J. K. Horn

HP-71-BASIC leicht gemacht

Deutsche Ausgabe von W. Stroinski

Heldermann Verlag Berlin

Wichtiger Hinweis

Dieser Kopie wird zur ausschließlich nicht-kommerziellen Verwendung mit freundlicher Genehmigung des Verlags bereitgestellt.
Sie wurde von Martin Hepperle im Jahr 2014 angefertigt.
Die in diesem Buch angegebenen Preise und Adressen sind nicht mehr aktuell.
Die HP-Bücher des Verlags sind seit langem vergriffen, Anfragen beim Verlag sind zwecklos. Der Heldermann Verlag verlegt allerdings weiterhin mathematische Bücher und Zeitschriften.

Die aktuelle (2014) Anschrift ist
Heldermann Verlag
Langer Graben 17
32657 Lemgo
http://www.heldermann.de/

Important Note

This copy is distributed for noncommercial purposes only with permission of the original publisher Heldermann Verlag.
It was created in 2014 by Martin Hepperle.
All prices and adresses in this book are out of date.
The HP books published by Heldermann are long out of print and not available from the publisher. However, the Heldermann Verlag still exists and publishes mathematical books and journals.

The current (2014) address is:
Heldermann Verlag
Langer Graben 17
D-32657 Lemgo
Germany
http://www.heldermann.de/

J.K. Horn

HP-71-BASIC leicht gemacht

Deutsche Ausgabe von W. Stroinski

Heldermann Verlag Berlin

Joseph K. Horn
1042 Star Rt
Orange, CA 92667

Werner Stroinski
Kampweg 7a
D-1000 Berlin 27

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Horn, Joseph K.: HP-71-BASIC, leichtgemacht / Joseph K. Horn.
Dt. Ausg. von Werner Stroinski. - Berlin : Heldermann, 1986.
Einheitssacht: HP-71-BASIC made easy (dt.)
ISBN 3-88538-807-3
NE: Stroinski, Werner (Bearb.)

Das in diesem Buch enthaltene Material ist mit keinerlei Verpflichtungen oder Garantie irgendeiner Art verbunden. Der Verlag sowie der Herausgeber übernehmen keine Verantwortung und keine als Folge auftretende oder sonstige Haftung, die auf irgendwelche Art aus der Benutzung dieses Materials oder Teilen davon entstehen könnten.

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 Verwendung, vorbehalten. Bei Vervielfältigung, im Ganzen oder in Teilen, für gewerbliche Zwecke ist eine Vergütung an den Verlag zu bezahlen, deren Höhe mit dem Verlag zu vereinbaren ist.

© 1986, Heldermann Verlag, Nassauische Str. 26, D-1000 Berlin 31.

ISBN 3-88538-807-3

Vorwort zur deutschen Ausgabe

Das Übersetzen technischer Literatur in die deutsche Sprache wird mitunter auch von den Landsleuten belächelt, die sich eigentlich darüber freuen sollten, weil sie mit den behandelten Geräten zu tun haben. Mag sein, daß es manchmal peinlich ist, unzureichende Kenntnisse der englischen Sprache zugeben zu müssen, wobei die in den Schulen vermittelten meist nur wenig hilfreich sind! Die Antipathie gegen "eingedeutschte" Handbücher ist allerdings auch zu verstehen, kennt doch jeder dafür erschreckende Beispiele!

Gute Übersetzungen (oder sollte man lieber von Bearbeitungen sprechen?) setzen voraus, daß der Übersetzer entweder mit dem behandelten Gerät selbst gut vertraut ist oder der "Rohtext" von einem Kundigen sachlich gründlich abgeklopft wird. Das kann mitunter in ordinäre Arbeit ausarten!

Bei der Bearbeitung des vorliegenden Buches war das jedoch ganz anders: Der Original-Text enthält so viel Humor (mitunter auch versteckte Spitzen), wie das sonst nicht üblich ist. Hinzu kam, daß der Übersetzer "seinen" HP-71 auch erst kurze Zeit besaß und sich schon einige Male über die manchmal recht geschraubte Ausdrucksweise der (sonst ausgezeichneten) deutschen Handbücher amüsiert hatte!

Die Übersetzung war also (überwiegend) ein Vergnügen, mühevoll wurde es nur dann, wenn der Verfasser mit Wortspielen arbeitete, für deren Übertragung es ja kein universelles Rezept gibt. Ähnliche Probleme boten einige der Programmbeispiele, die auch erst auf hiesige Verhältnisse zugeschnitten werden mußten.

Bei der Bearbeitung wurden erkannte Fehler nach bestem Wissen ausgemerzt und hoffentlich nicht durch andere ersetzt! Aber es ist ja bekannt, daß nur Bücher ohne Text vor Fehlern sicher sind! Der Original-Text wurde nur dort im Ausdruck (nicht aber im Sinn) verändert, wenn eine wörtliche Übersetzung zu steif geklungen hätte. Mitunter wurde auch ein Satz eingefügt, wenn sich damit ein Sachverhalt eindringlicher beschreiben ließ. Nur das Kapitel 10 geht ganz auf Rechnung des Bearbeiters: Die COPY-Anweisung hatte doch zu viele Erscheinungsformen und Ausnahmen, die sich nur aus der kompletten Form einleuchtend erklären lassen!

Mein Dank gilt auch diesmal wieder meinen Rechner-Freunden, die mich wirkungsvoll unterstützten und auch meiner Frau für die erwiesene Geduld und die geleistete Arbeit beim Lesen der Korrektur.

Werner Stroinski Juli 1986

Inhalts-Übersicht

Vorwort 7

Kapitel 1:
Die drei Betriebsarten des Rechners 9

Kapitel 2:
Grundfunktionen (und CALC-Modus) 13

Kapitel 3:
Der Tastenfeld-BASIC-Modus 41

Kapitel 4:
Die BASIC-Wortarten 47

Kapitel 5:
Variable im BASIC-Modus 51

Kapitel 6:
Speichermöglichkeiten im HP-71 77

Kapitel 7:
Kalender und Uhr des HP-71 103

Kapitel 8:
PEEK$ und POKE 110

Kapitel 9:
BASIC-Programme 115

Kapitel 10:
Der Umgang mit der COPY-Anweisung 128

Postscriptum:
Der HP-71-Antwort-Dienst 132

Anhang:
HP-71 Schlüsselwort- und Anweisung-Verzeichnis 134

Vorwort

Hewlett-Packard bezeichnet den HP-71 als "offenen" Rechner, weil für dieses Gerät alle Unterlagen erhältlich sind, die das interne Rechner-System beschreiben. "Jedermann" kann diese Unterlagen erwerben! Es ist nur schade, daß sich ein "normaler" (also durchschnittlicher) Benutzer dieses Rechners kaum als "Jedermann" fühlen wird! Die Dokumentation wird in nur mühsam lesbaren Papierbergen geboten, die teurer sind als der Rechner! Wer das Geld für diese hochgeistigen Wälzer aufbringt, braucht dann vermutlich den Rest seines Lebens, um alles zu verstehen. So "offen" ist der HP-71 im Moment für Sie.....

Darum entstand dieses Buch. Anders als die zwar umfangreichen, aber auch sehr trockenen "HP Internal Design Specifications" wurde "HP-71-BASIC, leicht gemacht" so geschrieben, daß Sie sich direkt angesprochen fühlen. Mit diesem Buch soll nicht jedes einzelne Byte im HP-71 beschrieben werden, hier sollen Sie genau die Tatsachen und Tips erfahren, die Ihnen sofort Nutzen bringen. Es gibt da schon Dinge, die Ihnen Benutzer- bzw. Referenz-Handbuch verschweigen! Weiter enthält dieses Buch einige Programme, in denen diese "Geheimnisse" angewendet werden. Für die Fragen, die dieses Buch unbeantwortet läßt, finden Sie am Schluß noch ein ganz besonderes Angebot:
den HP-71-Antwort-Dienst! ...und der arbeitet sogar kostenlos!

Bitte verstehen Sie: Mein Buch soll die mitgelieferten Handbücher nicht ersetzen, sondern ergänzen! Wenn Sie die bisher nur teilweise verstehen konnten, wird Ihnen dieses Buch schneller zum optimalen Umgang mit dem HP-71 verhelfen. Sollten Sie dagegen bei der Lektüre der Handbücher keine Schwierigkeiten gehabt haben, können Sie die leichten Kapitel hier getrost überspringen und sich den Informationen über PEEK$ und POKE am Schluß zuwenden und sich in den Programmen in BASIC oder Maschinen-Sprache austoben! Ich freue mich, wenn Ihnen dieses Buch dazu verhelfen kann!

Obwohl der Inhalt sorgfältig geprüft wurde, kann der Autor keine irgendwie geartete Garantie übernehmen. HP hat jederzeit die Möglichkeit, das Betriebs-System des HP-71 zu überarbeiten, wodurch es denkbar ist, daß sich zukünftige Ausführungen des HP-71 anders verhalten, als hier im Buch beschrieben. Wenn Sie also Programme in Maschinensprache schreiben möchten, die auch zu 'künftigen' Rechner-Versionen passen sollen, müssen Sie sich für den HP-71 das vorzügliche FORTH/ASSEMBLER-ROM (und leider auch die HP-71 Internal Design Specifications!) beschaffen. Damit erreichen Sie die Routinen des Rechners direkt an ihren Startpunkten, und die sollen, nach Zusage von HP, auch bei zukünftigen Ausführungen beibehalten werden.

Danken muß ich vor allem meinem Bruder James L. Horn, der mein Interesse für Computer weckte, mich mit dem PPC bekannt machte und mir noch immer moralische Stütze und intellektueller Ansporn ist. Deshalb ist dieses Buch ihm gewidmet.

Ein herzliches Dankeschön auch an Karen Dodson, die mir immer weiterhalf, wenn ich in der Mitte der 70er Jahre den College-Computer wieder mal abstürzen ließ! Karen ist für mich die Verkörperung des perfekten Programmierers: Eines Tages zog ich ganz lässig das unkommentierte Listing meines neuesten Programms hervor, von dem ich glaubte, daß es wegen vieler Verzweigungen und des sonstigen Durcheinanders für Nichteingeweihte ein ewiges Geheimnis bleiben müßte. Schon nach einem Blick wußte sie nicht nur, wozu dieses Programm dienen sollte, sondern nannte mir einen noch besseren und kürzeren Lösungsweg.

Dank gebührt auch James Davidson, der die berühmten Bücher über den HP-25 schrieb und uns alle lehrte, welch "elegante Lösungen" mit 49 (oder noch weniger) Bytes möglich sind.

Ohne Richard Nelson (Begründer von PPC und CHHU und seit mehr als zehn Jahren Herausgeber von Zeitschriften über Kleincomputer) wäre dieses Buch wahrscheinlich nicht entstanden. Herzlichen Dank, Richard, für Deine Arbeit und Dein Engagement! Möge Dein Verein weiter wachsen!

Ein herzliches Dankeschön auch an Jeremy Smith, der sich so herrlich freuen konnte, wenn einer von uns beiden wieder etwas Neues gefunden hatte. Diese Freude steckte auch mich an!

Schließlich muß ich auch Keith Jarett, dem Verleger dieses Buches, für seinen Gleichmut danken, mit der er mein Arbeits-"Tempo" ertrug: Dieses Buch ist ein konkretes Beispiel für Hofstadter's Gesetz: "Alles dauert länger als man denkt, besonders wenn man dabei Hofstadter's Gesetz schon berücksichtigt hat!" Eben deshalb werde ich nun keine unnützen Worte mehr machen und die letzten Klarheiten beseitigen. Auch der Leser sollte Keith etwas dankbar sein!

Joseph K. Horn (CHHU-Mitglied Nr. 13)
St. Michael's Prep.
1042 Star Route
Orange, California USA 92667

```
SUB ANYTIME @ DISP "(714) 633-2041" @ END SUB
```

Kapitel 1

Die drei Betriebsarten des Rechners

Wenn Sie ihren neuen HP-71 zum erstem Mal anfassen, glauben Sie bestimmt ein einziges Gerät in der Hand zu halten. (Ich meine hier nicht die Rechner-Fans, die grundsätzlich alles lieferbare Zubehör sofort mitkaufen!)

Tatsächlich halten Sie aber drei Rechner in der Hand, die ein gemeinsames Gehäuse haben. Sie können jeden davon benutzen, auch alle drei, wenn Sie wollen!

Betrieb als Rechner im "CALC"-Modus

Zuerst sollen Sie den HP-71 als beachtenswerten wissenschaftlichen Rechner kennenlernen, der mehr Möglichkeiten bietet, als andere Rechner dieser Epoche. Wenn Sie den HP-71 in den "CALC"-Modus schalten, ist die Handhabung so einfach, daß Sie ohne Erfahrung mit Ihren Berechnungen beginnen können. Trotzdem bietet diese Betriebsart soviel Möglichkeiten, daß Sie sich schon nach dem Lesen des Kapitels 2 als "Tasten-Zauberer" fühlen werden!

Im "CALC"-Modus lassen sich zwar keine Programme schreiben, es ist aber sehr leicht möglich, Werte mit Formeln auszurechnen, wobei die Formel nur einmal einzugeben ist. Für Wertetabellen sind nur die Werte der Variablen nacheinander einzusetzen. So ist es auch bei vielen ähnlichen Problemen: Es wird einmalig ein Grundschema eingegeben, in das die von Fall zu Fall wechselnden Werte einzufügen sind. Was bisher nur mit einen programmierbaren Rechner möglich war, können Sie jetzt im "CALC"-Modus erledigen, ohne ein Programm zu schreiben!

Betrieb als "BASIC"-Rechner per Tastenfeld

Diese Betriebsart nimmt der Rechner nach dem ersten Einsetzen der Batterien an. Es ist der Tastenfeld-"BASIC"-Modus. Die Zeichen, die Sie jetzt eingeben, erscheinen alle in der Anzeige. Wenn Sie anschließend die große ENDLINE-Taste drücken, versucht der HP-71 die eingegebenen Wünsche zu erfüllen. Wenn dies geschehen ist, wartet der Rechner auf Ihre nächsten "Befehle". Sie sind Aladin und der Rechner ist der Geist aus der Lampe, der ihre Wünsche ausführen muß, auch wenn es wesentlich mehr als drei sind!

Aber, Sie müssen ihre Befehle richtig formulieren, d.h. Sie müssen ohne Fehler schreiben und nur Wörter benutzen, die der HP-71 kennt. Wenn Sie z.B. "BEEP OR ELSE" eingeben, ist der HP-71 nicht in der Lage, diesen Wunsch zu erfüllen, weil der Befehl für "BEEP" anders aussehen muß.

Der HP-71 "versteht" BASIC. Diese Sprache ist wirklich "kinderleicht" zu lernen; deshalb können viele Kinder schon BASIC !

Auch wenn der HP-71 Ihnen ab und zu einen Schreck einzujagen scheint: Er meint das nicht ernst! Er wartet nur sehnsüchtig auf den nächsten richtigen Befehl von Ihnen! Lernen Sie deshalb seine Sprache! Das ist wirklich nicht schwieriger, als die Lektüre eines netten, kleinen Buches (z.B. dieses hier)!

Betrieb im "BASIC"-Programm-Modus

Wenn Sie etliche BASIC-Befehle zusammenfassen, haben Sie das, was man ein "Programm" nennt. Ein Programmheft sagt Ihnen im Konzert, welche Musikstücke in welcher Reihenfolge geboten werden. Ein Computer-Programm ist ähnlich: Es sagt, welche Dinge in welcher Reihenfolge zu tun sind. Sobald Sie BASIC-Befehle über das Tastenfeld eingeben können (Tastenfeld-BASIC-Modus), können Sie auch mit dem Schreiben von Programmen beginnen.

Dieses Buch wird Ihnen alles sagen, was Sie zum Programmieren wissen müssen. Sie werden erfahren, welche Befehle Ihnen zur Verfügung stehen, wie man sie am besten formuliert und wie man sie in den Speicher des Rechners einbringt.

Dann kommt der große Augenblick: Sie drücken auf die RUN-Taste: Schwupp! Schon sind alle Befehle ausgeführt! Wenn Sie es nochmal erleben wollen, drücken Sie wieder die RUN-Taste!

Das Schönste an einem BASIC-Programm ist nicht die Zeit, die es Ihnen einspart (das dachten Sie doch?), sondern der Umstand, daß diese Sammlung von Befehlen (das Programm) auf jedem HP-71 funktioniert. "Was hab' ich davon?" werden Sie fragen, "Ich hab' doch nur einen HP-71!" Ja, das stimmt, aber etliche tausend andere Menschen haben auch "ihren" HP-71 und schreiben Programme! Es ist nicht auszuschließen, daß ein anderer Benutzer eines HP-71 schon ein Programm hat, das genau zu Ihrem Problem paßt. Deshalb ist es günstig einem Club anzugehören, dessen Mitglieder sich auch mit dem HP-71 beschäftigen und auch die Mitgliedschaft in der "HP-71 User's Library" ist ernsthaft zu empfehlen.

Tatsächlich benutzen viele den HP-71 ausschließlich im BASIC-Modus, weil sie den Rechner mit (eigenen und fremden) Programmen so gefüllt haben, daß er ganz auf die individuellen Ansprüche des einzelnen zugeschnitten ist. Was glauben Sie, wieviele individuell zugeschnittene Programme ihr Rechner enthalten wird, wenn Sie die Lektüre dieses Buches abgeschlossen haben!

"Gleichzeitige" Benutzung aller drei Betriebsarten

Für dieses Buch ergeben sich aus den drei Betriebsarten einige Probleme: Es muß nicht nur den CALC-Modus eingehend erklären, danach den Tastenfeld-BASIC-Modus und schließlich den BASIC-Programm-Modus, es muß auch zeigen, an welchen Stellen sich die drei Betriebsarten überlappen.

