Hexadezimalzahl 5-7;17-4;17-5
Hexadezimalzahl in NNZ wandeln 5-7
"HG" 22-8
Histogramm 22-8
Histogrammsaeulen 22-22
HMS 16-5
"HN" 5-7
hochgestelltes T 1-3
Horner-Schema 15-29;17-2
HP-67/97 Programme 1-8
HP-IL-Drucker 22-9
HP-IL-Modul 22-9
HR 16-5
hyperbolische Funktionen 16-17;16-21;16-22
Hypothenuse 16-15
"I/" 15-11
"IB" 10-5
identische Namen 2-12;27-5
"IF" 6-4
"IG" 16-19
"IN" 3-6
"INB" 10-4
"INBL" 10-3
Inch 18-2
Indikatoren der Anzeige 4-17
indirekte Adressierung 1-5;2-11;2-28;13-5
indirektes GTO 2-11
indirektes Speichern 9-11
indirektes XEQ 2-11;3-17
inneres Produkt 10-28;19-11
Integration 16-19
Interferenz-Erscheinungen 21-16
internes STOP 2-22
inverse Funktion 2-7
Inverse einer Matrix 11-4
"IR" 11-2
"IS1" 24-1
"IS2" 24-2
"IS3" 24-3
"IS4" 24-4
"IS5" 24-5
"IS6" 24-6
"IS7" 24-7
"IT" 16-15
Iterationen 2-16
"IV" 15-6
ja/nein-Abfrage 3-8
ja/nein-Entscheidungen 3-9
"JC" 20-2
JD 20-1
JOSEF 11-2;26-1
Julianisches Datum 20-2
Julianischer Tag 20-1;20-2
JULJA 9-4
Kalenderdatum 20-2
Kalender-Druck 20-4
Kalender-Rechnung 20-1;20-2
Kartenleser 21-1
Kartenmischer 10-24
Karten-Stop 21-3
Karten teilweise einlesen 21-11
Katalog 2-29
Katalogdurchsicht 1-4
Kataloge drucken 22-3
KATJA 10-24
"KC" 1-26
"KE" 26-1
Kehrwerte 19-2
Kehrwert von Summen 19-2
Kelvin 18-21
Kenngroessen einer Division 15-11
Kettenrechnung 15-9
Kilometer 18-3
Klartext 26-1
Kleinbuchstaben 22-7
kleine Fehler 15-8
kleinste von 3 Zahlen 12-4
KNUT 10-24
Kombinationen 19-12;19-13;19-14
komplexe Zahlen multiplizieren 15-24
Kontrollzahl in L 7-12
Koordinatenwechsel 16-14
Kopien von Barcodes 21-17
Kryptographie 26-1
Kubikwurzel 15-21

Kugelkoordinaten 16-14
Kundendienst 19-3
Kuerzen von Bruechen 14-3;14-7
Kurzform-Exponenten 25-1
Kurzform-LBL 2-32
Kurzform-Marken 2-17;2-32
"KV" 26-1
"LA" 2-10
Laden von Bytes 25-1
Lage des Vorhangs 8-5
Laengenbegrenzung fuer Alpha-Ketten 4-6
lange Pause 2-26
laenger laufende Programme 2-19;2-25
Langform-GTO 2-32
LASTX-Fehler 19-3
Laufschrift 4-11;4-12;4-13;4-14;4-15
LAURENCE 26-1
"LB" 25-1
"LD" 9-1
Leerstellen 5-3
Leseblockade 21-16
Lesefehler 21-5
Lesestift 21-16;21-17
"LET" 23-2
letztes Programm 2-9;21-10
"LG" 22-19
"LINE" 22-13
linksseitige Laufschrift 4-13
"LO" 1-13
Loeschen 2-1
Loesung linearer Gleichungssysteme 11-4
logisches Oder 6-17
logisches Und 6-17
lokale Alpha-Marken 1-10;2-11;2-17
lokale Marken 1-22;1-24;2-17;25-1
Loeschen von Datenregistern 9-6
Loeschen von Programmen 8-2
Lotto spielen 10-24
LOW BAT 21-3
"LR" 2-33
LUTZ 10-24
"M1" 11-1
"M2" 11-1
"M3" 11-1
"M4" 11-1
"M5" 11-1
"MA" 5-1
magnetisierte Gegenstaende 21-5
"MAN" 15-16
"MANT" 15-18
Mantisse 15-16;15-18;15-19
Mantisse betrachten 15-17;15-19
MANual-Modus 22-6
"MARY" 6-12
Mary has a little lamb 6-12
Masseinheit 3-16
Massenloeschungen von Flags 6-3
Massensteuerung von Flags 6-8;25-6
Matrix betrachten 11-3
Matrix-Hilfsprogramm 10-18
Matrix laden 11-3
Matrixprodukt 19-11
Matrixprogramm 19-11
Matrix-Routinen 11-1
Matrizenrechnung 11-4
"MC" 15-24
mehrere Ausgaben in einer Zeile 22-24
mehrfach benutzter Programmname 2-12;27-5
Meile 18-3
Meldung 2-25;3-10;3-11;4-7;4-19
MEMORY LOST 1-9;1-23;6-7;8-1
Messwertpaare 19-1
Meter 3-16
metrische Einheiten 18-4
Minuten 16-5
"MIO" 11-3
"MO" 6-11
MOD 16-4;16-8
Moivresche Reihenentwicklung 15-27
"MOZ" 6-11
Mozart synthetisch 6-11
MRG 21-4;21-10
MRG ERR 21-4
"MS" 7-10
"MT" 15-19
Multiplikation von Bloecken 10-28
Multiplikation mit 10 15-10
Multiplikation mit kleiner Zahl 18-7
Multiplikation von Matrizen 19-11
Multipliktion mit SQRT(2) 18-9
Musterkaestchen der Byte-Tabelle 25-1
"MX" 14-7
"N-A" 5-8
Nachkommastellen 4-3;4-8;4-9
Nachtmusik eine kleine 6-11
Naeherungsbruch 14-1;14-2
Nautisches Jahrbuch 18-14
"NC" 5-5
neuer/alter Wert 3-13;3-15
Neugrad 18-1
Newton'sche Methode 16-18
"NF" 14-4
"NH" 5-7
nichtangeschlossene Peripheriegeraete 21-14;22-9
nichtausschliessendes Oder 6-17
nicht drucken 22-23
nicht normalisierte Zahl 5-7
nichtprogrammierbare Funktionen 21-15
NNZ 5-7;6-8
NNZ in Hexadezimalzahl wandeln 5-7
no operation 2-15