Der HP-71 ist wirklich eine "gespaltene" Angelegenheit: Es gibt natürlich keine drei Rechner in dem Gehäuse, aber manchmal scheinen wirklich mehrere Geräte im HP-71 zu stecken! Nach einer gewissen Zeit werden Sie das Rechnen im Tastenfeld-BASIC-Modus beherrschen. Sie werden dann auch sogar auf BASIC-Programme zugreifen können, während Sie den Rechner im CALC-Modus benutzen! Dann werden Sie den HP-71 nicht mehr als eine Kombination von drei Geräten ansehen, wohl aber als ein einziges Gerät mit einem dreigeteilten "Bewußtsein".

Erst wenn Sie den HP-71 als eine sehr leistungsfähige Einheit verstanden haben und nicht mehr als Vereinigung von drei auch einzeln schon recht respektablen Geräten empfinden, können Sie sich als HP-71-Experte bezeichnen!
Es ist wie beim Autofahren: Entscheidend ist, daß der Fahrer unter allen Umständen weiß, wie er in jedem Gelände zu fahren hat. Was dabei unter der Motorhaube geschieht, ist erst von zweitrangiger Bedeutung!

Hier ist mein persönliches Versprechen: Wenn Sie dieses Buch Seite für Seite gelesen und verstanden haben und die gebotenen Beispiele ausprobiert haben, werden Sie ein HP-71-Experte sein!

Beginnen wir jetzt mit dem CALC-Modus!

Notizen:

Kapitel 2

Grund-Funktionen (und CALC-Modus)

Schalten Sie als erstes den HP-71 mit der ON-Taste (unterste Reihe, ganz links!) ein. Nach dem Einschalten wechselt die ON-Taste ihre Bedeutung (der Rechner ist ja bereits eingeschaltet) und hat jetzt den Namen ATTN (Abkürzung von Attention=Achtung). Geben Sie nun die Ziffern 12345 (mit den Tasten des Ziffernblocks) ein und drücken dann die (neuernannte) ATTN-Taste: Die Ziffern verschwinden wieder! Betrachten Sie die ATTN-Taste deshalb in Zukunft als "Notbremse": Bei jeder Betätigung, wird die Anzeige gelöscht, egal was sonst gerade geschieht.

Sie sollten jetzt ganz links das Zeichen ">" sehen. Das ">"-Zeichen signalisiert Ihnen, daß sich der Rechner im Tastenfeld-BASIC-Modus befindet. Daneben steht als blinkendes Zeichen der "Cursor": Der zeigt an, wo das nächste Zeichen hinkommt. Wenn der Cursor die Form eines Rechtecks hat, nennen wir ihn den "ersetzenden" Cursor, weil ein neues Zeichen die Stelle des Cursors einnimmt, der dann um eine Stelle nach rechts rückt.

Versuchen Sie jetzt, Ihren Namen einzugeben. Die SPC-Taste (untere Reihe, etwa Mitte), dient zur Eingabe von Leerzeichen (wie die Leertaste der Schreibmaschine). Bevor Sie jetzt die Anzeige wieder löschen, erproben Sie noch die Links- und Rechts-Pfeil-Tasten (links von der SPC-Taste), mit denen sich der Cursor um jeweils eine Stelle in der gewünschten Richtung verschieben läßt. Wenn Sie aber die (blaue) g-Taste drücken, bevor Sie Pfeil-Tasten betätigen, benimmt sich der Cursor anders: er springt dann sofort an das linke bzw. rechte Ende der Anzeige. Auch die vorherige Betätigung der (goldgelben) f-Taste ändert die Wirkung der Pfeil-Tasten: die Links-Pfeil-Taste bewegt nun den Cursor um eine Stelle nach links, dabei wird das Zeichen auf der neuen Position gelöscht! Die Rechts-Pfeil-Taste beläßt zwar den Cursor in seiner Position, das Zeichen an seinem Standort wird aber gelöscht und die weiter rechts stehenden Zeichen rücken auf, um die Lücke zu schließen.

Wir wollen nun eine vereinfachte Benennung verabreden: Statt "Drücken Sie die die goldgelbe f-Taste und danach die Links-Pfeil-Taste!", könnten wir in Zukunft einfach "f-LINKS" schreiben. Noch besser ist es aber, wenn wir uns auf "f-BACK" einigen, weil die f-Taste ja die Funktion vorbereitet, die (goldgelb) über der LINKS-Taste steht. Entsprechend aktiviert die g-Taste die Funktionen, die (blau) auf den Vorderseiten der Tasten angegeben sind. Bei den Buchstaben-Tasten fehlen jedoch die blauen Aufschriften, weil diese Tasten durch die g-Taste zwischen Groß- und Klein-Schreibung umgeschaltet werden (wie bei einer Schreibmaschine). Wenn Ihnen die momentane Zuordnung nicht zusagt, brauchen Sie nur f-Aufwärts-Pfeil (kürzer: f-LC) zu drücken, um die Verhältnisse umzukehren. Die LC-Taste tauscht die Zuordnung von Groß- und Kleinschreibung für dauernd.

Beachten Sie, daß sich auch Form (und Wirkung) des Cursors durch f-I/R ändern läßt: der dreieckige Cursor hat "einfügende" Funktion, weil er die Stelle bezeichnet, an der das neue Zeichen eingefügt wird. Ist die Stelle schon besetzt, wird eine Verschiebung schon vorhandener Zeichen nach rechts vorgenommen. Dieser Cursor ist also sehr praktisch, wenn innerhalb eines Textes nachträglich etwas einzufügen ist.

Um den Rechner auszuschalten, drücken Sie f-ATTN (oder kürzer: OFF).

Der HP-71 im CALC-Modus

Wenn Sie den HP-71 in den CALC-Modus schalten wollen, müssen Sie erst die f-Taste und danach die Komma-Taste (unterste Reihe, zweite von rechts) drücken, was sich kürzer als f-CALC oder einfach CALC schreiben läßt. Auf die gleiche Weise können Sie den CALC-Modus auch wieder verlassen. Der Rechner behält auch nach dem Aus- und wieder Einschalten den gewählten Modus bei.

Jetzt wollen wir einige Aufgaben im CALC-Modus rechnen: Schauen Sie bitte erst dann auf die Anzeige, wenn Sie folgendes eingegeben haben:

12+34+56 (ergibt Anzeige):

46+56

46 ist offensichtlich das Zwischenergebnis von 12+34, während die letzte Addition noch nicht ausgeführt wurde. Wenn wir jetzt merken, daß uns ein Fehler unterlaufen ist und wir 52 statt 56 eingeben wollten, könnten wir diesen Fehler durch ATTN und Neueingabe aller Werte beheben. Es geht aber auch einfacher: Wir drücken BACK und sehen in der Anzeige:

46+5

Nun können wir eingeben:

2 (ergibt Anzeige):

46+52

So lassen sich Fehler am rechten Ende der Anzeige mit der BACK-Taste leicht korrigieren. Dazu noch eine weitere Erleichterung: Wenn Sie BACK mehrmals ausführen müssen, brauchen Sie die f-Taste nicht jedesmal zu drücken. Halten Sie entweder die f-Taste fest, und betätigen Sie die BACK-Taste so oft, wie Sie es wünschen, oder drücken Sie die f-Taste nur einmal und halten dann die BACK-Taste fest. Dann wird gelöscht, bis Sie BACK (hoffentlich im rechten Moment) wieder loslassen!

Die Wirkung der (goldgelben) f- bzw. (blauen) g-Taste bleibt erhalten, bis die danach gedrückte Taste wieder losgelassen wird. Das gilt für alle Tasten, die automatisch repetieren, es gibt also nur wenige Ausnahmen davon, die dazu kaum jemals bemerkt werden!

Bevor wir nun den Rechner das "Problem" 12+34+52 endgültig lösen lassen, noch ein nützlicher Trick: Immer, wenn wir während einer Berechnung nochmal die ursprüngliche Aufgabenstellung sehen wollen, hilft der Griff zur Aufwärts-Pfeil-Taste! Mit ENDLINE (nicht mit der Abwärts-Pfeil-Taste!) kommen wir wieder zum aktuellen Stand zurück:

Aufwärts-Pfeil (ergibt Anzeige):

12+34+52

END LINE (ergibt Anzeige):

46+52

Wie können wir nun Fehler korrigieren, die irgendwo in der Zeile stehen? Wie läßt sich z.B. die 12 in eine 19 ändern? Bitte, jetzt nicht BACK benutzen, denn das würde die ganze "Berechnung" löschen! Die Aufwärts-Pfeil-Taste (kurz AUF) genannt hilft uns weiter, weil wir dann den Cursor mit den Pfeil-Tasten RECHTS und LINKS (auch in Verbindung mit der g-Taste) an jede Stelle der Anzeige bringen können, um dort etwas zu korrigieren. Jetzt dürfen wir auch BACK, -CHAR und I/R wieder benutzen, wenn die neuen Werte mehr oder weniger Platz benötigen. So einfach ist das!

AUF RECHTS 9 (ergibt Anzeige):

19+34+52

Um zu unserem (korrigierten) Zwischenergebnis zu kommen, benutzen wir END LINE:

END LINE (ergibt Anzeige):

53+52

wie wir es auch erwarten durften. Das endgültige Ergebnis liefert wiederum die END LINE-Taste:

END LINE (ergibt Anzeige):

105

Damit haben wir unsere erste Berechnung glücklich zu Ende gebracht! Aber nun geht es erst richtig los, wie Sie in den nächsten Absätzen sehen werden!

Durchführung von Ketten-Rechnungen

Es kann sein, daß wir das Ergebnis der letzten Berechnung in die nächste übernehmen wollen. Dazu haben wir vier Möglichkeiten: Selbstverständlich können Sie den eben ermittelten Wert erneut eingeben, wie Sie es für gewöhnlich tun. Das geht zwar, ist aber unnötige Mühe, und außerdem können Sie sich dabei noch vertippen! Gründe genug, diese Methode nicht zu benutzen!

Aufruf des Ergebnisses durch RES

Besser ist da schon die Möglichkeit, das Ergebnis der letzten Berechnung mit der RES-Taste (f-END LINE) zu benutzen: RES (wie RESultat) greift auf das letzte Ergebnis zurück, das der Rechner angezeigte (im Beispiel war das 105). RES arbeitet ähnlich wie die "LAST X"-Funktion in den HP-Rechnern mit UPN.

Sie können RES über das Tastenfeld eingeben (das sind drei Tasten) oder f- und END LINE-Taste benutzen (das sind nur zwei Tasten). In beiden Fällen erscheinen die Zeichen RES in der Anzeige, bzw. die Zeichen res, falls Sie vorher den Kleinschreib-Modus gewählt hatten.

Wir wollen jetzt unter Benutzung der RES-Funktion weiterrechnen: 105+5*105/(105-30)

RES+ (ergibt Anzeige):

105+

Wie Sie bemerken konnten, wird der alte Wert erst nach der Eingabe des Additions-Zeichens sichtbar! Tippen wir nun weiter ein:

5*RES/ (ergibt vorübergehend 105+5*RES und schließlich):

105+525/

Sie sehen, nach Eingabe des Divisionszeichens wird die Multiplikation ausgeführt! Und nun der Rest:

(RES-30 (ergibt Anzeige):

105+525/(105-30)

Wenn statt dessen 110-30 angezeigt wird, haben Sie die führende Klammer vergessen! Fügen Sie diese mit **AUF I/R** (neunmaligem) **RECHTS (** und **END LINE** ein!

Mit END LINE könnten wir jetzt die Rechnung abschließen, wir haben es aber nicht so eilig, wir wollen lieber genau sehen, was passiert. Sehen Sie, wie der Cursor "über" der schließenden Klammer blinkt? Er "meint", daß hier eine rechte Klammer stehen muß!

Wenn wir der "Vermutung" des Cursors durch Eingabe einer rechten Klammer zustimmen, dann sieht die Anzeige so aus:

105+525/(75)

Nun gibt es noch einen herrlichen Trick im CALC-Modus: die "magische" RUN-Taste! Sie ist eine "gemäßigte" Alternative zur "brutalen" END LINE-Taste, die immer "alles auf einen Streich" erledigt!

RUN (ergibt Anzeige):

105+7

Prima! Wir sehen das Teilergebnis der Division 525/7. Nun bringen wir die Sache mit END LINE zum Abschluß:

END LINE (ergibt Anzeige):

112

Aufruf des Ergebnisses durch ein leeres Klammerpaar

Es ist im CALC-Modus möglich, durch ein Paar leerer Klammer-Symbole den RES-Wert aufzurufen, der dann zum Inhalt der Klammern wird. Weil vom vorigen Beispiel RES noch mit dem Wert 112 besetzt ist, wollen wir lieber zum alten Wert 105 zurückkehren. Dazu ist folgendes nötig:

105 END LINE (ergibt Anzeige):

105

Wir wollen nochmal das gleiche Beispiel wie vorhin berechnen: 105+55*105/(105-30). Jetzt werden wir aber nicht RES sondern leere Klammer-Paare benutzen:

()+ (ergibt Anzeige):

105+

Haben Sie gesehen, wie sich 105 in die leere Klammer drängte? Nun geht's weiter:

5*()/ (ergibt Anzeige):

105+525/

Ein weiterer Trick als nächstes: Wir müssen eine Klammer für den Nenner aufmachen, brauchen aber ein weiteres leeres Klammerpaar

für den RES-Wert: Dazu brauchen wir nur die (linke) Klammer doppelt einzugeben:

(()-30 (ergibt Anzeige):

105+525/(105-30)

Das kommt uns doch sehr bekannt vor! Mit END LINE kommen wir wieder zum (gleichen) Ergebnis wie vorher: 112

Das "Wundermittel" () läßt sich im CALC-Modus immer benutzen, besonders bequem ist es aber in den Fällen, in denen der HP-71 von sich aus die abschließende Klammer setzt.

In unserem Beispiel ist nun 112 das letzte durch die END LINE-Taste ermittelte RESultat. Um aus diesem Wert die Quadratwurzel zu berechnen, ist nur folgende Eingabe nötig:

SQR(END LINE (ergibt Anzeige):

10.5830052443

Das ging uns aber doch zu schnell! Wir geben nochmals 112 und END LINE ein, um RES wieder mit 112 zu besetzen und geben dann nur ein:

SQR((ergibt Anzeige):

SQR()

Jetzt blinkt der Cursor "über" der rechten Klammer die "sanfte" RUN-Taste ist momentan wirkungslos: Erst wenn wir die rechte Klammer bestätigen, kommt die 112 in die Klammer:

SQR(112)

Haben Sie gesehen, wie sich die 112 zwischen die Klammern gezwängt hat? Nun funktioniert auch die RUN-Taste wieder:

RUN (ergibt Anzeige):

10.5830052443

Dieses Verhalten offenbart eine sehr wichtige Eigenschaft der RUN-Taste im CALC-Modus: das Ergebnis auf die Frage "Quadratwurzel aus 112" steht zwar in der Anzeige, aber immer noch auf der Eingabeseite! Der HP-71 wartet, was weiter an mathematischen Dingen geschehen soll, weil wir noch nicht END LINE gedrückt haben.

Jetzt überlegen Sie bitte, was das für RES bedeuten müßte: Der dort gespeicherte Wert kann sich noch nicht geändert haben, da müßte immer noch 112 stehen! Vergewissern wir uns, indem wir mit ATTN (der "ehemaligen" ON-Taste) die Anzeige löschen und ein neues leeres Klammern-Paar eingeben:

ATTN ( ) (ergibt Anzeige):

(112)

Das sollten wir uns gut merken! Wenn also mehrere Rechen-Operationen den zu Beginn der Rechnung in RES stehenden Zahlenwert unverändert als Argument brauchen, sollten wir die Lösungen stets mit der RUN-Taste und nie mit der END LINE-Taste ermitteln! Nur dann bleibt der Wert in RES erhalten, weil er nur durch END LINE aktualisiert wird.

Aufruf des Ergebnisses nur zur Anzeige

Noch ein Trick im CALC-Modus: Sie brauchen weder RES noch ein leeres Klammerpaar einzugeben, um den Wert in RES zu sehen! Löschen der Anzeige und END LINE reichen aus! Also:

ATTN END LINE (ergibt Anzeige):

112

Benutzung von Variablen im CALC-Modus

Wenn Sie einen Zahlenwert sehr häufig und zu sehr verschiedenen Zeitpunkten in Berechnungen brauchen, kann der Umgang mit RES oder dem Klammern-Paar doch zu riskant sein: Ein einziges versehentliches END LINE und der Wert ist futsch! Unter einem eigenen Namen ist der wichtige Wert dann doch sicherer aufgehoben. Immer wenn Sie dann diesen Namen rufen, erhalten Sie mit Sicherheit den richtigen Wert.

Wenn z.B. der Wert der Quadratwurzel aus 2 mehrmals benötigt wird, werden Sie jedesmal weder SQR(2), nOch 1.41421356237 eintippen wollen. Es ist statt dessen ein Name (das muß ein Buchstabe sein) für diesen Wert festzulegen, damit er bei Bedarf mit diesem Namen gerufen werden kann. Sagen wir einfach "Q" zu diesem Wert (wie "Q"uadratwurzel):

Q=SQR(2 END LINE (ergibt Anzeige):

1.41421356237

Sie wissen doch noch, daß im CALC-Modus die am Schluß der Aufgabe fehlende Klammer automatisch vom END LINE gesetzt wird, darum brauchen Sie sich nicht zu kümmern. Es sind dadurch drei Dinge geschehen: Der Wert für die Quadratwurzel wurde errechnet, angezeigt und unter dem Namen "Q" gespeichert. Das "Q=" veranlaßte den Rechner zur Speicherung. Immer, wenn Sie einen Zahlenwert bewahren wollen, tippen Sie einen Namen ein, gefolgt von einem Gleichheitszeichen und danach den Wert (oder die Rechenvorschrift).

Jedesmal, wenn Sie nun die Quadratwurzel aus 2 benötigen, ist nur noch der Buchstabe Q nötig! Wenn Sie z.B. SQR(2)+17-SQR(2)/12 berechnen wollen, ist nun einzugeben:

Q+17-Q/12 (ergibt Anzeige):

18.2963624322

Haben Sie bemerkt, wie jedesmal das Q durch 1.41421356237 ersetzt wurde? Das ist doch wirklich viel einfacher als die Zifferntipperei! Wir haben Q mit diesem Wert besetzt, und dabei bleibt es so lange, bis wir es ändern. Auch wenn der Rechner zwischendurch ausgeschaltet wird, bleibt die Zuordnung erhalten. Wenn Sie den CALC-Modus jedoch verlassen und/oder zwischendurch ein BASIC-Programm ablaufen lassen, kann die Zuordnung allerdings verloren gehen.