NOP 2-15;2-16
NORMal-Modus 1-24;22-3
"NP" 15-15
n-te Wurzel 15-21
NULL 1-12
Null 18-5
Nullstellen-Bestimmung 16-18
"NUM?" 3-18
numerische Marken 2-17
"N↑R" 16-12
"N↑S" 16-12
"OE" 10-25
OFF 2-19
offene Ruecksprueng 1-2
Oktaldarstellung 17-3
optische Effekte 4-14
"OUT" 3-7
"P*" 15-31
"P+C" 19-13
P-R 2-4
"P-S" 10-15
"P2" 22-15
PAPER ADVANCE 22-3
Papierreflexion 21-16
Papier sparen 22-6
Papiervorschub 22-3
Parabel 16-9
Parabelsegment 16-11
partielles Einlesen von Karten 21-11
partielle Pivotierung 11-4
Pause 2-26
Pausen variabler Laenge 2-6
pausieren 22-23
"PC" 20-4
"PD" 15-14
"PE" 15-30
Peripheriegeraete 21-14;22-9
"PERM" 19-12
Permutationen 19-12;19-13;19-14
"PF" 15-14
"PHONE" 9-4
PI 18-10;18-11
Pivot-Elemente 10-18
Pivotierung 11-4
Planetensymbole 22-27
Plot-Programm 22-20
Plotten 22-3
Plotten mit zwei Variablen 22-20
Plot-Zeichen 22-3
"PM" 19-14
"PN" 16-9
"PO" 22-2
Polarkoordinaten 16-14
"POLY" 15-29
Polynom-Berechnung 15-30
Polynome 15-29
Polynom-Multiplikation 15-31
Populationsparameter 19-15
Positionierung 1-2
potenzieren 18-12;18-13
PPC-Siegel 22-19
"PR" 9-7
PRA 22-3;22-7
PRBUF 22-7
PRFLAGS 22-3;22-7
PRGM-Indikator 2-6;4-20
PRGM-Modus 2-24
Primfaktorzerlegung 15-14;15-15
Primteiler 15-14
Primzahlpotenzen 15-23
Primzahltabelle 15-14
Primzahltest 15-14
PRINT 22-3
PRKEYS 1-21
"PRL" 22-4
"PRM" 19-13
Programmaenderungen 2-2
Programm-Dokumentation 1-22;22-11
Programme ohne Marken 2-29
Programm-Segmentierung 2-13
Programmspeicher 1-10
Programmueberschriften 3-1
Programmunterbrechung 2-6;22-23
Programmzeiger 2-30
PROMPT 3-6
Prozentanteil 18-22
PRP 22-7
PRPLOT 22-3;22-17
"PRQE" 15-26
PRSTK 22-4
"PRSW" 22-21
Pruefung der Anzeige-Segmente 4-17
PRX 22-3;22-7
PSE 2-6
"PT" 16-15
Puffer-Ueberlauf 22-7
pythagoraeische Tripel 16-15
"QE" 15-26
"QEQ" 15-25
"QR" 15-12
"QS" 12-1
"QSS" 12-6
quadratische Gleichung 15-25;15-26
Quadratsummen 15-22
Quadratwurzel 2-4;15-22
quergestellte Druckzeichen 22-21
quergestellte Zahlen 22-22
Querschrift 22-21
Quersumme 15-4
R-D 2-4
R-P 2-4
"R-S" 16-14
R/S 1-2
"R1" 15-20
"R2" 15-20
"R3" 15-20
"RAN" 13-1
Rangfolge von Funktionen 1-24