Sie können andererseits den Wert von Q jederzeit ändern, indem Sie eine neue Zuordnung Q= eingeben. Weil sich der Wert von Q so leicht 'variieren' läßt, nennen wir Q eine 'Variable', wie Sie es aus der Algebra kennen.

Eine Bemerkung für Scharfsinnige: Im Beispiel enthält Q den Wert 1.41421356237, nicht aber die Rechenvorschrift SQR(2). Der Aufruf von Q bewirkt also nur das Präsentieren des Zahlenwertes und keine erneute Berechnung! Wenn Sie festlegen: Q=A+B, dann nimmt Q den Wert der Summe der augenblicklichen Werte von A und B an. Die Zuordnung bewirkt nicht, daß Q immer und ewig den Wert von A+B hat. Wenn das so wäre, würde jede Änderung von A und/oder B auch Q beeinflussen. Das könnte große Verwirrung stiften und würde den Aufruf von Q erheblich langsamer machen, weil dann erst andere Werte aufgerufen und zu Q verknüpft werden müßten.

Wenn Sie wirklich wollen, daß sich der Wert einer Variablen in Abhängigkeit von anderen Variablen ändert, dann läßt sich das leicht mit den "benutzerdefinierten Funktionen" erreichen, die Sie im BASIC-Teil kennenlernen werden. Solche Funktionen arbeiten auch im CALC-Modus!

Für die Variablen können alle Buchstaben des Alphabets benutzt werden. Wenn das noch nicht reicht, kann auf den Buchstaben noch eine Ziffer von 0 bis 9 folgen. Variable lassen sich also korrekt mit L, U, K oder E, aber auch mit C3, P0, R2 oder D2 oder gleich aufgebauten Kombinationen bezeichnen. DD ist dagegen nicht zulässig, weil es zwei Buchstaben enthält. Der Rechner wird solche Versuche mit einer Warnung quittieren. Ebenso wird er auf 3M reagieren, weil hier der Name mit einer Ziffer beginnt und auch T42 ist ihm nicht recht, weil zwei Ziffern folgen. Sie müssen sich also auf (11*26 =) 286 Variable "beschränken", denen Sie die von Ihnen benötigten Werte zuordnen können.

Zwar lassen sich auch indizierte Variable, wie X(8) im CALC-Modus verwenden, diese beanspruchen aber viel Speicherplatz und sind unbequem beim Aufruf. Über diese Variablen-Art werden wir uns besser im BASIC-Teil dieses Buches unterhalten.

Wenn Sie das Resultat einer vorangegangenen Berechnung (nachträglich) einer Variablen zuordnen wollen, brauchen Sie nicht etwa X=RES oder X=() einzugeben: Die Eingabe von X= löscht das Ergebnis (links in der Anzeige) und END LINE ruft dann bekanntlich RES automatisch auf, womit die Zuordnung erfolgt ist! Versuchen Sie es: 1+2 END LINE (setzt RES auf 3, Anzeige links): 3. Wenn Sie jetzt X= END LINE eingeben, erscheint links: 3

Jetzt ist 3 der Variablen X zugeordnet, wie Sie durch die folgende Probe leicht feststellen können: X*2+X/3 END LINE ergibt die Anzeige: 7

Beim Abarbeiten von schwierigen Berechnungen ist es gut, alle diese Dinge zu kennen.

Umgang mit negativen Zahlen

Wie würden Sie die Aufgabe "zwölf mal minus 5" eingeben? Sicher als 12*(-5) END LINE, aber das sind ein bis zwei überflüssige Tastenbetätigungen! Leichter geht es so:

12*-5 END LINE (ergibt Anzeige):

-60

Zugegeben, 12*-5 sieht ungewohnt aus, aber wenn's geht, warum nicht? Beim Teilen ist es ähnlich, nur sieht das nicht so seltsam aus: 12/-5 findet jeder fast normal.

Das große ABER kommt jetzt!!! (Und das ist wirklich tückisch!): Versuchen Sie nie, mit dieser "schlampigen" Schreibweise eine Zahl zu einer negativen Potenz zu erheben! Das geht schief, wie der Versuch zeigt:

12^- (ergibt Anzeige):

WRN:Illegal Context

Wenn Sie diese Meldung überhaupt nicht oder nur zu kurz sehen konnten, drücken Sie ERRM (das ist blau auf der SPC-Taste). Dann ist die Warnung für die Dauer des Tastendrucks sichtbar! Eine sehr nützliche Taste!

Ein negativer Wert für den Exponenten muß in Klammern gesetzt werden:

12^(-5 END LINE (ergibt Anzeige):

4.01877572106E-6

Diese ungewohnte Art der Anzeige weist darauf hin, daß der errechnete Wert zu klein ist, um sich im üblichen Anzeigeformat mit voller Genauigkeit darstellen zu lassen. Der Rechner geht von sich aus zur wissenschaftlichen Schreibweise der Zahlen über. Der Wert ist 4.01877572106 geteilt durch 10^6 (was man üblicherweise als 1 Million bezeichnet). Gewiß, diese Zahl ließe sich als

0,00000401877572016

darstellen, aber mit so langen Zahlen kann der 'arme' HP-71 nun mal nicht umgehen! 12^(-5) ist nun mal so klein, weil es das gleiche ist, wie 1/(12^5).

Wenn Sie wissen wollen, weshalb man negative Zahlen ohne Klammer nur als Faktor oder Divisor benutzen darf, nicht jedoch als Exponent, dann müssen wir uns ansehen, wie der HP-71 mathematische Probleme bearbeitet. Hier ein erster Einblick:

Wie müßten Sie denn die Aufgabe **1+2*3^4=?** mit Papier und Bleistift lösen? Sicherlich doch so, daß Sie zuerst **3^4** ausrechnen (**81**), dann diesen Wert mit **2** malnehmen (**162**) und schließlich noch die **1** addieren (Ergebnis **163**).

Haben Sie bemerkt, daß Sie diese Aufgabe nicht in der üblichen Richtung, d.h. von links nach rechts, bearbeitet haben? Warum nahmen Sie die entgegengesetzte Richtung? Sicher doch deshalb, weil Sie mal gelernt haben, daß Potenzierungen Vorrang vor Multiplikationen und diese wiederum Vorrang vor Additionen haben. Diese zeitliche Rangordnung im Ausführen von Rechenoperationen wird "algebraische Hierarchie" genannt.

Auch im HP-71 ist eine solche Hierarchie festgelegt, mit der Sie vertraut sein sollten, wenn Sie den Rechner optimal verwenden wollen. In der folgenden Übersicht sind die Operationen nach ihrem Rang geordnet:

Die algebraische Hierarchie des HP-71

1.) **Klammern.** Der Klammerinhalt wird zuerst errechnet, verschachtelte Klammern werden von innen nach außen aufgelöst.

2.) **Funktionen** (SQR, SIN, MOD u.ä.).

3.) **Exponenten (^).**

4.) **Negatives (-), positives (+) Vorzeichen** und **Negation (NOT).**

5.) **Multiplikation (*), Division (/), Prozent (%)** und **ganzzahlige Division (\)** bzw. **(DIV).**

6.) **Addition (+)** und **Subtraktion (-).**

7.) **kleiner als (<), gleich (=), größer als (>), ungleich (#)** und **ungeordnet (?)** und **sämtliche Kombinationen** daraus.

8.) **Logisches UND (AND).**

9.) **Logisches Inklusiv-ODER (OR)** und **Logisches Exklusiv-ODER (EXOR).**

Merken Sie, daß damit das Dilemma des negativen Exponenten bereits erklärt ist? Nun, 12*-5 klappt ohne weiteres, weil die Vorzeichenumkehr Vorrang vor der Multiplikation hat: der Rechner negiert erst die 5, erhält also 12*(-5) und multipliziert dann.

Aber, Exponenten haben Vorrang vor der Vorzeichenumkehr! Deshalb geht es mit 12^-5 so gründlich schief! 12^-5 gibt für den Rechner genau so viel "Sinn" wie 12^*5, nämlich"Blödsinn"! Deshalb auch die Warnung, die im CALC-Modus schon die Eingabe unterbricht.

Der Exponent bekam Vorrang vor den Vorzeichen, damit Ausdrücke wie -12^4 richtig berechnet werden. Dieser Ausdruck besagt (schon immer, auch vor dem Zeitalter der Computer!) nicht, daß -12 in die vierte Potenz erhoben werden soll, sondern, daß die vierte Potenz von 12 ein negatives Vorzeichen erhalten soll! Probieren Sie doch: -12^4 ergibt -20736, (-12)^4 dagegen +20736, das ist doch schon ein Unterschied. Auch das positive Vorzeichen ist in der Rangordnung aufgeführt (es muß ja mindestens erwähnt werden), es ist aber eine völlig überflüssige Funktion! Die Eingabe von 12*+5 ist zwar korrekt, wir können das positive Vorzeichen ebensogut weglassen. Sie werden erleben, daß überflüssige positive Vorzeichen im CALC-Modus garnicht erst angenommen werden, und der Rechner diese auch aus BASIC-Anweisungen freundlicherweise entfernt, was Sie aber erst beim nächsten Auflisten der Anweisung bemerken können!

Funktionen mit mehreren Argumenten

Die meisten Funktionen wie LOG, SIN, FACT usw. benötigen eine einzige Zahl als Eingabe. So z.B. die Eingabe von:

FACT(14 END LINE (ergibt Anzeige):

87178291200

Berechnet wurde die Fakultät von 14 (geschrieben 14!). Die FACT-Funktion nahm den Wert 14 und gab 14! dafür zurück, eine sehr große Zahl, wie wir sehen! Den Wert, den wir (in Klammern) der Funktion anbieten, nennt man "Argument" (im Beispiel die 14).

Einige Funktionen brauchen zwei oder mehr Argumente, wobei die Reihenfolge der Argumente sehr wichtig ist.

Die RMD-Funktion führt z.B. (intern) eine ganzzahlige Division durch und gibt nur den "Rest" als Antwort zurück. Nehmen wir an, daß eine große Pizza in 15 Teile geschnitten wurde. Der Teller geht in einer Runde von vier Studenten einige Male reihum, jeder nimmt sich immer nur ein Stück, bis die Zahl der restlichen Stücke nicht mehr für die nächste Runde reicht. Wieviel Pizza-Stücke liegen dann noch auf dem Teller? Es ist der Rest, der übrig bleibt, wenn 15 durch 4 geteilt wird und man keine Nachkommastellen zuläßt. Mit dem HP-71-BASIC (auch im CALC-Modus) sieht das so aus:

RMD(15,4 END LINE (ergibt Anzeige):

3

Es sind also drei Pizza-Stücke übriggeblieben, vier arme, noch hungrige Studenten sitzen drum herum und schauen zu, wie der Rest kalt wird! Sie gehen dann in ihr Kolleg, um nachzudenken, wie dieses hochgeistige Problem zu lösen ist!

Beachten Sie, daß wir RMD(15,4) geschrieben haben! Die Reihenfolge der Argumente ist wichtig! Versuchen Sie es doch mal mit RMD(4,15) und warten, was passiert: Es wird nun der Rest der Division 4:15 errechnet! Das ergibt 4.

Der Befehls-Stack

Wie wir gesehen haben, können wir im CALC-Modus mit der AUF-Taste die momentane Berechnung wiederholen bzw. bearbeiten. Tatsächlich wird dabei das unterste "Fach" des Befehls-Stacks angezeigt, in dem der vorige Inhalt der Anzeige abgelegt ist. Jedesmal, wenn Sie im CALC-Modus zur END LINE-Taste greifen, wird nicht nur RES auf den aktuellen Wert gebracht, sondern auch der Befehls-Stack beeinflußt. Ihre gesamte Rechnung, wie sie auch aussehen mag, wird in den Befehls-Stack übernommen und dort in das unterste Fach gesteckt, wodurch die Inhalte aller Fächer um eine Etage angehoben werden. Natürlich ist der Befehls-Stack nicht beliebig groß, irgendwann ist er voll! Beim erneuten Anheben schwappt der Inhalt des obersten Faches über und fällt in ein gefräßiges, schwarzes Loch, das sich von Befehlen ernährt, deren Aufbewahrung nicht mehr sinnvoll erscheint.

Wenn Sie nun die AUF-Taste mehrmals betätigen, werden Sie einige der vorher gemachten Eingaben wiederfinden (die ältesten zuletzt), bis Sie an das hungrige, schwarze Loch kommen (Vorsicht, fallen Sie da nicht selbst hinein!)

Sie können sich diese älteren Befehle nicht nur ansehen, sondern auch erneut (auch mit geänderten Werten) ausführen. Dazu gehen Sie mit den AUF- und AB-Tasten zur gewünschten Anweisung (führen mit den LINKS-, RECHTS-Tasten, mit BACK und mit ersetzendem oder einfügendem Cursor eventuelle Änderungen durch) und drücken die END LINE-Taste. Die Anweisung wird dann nicht nur ausgeführt, sondern gleichzeitig auch in das unterste Fach des Befehls-Stacks getan, wie es für Sie bequem und zweckmäßig ist. Durch die Änderungsmöglichkeit ersparen Sie sich die Neu-Eingabe der ganzen Anweisung.

Der Befehls-Stack des HP-71 hat für gewöhnlich fünf Fächer. Es besteht aber die Möglichkeit, den Stack bis auf 16 Fächer zu vergrößern! Die Theorie darüber und ein geeignetes Programm dazu finden Sie weiter hinten in diesem Buch. Vorher müssen Sie aber erst noch BASIC lernen! Das ist die nächste größere Aufgabe.

Wenn Sie erst gewohnt sind, den Befehls-Stack zu benutzen, werden Sie merken, daß der HP-71 im CALC-Modus den beiden bisher angewandten Rechner-Prinzipien überlegen ist. Nach einer kurzen Übungszeit werden Sie ihre Berechnungen wie im Fluge erledigen, so schnell, wie Sie es sich vorher kaum vorstellen konnten: Hier ist alles sichtbar, es gibt keine verborgenen Werte. Alles ist logisch, keine bizarre Mischung, bei der die Argumente mal vor und mal hinter den Funktionen einzugeben sind (wie bei den AOS-Rechnern), aber auch keine (klare aber ungewohnte) neue Logik, wie bei den UPN-Rechnern.

Sind Sie jetzt bereit, einige handfeste Beispiele durchzurechnen?

Anwendungsbeispiele aus der Praxis

Beispiel 1:
Farmer Silas liefert Tomaten an eine Konservenfabrik: Am Montag fährt er zweimal und liefert 25 und 27 Tonnen. Am Dienstag liefert er weitere 19 und 23 Tonnen in zwei Fuhren. Für diese Lieferungen war ein Preis von 55 $ pro Tonne vereinbart, die Fabrik drückt den Preis jedoch um 2%, weil die Ware Mängel aufwies. Am Mittwoch kann Farmer Silas weitere 26 und 28 Tonnen liefern, für die er 57,50 $ pro Tonne erwartet. Auch bei dieser Lieferung werden Mängel festgestellt, die den Preis um 3% verringern. Welchen Betrag kann Farmer Silas für seine Lieferungen von der Fabrik erwarten?

Schreiben wir das Problem zuerst in algebraischer Form hin:

Wert der Lieferungen von Mo. u. Di.:	M=55*(25+27+19+23)
Abzug bei diesen Lieferungen:	2%M
Wert der Lieferungen vom Mi.:	N=57.5(26+28)
Abzug bei dieser Lieferung:	3%N
Gesamtsumme:	M-2%M+N-3%N

Dann ist die Berechnung in drei Zeilen in den Rechner einzugeben und nach Ende jeder Zeile mit END LINE abzuschließen, was von nun an durch "--->" ausgedrückt wird:

M=55*(25+27+19+23) ---> (erwartete Summe für Mo. u. Di.)
N=57.5*(26+28) ---> (erwartete Summe für Mi.)
M-2%M+N-3%N ---> 8058.45 Soviel Dollar zahlt die Fabrik!

Beispiel 2:
Ingenieur P. Rintplatte hat gemessen, daß ein RC-Glied einen Scheinwiderstand von 77,8 Ohm hat und daß die Spannung dem Strom um 36,5 Grad nacheilt. Welche Werte haben Wirk- und Blindwiderstand dieser Reihenschaltung?

Die Lösung benutzt die zwei Formeln R=Z*cos(a) und X=Z*sin(a), wobei Z der Scheinwiderstand und a der Winkel zwischen Strom und Spannung ist.
Dann ist die Berechnung in vier Zeilen in den Rechner einzugeben und nach Ende jeder Zeile END LINE zu drücken:

Z=77.8 ---> (Eingabe der Impedanz)
A=-36.5 ---> (Eingabe des Winkels)
Z*COS(A) ---> **62.54** (Wirkwiderstand in Ohm)
Z*SIN(A) ---> **-46.28** (Blindwiderstand in Ohm)

Wenn der HP-71 mit dem Math Pac ROM bestückt ist, bietet sich ein alternativer, kürzerer Lösungsweg, weil dieses ROM die unmittelbare Umwandlung von Polar-Koordinaten in rechtwinklige Koordinaten (und umgekehrt) ermöglicht:

RECT((77.8,-36.5)) ---> (62.54,-46.28)

Beispiel 3:
Fünf Studenten erzielten bei einem Test 95%, 90%, 88%, 94% und 93% der maximal möglichen Punkte. Welcher Durchschnitt ergibt sich daraus?

Lösung:
Durchschnitt = Summe der Einzeldaten/Anzahl der Einzeldaten.

Es ist also einzugeben:

(95+90+88+94+93)/5 ---> 92

Ergebnis: Der Durchschnitt liegt bei 92%.

Beispiel 4:
Sie wollen die Summe aller ganzen Zahlen von 1 bis 50 ermitteln und haben die Formel "vergessen". Anstatt nun die Formel zu suchen, greifen Sie nach dem HP-71 und fangen an, die Werte im CALC-Modus zu addieren 1+2+3+5+6....

Dummerweise, als Sie eben die Zahl 36 eingeben wollen, piepst der HP-71, zeigt "Line Too Long" an, wenn Sie ihn mit ERRM nach seinen Sorgen fragen, und scheint den Dienst einstellen zu wollen.

Der HP-71 kann nur Zeilen mit maximal 96 Zeichen annehmen, das gilt auch für den CALC-Modus, auch wenn die "Zeile" Ihnen viel kürzer erscheint! Aber Sie wissen doch, daß die Zeile in der eingegebenen, unausgerechneten Form auch im Befehls-Stack Platz finden muß....

Wenn so etwas eintritt, müssen Sie eben Zwischenergebnisse in RES abspeichern, wie das nachstehend vorgeführt wird:

```
1+2+3+4+5+6+7+8+9+10  --->  55
()+11+12+13+14+15+16+17+18+19+20  --->  210
()+21+22+23+24+25+26+27+28+29+30  --->  465
()+31+32+33+34+35+36+37+38+39+40  --->  820
()+41+42+43+44+45+46+47+48+49+50  ---> 1275
```

Ergebnis: 1+2+3+...+48+49+50=1275

Die alternative Lösung setzt voraus, daß Ihnen plötzlich die Formel wieder "einfällt": Summe der ganzen Zahlen von 1 bis N ist (N+1)*N/2, was sich so eingeben läßt:

(50+1)*50/2 ---> 1275

Das ist doch wesentlich einfacher! Immer, wenn ihnen "Line Too Long" entgegengehalten wird, müssen Sie einen mühsamen Weg gehen oder nach einer Lösung suchen, die sich kürzer eingeben läßt.

Glauben Sie mir, der HP-71 tut trotz seiner Nöte mit "Line Too Long" sein Bestes für Sie, um Ihnen Zeit zu sparen! Er zeigt Ihnen den Beginn der langen Zeile, nun aber als Inhalt des untersten Faches des Befehls-Stacks. Mit der RECHTS-Taste können Sie sich schnell oder auch langsam die übrigen Teile der Zeile ansehen und dann den richtigen Entschluß fassen: Wenn Sie END LINE drücken, wird das Zwischenergebnis in RES gespeichert und kann dann mit () zur Fortsetzung der Rechnung aufgerufen werden.

Beispiel 5:
Sie wollen für die Funktion y=7x^5-12x^4+54x^3-22x^2+11x-1 die Werte für x=0.1 und x=0.2 berechnen. Bisher benutzten Sie dazu einen Rechner mit UPN-Logik, der sich gut für das Horner-Schema eignete. Dieses Rechenverfahren wählten Sie, um (langsame) Potenz-Operationen und Rundungsfehler zu vermeiden. Ist es Ihnen nicht klar, wie der HP-71 das lösen kann?

Ganz einfach! Auch mit dem Horner-Schema! Sie schauen nach dem höchsten Exponenten (5) und setzen 5-1 Klammern, dann den Koeffizienten, der zur höchsten Potenz gehörte, mit dem einfachen Argument, fügen den Koeffizienten der zweithöchsten Potenz an und schließen die innerste Klammer. Dann folgt als Faktor wieder das Argument, dann der Koeffizient der zweithöchsten Potenz usw...:

```
X=0 END LINE------>  (setzt X auf 0)
((((7*X-12)*X+54)*X-22)*X+11)*X-1 END LINE ---> -1
X=.1 END LINE----->  (setzt X auf .1)
AUF AUF END LINE-->  -.06713
X=.2 END LINE----->  (setzt X auf .2)
AUF AUF END LINE-->   .73504
```

Ergebnis: f(0)=-1; f(0.1)=-0.06713; f(0.2)=0.73504

Merken Sie es? Wir haben den Befehls-Stack hier recht raffiniert benutzt: Der Variablen X werden nacheinander die Werte 0, 0.1 und 0.2 zugeordnet. Die Formel selbst wird nur einmal komplett eingegeben und dann immer wieder aus dem Befehls-Stack aufgerufen. Wenn Sie eine Berechnung mehrere Male mit veränderten Eingaben durchführen müssen, ist das die richtige Methode. Sie werden bald nicht mehr ohne den Befehls-Stack rechnen wollen!

Beispiel 6:
Der Raumgestalter Derek Lobos hat in seinem Wohnzimmer vier Bilderhaken und besitzt sieben gerahmte Gemälde. Damit sein Wohnraum von Zeit zu Zeit verändert erscheint, arrangiert er jeden Monat die Gemälde neu, wobei er immer eines der Gemälde gegen ein anderes aus seinem Vorrat austauscht. Das ergibt viele Möglichkeiten! Wie lange dauert es, bis alle möglichen Anordnungen aufgetreten sind und Derek Lobos sich ein neues Bild kaufen muß?

Lösungsweg:
Die mögliche Zahl der Anordnung von x Gegenständen, von denen immer nur y gleichzeitig gezeigt werden, ist x!/(x-y)!. Das Ausrufungszeichen ist dabei das mathematische Zeichen für die Fakultät-Funktion. Weil Derek Lobos die Anordnung seiner Bilder jeden Monat einmal neu arrangiert, ist das Ergebnis noch durch 12 zu teilen, um die Zahl der Jahre zu ermitteln:

FACT(7)/(FACT(7-4)*12) END LINE----> 70

Ergebnis: Derek Lobos ist für die nächsten 70 Jahre versorgt!

Als alternative Lösung könnten Sie hier auch gut eine selbstdefinierte Funktion verwenden.

Beispiel 7:
An der Front des Fonch-Hotels in San Placebo hat sich das obere Ende eines Regenfallrohres (7 m lang) von der Regenrinne gelöst und sich in ganzer Länge (Drehpunkt am Boden) gegen das gegenüberstehende Gebäude der Clutch Cargo Bank gelegt. Zum Glück ist das Rohr weder gebrochen noch verbogen und sehr stabil. Die Breite der Straße beträgt hier 5.5 m (von Wand zu Wand gemessen). Um ein Feuer zu bekämpfen, das im Fonch-Hotel ausgebrochen ist, muß die Feuerwehr mit einem Fahrzeug von 2.2 m Höhe und 2.45 m Breite unter dem Rohr durchfahren. Kann das ohne Berührung zwischen Rohr und Fahrzeug geschehen?

Lösungsweg:
Das Rohr bildet die Hypotenuse in einem rechtwinkligen Dreieck, dessen eine Kathete durch die Straßenbreite festgelegt ist. Dank Pythagoras können wir dann die andere Kathete ausrechnen, nämlich die Höhe, in der das Rohr die Wand des Bankgebäudes berührt. Durch eine Proportion können wir berechnen, welche lichte Höhe das Rohr in 2.45 m Abstand von der Mauer der Bank freigibt. Dieser Wert entscheidet, ob das Fahrzeug drunter weg kann, ohne das Rohr zu berühren.

W=SQR(6.8^2-5.5^2) END LINE----> 3.9987 (gerundet)
(5.5-2.45)*W/5.5 END LINE------> 2.21 (gerundet)

Ergebnis: Da das Löschfahrzeug nur 2.15 m hoch ist, kann es unter dem Rohr durchfahren, wenn es sich nahe an der Wand der Bank hält. Der Hotelbrand kann gelöscht werden und alle leben glücklich weiter!

Im CALC-Modus benutzbare Funktionen

Auch im CALC-Modus kann der HP-71 auf eine große Zahl von Funktionen und Befehlen reagieren. Um die Möglichkeiten, die der CALC-Modus bietet, gut kennenzulernen, sollten Sie diese Liste aufmerksam lesen und alle Beispiele ausprobieren. Nur wenn Sie diese Befehle und Funktionen alle kennen, dürfen Sie behaupten, den CALC-Modus zu beherrschen! Wenn Ihr Rechner außerdem noch ein Math Pac ROM oder andere Einschübe enthält, stehen Ihnen weitere Funktionen (auch im CALC-Modus) zur Verfügung. Schauen Sie in diesem Fall in die zu den Einschüben gehörenden Kurz-Anweisungen, um Informationen und Beispiele zu finden!

+ Veranlaßt die Addition von zwei Zahlen.
1+2 ergibt 3

- Veranlaßt die Subtraktion der rechten Zahl von der linken oder kehrt Vorzeichen des unmittelbar folgenden Wertes um.
7-3 ergibt 4
-7 ergibt -7
-7-3 ergibt -10
-7--3 ergibt -4 (-7+3=-4)

***** Veranlaßt die Multiplikation zweier Zahlen.
3*4 ergibt 12
(1+2)*(3*4) ergibt 21 (3 mal 7)
12/2*2 ergibt 12 (12/2, anschließend mal 2)

/ Veranlaßt das Teilen der linken Zahl durch die rechte. Der Wirkungsbereich des Symbols endet bei der nächsten Rechenvorschrift, ausgenommen Klammerpaar, Funktion oder Exponent.
12/2/2 ergibt 3 (12 zweimal durch 2 geteilt)
12/2^2 ergibt 3 (12 geteilt durch 4)
12/2+2 ergibt 8 (12 geteilt durch 2, plus 2)

^ Veranlaßt die Erhebung der linken Zahl in die durch die rechte Zahl bezeichnete Potenz.
2^5 ergibt 32 (fünfte Potenz von 2)
FACT(10)/2^8 ergibt 14175 (10 Fakultät, geteilt durch achte Potenz von 2)
Beachten Sie, daß sich mit ^ auch Wurzeln berechnen lassen:
65536^(1/4) ergibt 16 (vierte Wurzel aus 65536).

% Veranlaßt die Multplikation der linken Zahl mit der rechten mit anschließender Division durch 100 (Wirkung ähnlich wie *).
6%24 ergibt 1.44 (6 Prozent von 24 sind 1.44)
24+6%24 ergibt 25.44 (24 plus 6% von 24)

ABS(X) Ermittelt den Betrag von X (stets >=0).
ABS(-3) ergibt 3
ABS(0) ergibt 0
ABS(5) ergibt 5

ACOS(X) Ermittelt den Winkel, dessen Kosinus X ergibt.
ACOS(.5) ergibt 60 (Beispiele im Grad-Modus)
ACOS(0) ergibt 90
ACOS(-1) ergibt 180

ACS(X) Hat gleiche Wirkung wie ACOS(X)

AF Veranlaßt die Anzeige des momentanen Korrekturfaktors der Uhr.

AF(X) Setzt den Korrekturfaktor der Uhr auf den in der Klammer stehenden Wert und bringt den bis daher gültigen Wert in die Anzeige
AF(1537) setzt 1537 als neuen Korrekturfaktor und bringt den bisherigen Wert in die Anzeige.

AND Unterwirft zwei Zahlen der logischen UND-Verknüpfung.
0 AND 0 ergibt 0
0 AND 2 ergibt 0 (2 als Beispiel für eine Zahl # 0)
3 AND 0 ergibt 0
4 AND 5 ergibt 1
AND prüft also, ob beide Werte # Null sind.

ANGLE(X,Y) Ermittelt den Winkel zwischen positiver X-Achse und der Strecke vom Ursprungspunkt zum Punkt (X,Y)
ANGLE(3,4) ergibt 53.13 (Beispiele im Grad-Modus)
ANGLE(.5,SQR(3)) ergibt 60
ANGLE(-1,0) ergibt 180

ASIN(X) Ermittelt den Winkel, dessen Sinus X ergibt.
ASIN(.5) ergibt 30 (Beispiele im Grad-Modus)
ASIN(0) ergibt 0
ASIN(-1) ergibt -90

ASN(X) Hat gleiche Wirkung wie ASIN(X)

ATAN(X) Ermittelt den Winkel, dessen Tangens (Anstieg) X ergibt.
ATAN(1) ergibt 45 (Beispiele im Grad-Modus)
ATAN(SQR(3)) ergibt 60
ATAN(INF) ergibt 90

ATN(X) Hat gleiche Wirkung wie ATAN(X)

CEIL(X) Ermittelt die kleinste ganze Zahl, die >=X ist.
CEIL(-1.7) ergibt -1
CEIL(-1) ergibt -1
CEIL(1) ergibt 1
CEIL(1.7) ergibt 2

CLASS(X) Ermittelt den Klassen-Code des Wertes in X.
CLASS(INF) ergibt 4 (Überlauf im Positiven)
CLASS(-INF) ergibt -4 (Überlauf im Negativen)
CLASS(MINREAL) ergibt 1 (Unterlauf im Positiven)
CLASS(-MINREAL) ergibt -1 (Unterlauf im Negativen)

CORR(X,Y) Ermittelt den Korrelations-Koeffizienten.
CORR(2,4) ermittelt die Korrelation der Variablen #2 und #4 des momentanen statistischen Feldes. Das Feld muß existieren und Daten enthalten.
Das Anlegen des Feldes ist im CALC-Modus nicht möglich!

COS(X) Ermittelt den Kosinus des Winkels X.
COS(60) ergibt .5 (Beispiele im Grad-Modus)
COS(180) ergibt -1
COS(0) ergibt 1

DATE Ermittelt das aktuelle Datum im JJTTT-Format (JJ=Jahreszahl, TTT=laufende Tageszahl).

DEG(X) Wandelt den Winkel X (im Bogenmaß) in Winkelgrade um.
DEG(PI) ergibt 180 (2*Pi = Vollkreis = 360 Grad)

DIV Veranlaßt das Teilen der linken Zahl durch die rechte, gibt nur ganzzahligen Teil des Quotienten zurück.
12 DIV 5 ergibt 2 (12:5=2 Rest 2)
DATE DIV 1000 ergibt die aktuelle Jahreszahl.
Achtung: das Zeichen "\" (ASCII-Code 92) kann an Stelle von DIV benutzt werden und (als Eingabehilfe) einer Taste zugewiesen werden.

DVZ Gibt -7 aus, die Nummer des Flags, das beim Versuch der Division durch Null gesetzt wird.
DVZ ergibt -7
FLAG(DVZ) ergibt 1, wenn eine Division durch Null versucht wurde, sonst 0
FLAG(DVZ,0) bewirkt gleiches, löscht jedoch zusätzlich das DVZ-Flag
FLAG(0,FLAG(DVZ)) übernimmt Zustand vom DVZ-Flag in das Flag 0

TRAP(DVZ) ergibt 0 oder 1 oder 2 als Kennzeichen der momentan vereinbarten Fehlerbehandlung beim Versuch der Division durch Null
TRAP(DVZ,2) wirkt ebenso, setzt aber das Kennzeichen für die Fehlerbehandlung von nun an auf 2

EPS Bezeichnet 1E-499, den kleinsten positiven Wert, den der Rechner normalisiert darstellen kann.
ABS(CLASS(X)=2)=(ABS(X)<EPS) ergibt 1, weil es sich um einen positiven Wert handelt, der zwischen EPS und Null liegt, nur denormalisiert darstellbar ist und deshalb der Klasse 2 angehört.

ERRL Ermittelt in einem BASIC-Programm die Nummer der Zeile, die den jüngsten Fehler auslöste.
ERRL ergibt 0, wenn das BASIC-Programm keinen Fehler ausgelöst hat.

ERRN Ermittelt die Nummer des jüngsten Fehlers.
ERRN ergibt 4, wenn TAN(90) ausgeführt werden sollte, weil 4 die Fehlernummer für TAN=INF ist.
ERRN ergibt 2, wenn ASIN(2) ausgeführt werden sollte.

EXOR Unterwirft zwei Zahlen der logischen Exklusiv-ODER-Verknüpfung.
0 EXOR 0 ergibt 0
0 EXOR 1 ergibt 1 (1 als Beispiel für eine Zahl # 0)
1 EXOR 0 ergibt 1
1 EXOR 1 ergibt 0
EXOR ist also nur dann wahr, wenn lediglich eine der beiden Größen existiert.

EXP(X) Ermittelt den Wert von e (2.71828182816...) erhoben in die Potenz X.
EXP(1) ergibt e (Näherung auf 12 stellen)
EXP(27.6310211159) ergibt 99999999971

EXPM1(X) Im Prinzip gleich EXP(X)-1, für kleine Funktions-Werte jedoch genauer:
EXPM1(PI/1E7) ergibt 3.14159314707E-7 (12 signifikante Stellen)
EXP(PI/E7)-1 ergibt .00000031416 (nur 5 signifikante Stellen)

EXPONENT(X) Ermittelt die (dekadische) Größenordnung von X.
EXPONENT(X) ist zuverlässiger, als IP(LOG10(X)).
EXPONENT(10) ergibt 1
EXPONENT(100) ergibt 2
EXPONENT(999) ergibt 2
EXPONENT(1000) ergibt 3
EXPONENT(10/3*3) ergibt 0 (das ist richtig!!!!)
EXPONENT(10*3/3) ergibt 1 (das ist auch richtig!!)
IP(LOG10(10/3*3)) ergibt 1 (das ist falsch!)
IP(LOG10(10*3/3) ergibt 1 (das ist wieder richtig!)
Um diese scheinbaren Widersprüche zu verstehen, ist es notwendig, die Zwischenergebnisse zu betrachten:

10*3/3 ergibt 10, aber 10/3*3 ergibt 9.99999999999

FACT(X) Ermittelt die Fakultät von X für ganzzahlige Argumente (X>=0).
FACT(6) ergibt 720 d.h. 6*5*4*3*2*1
FACT(8) ergibt 40320 d.h. 8*7*6*5*4*3*2*1
FACT(0) ergibt 1 (per Definition!)