Rankine 18-21
"RB" 10-10
RCL T 7-4
RCL X 7-4
RCL Y 7-3;7-4
RCL Z 7-4
RCL Σ 19-5;19-6
"RD" 4-4
"RDEC" 17-3
RDN 2-20
"RE" 14-7
rechtsseitige Laufschrift 4-14
rechtwinklige Dreiecke 16-15
rechtwinklige Koordinaten 16-14
"RED" 14-3
reduzierte Matrix 11-4
Registerarithmetik 9-5;9-9
Registeraustausch 9-9
regelmaessiges Vieleck 16-12;16-13
Registerinhalte aendern 9-2;9-3
Registerinhalte austauschen 10-25
Registerinhalte betrachten 9-1
Registerinhalte kopieren 9-10
Registerinhalte verdichten 9-7
Registermischer 10-24
REGMOVE 10-12
REGPLOT 22-20
REGSWAP 10-14
regulaeres Polygon 16-12;16-13
rein gebrochene Dezimalzahl 15-5
Rest 15-11;15-12
"RF" 6-7
"RI" 9-11
"RK" 1-18
"RN" 13-3
RND 14-9
"RNG" 13-5
"ROCT" 17-3
roemische in arabische Zahlen wandeln 26-2
Rollen der Buchstaben 2-31
ROM-Routinen ohne Ausgabeunterbrechung 6-6
ROM-Routinen als Unterprogramme 6-6
"RRM" 11-4
"RS" 3-2
RSUB 21-4
RTN 2-9;2-22
Rueckkehradressen 2-33
Ruecksetzen der Flags 6-7
Ruecksprueng 1-2
Ruecksprungstapel 2-33
Rundlauf von Text 4-14;4-15
Rundung 14-4;14-5;14-6;14-9
"RV" 9-1
R↑ 2-20
"R↑A" 26-2
"RΣ" 19-5
"S-R" 16-14
"S1" 12-2
"S2" 12-3
"S3" 12-3
"S?" 1-17
"SA" 7-7
Saeulenstatistik 22-8
"SC" 22-25
"SCE" 4-13
schiefwinklige Dreiecke 16-15
Schleife 4-19
Schleifenendwert 2-15
Schleifenkontrolle 2-14
Schleifenkontrolle mit L 7-12
Schleifenzaehler 2-16
schnelle Eins 18-6
schnelle Null 18-5
Schreibfehler 21-15
Schrift-Fahnen 23-1;23-2
Schrittfolge mehrfach ausfuehren 2-14
Schutzfolie 21-17
schwache Batterien 21-3
SCI-Modus 1-7
"SCL" 22-25
"SCR" 4-14
"SCT" 22-25
"SCX" 22-25
"SD" 4-4
"SE" 10-24
Segmente 4-17
Sekunden 16-5
"SET" 6-15
"SI" 9-11
SIGN 1-25;2-4;7-5
sin 16-6
Singularitaet 16-2
"SINH" 16-17;16-22
sinh 16-22
SIZE 1-14;1-15;1-16;1-17;3-1;8-1;9-6
"SIZE?" 3-1
SIZE pruefen 3-2;3-3;3-5
SIZE synthetisch pruefen 3-4
"SK" 1-18
SKPCHR 22-7
SKPCOL 22-3;22-7
"SLC" 10-27
"SLD" 10-2
"SM" 7-10
Sonderzeichen 22-18;22-21;22-25;22-28
SONJA 10-24
"SORT" 12-4
Sortierprogramm 12-1;12-3
Sortierung des Stapels 12-2;12-6
"SP" 16-11
Spalten-Druck-Zahlen 22-18;25-5
Speicherzellen 22-7
sphaerische Koordinaten 16-14
Splittingtarif 20-5

Sprungabstand 2-17
Spruenge 2-1
Sprunglaengenbegrenzung 2-17;2-32
Sprunglaengeninformation 2-24
Sprunglaengenkode 2-24
"SR" 2-33
"ST" 7-8
ST* 9-5;9-9
ST+ 9-5;9-9
ST- 9-5;9-9
ST/ 9-5;9-9
"STACK" 7-11
Staffelform 11-4
Standardabweichung 19-15
Standardzeichen 1-1;22-10
staendiges .END. 1-10;2-9;8-4;21-4;21-10
Stapel-Analyse 7-7
Stapel-Austausch 7-9
Stapel betrachten 7-8
Stapelfunktion 7-11
Stapelinhalt speichern 7-9;7-10
Stapel-Lift-Probleme 2-18
Stapel-Sortierung 12-2;12-6
Stapel-Umordnungen 2-20;7-11;27-3
Startvorbereitungen 3-10
Statistik mit 3 Variablen 19-16
Statistik-Block 1-17;19-4;19-6;19-10;19-11;19-17
Statistikregister 8-6
statistische Summation 19-1
Status-Karte 21-7
Stern-Tripel 22-3
Stirlingsche Naeherung 15-27
STO 9-9
STOP 1-2;2-22
Stoppuhr 20-3
"STR" 7-9
Stromausfall 2-19
Stromverbrauch 21-1
"ST5" 7-9
"STX" 7-9
ST*X 7-1
ST+X 7-1
ST-X 7-1
ST/X 7-1;7-5
ST*Y 7-12
ST+Y 7-12
"SU" 5-5
Sucher fuer den Statistik-Block 19-4;19-6
Summation von x und y 19-7
Summe ganzer Zahlen 15-1
Summe von Kubikzahlen 15-3
Summe von Quadratzahlen 15-2;19-8
"SV" 16-18
SYLVIA 26-1
"SYM" 23-2
symbolische Aufforderung 3-10
synthetisches BLDSPEC 22-18;22-28;25-5
synthetisches Flag-Setzen 25-6
synthetische Funktionen 25-1
synthetische GTOs 2-32
synthetisches Programmieren 25-1
synthetische Statistik-Bloecke 19-17
synthetische Textzeilen 25-1
synthetische Toene 6-10;6-16
Systemflag 27-4
"SZ" 1-15
"SZE" 1-14
"T+S" 3-1
"T1" 6-10
"T2" 6-10
"T3" 6-10
"T4" 6-10
"T5" 6-10
"T6" 6-10
"T7" 6-10
Tabelle von Funktionswerten 22-17
TAN 16-2
"TANH" 16-17
tanh 16-21
Tastaturschablonen 1-19
Tastenkodes 1-21
Tastenzuweisungen 1-18.21.22.24;2-1;6-9;21-13;22-9
Tastenzuweisungen loeschen 1-26;21-8
Tastenzuweisungsregister 1-26
"TB" 17-1
"TC" 18-21
Teilprogramme 2-13
Telefonverzeichnis 9-4
"TEM" 18-19
"TEMP" 18-17
Temperatureinheiten 18-17;18-18;18-19;18-20;18-21
Test 2-7;2-23;4-7
Testen des Lesestiftes 21-16
Test auf numerische Eingabe 3-14
Test von Zufallsgeneratoren 13-5
Text drucken 22-12;22-16
Text-Fahnen 23-1;23-2
Text-Folge-Zeichen 1-3
Text-T 1-3
"TH" 16-21
Tierkreiszeichen 22-26
TILL 9-4
"TITLE" 3-1
"TM" 20-3
"TMP" 18-18
"TO" 6-16
Toccata von J. S. Bach 6-10
"TON" 6-16
TONE 6-9
Ton-Programme 6-10
Totalloeschung 1-9;6-7;27-4
"TP" 15-14