FLAG(X) Ermittelt den Zustand (0 oder 1) vom Flag X.
FLAG(0) bringt den Wert vom Flag 0 in die Anzeige
FLAG(INX) bringt den Wert vom Unexakt-Flag in die Anzeige
FLAG(-3) bringt den Wert vom Dauer-Einschalt-Flag in die Anzeige

FLAG(X,Y) Ermittelt den Zustand vom Flag X und setzt Flag X anschließend auf Wert Y.
FLAG(2,0) schaltet akustische Signale ab.
FLAG(2,1) schaltet akustische Signale wieder an
FLAG(-10,1) schaltet Winkel-Modus auf Bogenmaß
FLAG(-10,0) schaltet Winkel-Modus auf Grad

FLOOR(X) Ermittelt die größte ganze Zahl, die <=X ist.
FLOOR(-1.7) ergibt -2
FLOOR(-1) ergibt -1
FLOOR(1) ergibt 1
FLOOR(1.7) ergibt 1

FNvar Vom Anwender definierte Funktion, wobei "var" irgendein gültiger Name für eine Variable ist.

FP(X) Ermittelt den nicht-ganzzahligen Teil von X.
FP(-1.7) ergibt -.7
FP(-1) ergibt -0
FP(1) ergibt 0
FP(1.7) ergibt .7

INF Bringt das Symbol für "unendlich" (im mathematischen Sinn) in die Anzeige.
INF ergibt Inf
SQR(INF) ergibt Inf

1/0 ergibt mit TRAP(DVZ,0) Fehlermeldung und wird nicht ausgeführt
1/0 ergibt mit TRAP(DVZ,1) Warnung und die Anzeige 9,99999999999E499
1/0 ergibt mit TRAP(DVZ,2) Warnung und die Anzeige Inf

FACT(255) ergibt mit TRAP(OVF,0) Fehlermeldung und wird nicht ausgeführt
FACT(255) ergibt mit TRAP(OVF,1) Warnung und die Anzeige 9,99999999999E499
FACT(255) ergibt mit TRAP(OVF,2) Warnung und die Anzeige Inf

INT(X) Ermittelt die größte ganze Zahl, die <=X ist (gleiche Wirkung wie FLOOR).

INX Gibt -4 aus, die Nummer des Flags, das bei unexaktem Ergebnis gesetzt wird.
Hinsichtlich der Anwendung siehe DVZ!

IP(X) Ermittelt den ganzzahligen Teil von X.
IP(-1.7) ergibt -1
IP(-1) ergibt -1
IP(1) ergibt 1
IP(1.7) ergibt 1

IVL Gibt -8 aus, die Nummer des Flags, das durch unerlaubte Operationen gesetzt wird.
Hinsichtlich der Anwendung siehe DVZ!

KEYDOWN Gibt eine 1 zurück, sobald eine Taste betätigt wurde, sonst eine 0.

LET Ordnet einen Wert einer Variablen zu. Anwendung ist freigestellt!
LET X=1+2+3+4 oder auch
X=1+2+3+4 ergibt 10 und besetzt X mit diesem Wert
LET F3=FLAG(3) oder auch F3=FLAG(3) besetzt die Variable F3 mit dem Zustand des Flags 3
LET N(3,4)=153 oder auch N(3,4)=153 setzt 153 auf den Platz in der 3. Reihe und der 4. Spalte des Variablenfeldes N

LGT(X) Ermittelt den dekadischen Logarithmus (Basis 10) von X.
LGT(10) ergibt 1
LGT(100) ergibt 2
LGT(1000) ergibt 3
LGT(10^5) ergibt 5

LN(X) Ermittelt den natürlichen Logarithmus (Basis e) von X.
LN(10) ergibt 2.30258509299
LN(100) ergibt 4.60517018599
LN(1000) ergibt 6.90775527898
LN(EXP(5)) ergibt 5

LOG(X) Gleiche Wirkung, wie LN(X). Mit LOG(X) wird nicht der Logarithmus zur Basis 10 errechnet!

LOGP1(X) Bedeutet nur LOG(X+1), für kleine Funktionswerte jedoch genauer:
LOGP1(PI/1E12) ergibt 3.14159265359E-12 (12 signifikante Stellen)
LOG(PI/1E12+1) ergibt 0 (das ist falsch!)

LOG10(X) Gleiche Wirkung, wie LGT(X).

MAX(X,Y) Wählt den größeren der beiden Werte aus.
MAX(2,3) ergibt 3
MAX(3,2) ergibt 3 (die Reihenfolge hat keinen Einfluß)
MAX(-7,-8) ergibt -7
MAX(A,MAX(B,C)) gibt den größten Wert der Gruppe A, B, C zurück
MAX(X,0) ergibt das gleiche, wie X*(X>0)

MAXREAL Bezeichnet die größte positive endliche Zahl, die der Rechner darstellen kann.
MAXREAL ergibt 9.99999999999E499

MEAN Ermittelt den Durchschnitt der für statistische Zwecke eingegebenen Werte. Im momentanen Statistik-Feld ermittelt MEAN den Durchschnitt der ersten Variablen dieses Feldes. Das Feld muß mit Daten existieren.
Das Anlegen des Feldes ist im CALC-Modus nicht möglich!

MEAN(X) Ermittelt den Durchschnitt der Variablen X. Näheres siehe MEAN.

MEM Ermittelt die Zahl der unbenutzten Bytes im Hauptspeicher.

MEM(X) Ermittelt die Zahl der unbenutzten Bytes im Port X (sinnlos für Ports, die mit ROM's besetzt sind!).

MIN(X,Y) Wählt den kleineren der beiden Werte aus.
MIN(2,3) ergibt 2
MIN(3,2) ergibt 2 (die Reihenfolge hat keinen Einfluß)
MIN(-7,-8) ergibt -8
MIN(A,MIN(B,C)) gibt den kleinsten Wert der Gruppe A, B, C zurück
MIN(X,0) ergibt das gleiche, wie X*(X<0)

MINREAL Bezeichnet die kleinste positive Zahl, die der Rechner überhaupt darstellen kann. Diese Zahl ist nicht normalisiert!
MINREAL ergibt mit TRAP(UNF,0) Fehlermeldung und wird nicht ausgeführt
MINREAL ergibt mit TRAP(UNF,1) Warnung und die Anzeige 0
MINREAL ergibt mit TRAP(UNF,2) Warnung und Anzeige 0.00000000001E-499

MOD(X,Y) Ermittelt X modulo Y (als X-Y*INT(X/Y) definiert).
MOD(X,Y) ergibt immer einen Wert zwischen 0 und Y
MOD(12,5) ergibt 2 (den Rest nach der Teilung 12/5)
MOD(-12,5) ergibt 3 (weil -2 noch nicht zwischen 0 und 5 liegt!)
MOD(12,-5) ergibt -3 (weil 2 noch nicht zwischen 0 und -5 liegt!)

NAN Bezeichnet eine Zahl mit unbestimmtem Wert (mit der es sich nicht weiterrechnen läßt!).
INF*0 ergibt mit TRAP(IVL,0) oder TRAP(IVL,1) eine Fehlermeldung und wird nicht ausgeführt
INF*0 ergibt mit TRAP(IVL,2) eine Warnung und die Anzeige NaN

NOT X Ermittelt das Existieren von X und negiert das Ergebnis.
NOT X ergibt 1, wenn X=0 ist
NOT X ergibt 0, wenn X#0 ist
FLAG(0,(NOT FLAG(0)) ergibt bei jeder Ausführung die Änderung des Zustandes des Flags 0

OR Unterwirft zwei Zahlen der logischen ODER-Verknüpfung.
0 OR 0 ergibt 0
0 OR 2 ergibt 1 (2 als Beispiel für eine Zahl #0)
3 OR 0 ergibt 1
4 OR 5 ergibt 1
OR prüft also, ob mindestens einer der beiden Werte von Null verschieden ist.

OVF Gibt -6 aus, die Nummer des Flags, das beim Versuch des Überschreiten des Zahlenbereichs (in positiver oder negativer Richtung) gesetzt wird
Hinsichtlich der Anwendungen siehe DVZ!

PI Gibt 3.14159265359 aus, den auf 12 Stellen gerundeten Wert der Irrationalzahl Pi aus, die dem Verhältnis von Umfang und Durchmesser eines Kreises entspricht.
Weil es sich nur um einen Näherungswert handelt, kann der Rechner (im RAD-Modus) für SIN(PI) nur den Wert -2.06761537357E-13 liefern, wo doch sin(Pi)=0 richtig wäre.

PREDV(X) Ermittelt mit der aus statistischen Daten errechneten Näherungsfunktion den zum Argument X gehörenden Funktionswert.
PREDV(X) benutzt das momentane statistische Feld. Feld, Daten und Näherungs-Funktion müssen existieren.
Anlegen des Feldes und Berechnen der Näherungs-Funktion sind im CALC-Modus nicht möglich!

RAD(X) Wandelt den Winkel X (in Winkelgraden) in das Bogenmaß um.
RAD(180) ergibt 3.14159265359, den Näherungswert für Pi.
(360 Grad = Vollkreis = 2*Pi)

RED(X,Y) Ermittelt den Rest, der sich bei der Division X/Y ergibt so, daß dieser Rest stets zwischen -Y/2 und +Y/2 liegt.
Wenn zwei Werte möglich sind, die sich nur durch das Vorzeichen unterscheiden, wird der Rest ausgegeben, der sich bei Teilung von X durch eine gerade Zahl ergibt.

RED(12,5) ergibt 2
RED(13,5) ergibt -2 (-2 näher an Null als +3)
RED(25,10) ergibt 5
RED(35,10) ergibt -5 (+5 zwar gleichwertig, hat aber ungeraden Teiler!)

RES Ruft den Wert der letzten ausgeführten Berechnung auf. Im CALC-Modus wird RES bei Betätigen der END LINE-Taste jedesmal aktualisiert.
Der RES-Wert läßt sich ebenso durch (), d.h. ein leeres Klammerpaar aufrufen.
13^PI ergibt 3159.04819859
RES^(1/PI) ergibt (nach der vorigen Zeile!) 13

RMD(X,Y) Ermittelt den Rest, der sich bei der Division X/Y ergibt so, daß dieser Rest das gleiche Vorzeichen wie X hat und der Betrag des Restes zwischen 0 und dem Betrag von Y liegt.
RMD(12,5) ergibt 2
RMD(12,-5) ergibt 2
RMD(-12,5) ergibt -2
RMD(-12,-5) ergibt -2

RND Ermittelt eine Zufallszahl zwischen 0 und 1 (exakt zwischen .000000000000 und .999999999999 mit 12 Stellen).
RND<RND ergibt statistisch verteilt etwa gleichhäufig 0 oder 1

SDEV Ermittelt die Standardabweichung der ersten Variablen des momentanen statistischen Feldes. Dieser Wert ist nur gerechtfertigt, wenn das Feld nur eine Auswahl und nicht alle möglichen Werte enthält:
Bei der Berechnung der Standardabweichung wird bekanntlich die Summe aller Fehlerquadrate durch die um 1 verringerte Zahl der Daten dividiert. Wenn das statistische Feld die Gesamtheit aller möglichen Werte enthält, muß deshalb vor der Berechnung von SDEV einmal MEAN addiert werden, um die Zahl der Daten um 1 zu erhöhen, ohne die Summe der Fehlerquadrate zu verändern.
Das Feld muß vorhanden und mit Daten besetzt sein. Das Anlegen des Feldes ist im CALC-Modus nicht möglich!

SDEV(X) Hat gleiche Wirkung wie SDEV, bearbeitet jedoch die Variable X des momentanen statistischen Feldes.

SGN(X) Ermittelt das Vorzeichen des Wertes X.
SGN(-37) ergibt -1
SGN(0) ergibt 0
SGN(-0) ergibt -0
SGN(153) ergibt 1

SIN(X) Ermittelt den Sinus des Winkels X.
SIN(30) ergibt .5 (Beispiele im Grad-Modus)
SIN(90) ergibt 1
SIN(180) ergibt -0
SIN(270) ergibt -1

SQR(X) Ermittelt die Quadratwurzel aus X.
SQR(9) ergibt 3
SQR(25) ergibt 5
SQR(12345654321) ergibt 111111

SQRT(X) Gleiche Wirkung wie SQR(X).

TAN(X) Ermittelt den Tangens des Winkels X.
TAN(45) ergibt 1 (Beispiele im Grad-Modus)
TAN(90) ergibt mit TRAP(DVZ,0) Fehlermeldung und wird nicht ausgeführt
TAN(90) ergibt mit TRAP(DVZ,1) Warnung und die Anzeige 9.99999999999E499
TAN(90) ergibt mit TRAP(DVZ,2) Warnung und die Anzeige Inf

TIME Ermittelt die Zahl der Sekunden, die seit dem letzten Wechsel des Datums vergangen sind.
TIME ergibt eine Zahl zwischen 0 und 86399.99
RMD(TIME,60) ergibt die Zahl der Sekunden, die seit dem letzten Wechsel der Minute vergangen sind
RMD(TIME,3600) DIV 60 ergibt die Zahl der Minuten, die seit dem letzten Wechsel der Stunde vergangen sind
TIME DIV 3600 ergibt die momentan ablaufende Stunde im 24-Stunden-Format

TOTAL Ermittelt die Summe aller Werte der ersten Variablen des momentanen statistischen Feldes.
TOTAL ergibt die Summe aller Posten, die im momentanen Statistik-Feld als erste Variable abgelegt sind.
Das Feld muß vorhanden und mit Daten besetzt sein. Das Anlegen des Feldes ist im CALC-Modus nicht möglich!

TOTAL(X) Gleiche Wirkung wie TOTAL, bezieht sich jedoch auf die Variable X des momentanen statistischen Feldes

TRAP(X) Ermittelt die Kennzahl für die momentan vereinbarte Fehlerbehandlung durch das Fehler-Flag X.
TRAP(INX) ergibt 0, 1 oder 2 (die momentan gültige Kennzahl)

TRAP(X,Y) Wirkung wie TRAP(X), ersetzt jedoch die bisherige Kennzahl durch Wert Y.
TRAP(INX,2) ergibt bisher gültige Kennzahl und setzt neuen Wert auf 2

UNF Gibt -5 aus, die Nummer des Flags, das beim Versuch des Unterschreitens des vom Rechner darstellbaren Zahlenbereichs (in positiver oder negativer Richtung auf den Wert Null) gesetzt wird.
Hinsichtlich der Anwendung siehe DVZ!

Kapitel 3

Der Tastenfeld-Basic-Modus

In der rechten unteren Ecke der Anzeige ist das Wort CALC sichtbar, wenn sich der Rechner im CALC-Modus befindet. Das ist der 'CALC-Melder'. Es gibt auch noch weitere Meldungen in der Anzeige, auf die wir jedoch erst dann hinweisen wollen, wenn sie für uns wichtig werden Wir wollen jetzt den Rechner nicht mehr im CALC-Modus betreiben und schalten ihn mit der f-Taste und der CALC-Taste (untere Reihe, zweite von links bzw. rechts) ab. Der CALC-Melder muß nun verschwinden. Wenn der CALC-Melder nicht zu sehen ist, befindet sich der Rechner im BASIC-Modus. Im BASIC-Modus wartet der Rechner auf unsere Befehle, so wie er bisher im CALC-Modus auf unsere Rechenanweisungen wartete.

Wenn Sie den Rechner im BASIC-Modus zu einer Berechnung benutzen wollen, haben Sie zwei Möglichkeiten: Entweder Sie flüchten wieder in den (vertrauten) CALC-Modus, berechnen so, was Sie wollen und kehren dann in den BASIC-Modus zurück. Sie können aber Berechnungen auch direkt im BASIC-Modus durchführen!

Wir wollen dazu eine "wohlbekannte" Aufgabe durchrechnen:

105+5*105/(105-30)

Im BASIC-Modus müssen wir dazu nur die gesamte Zeile eintasten. Sie erscheint dann so in der Anzeige (bitte nicht END LINE drücken!):

>105+5*105/(105-30)

Hier sagt uns das ">", daß wir uns im Tastenfeld-BASIC-Modus befinden. Merken Sie, daß es im BASIC-Modus keine Zwischen-Ergebnisse gibt? Deswegen hier eine Grundregel:

Tastenfeld-BASIC startet erst mit der END LINE-Taste!

Weil 105+5*105/(105-30) ein Befehl in HP-71-BASIC ist, wird der Rechner nach Betätigen von END LINE diesen Befehl lesen und ausführen:

END LINE (ergibt Anzeige):

112

Wenn Sie im BASIC-Modus den Befehls-Stack benutzen wollen, reicht es nicht aus, nur die AUF-Taste zu drücken, wie Sie es vom CALC-Modus her gewohnt waren: Nur über die g-Taste und die END LINE-Taste (kurz also CMDS) erreichen Sie den Befehls-Stack. Nun können Sie mit den AUF- und AB-Tasten im Befehls-Stack von Zeile zu Zeile springen!

Alle vier Cursor-Tasten sind (auch in Verbindung mit der g-Taste) voll wirksam, und auch die Cursor-Umschaltung (Ersetzen/Einfügen) sowie -CHAR und BACK dürfen Sie nach Belieben benutzen.

Hier noch eine weitere Tatsache, die Sie sich gut einprägen sollten:

Der HP-71 ist wegen seines Befehls-Stacks so einfach zu benutzen!

Je schneller Ihnen der Befehls-Stack so vertraut wird, wie ihre Brieftasche, umso eher werden Sie erkennen, wie bequem Sie mit dem HP-71 umgehen können!

Das HP-71-BASIC

Die Anzeige sollte jetzt so aussehen:

> (rechts daneben ein blinkender Cursor)

Das ">" zeigt Ihnen an: "Ich verstehe BASIC! Was wird gewünscht?" Der blinkende Cursor zeigt Ihnen: "Hier kommt das nächste Zeichen hin!"

Bevor wir irgendeinen Befehl ausprobieren, sollten wir uns überzeugen, daß der Rechner sich im Kleinschreib-Modus befindet. Dazu drücken wir eine beliebige Buchstaben-Taste: Wenn jetzt ein Großbuchstabe in der Anzeige erscheint, sollten wir die f-Taste und danach die LC-Taste (unten, etwa Mitte) drücken. Jetzt wird noch mit ATTN die Anzeige gelöscht, um klare Verhältnisse zu haben. Von nun an (bis zur Seite 115) setzt dieses Buch immer voraus, daß Sie den Rechner im Kleinschreib-Modus belassen.

Jetzt geht es los! Wir drücken die Tasten:

BEEP (ergibt Anzeige):

>beep

Das ist ein BASIC-Befehl, der den Rechner veranlassen soll, einen Laut von sich zu geben, den nur ein unmusikalischer Mensch als Ton bezeichnen wird. Aber warum quakt der Rechner nicht? Wir erinnern uns: Der Befehl wird erst ausgeführt, wenn die END LINE-Taste gedrückt wird! Tun Sie dem Rechner diesen Gefallen und Sie werden das erste wunderschöne Quaken hören.

Jetzt finden Sie auch "BEEP" goldgelb über der Taste "L". Wie Sie richtig vermuten, kann man den BEEP-Befehl auch mit der f-Taste und der L-Taste eingeben. Wenn Sie diese "Eingabehilfe" benutzen, steht aber >BEEP mit großen Buchstaben in der Anzeige! Auf diese Weise können Sie in diesem Buch immer unterscheiden, was Sie selbst mit Einzelbuchstaben eingeben mußten (das gibt kleine Buchstaben) und wann Sie Eingabehilfen verwenden konnten (das gibt große Buchstaben).

Weil zur Ausführung jedes Befehls die END LINE-Taste zu drücken ist, reden wir von dieser Taste als Selbstverständlichkeit nicht mehr! Wenn Sie lesen, daß Sie BEEP ausführen sollen, dann ist damit immer die Eingabe des Wortes BEEP (oder auch beep) mit anschließender END LINE-Betätigung gemeint. Dabei brauchen Sie selber aber keinen Ton von sich zu geben, es sei denn, es macht Ihnen Freude!

Um von den Möglichkeiten, die BASIC bietet, eine Kostprobe zu bekommen, tippen Sie mal folgendes ein:

FOR x=1 TO 10 @ x @ NEXT x

Sie haben den HP-71 eben angewiesen, von 1 bis 10 zu zählen, und er tat es! (Was es mit FOR, TO und NEXT auf sich hat, werden wir später noch sehen.) Wenn Ihnen das Zählen zu langsam erschien, dann nur deshalb, weil der HP-71 ihnen genügend Zeit zum Lesen geben wollte! Wenn Sie jede Zahl nur noch 0,1 Sekunde lang sehen wollen, geben Sie diesen Befehl:

delay .1

Wenn Sie jetzt in den Befehls-Stack schauen (g, CMDS) und die AUF-Taste einmal betätigen, finden Sie die FOR....-Zeile wieder. Sie können Sie mit END LINE erneut ablaufen lassen. Jetzt geht es schon wesentlich schneller! Geben Sie aus Neugier delay 0 ein, lassen Sie die FOR...-Zeile nochmal ablaufen, bzw. vorbeirauschen! Der DELAY-Befehl sagt also mit der nachfolgenden Zahl dem Rechner nur, wieviel Sekunden die Anzeige stehen bleiben soll.

Es gibt auch noch andere Befehle in BASIC, die Einfluß auf die Anzeige haben. Sehr nützlich ist CONTRAST, um die Anzeige an die Beleuchtung und an die Blickrichtung anzupassen. Wenn Ihnen die Anzeige zu flau erscheint, können Sie das ändern:

contrast 12

Jetzt haben Sie bei senkrechter Blickrichtung eine kontrastreiche Anzeige, allerdings ist diese Einstellung ungünstig, wenn Sie sehr flach auf die Anzeige schauen: Dann werden auch die unbenutzten Punkte der Anzeige sichtbar und der Text versinkt darin! Also ändern wir nochmals:

contrast 6

Jetzt ist es umgekehrt: Bei flachem Blickwinkel ist es nun besser, bei senkrechter Blickrichtung aber schlechter! Der "normale" Wert (nach erstmaligem Einsetzen der Batterien) ist CONTRAST 9. Wenn ich es mir in meinem Sessel bequem gemacht habe, erscheint mir CONTRAST 7 angenehm; sicher werden auch Sie bald wissen, welcher Wert Ihnen am besten zusagt; wenn sich die Blickrichtung stark ändert, müssen Sie auch den Kontrast ändern: Niedrige Werte für flache und hohe Werte für steile Blickrichtungen.

Bis jetzt wurde bei allen Berechnungen das Standard-Format des HP-71 benutzt. Dabei werden grundsätzlich alle 12 Stellen angezeigt, nur nachfolgende Nullen werden weggelassen. So ergab z.B. 1/2 die Anzeige .5 (und nicht .500000000000), aber 1/7 wurde als .142857142857 angezeigt (also mit 12 Stellen).

Das Anzeigeformat läßt sich nun in mehrfacher Weise verändern: Wenn es sich bei unseren Berechnungen um Geld handelt, sollten Sie folgendes eingeben:

fix 2

Damit weisen Sie den HP-71 an, immer genau 2 Stellen hinter dem Dezimalpunkt anzuzeigen, ohne Rücksicht auf die tatsächliche Zahl der Dezimalstellen. 1/2 wird jetzt zu 0.50, aber bei 1/7 erscheint nun 0.14 der Rest bleibt unsichtbar! Und das sollten Sie sich gut merken: Das gewählte Anzeige-Format beeinflußt wirklich nur die Stellenzahl der Anzeige, der in die Anzeige gebrachte Wert behält im Rechner seine Stellenzahl! Allerdings tut uns der Rechner noch einen weiteren Gefallen: Er rundet die Anzeige in der üblichen Weise. Wahrscheinlich kennen Sie das FIX-Format schon von anderen Rechnern. Die Rückkehr zum Standard-Format können Sie so bewirken:

std

Damit kehrt der Rechner wieder auf sein Standard-Format zurück. Außerdem gibt es noch die Formate SCI und ENG, die Ihnen sicher von anderen wissenschaftlichen Rechnern bekannt sind. Für ein wissenschaftliches Format mit vier Nachkommastellen ist einzugeben:

sci 4

Ähnlich können Sie auch ein technisches Format wählen, bei dem die Zahl der Vor- und Nachkommastellen zusammen immer um 1 größer ist, als die im ENG-Befehl stehende Zahl. Das ENG-Format unterscheidet sich vom SCI-Format dadurch, daß es nur Exponenten zuläßt, die ohne Rest durch 3 teilbar sind. Man räumt damit ein, daß vor dem Komma ein bis drei Stellen auftreten. Das ENG-Format paßt sich damit den technischen Vorsilben Mega-, Kilo-, Milli-, Mikro-, Piko- usw. an und erleichtert die jeweilige Umrechnung. Für das Format ENG 3 ist einzugeben:

eng 3

Dann erscheinen die Zahlenwerte 1/2, 1/20 und 1/200 als 500.0E-3, 50.00E-3 und 5.000E-3.

Beachten Sie, daß der gewählte Kontrast und das gewählte Format erhalten bleiben, wenn Sie den Rechner in den CALC-Modus umschalten. Änderungen dieser Einstellungen können aber nur im Basic-Modus vorgenommen werden.

Bevor wir zum BASIC-Programm-Modus übergehen können, muß ich voraussetzen, daß Sie die nachfolgend gebotenen Beispiele im Tastenfeld-Modus erprobt haben. BASIC-Programme können Sie erst schreiben, wenn Sie das Tastenfeld-BASIC beherrschen.

Sie wissen jetzt, daß sich der Rechner im BASIC-Modus befinden muß, wenn Sie BASIC-Befehle eingeben wollen und daß erst nach dem Druck auf die END LINE-Taste etwas geschieht. Im nächsten Kapitel werden die Grundregeln der BASIC-"Grammatik" erklärt.

Notizen:

Kapitel 4

Die BASIC-Wortarten

Die -zigtausend Wörter der Umgangssprache werden bekanntlich in eine ganze Reihe von Wortarten eingeteilt (Substantive, Adjektive, Verben, Adverben, Fronomen, Partikel usw.). Das HP-71-Basic kennt glücklicherweise nur sehr wenige "Wortarten":

1. Anweisungen (bzw. Befehle)

Wenn Sie BEEP eingeben, dann weisen Sie den HP-71 an, intern etwas zu tun, nämlich einen Ton zu erzeugen. Eine weitere Tätigkeit erwarten Sie nicht. Alle Worte oder Ausdrücke, die nur im Rechner etwas bewirken, ohne daß eine Zahl oder ein Text ausgegeben wird, nennen wir "Anweisungen". In diesem Buch wird dafür häufig auch das Wort "Befehl" gebraucht, was praktisch das gleiche besagt. Wenn Ihnen der Vergleich mit der Umgangssprache hilfreich ist, kann man Anweisungen (oder Befehle) als Befehlsform der Verben ansehen: He, Du, mach BEEP!

Einige Anweisungen sind schon allein in festgelegter Form wirksam, andere können (und viele müssen) durch Zahlenwerte ergänzt werden, wodurch sich die Wirkung variieren läßt. So bringt z.B. BEEP 1000 einen doppelt so hohen Ton hervor, wie das einfache BEEP. Beachten Sie, daß die 1000 nicht in Klammern eingeschlossen war, wir merken uns also:

Anweisungen benötigen keine Klammern!

2. Funktionen

Wenn Sie SIN(45) eingeben, erwarten Sie die Beantwortung eines mathematischen Problems: Sie wollen den Wert der Sinusfunktion für 45 Grad erfahren. Der Teil "SIN" beschreibt die gewünschte Funktion, die "45" ist das Argument, mit dem die Funktion ausgeführt werden soll, und die Klammern zeigen nur an, wo das Argument beginnt und endet. So ist z.B. SIN(45+45) etwas ganz anderes als SIN(45)+45.

Wenn Sie die Analogie zur Umgangssprache weiter benutzen wollen, dann sind Funktionen mit Adjektiven und Argumente mit Substantiven vergleichbar: Wenn ich von einem "blauen Haus" spreche, ist es eben kein x-beliebiges, sondern ein Haus mit einer bestimmten Farbe. Ähnlich ist SIN die Funktion, die die Zahl 45 auf ganz bestimmte Weise verwandelt. Wir merken uns:

Argumente für Funktionen müssen in Klammern eingeschlossen sein!

Nur wenige Funktionen (z.B. PI, DVZ, RES usw. brauchen kein(e) Argument(e).

3. Postfixe (Anhängsel)

Wenn Sie CFLAG ALL eingeben, weisen Sie den HP-71 an, alle Benutzer-Flags (von 0 bis 63) zu löschen. Das Wort CFLAG ist eine Anweisung, wie oben beschrieben, allein aber nicht wirksam. Erst der Zusatz einer Zahl oder einer Variablen (mit passendem Wert oder eine umfassende Angabe (in unserem Beispiel ALL) macht die Anweisung ausführbar. Der Versuch, CFLAG oder auch ALL, für sich allein auszuführen, ergibt die Anzeige "ERR:Invalid Expr".

Wenn Sie anordnen, ein Haus anzustreichen, ist diese Anweisung ohne Angabe der Farbe unvollständig. Die Postfixe sind so etwas wie die Adverben (oder adverbiale Ergänzungen): "Streiche das Haus rot an!" ist eine klare ausführbare Anordnung ("Streiche das Haus mit der vorhandenen Farbe an!" aber auch!).

Ähnliches gilt für das obige Beispiel, bei dem Sie entweder anordnen können, welche Flags zu löschen sind, aber auch, daß alle Flags gelöscht werden sollen.

Postfixe können in BASIC nur Anweisungen ergänzen, niemals Funktionen!

4. Operatoren

Die Operatoren (+, -, *, /, & usw.) lassen sich mit den Präpositionen der Umgangssprache vergleichen: Sie treten niemals am Ende eines Satzes auf, sondern leiten einfache oder komplizierte Wortfolgen ein. In BASIC werden die Operatoren meistens in der gleichen Weise benutzt, die uns von der Algebra bekannt ist: z.B. 1+2*(3-7). was wir in normaler Sprache als "Eins mit dem Doppelten von der Differenz von drei und sieben" ausdrücken. Sie sehen, jeder Vergleich hinkt etwas!

Dummerweise (?) gehören manche BASIC-Worte mehr als einer Wortart an. Wundern darf uns das nicht, denn BASIC wurde von Menschen gemacht, und in der Umgangssprache ist das auch nicht anders! So ist z.B. OFF einerseits eine komplette Anweisung (Rechner wird ausgeschaltet), es kann aber auch als Postfix dienen: BEEP OFF unterdrückt die akustischen Signale des Rechners. ANGLE kann als ANGLE(X,Y) einmal ein Funktion sein, in OPTION ANGLE DEGREES aber auch Teil eines Postfixes. Das ist nun bei BASIC nicht mehr zu ändern!

Den geringsten Kummer werden Ihnen die Operatoren bereiten, weil Sie diese (fast) alle schon kennen. Es wäre aber Zeitverschwendung, auswendig zu lernen, welche BASIC-Wörter Anweisungen, welche Funktionen und welche Postfixe sind, weil allein die Anwendung bestimmt, welche Wirkung die BASIC-Wörter an bestimmten Stellen haben. Außerdem sagt Ihnen der Rechner sehr deutlich, wenn Sie gegen die festgelegten BASIC-Regeln verstoßen!

Alle bisher erwähnten Wortarten waren Ihnen geläufig, es gibt aber noch ein BASIC-Symbol, das ich Ihnen bis jetzt vorenthalten habe: Es ist das "@"-Symbol. In der Umgangssprache entspricht das Symbol den Worten "und dann". Angenommen, ich will dem Rechner auftragen "BEEP, und dann BEEP 1000" auszuführen. Anstatt dafür zwei Zeilen zu nehmen, geht das (einfacher!) auch so:

```
BEEP @ BEEP 1000
```

Das "@"-Symbol "verklammert" die beiden kurzen Anweisungen zu einer längeren, weshalb das Symbol oft auch als "Klammeraffe" bezeichnet wird. Sie können @ so oft benutzen, wie Sie wollen, lediglich die Länge der gesamten Anweisung darf 96 Zeichen nicht überschreiten!

Aber, **Aber, ABER:** "@" in BASIC ist genauso bedenklich, wie "und dann" in der Umgangssprache! Sicher kennen auch Sie Leute, die mit einer Geschichte beginnen und dann vom Hundertsten ins Tausendste kommen und Ihnen alles mögliche von allen möglichen Leuten erzählen! Wie leicht geschieht es da, daß Ihnen beim Zuhören die Zuordnung von Leuten und Fakten durcheinandergerät! Solche Unterhaltungen sind bestenfalls ermüdend und langweilig, wenn man nicht weiter darüber nachdenkt. Oft ergeben sich aber aus den Verwechslungen Streit, Wutausbrüche, Ehescheidungen und womöglich Mord!

Halten Sie sich also zurück, wenn Sie mit "@" Ihren Rechner ansprechen! Der Rechner kann Vieles (fast) gleichzeitig tun, konnten aber Sie (vorher) auch alles logisch durchdenken?

Scherz beiseite! Benutzen Sie @ nicht zu oft und vor allem nicht unüberlegt! @ ist sehr bequem und für manche Dinge unentbehrlich, wie z.B für FOR-NEXT-Schleifen im Tasten-Modus (was das ist, erfahren Sie später!). Sie können mit @ auch eine beachtliche Anzahl von Einzelanweisungen in den Befehlsstack laden. Gewöhnen Sie sich aber bitte nicht an, immer @ zu benutzen, wenn eigentlich END LINE fällig wäre! Schreiben Sie später Ihre Programme im gleichen Schlendrian, dann wird das Aufspüren und Beseitigen von Fehlern im Programmablauf viel Kummer und Sorge machen! Weitere Einzelheiten darüber, wann @ unbedingt nötig ist, werden Sie bei den BASIC-Programmiertechniken erfahren.

Nun haben Sie einiges über die "Grammatik" von BASIC erfahren, im nächsten Kapitel werden Sie nun sehen, wie man im BASIC-Modus Variable auf eine Weise manipulieren kann, die im CALC-Modus unmöglich ist.

Notizen:

Kapitel 5

Variable im BASIC-Modus

Im CALC-Modus haben wir erfahren, wie man Variable mit Werten besetzen kann: Wir gaben W=SQR(6.8^2-5.5^2) ein, um W mit dem Wert 3.9987 zu besetzen. (Das war die Höhe, in der sich das abgeknickte Regenfallrohr an die Mauer des Bankgebäudes lehnte.) Im BASIC-Modus funktioniert das ebenso! Es gilt allgemein, daß Variable ihre Werte behalten, wenn Sie vom CALC-Modus in den BASIC-Modus (oder umgekehrt) wechseln. Das ist eines der Gebiete, auf denen Zusammenhänge zwischen beiden Betriebsarten bestehen.

Mit Variablen kann man aber im BASIC-Modus viel mehr anstellen, als im CALC-Modus! So können Sie im BASIC-Modus dem HP-71 sagen, daß unter einer Variablen eine ganze Liste von Daten (z.B. 100 Werte) abzulegen sind und nicht nur ein einzelner Wert. Und alle Werte lassen sich mit einem einzigen (Familien-) Namen aufrufen! Oder Sie können alle Variable mit einem Befehl löschen. Sie können Variable aber auch in ein "File" verpacken und dieses in einem "Speicher" ablegen, wo die Daten für spätere Verwendung geschützt aufgehoben sind (so als wenn ein Disketten-Laufwerk im Rechner eingebaut wäre, nur schneller!). Und noch etwas: Sie können nicht nur Zahlen, sondern auch Texte in Variablen ablegen.

Einige Bemerkungen über die "Formate" von Zahlen

Der HP-71 ist immer bemüht, unsere Wünsche aufs Beste zu erfüllen. Dazu gehört, daß er nach dem ersten Einsetzen der Batterien alle Ergebnisse mit 12 signifikanten Stellen ausgibt, wobei er das Komma an die richtige Stelle setzt. Allerdings unterdrückt er dabei "langweilige" nachfolgende Nullen rechts vom Komma. Wenn der errechnete Wert sehr groß (oder sehr klein) wird, nimmt der Rechner von sich aus Zuflucht zu der sogenannten "wissenschaftlichen" Darstellungsweise, indem er den 12-stelligen Zahlenwert nur mit einer Stelle links vom Komma ausgibt und einen positiven oder negativen Exponenten folgen läßt, der die Größenordnung der Zahl bestimmt. Diese Grundeinstellung der Anzeige sollten Sie für die folgenden Beispiele durch Eingabe von **std** sicherstellen.