"TR" 13-5
TRACE-Modus 22-3
Trennlinien 22-13;22-14
Trennungszeichen 5-3
Treppenfunktion 14-6
"TX" 22-12
uebereinstimmende Namen 2-12;27-5
"UD" 8-6
Uhr 20-3
ULF 10-24
Umkehrung einer Ziffernfolge 15-6
Umordnungen des Stapels 2-20;7-11;27-3
Umrechnung von Masseinheiten 18-3;18-4
Umrechnung von Temperatureinheiten 18-17;18-18;18-19;18-20;18-21
Umschalten von Flags 6-1;6-4;6-6
Umwandlung dezimal in binaer 17-6
Umwandlung dezimal in hexadezimal 17-4
Umwandlung Grad in Dezimalgrad 18-14;18-16
Umwandlung hexadezimal in dezimal 17-5
unbedingtes Dekrementieren 2-15
unbedingtes Inkrementieren 2-15
unechter Bruch 14-7
unterbrechen 22-23
Unterprogramm 1-10;1-27;2-10;2-11
Unterprogrammebene 2-33
Unterstriche 2-34
"UR" 9-7
USER-Modus 1-24
USER-Taste 1-11
"V2" 22-22
"VA" 4-1
Variablenwechsel 24-1;24-2;24-3;24-4;24-5;24-6;24-7
Variationen 19-12;19-13;19-14
"VB" 10-7;10-8
Vektor 16-1
Vektor-Analysis 19-5
VER 21-15
verdichtete Datenspeicherung 9-7;9-8
verdichtete Zahlen 9-7
vertikale Zahlen 22-22
verkehrte Graugans 1-3
Verlaengerung des Rueckkehrstapels 2-33
Verlust von Alpha-Zeichen 5-9
Verlust von LASTX 19-3
Verschliessen des HP-41 1-13
Verschluesselung von Klartext 26-1
verschiedene Masseinheiten 3-16
Versetzen des Vorhangs 8-1;8-6
verzoegerungsfreie Berechnungen 4-20
"VF" 6-5
VIEW 2-6;3-12;4-2;6-14;22-1
"VK" 1-21
"VM" 15-19
"VMAN" 15-17
Vollzeichen 1-3
Vorhang 1-17;8-1;8-5;8-6
Vorschalttaste 1-11
"VP" 2-6
"VS" 3-5
Wahlmoeglichkeiten mit und ohne Drucker 22-23
WALL 21-12
WALL-Karten 21-11
Warnton 5-9;26-1
Wechsel der Basis 17-1;17-2;17-7
wiederholtes Ausfuehren 2-5
wiederholter Programmlauf 2-5;2-21
Winkel 16-3;16-4
WNDSCN 21-17
WNDTST 21-17
Wortspiele 2-28
WPRV 21-15
"WRD?" 3-18
WSTS 21-8
Wurzelfunktionen 16-16
"X?Y" 4-16
"X↑3" 15-34
X↑Y 15-28
"XB" 10-13
"XD" 17-5
XEQ IND 9-4
"XL" 25-4
X operiert auf X 7-1
"XP" 15-35
"XPN" 15-16
"XR" 25-3
XROM 25-1
XROM-Zahlen berechnen 25-3;25-4
"XY" 4-16
"XYΣ" 19-7
Yard 18-2
"YN" 3-8
Zahlen durch 1 oder 0 ersetzen 18-15
Zahlen-Eintrag 1-12;25-1
Zahlenpaare anzeigen 4-16
Zeichen in Dezimalzahl wandeln 5-6
zeilenaequivalent 11-4
Zeilen auffuellen 22-24
zeilenreduzierte Staffelform 11-4
Ziffern-Eintrag 1-12
Ziffernsumme 15-4
"ZODIAC" 22-26
Zufallsgenerator 13-1;13-2;13-3;13-4;13-5
Zufallszahlen 13-1;13-2;13-3;13-4;13-5
Zustandsregister 27-2
Zwei-Byte-Funktionen 25-1
zwei Schritte ueberspringen 2-8
"Σ+Z" 19-16

"Σ-Z" 19-16
"Σ3" 15-3
"Σ?" 1-17;19-4
"ΣB" 19-10
"ΣC" 8-6
"ΣD" 15-4
"ΣF" 19-1
"ΣI" 15-1
"ΣR" 19-6
"ΣRECIP" 19-2
ΣREG 1-17;8-6
ΣREG T 19-17
ΣREG X 19-17
ΣREG Y 19-17
ΣREG Z 19-17
"ΣS" 15-2

NACHWORT

Der neue HP-41CX

Gleich einer Aufmunterung zur Verwirklichung der im Vorwort angedeuteten Absicht, u.U. einen zweiten Band mit Tips und Routinen, welche die X-Funktionen und den Time-Modul berücksichtigen, herauszugeben, erscheint in diesen Tagen ein neuer HP-41, der

HP-41CX

Wie nicht anders zu erwarten, überrascht HP den unübersehbar großen Kreis der HP-41 Freunde mit neuen Möglichkeiten, die, gerade wegen der unübertroffenen Leistungsfähigkeit dieser kleinen Maschine, langsam zwar aber unabweisbar deutlich einen Wunsch zutage treten lassen - doch davon ganz am Schluß.

Mit dem HP-41CX steht ein unglaublich leistungsstarker Kleinstrechner zur Verfügung, der seinesgleichen nicht hat. Ihm sind die bislang nur extern verfügbaren Zeit- und X-Funktionen fest eingebaut. Zusätzlich enthält er eine Anzahl neuer Befehle. Die wesentlichen Neuerungen sind diese:

Es gibt nunmehr *6 Kataloge:*

'CAT 1' Anwenderprogramme, jetzt mit Angabe der vom Programm belegten Bytes.