Für die kommenden Überlegungen sollten Sie sich merken, daß bei internen Rechenoperationen durchaus 15 (oder manchmal noch mehr) Stellen auftreten können. Erst das Ergebnis (das Sie mit RES aufrufen können) wird auf 12 Stellen gerundet. Wenn weniger Stellen angezeigt werden, bedeutet das nur, daß sich das Ergebnis ohne Verfälschung mit weniger Stellen ausdrücken läßt.

Nun sind 12 Stellen nicht unter allen Umständen notwendig, mitunter sogar störend. Die Stellenzahl läßt sich durch einige Befehle beeinflussen, die aber auf sehr unterschiedliche Weise in die Arbeit des Rechners eingreifen. Diese Befehle und ihre Auswirkungen sollen uns als nächstes beschäftigen.

Im CALC-Modus haben wir Variable als Kurzbezeichnungen für Zahlenwerte benutzt. Diese Variablen wiesen immer 12 Stellen auf. Wenn wir X=1/3 eingaben, wurde X auf .333333333333 gesetzt, das sind aber nur die ersten 12 Stellen des unendlichen, periodischen Dezimalbruches, weil der Rechner den Wert weder genauer darstellen noch speichern kann. Wenn wir aber X=1/4 eingaben, erschien .25 in der Anzeige, weil dieser Wert mit 2 Stellen exakt beschrieben ist.

Die Variable mit 12 Stellen ist das Genaueste, was der HP-71 uns bieten kann. Wenn wir diese Präzision für die Variable X wünschen, können wir das dem Rechner auf folgende Weise mitteilen:

real x

Wir bezeichnen diese Variablen als "real", weil sie die vom Rechner gebotene Stellenzahl "wirklich" ausnutzen.

In vielen Fällen sind aber 12 Stellen nicht nur unnötig, sondern mitunter sogar störend. Wir können dann den Rechner anweisen, die Werte mit nur 5 Stellen zu speichern:

short y

bewirkt für die Variable Y diese Festlegung. Auch in diesem "Speicherformat" kann der Rechner Werte zwischen -9.9999E499 und +9.9999E499 verarbeiten.

Wenn wir dagegen sicher sind, daß die zu speichernden Werte nur ganzzahlig sein können und im Bereich von -99999 bis +99999 liegen werden, dann steht noch ein weiteres Speicherformat zur Verfügung, daß sich für die Variable Z so festlegen läßt:

integer z

Wenn Sie diese Eingaben so vorgenommen haben, können wir jetzt mit den Dingen etwas spielen:
Wir besetzen: x=1/3 @ y=x @ z=y und schauen nach, was der Rechner gespeichert hat: Der Aufruf von x ergibt .333333333333 (12 Stellen), y ergibt .33333 (5 Stellen) und z ergibt 0. Während die Anzeigen für x und y plausibel sind, sollten wir bei z daran denken, daß 1/3 näher an der 0 als an der 1 liegt!

Wenn wir nun x=2/3 @ y=x @ z=y eingeben, erhalten wir für x die Anzeige .666666666667 (12 Stellen, die letzte gerundet), für y .66667 (5 Stellen, die letzte gerundet) und für z eine schlichte 1 (weil 2/3 näher an der 1 als an 0 liegt).

Wir sehen, daß der Rechner bei der Zuordnung der Werte an die Variablen die nötigen Rundungen automatisch ausführt.
Dieses Festlegen des Formates der einzelnen Variablen wird häufig als "Deklarieren" bezeichnet.

Die Zahl der angezeigten Stellen läßt sich aber auch durch die Wahl eines passenden Anzeigeformates erreichen, wie Sie das schon am Ende des 3. Kapitels kennengelernt haben. Diese Formatierung ist zwar die "sanfteste" von allen, weil damit im Grunde keiner Variablen auch nur eine Stelle "gekrümmt" wird. Trotzdem können auch durch die Anzeigeformatierung Dinge geschehen, die auf den ersten Blick verblüffen: Wenn wir das Format **fix 5** einstellen und danach der "realen" Variablen X den Wert 1/3 zuordnen, und anschließend 3*x berechnen, erhalten wir die Anzeige 1.00000 (also 5 Stellen rechts vom Komma). Nach Eingabe von RES-1 werden Sie jetzt die Anzeige 0.00000 erwarten, tatsächlich zeigt Ihnen der Rechner aber -1.00000E-12 an, die wissenschaftliche Darstellung von -0.000000000001. Die Ursache für diese (scheinbare) Diskrepanz läßt sich im std-Format leicht klären.

Das Format von Variablen können Sie ändern, so oft Sie es wünschen! Sie müssen dabei aber an drei Dinge denken: Erstens wird bei jeder Neufestlegung des Formates einer Variablen ihr Wert auf 0 gesetzt, der alte Wert geht also verloren! Zweitens wirkt die Neufestlegung des Formates nur auf die dabei angegebenen Variablen, weiter geschieht nichts! Drittens (und das ist sehr wichtig!) behalten alle Variablen sowohl ihr Format als auch ihre Werte, wenn Sie vom BASIC-Modus in den CALC-Modus überwechseln! Wundern Sie sich also nicht, wenn Sie im CALC-Modus für z=1/3 die Anzeige 0 erhalten, denn Z ist vom letzten Beispiel noch als INTEGER-Format deklariert!

Nun eine Testfrage: Was erscheint in der Anzeige, wenn Sie im Tastenfeld-Basic-Modus folgendes eingeben?

```
short y
y=1/3
1/3-y
```

Das Dimensionieren von Variablen

Bisher benutzten wir die Variablen immer nur als Kurzbezeichnungen für einzelne (mitunter wechselnde) Zahlenwerte. Wenn wir aber eine ganze Zahlen-Liste speichern wollen, dann müßten wir für jeden Wert eine einzelne Variable benennen, wie das folgende Beispiel zeigt:

```
a=20
b=37
c=54
d=71
e=88
f=105
g=122
:  :
usw.
```

Gewiß, so können Sie es machen, es gibt aber einen besseren Weg! In der Mathematik benutzen wir oft Ausdrücke wie P_1 , P_2 und P_3, um damit verschiedene Werte zu kennzeichnen. Wir sagen dann, daß P_n der n-te Wert in unserer Werte-Liste "P" ist. Diese Darstellung benutzt Variable mit Index (oder indizierte Variable), weil man die tiefgestellte Zahl als Index bezeichnet. Eine Liste von Werten, die als indizierte Variable abgelegt sind, nennt man ein "Feld" oder ein "Array".

Auch BASIC läßt derartige Variable zu und übernimmt die Begriffe "Index" und "Feld" bzw. "Array". Bevor aber ein solches Feld benutzt werden kann, muß es angelegt werden, d.h. der Rechner muß einen genügend großen Bereich seines Speichers für das Feld reservieren. Die Größe dieses Feldes hängt natürlich von der Zahl der Werte und vom gewünschten Format ab.

Wenn Sie für die oben genannten sieben Werte ein Feld anlegen wollen, muß zuerst festgelegt werden, in welchem Format die Zahlen gespeichert werden sollen. Da es sich nur um ganzzahlige Werte handelt, ist das (Speicherplatz sparende) INTEGER-Format angebracht. Wir geben Format, Listen-Namen und Anzahl ein:

```
integer p(7)
```

Nun können wir die einzelnen Werte eingeben:

```
p(1)=20
p(2)=37
p(3)=54
p(4)=71
p(5)=88
p(6)=105
p(7)=122
```

Jetzt läßt sich (leider nur für die USA!) ganz leicht das Porto ermitteln, wenn eine Sendung von n Unzen zu verschicken ist! Für 3 Unzen ergibt p(3) eine Anzeige von 54 (Cent). Oder Sie geben für eine 5-Unzen-Sendung n=5 ein und anschließend p(n), um in der Anzeige 88 zu sehen!

Nun gibt es einige Leute, denen Tabellen mit einfachen Indizes noch nicht genügen: Ein solche (ein-dimensionale) Aufzählung von Werten wie auf einer Schnur kann doch kein "Feld" sein, sondern bestenfalls der "Feldrand"! Diese Leute wollen also eine Fläche mit Werten besetzen und müssen deshalb die Werte mit Doppel-Indizes kennzeichnen. Und eine solche (zwei-dimensionale) Anordnung, die wirklich die Bezeichnung "Feld" verdient, nennt man dann (wie logisch!) eine "Matrix".

Der Umfang einer Matrix läßt sich einfach dadurch beschreiben, indem man angibt, wieviel (waagerechte) Reihen und (senkrechte) Spalten (oder Kolonnen) sie aufweist.

Nehmen wir doch einmal an, daß Sie eine bestimmte Anordnung von Werten in Form einer Matrix abspeichern wollen, wie es in der höheren Mathematik üblich und zweckmäßig ist:

```
1   2  -3  -4
2   3  -4   5
5  -6   7  -8
```

Diese Anordnung soll unsere Matrix M werden. Die erste Zahl in der obersten Reihe können wir dann als $M_{1,1} = 1$ beschreiben und die letzte Zahl in der untersten Reihe als $M_{3,4} = -8$. Sie werden es kaum glauben, aber der HP-71 kann mit Matrizen genauso gut umgehen wie mit Feldern! Sie brauchen nur anzugeben, wieviel Reihen und wieviel Spalten und welche Bezeichnung die Matrix bekommen soll. Weil es beim Umgang mit Matrizen meist recht genau zugehen muß, dimensionieren wir die Matrix M auf folgende Weise und geben dann die Werte ein, wie bei einem Feld:

```
real m(3,4)

m(1,1)=1
m(1,2)=2
m(1,3)=-3
m(1,4)=-4

m(2,1)=2
m(2,2)=3
m(2,3)=-4
m(2,4)=5

m(3,1)=5
m(3,2)=-6
m(3,3)=7
m(3,4)=-8
```

Sie wollen kontrollieren, welchen Wert $M_{2,3}$ hat? Geben Sie m(2,3) ein und der Rechner wird Ihnen -4 anzeigen!

Wenn eine Variable mit einem einfachen Index "Feld" genannt wird und eine Variable mit doppeltem Index "Matrix" heißt, wie nennt man dann eine Variable mit dreifachem Index?? Diesen Namen brauchen Sie sich nicht zu merken, weil der HP-71 solche Variablen nicht kennt! Jetzt verstehen Sie, weshalb die HP-Bücher oft die Zahl der Indizes als "Dimension" von Variablen bezeichnen: Felder sind "eindimensional" und Matrizen "zweidimensional". Daher rührt auch der Ausdruck "Dimensionieren einer Variablen" (an Stelle von "Deklarieren"). Die Befehle **real x(100)** und **dim x(100)** bewirken exakt das gleiche. Sie können REAL beliebig gegen DIM tauschen, wenn es ihnen besser gefällt. Vergessen Sie aber nicht, daß DIM nicht nur die Zahl der Plätze bestimmt, sondern außerdem auch das Format der Variablen und damit die gespeicherte Stellenzahl auf 12 Stellen (also höchste Genauigkeit) festlegt.

Wie nennen wir denn nun unsere altbekannten Wald-und-Wiesen-Variablen ("Feld" sage ich lieber nicht!), die keinerlei Index ziert?

Wenn wir x=1/3 eingeben, dann ist X weder ein Feld noch eine Matrix, sondern eine "skalare" Variable, die nur einen einzigen Wert ohne jeglichen Schnörkel oder Zusatz enthält. Diese Variablen sind es, mit denen die meisten Benutzer des Rechners in fast allen Fällen auskommen. Wenn Sie aber ab und zu eine Variable als Feld oder gar als Matrix brauchen, ist es beruhigend zu wissen, daß der HP-71 damit umgehen kann.

Bei dieser Gelegenheit sollte erwähnt werden, daß das HP-71 MATH PAC ROM ganz hervorrragend mit Feldern und Matrizen umgehen kann! Sie können mit diesem ROM mit einfachen und kurzen Befehlen Felder addieren, Matrizen miteinander multiplizieren oder invertieren. Gleichungssysteme lassen sich durch einen Befehl auflösen, Fourier-Transformationen sind ebenso leicht vorzunehmen wie das Ermitteln der Nullstellen einer Funktion und vieles andere mehr. Wenn Sie derartige Probleme oft beschäftigen, wird sich für Sie die Anschaffung des MATH PAC ROM's sicher lohnen, weil es Ihnen die Arbeit sehr erleichtern kann.

Bemerkung: Wenn Sie eine Variable in einer Weise aufrufen, die der Deklaration widerspricht, wird eine Fehlermeldung ausgelöst: Wenn z.B. P als Feld deklariert wurde, dann kann die Eingabe von p=4 nicht ausgeführt werden. Es erscheint die "Data Type"-Fehlermeldung in der Anzeige.

Wenn Sie ein Feld oder eine Matrix aus dem Speicher restlos entfernen wollen, können Sie den DESTROY-Befehl dafür benutzen. Um das Feld P und die Matrix M zu "liquidieren" geben Sie bitte ein:

destroy p,m

Der DESTROY-Befehl löscht nicht den Inhalt der angegebenen Variablen, sondern tilgt die Variable selbst aus dem Speicher! Sie existiert dann nicht mehr!

Wenn viele Variable (besonders Felder und Matrizen) völlig unnütz den Hauptspeicher belagern, kann es sehr hilfreich sein, einige der überflüssigen Variablen zu löschen, um damit die Fehlermeldung "Insufficient Memory" zu vermeiden. Wenn der Platz im Hauptspeicher sehr eng wird, lassen sich sogar alle Variablen mit diesem Befehl "ausrotten":

destroy all

Obwohl dieser Befehl sehr gefährlich aussieht (nur END ALL wirkt noch furchterregender, wir lernen das später kennen!) zögern Sie nicht, den Befehl trotzdem zu benutzen: Er wirft beim Aufräumen des Hauptspeichers wirklich nur alle Variablen hinaus und "zerstört" bestimmt nichts anderes!

Komplexe Variable

Das Grundkonzept des HP-71 sieht grundsätzlich fünf verschiedene Arten von Variablen vor, die alle indiziert benutzt werden dürfen und von denen Sie bisher nur drei (REAL, SHORT und INTEGER) kennengelernt haben.

Die vierte Art der Variablen wird mit COMPLEX bezeichnet, kann nur in den Speicherformaten REAL und SHORT benutzt werden und dient dem Umgang mit komplexen Zahlen, wie 3 - i4 (wobei das Symbol "i" bekanntlich die "Quadratwurzel aus -1" bezeichnet). Das "Schicksal" dieser Variablenart ist etwas eigenartig:

Als das Entwurfs-Team das endgültige Betriebssystem für den HP-71 schrieb, stellte sich heraus, daß hierfür etwa 80 KByte nötig waren, während das Gesamtkonzept nur Platz für 64 KByte vorsah: Man mußte also notgedrungen einige interessante Routinen aus dem zu langen Betriebssystem hinauswerfen. Dabei ging (mit anderem!) auch die Behandlung von komplexen Variablen den Bach hinunter. Der HP-71 (ohne jegliches Zubehör) bietet den merkwürdigen Zustand, daß er komplexe Variable enthalten kann, die aber ohne das MATH PAC ROM nicht zu bearbeiten oder anzuzeigen sind! Der Grund: Die Befehle zum Deklarieren, Dimensionieren, Bearbeiten und Anzeigen komplexer Variabler stecken im MATH PAC ROM! Wenn Sie also mit komplexen Zahlen rechnen wollen, kaufen Sie sich dieses ROM, das mit komplexen Zahlen noch besser als der HP-15C umgehen kann.

Es soll hier nochmals wiederholt werden: Beim Übergang vom BASIC- zum CALC-Modus ändern Variable weder ihre Speicherformate noch ihre Dimensionen! Wenn Sie also im CALC-Modus mal ein Feld benötigen sollten, schalten Sie in den BASIC-Modus, dimensionieren das Feld und schalten in den CALC-Modus zurück. Wenn ein MATH PAC ROM im Rechner steckt, können Sie auf entsprechende Weise auch komplexe Variable im CALC-Modus benutzen.

Schluß-Bemerkung: Wenn Sie ein Feld in seiner Größe verändern, ohne das Speicherformat neu anzugeben, dann wird das Feld nicht automatisch mit Nullen besetzt! Wenn das neue Feld kleiner ist, gehen selbstverständlich die Werte verloren, die keinen Platz mehr finden. Wenn das neue Feld größer ist, bleiben alle alten Werte erhalten, nur die neu hinzugekommenen Plätze werden mit Nullen besetzt. Entsprechendes gilt auch für Matrizen: Ist die neue Matrix kleiner, gehen Werte verloren, ist sie größer, werden die neuen Plätze mit Nullen besetzt. Die Umsetzung der alten Werte in die neue Matrix geht zeilenweise vor sich, nicht spaltenweise!

Wenn Sie dagegen das Speicherformat für eine bereits benutzte Variable erneut angeben, werden bei Skalaren, Feldern und Matrizen alle Werte auf Null gesetzt. Experimentieren Sie mit kleinen Feldern und Matrizen, um zu erleben, wie das alles abläuft.

Statistische Funktionen

In diesem Buch werden die durch den HP-71 gebotenen statistischen Funktionen wie STAT, CLSTAT, ADD, DROP, TOTAL, MEAN, SDEV, CORR, LR und PREDV nicht behandelt, weil der entsprechende Abschnitt im Benutzerhandbuch für den HP-71 kaum Wünsche offen läßt. Wenn Sie diese Funktionen benutzen wollen, schauen Sie bitte in das Handbuch und erproben Sie die Beispiele.

Dort ist auch beschrieben, wie sich Datenmengen durch unterschiedliche Regressionsverfahren so verarbeiten lassen, daß die Abhängigkeit der Werte voneinander als mathematische Funktion angegeben werden kann.

String-Variable (für Zeichenketten)

Allen bisher besprochenen Variablen ist eines gemeinsam: sie sind nur zum Speichern von numerischen Daten brauchbar. Was aber, wenn Sie einen Namen, ein Wort oder einen längeren Satz speichern wollen? Die dazu fähige Variablenart fehlt noch!