'CAT 2' Externe Funktionen. Dazu rechnen die Zeitfunktionen, die X-Funktionen und die in eingesteckten ROMs ansässigen Funktionen. Es wird zwischen Hauptkatalog (Auflistung der Kopfzeilen) und Teilkatalog (Auflistung der unter einer Kopfzeile stehenden Funktionen-Gruppe) unterschieden.

'CAT 3' Standardfunktionen.

'CAT 4' X-Memory. Es werden die bisher nur über 'EMDIR' erreichbaren Dateien aufgelistet.

'CAT 5' Es werden die bisher nur über 'ALMCAT' erreichbaren Weckaufträge aufgelistet.

'CAT 6' Verzeichnis der vom Benutzer für den Gebrauch im USER-Modus belegten Tasten.

Die neuen *Zeitfunktionen* lauten:

'CLALMA' Weckauftrag entsprechend dem Eintrag im Alpha-Register löschen.

'CLALMX' Weckauftrag entsprechend dem Eintrag im X-Register löschen.

'CLALMS' Sämtliche Weckaufträge löschen.

'RCLALM' Parameter des durch den Inhalt von X bestimmten Weckauftrages in den Stapel und das Alpha-Register bringen.

'SWPT' Stoppuhr und Stoppuhrzeiger setzen.

Die neuen *X-Funktionen* lauten:

'RESZFL' Anzahl der einer im X-Memory liegenden Daten- oder Textdatei zugeordneten Register ändern.

'ASROOM' Anzahl der freien Bytes der aktuellen Datei ermitteln.

'EMDIRX' Die durch den Inhalt von X bestimmte (n-te) Datei im X-Memory ermitteln.

'EMROOM' Anzahl der freien Register im X-Memory ermitteln.

'CLRGX' Löschen von Registern entsprechend einer in X liegenden Kontrollzahl.

'ΣREG?' Adresse des ersten Statistik-Registers ermitteln.

'GETKEYX' Diese Funktion arbeitet ähnlich wie 'GETKEY', läßt aber variable Zeitabläufe bei der Entgegennahme von Tastenkodes zu und nimmt gemäß einer einleuchtenden Logik wesentlich mehr Kodes als 'GETKEY' (Tastenkodes *und Zeichenkodes*) vom Tastenfeld entgegen. Sie verspricht somit, insbesondere für Spiele auf dem HP-41 *die Bedienungsfunktion* zu werden.

Schließlich gibt es noch eine neue Gruppe von *Vergleichsfunktionen*, die endlich auch den Vergleich von Alpha-Ketten (für Kenner sei hinzugefügt: sogar von 'NNZ's) und damit das Alphabetisieren auf dem HP-41 erlaubt. Diese Funktionen lauten:

'X<NN?' 'X<=NN?' 'X=NN?' 'X≠NN?' 'X>=NN?' 'X>NN?'

Sie vergleichen den Inhalt von X mit dem Inhalt des Registers, dessen Adresse in Y liegt.

Die erfreulichste Neuerung aber ist ein äußerst benutzerfreundlicher *Text-Editor*, die Funktion 'ED'. Sie macht das Tastenfeld zu einer Steuerkonsole für die Behandlung von Textdateien im X-Memory. Ein blinkender Cursor unterstützt dabei die Arbeit. Man kann von Zeichen zu Zeichen, von Zeichengruppe zu Zeichengruppe oder von Datensatz zu Datensatz springen. Es gibt die Betriebsarten 'Einfügen' und 'Ersetzen', die sich auf einzelne Zeichen beziehen, und selbstverständlich auch die Möglichkeit ganze Datensätze zu löschen oder einzufügen. All das durch einfache und sinnfällige Tastenbedienung.

Ohne Zweifel ist mit dem HP-41CX ein neuer Höhepunkt erreicht. Dennoch behält jemand recht, der an Mikroelektronik ganz gewiß nicht dachte, als er feststellte:

Wonach du sehnlich ausgeschaut,
Es wurde dir beschieden.
Du triumphierst und jubelst laut:
"Jetzt hab' ich endlich Frieden!"

Ach, Freundchen, rede nicht so wild.
Bezähme deine Zunge.
Ein jeder Wunsch, wenn er erfüllt,
Kriegt augenblicklich Junge.

Und so mögen denn folgende Wünsche den Hersteller des besten Taschenrechners der Welt erreichen:

- Wenn man dem Gehäuse des HP-41 soviel einverleiben kann, dann hat auch noch das vollständige X-Memory darin Platz.
- Die neuen Vergleichsfunktionen sind äußerst willkommen. Sie wären vollkommen, wenn sie nicht den Stapel mit Registeradressen belasteten, sondern stattdessen mit Ziffern-Parametern arbeiteten, wie 'RCL _ _' oder 'X<>_ _', einschließlich indirekter Steuerung. Und schließlich:
- Schneller müßte der HP-41 sein, in vertrauter Gestalt ansonsten, aber erheblich schneller.

Im Dezember 1983 H.D.

PS: Wer findet das Hoch auf den Autor, das im Buch versteckt ist? Die erste richtige Antwort, die den Herausgeber der deutschen Ausgabe erreicht, wird mit einem kostenlosen Exemplar des Buches von Keith Jarett, "Synthetisches Programmieren auf dem HP-41 - leicht gemacht", belohnt.

COMPUTERBÜCHER IM HELDERMANN VERLAG

K. Jarett: Synthetisches Programmieren auf dem HP-41 - leicht gemacht.

Der Autor wendet sich an HP-41 Benutzer, denen die gründliche aber anspruchsvolle Darstellung der synthetischen Programmierung durch W.C.Wickes ("Synthetische Programmierung auf dem HP-41C/CV", Heldermann Verlag) Schwierigkeiten bereitet. Es gelingt ihm, in ausführlicher Weise einen Zugang zu diesem Gebiet zu bereiten, der zugleich abwechslungsreich und spannend ist. Als Hilfsmittel benutzt er den "Byte-Schnapper", der leichter als der "Byte-Hüpfer" handzuhaben ist. Andererseits berücksichtigt er neueste Erweiterungen des HP-41 durch den Hersteller und bringt Programme, welche die Funktionen aus dem X-Modul und dem Time-Modul beinhalten, wodurch viele synthetische Programme - z. Bsp. das Tastenzuweisungsprogramm - wesentlich kürzer und schneller werden. Für Kundige ist dieses Buch somit eine spielend lesbare Ergänzung des Buches von Wickes, für Anfänger ist es die ideale Einführung. Die Übersetzung besorgte in gewohnt exzellenter Manier Heinz Dalkowski.
(1984, 34.00 DM, ISBN 3-88538-802-2)

K. Jarett: Erweiterte Funktionen des HP-41 - leicht gemacht.

Das Buch beschreibt die Eigenschaften des erweiterten Speichers und der X-Funktionen, mit denen der HP-41C/CV aufgerüstet werden kann und die im HP-41CX fest eingebaut sind. Da das Bedienungshandbuch zum Umgang mit X-Modulen nur knappe Hinweise gibt, war ein Buch nötig, das die Fähigkeiten der X-Funktionen und des HP-41CX vollständig beschreibt. Der Autor, ein führender Experte des HP-41 Systems und Pionier der synthetischen Programmierung, hat dieses Buch in seinem unnachahmlichen Stil - einfach, klar und doch präzise - geschrieben. Dem deutschen Leserkreis wird dieser amerikanische Bestseller durch Heinz Dalkowski zugänglich gemacht, der sich schon als Übersetzer des 'Wickes' auszeichnete. Nach der Lektüre kann der Leser die X-Funktionen wirkungsvoll einsetzen und, wenn er Kenntnisse in synthetischer Programmierung besitzt, die Kraft dieser Kunst mit der der X-Funktionen verbinden. Insbesondere bekommt er über 30 ausgereifte Programme, die von den führenden Experten auf dem Gebiet stammen, an die Hand, darunter einen umfangreichen Text-Editor für den HP-41C/CV, ein Adressenverzeichnis-Programm, eine Simulation des HP-16, mathematische Programme, Programme zum Übertragen von Textdateien auf Magnetkarten, sowie - in Verbindung mit synthetischer Programmierung - Programme zum Reparieren fehlerhaften Verhaltens einiger spezieller Vorgänge beim Einsatz von X-Funktionen (Betriebssystemfehler). Sämtliche Programme sind als Barcodes abgedruckt, insgesamt 4181 Bytes! Wer aus seinem Kraftpaket herausholen will, was drin steckt, benötigt dieses Buch.
(1984, ca. 34.00 DM, ISBN 3-88538-803-0)

W.C.Wickes: Synthetische Programmierung auf dem HP-41C/CV

Die englische Originalausgabe dieses Buches ist in den U.S.A. ein Bestseller unter der Literatur über Kleinrechner geworden und auch in Deutschland wurden über 7000 Exemplare verkauft. Die deutsche Ausgabe enthält gegenüber dem Original zahlreiche Verbesserungen, Verfeinerungen und Ergänzungen und wird zu Recht die "Bibel der Synthetischen Programmierung" genannt.

Der Autor führt den Leser in leicht verständlicher Weise "durch" den HP-41C/CV, entdeckt ihm alle seine verborgenen Fähigkeiten und geht so inhaltlich weit über das hinaus, was das "Handbuch" bietet. Der Leser lernt die Synthetische Programmierung kennen, durch die der Rechner zu unglaublichen Taten veranlaßt werden kann: Erzeugung neuer Zeichen in der Anzeige; Verwendung des Alpha-Registers als arithmetisches Daten-Register; vollständige Benutzerkontrolle über alle Flags (einschließlich der normalerweise unzugänglichen Systemflags); Zugriff auf sämtliche Informationen über den Zustand des Rechners; schnelle Alphabetisierung von Alpha-Daten; Erzeugung neuer Töne; Verwandlung von Programmzeilen in Daten und umgekehrt; programmierter Zugriff auf beliebige Zeilen in ROMs; Herstellung programmierender Programme.

Synthetische Programmierung ist nicht nur für Hobby-Anwender, sondern in gleicher Weise für professionelle Benutzer von Interesse, wenn es darum geht, Speicherplatz zu sparen oder die Bearbeitungsgeschwindigkeit zu erhöhen.

Die zweite Auflage des Buches ist um inzwischen bekannt gewordene Fortschritte der synthetischen Programmierung erweitert worden: Der Byte-Schnapper, Programme zur automatischen Erzeugung synthetischer Programmzeilen, die Funktion eGØBEEP, Programme zur Herstellung vollständiger hexadezimaler Speicherauszüge, synthetischer Vorstoß ins X-Memory.

Die Synthetische Programmierung wird - laut Autor - nun auch von Hewlett Packard unterstützt und Programme, die synthetische Zeilen enthalten, werden von den Programm-Bibliotheken akzeptiert.

Obwohl dieses Buch manche Schwierigkeit enthält, was für den einen oder anderen Leser die vorausgehende Lektüre des Buches von K. Jarett, "Synthetische Programmierung auf dem HP-41 - leicht gemacht" (Heldermann Verlag) empfehlenswert macht, ist dieses Buch unerläßlich für jeden HP-41C/CV/CX Benutzer, der die Fähigkeiten und Möglichkeiten des Rechners voll ausschöpfen will.

(1983, 34.00 DM, ISBN 3-88538-800-6)

DIE HP-41-PALETTE DES HELDERMANN VERLAGES

J.S.Dearing: Tricks, Tips und Routinen für Taschenrechner der Serie HP-41. Deutsche Ausgabe von Heinz Dalkowski. 220 Seiten, 34.00 DM, ISBN 3-88538-801-4 (1984).

K.Jarett: Synthetisches Programmieren auf dem HP-41 - leicht gemacht. Deutsch von Heinz Dalkowski. Ca. 34.00 DM, ISBN 3-88538-802-2 (1984).

K.Jarett: Erweiterte Funktionen des HP-41 - leicht gemacht. Deutsch von Heinz Dalkowski. Ca. 34.00 DM, ISBN 3-88538-803-0 (1984).

W.C.Wickes: Synthetische Programmierung auf dem HP-41C/CV. Deutsch von Heinz Dalkowski. 165 Seiten, 34.00 DM, ISBN 3-88538-800-6 (1983).

HP-41 Kombinierte Hex/Dezimale Byte Tabelle, 7 x 11.5 cm Plastikkarte, 6.00 DM (1983).

Alle Produkte sind direkt vom Verlag erhältlich, die Plastikkarte nur auf diese Weise. Richten Sie Ihre Bestellung bitte an Heldermann Verlag Berlin, Herderstr. 6-7, D - 1000 Berlin 41.

Dieses Buch enthält über 350 Routinen und Tips für den HP–41. Von elementaren Tricks und pfiffigen Abkürzungen bis hin zu komplizierten synthetischen Programmen aus dem PPC-Modul und umfangreichen Druck-Routinen ist alles vertreten, was die grosse Gemeinde der HP–41-Benutzer im Laufe von Jahren herausgefunden und zusammengetragen hat. Dem Autor ist die herkulische Leistung zu verdanken, die Fülle dieser Ergebnisse gesammelt, gesichtet und geordnet zu haben. Der Übersetzer hat in Zusammenarbeit mit dem Autor das Original erweitert und in einem Nachwort die Funktionen des neuesten Rechners aus der Serie HP–41, des HP–41CX, beschrieben. Ein umfangreiches Stichwortverzeichnis von über 1000 Einträgen erschliesst die Sammlung lückenlos und lässt zielsicher auffinden, worüber man sich zu informieren wünscht. Man spart so manche Programmierstunde und kommt jedem merkwürdigen Verhalten des HP–41 auf die Spur.

ISBN 3-88538-801-4