Wieviel Stellen speicherte der Rechner, wenn Sie einen Zahlenwert mit der höchsten (möglichen) Genauigkeit aufbewahren wollten? Es waren 12 Stellen, und zwar immer. Auch wenn nur eine einsame Ziffer 3 der Variablen X zugeordnet wurde. Das wissen Sie bereits: X besteht dann aus der 3 und elf Nullen.

Wenn Sie aber das Wort "Hallo" speichern wollen, dann haben die einzelnen Buchstaben keine Stellenwerte, sondern sollen als fünf zusammengehörende Buchstaben als "Zeichenkette" angesehen werden. Dieses "Zusammenhängen" läßt sich (in Kürze) am besten durch das englische Wort "String" beschreiben (das Wort "Adstringens" kennen Sie vielleicht von. "zusammenziehenden" Medikamenten!). Wir wollen deshalb in Zukunft bei "String" bleiben (Manche Fremdwörter lassen sich eben nur mit umständlichen und gekünstelten Umschreibungen vermeiden).

String-Variable sehen den schon bekannten (numerischen) Variablen sehr ähnlich, legiglich ein angehängtes "$"-Zeichen kennzeichnet sie. Versuchen Sie mal:

N$="Hallo"

N$ (meist als "N-String" oder "N-Dollar" gesprochen) ist der Name der String-Variablen, die wir mit dem String "Hallo" besetzt haben. Um zu sehen, was N$ enthält, verfahren wir genau so, wie bei numerischen Variablen: wir geben den Namen der Variablen ein und drücken END LINE.

Es mag Sie etwas verwundern, daß der String jetzt ohne die ursprünglich eingegebenen Anführungszeichen angezeigt wird: die Anführungszeichen in der Zuordnung sollten dem Rechner nur zeigen, daß ein String (und kein Zahlenwert) zu bearbeiten ist. Wenn Sie (im std-Format) nur die Buchstaben pi eingeben, erhalten Sie den Zahlenwert 3.14159265359 als Anzeige, geben Sie dagegen "pi" (also mit Anführungszeichen) ein, erscheint (nach END LINE) nur pi in der Anzeige! Probieren Sie es aus!

Deklarieren der String-Variablen

Wenn Sie bei einer numerischen Variablen kein Speicherformat angegeben hatten, setzte der Rechner von sich aus das REAL-Format fest. Bei einer String-Variablen (ohne weitere Formatierung) ist der Rechner bereit, bis zu 32 Zeichen dieser Variablen zuzuordnen. Sie können also n$="Hallo" eingeben, ohne N$ zu deklarieren, weil die Eingabe kürzer als 32 Zeichen ist. Wenn Sie aber einen Satz wie diesen eingeben:

n$="Ein einzelnes Faedchen kann einen Mann fesseln"

dann erhalten Sie die Fehlermeldung "String Ovfl", die anzeigt, daß die String-Variable nicht genügend Platz hat, um den Text aufzunehmen. Für jeden String, der mehr als 32 Zeichen enthält, muß also die String-Variable entsprechend deklariert werden. Für einen String mit 100 Zeichen ist folgende Vorbereitung nötig:

dim n$[100]

Beachten Sie die eckigen Klammern! Eckige Klammern deuten beim HP-71 immer auf Dinge, die mit der Stellung der Zeichen in der String-Variablen zu tun haben. So bedeutet dim n$[100] für den Rechner die Aufforderung, im Hauptspeicher Platz für 100 Zeichen zu reservieren.

Wenn Sie ein ganzes Feld für eine String-Variable einrichten wollen, müssen Sie als erstes den Namen der Variablen, danach die Zahl der benötigten Plätze für die Strings und letztlich die notwendige Länge des (längsten) Einzelstrings angeben. Das sieht dann so aus:

dim n$(7)[100]

Damit reservieren Sie im Hauptspeicher sieben Plätze, von denen jeder einen String mit bis zu 100 Zeichen aufnehmen kann. Diese Plätze können Sie mit n$(1), n$(2) usw. besetzen und auch aufrufen.

Der HP-71 kann nur String-Felder verwalten, keine String-Matrizen! Wenn Sie etwas derartiges nötig haben, müssen Sie sich gedulden, bis wir bei den BASIC-Programmen sind, dann läßt sich auch darüber reden!

Nun rasch ein kurzer Test: Was bewirkt die folgende Eingabe?

dim n$[96],x,y,z

Ganz einfach: Es wird eine String-Variable N$ für 96 Zeichen eingerichtet und die Variablen X, Y und Z werden für das Speicherformat REAL deklariert.

Die besonderen String-Funktionen

Numerische Funktionen verarbeiten numerische Ausdrücke (darum heißen sie ja auch so!). Dann ist es wohl logisch, die Funktionen, die auf Strings wirken "String-Funktionen" zu nennen. Es gibt nun viel weniger String-Funktionen als numerische Funktionen. Das ist kein Wunder, weil der HP-71 in erster Linie ein Gerät ist, daß sich an Zahlen seine Zähne wetzt! Er ist z.B. glücklich, wenn er SQR(X) bearbeiten darf, er wird aber nervös, wenn er SQR("Alfred") oder FACT("Hallo") bearbeiten soll. Das sind für ihn sinnlose Eingaben und er sagt Ihnen das auch mit der Fehlermeldung ERR:Invalid Expr. Versuchen Sie es! Und merken Sie sich auch, daß der Rechner im CALC-Modus nur numerisch arbeiten kann. String-Funktionen und Strings sind dort nicht zugelassen.

Eine sehr nützliche String-Funktion ist LEN. Sie ermittelt die Länge eines Strings. Die Eingabe von

len(n$)

gibt die Zahl 5 zurück, weil n$ 5 Zeichen enthält (nämlich noch immer das Wort "Hallo").

Fast allen bisher besprochenen numerischen Funktionen war eines gemeinsam: sie bearbeiteten einen numerischen Ausdruck, d.h. eine numerische Variable oder eine unmittelbar lesbare Zahl (man könnte dazu "Numeral" sagen) oder eine Kombination aus diesen Möglichkeiten. Entsprechendes gilt auch für die String-Funktionen sie bearbeiten einen String-Ausdruck, d.h. eine String-Variable (z.B. n$) oder einen unmittelbar lesbaren und in verabredete Zeichen eingeschlossenen String (z.B. "Hallo"), für den auch die Bezeichnung "Literal" üblich ist, oder eine Kombination aus diesen beiden Möglichkeiten. Um den Gegensatz herauszustellen: 1537 ist ein Numeral, "Hallo" (aber auch "1537") sind Literale!

Als Abgrenzung vor und hinter Literalen können Sie entweder das Anführungszeichen (") oder den Apostroph (') benutzen, Sie dürfen diese beiden Zeichen aber nicht mixen, d.h. vor und hinter einem Literal müssen gleiche Zeichen stehen! Richtig ist also "Text" oder auch 'Text', falsch ist aber "Text' oder 'Text". In einigen Fällen scheint der HP-71 das Anführungszeichen (nachträglich) gegen den Apostroph auszutauschen (bei Label-Namen und bei den Tasten-Zuordnungen werden wir das noch sehen!), lassen Sie sich dadurch nicht beeinflussen und wählen Sie das Zeichen, das Sie wollen!

Nun etwas Spitzfindiges: Haben Sie bemerkt, daß PI weder eine Variable (der Wert liegt fest), noch ein Numeral (nicht unmittelbar lesbar) ist? Es wird "daher" als Funktion bezeichnet! Für einen Mathematiker ist auch das ein Widerspruch, denn eine Funktion hängt mit ihrem Wert immer von einer Variablen ab! PI hat keine Variable und immer den gleichen Wert, es ist also keine Funktion sondern eine Konstante! Ich stimme Ihnen da voll zu!

PI sollte also als Konstante bezeichnet werden! Wird es aber nicht, man nennt es eben eine "Funktion"! Das "klärt" auch das Dilemma, weshalb wir auch zu einigen anderen Dingen "Funktion" sagen sollen, z.B. bei RES. RES arbeitet (fast) wie eine Konstante, denn es liefert ohne Argument einen Wert, dieser Wert ist jedoch nicht unveränderlich. Eine Variable ist RES aber auch nicht, weil man RES keinen Wert zuordnen kann! Versuchen Sie doch RES=12345 und schauen Sie nach, was geschieht: RES wird auf Null gesetzt! Wir müssen also auch RES als "Funktion" hinnehmen!

Da wir uns gerade über RES unterhalten: Beachten Sie, daß RES niemals einen String enthalten kann! Der Inhalt von RES wird jedesmal aktualisiert, wenn eine numerische Variable besetzt oder ein Wert angezeigt oder gedruckt wird. RES wird von keiner String-Operation verändert.

Eckige Klammern und Teil-Strings

Geben Sie einmal folgendes ein:

```
dim n$[96]
n$="Ein einziges Faedchen kann einen Mann fusseln"
n$[14,32]
```

Was sehen Sie? Sie sollten drei Worte sehen. Die eckigen Klammern, die auf die String-Variable folgen, bestimmen den angezeigten Teil des Textes. Die 14 bedeutet: "beginne beim 14. Zeichen". Die 32 besagt: "beachte nichts, was hinter dem 32. Zeichen steht". So wurde von N$ nur der Teil vom 14. bis zum 32. Zeichen (einschließlich) dargestellt. Den Teil eines längeren Strings nennen wir "Teil"-String (hier kann man das Wort "Substring" wirklich entbehren).

Wenn in den eckigen Klammern nur eine Zahl steht, wertet der Rechner diese als "von da ab, bis zum Ende des Hauptstrings".

Benutzen Sie jetzt den Befehlsstack, damit Sie nicht alles neu eingeben müssen und ändern Sie die Teilstring-Anweisung in:

```
n$[14]
```

Sie sehen jetzt den String vom 14. Zeichen bis zum Ende!

Nun kommen wir zu der wichtigsten Anwendung von Teilstrings: Wenn Sie jetzt das Wort "fesseln" statt "fusseln" in den Text setzen wollen, sind nur folgende Eingaben nötig:

```
n$[40,40]="e"
n$
```

Jetzt sehen Sie "Ein einziges Faedchen kann einen Mann fesseln".

Wenn sie meinen, das dieser Mann ein Schwächling ist, hilft folgende Eingabe:

```
n$[14,14]="M"
n$
```

Das gefällt Ihnen sicher schon besser! Etwas später läßt sich die Situation vielleicht so beschreiben:

```
n$[39,40]="bu"
n$
```

Aber auch die "Probleme" sollen nicht zu kurz kommen:

```
n$[39,40]="stre"
n$
```

Sie sehen, es ist doch nicht ganz so ernst gemeint!

Obwohl es in den HP-Unterlagen nicht angegeben ist, lassen sich die eckigen Klammern nicht nur auf den Hauptstring, sondern auch auf einen eben definierten Teilstring nochmals anwenden. Versuchen Sie es:

```
n$[len(n$)/2][11,14]
```

Im ersten eckigen Klammerpaar wird die Stringlänge ermittelt, halbiert (aufgerundet, falls der Hauptstring eine ungerade Zahl von Zeichen aufweist) und so ein Teilstring gebildet, der entweder mit dem mittleren Zeichen des Haupstrings (bei ungerader Gesamt-Zeichenzahl) oder mit dem letzten Zeichen der ersten Hälfte des Hauptstrings (bei gerader Gesamt-Zeichenzahl) beginnt. Das zweite eckige Klammerpaar selektiert aus diesem Teilstring dann das 11. bis 14. Zeichen, also "Mann". Nach den Angaben der HP-Handbücher hätten wir folgendes eingeben müssen:

```
n$[len(n$)/2+10,len(n$)/2+13]
```

Wie Sie sehen, unterscheiden sich die beiden Verfahren in den Angaben für die zweite Selektierung, was an der unterschiedlichen Zählweise liegt: einmal muß das n-te Zeichen im Teilstring und einmal das (n-1)-te Zeichen nach dem ersten Zeichen des Teilstrings angesprochen werden! Am besten schreiben Sie sich den String auf und verfolgen dann die Wirkung der einzelnen Operationen. Selbstverständlich können Sie die Stringlänge auch durch 3, 4 oder sonst etwas teilen!

Merken Sie sich bitte, daß bei der Definition von Teilstrings nur ein Paar eckiger Klammern links von Gleichheitszeichen stehen darf: **n$[10,20]="xyz"** ist korrekt, **n$[10,20][2,4]="!"** führt zwar zu keiner Fehlermeldung, bewirkt aber auch nichts im Rechner!

Nun gibt es da noch eine nettes Verfahren, von dem nichts im Handbuch steht: Wenn Sie irgendetwas mitten in einen String einfügen wollen, dürfen Sie einen Teilstring formulieren, der genau an der gewünschten Stelle beginnt und der "eigentlich" gar keine Länge hat, weil sein Ende auf oder vor seinem Anfang liegt:

```
n$[34,33]="schoenen "
n$
```

Das funktioniert sogar, wenn Sie den Teilstring vor den schon existierenden String setzen wollen:

```
n$[0,1]="Nur e"
n$
```

Und schließlich können Sie überflüssige Teile des Hauptstrings durch einen Teilstring, der "nichts" enthält, auch löschen:

```
n$[5,8]=""
n$
```

Zum Schluß wieder einige Quiz-Fragen: Wenn Sie i$="123456789" eingegeben haben, was ergibt i$[3,6] ? Was ergibt i$[6] ? Und was geschieht mit i$, wenn Sie i$[3,7]="-" eingegeben haben?
"Erdenken" Sie das Ergebnis, bevor Sie es ausprobieren!

Auf eine Kuriosität soll am Ende dieses Abschnitts hingewiesen werden: Im CALC-Modus ist der HP-71 in algebraischen Dingen sehr konsequent, denn es müssen alle Funktionen vor den als Argument dienenden Ausdrücken eingegeben werden und die Operatoren stehen zwischen den numerischen Ausdrücken. Das gilt "natürlich" für alle HP-Funktionen, auch für die String-Funktionen . . . Nur die eckigen Klammern machen da eine Ausnahme: Sie stehen immer hinter dem String, der für sie das Argument bildet (die Angaben innerhalb der Klammern sind doch nur Parameter).

Wenn Sie von anderen BASIC-Dialekten den Umgang mit eckigen Klammern schon kennen, um mit Funktionen wie MID$, LEFT$ und RIGHT$ Teilstrings zu bilden, werden Sie einige Zeit zum Umgewöhnen brauchen. Wenn Sie bisher noch keinen Umgang mit BASIC hatten, werden Sie die eckigen Klammern bald zu schätzen wissen!

Das Et-Zeichen (&), auch Ampersand genannt

Mit den eckigen Klammern konnten wir einen String in beliebige kürzere Teile zerlegen, das "&"-Zeichen ermöglicht die umgekehrte Funktion, nämlich aus mehreren Strings einen einzigen zu machen:

"ABC"&"123"

ergibt eine Anzeige von ABC123. Das "&" hat also etwa die Wirkung von "und", weil es zwei Strings miteinander verklammert, wenn es zwischen ihnen steht. Man sollte sich aber davor hüten, das "&" in seiner Wirkung mit "+" zu vergleichen, Mißverständnisse sind dann unvermeidbar! Wenn Sie 1+2 rechnen, erhalten Sie nicht 12 sondern 3, nur "1"&"2" ergibt "12". Besser ist da schon der Vergleich mit dem "@"-Zeichen (dem "Klammeraffen"), das zwei kurze Befehle miteinander zu einem längeren Befehl "verklammert", wenn es zwischen ihnen steht. Wir können also "&" auch als "String-Verbinder" bezeichnen.

Wichtig ist es, daß wir eckige Klammern und den String-Verbinder auch gemischt anwenden dürfen und damit häufig überraschende Lösungen für unsere Probleme finden können:

"ABCDEF"[2,6]&"1234567"[3,5]

ergibt BCDEF345 in der Anzeige, weil wir das 2. bis 6. Zeichen des linken Strings (ABCDEF) mit dem 3. bis 5. Zeichen des rechten Strings (1234567) verbunden haben.

Wenn wir die Eingabe jedoch (unter Benutzung des Befehlsstacks) "geringfügig" ändern in:

("ABCDEF"[2,6]&"1234567")[3,5]

dann erscheint nur DEF in der Anzeige, nämlich das 3. bis 5. Zeichen von "ABCDEF"[2,6]&"1234567". Durch die gewöhnlichen Klammern können wir also erreichen, daß der "Machtbereich" der eckigen Klammern über den unmittelbar davor stehenden String hinaus erweitert wird. Erinnern Sie sich noch an die Hierarchie der Rechnungsarten? Multiplikation kam "vor" der Addition! Ähnliches gilt hier:

Eckige Klammern kommen "vor" dem String-Verbinder. Wir könnten das eine "Hierarchie der String-Funktionen" nennen, aber das tut niemand! Also lassen wir's auch sein!

Sie merken sich nur, daß gewöhnliche Klammern wichtiger sind als eckige und diese wiederum wichtiger als der String-Verbinder.

Die Funktionen CHR$ und NUM

Die CHR$-Funktion (meist "Charakter-String" gesprochen) ist oft sehr nützlich: sie wandelt eine Zahl (zwischen 0 und 255) in das unter diesem Wert definierte ASCII-Zeichen um. Wenn Sie CHR$(74)&CHR$(72) eingeben, erscheint JH in der Anzeige, weil 74 der ASCII-Code für J und 72 der für H ist. Wenn der HP-71 einen String speichern muß, dann speichert er die ASCII-Code-Zahlen der Zeichen, die den String bilden. Wenn Sie den String sehen wollen, muß der Rechner die ASCII-Codes in die für Sie lesbaren Zeichen umwandeln.

In diesem Buch ist keine Tabelle mit den ASCII-Codes abgedruckt, weil Sie diese Tabelle (in aller Ausführlichkeit) auf den Seiten 41 bis 45 in der Kurzanleitung zum HP-71 finden, die Sie immer zur Hand haben sollten, zumal sie doch in der Schutztasche so gut Platz findet.

Sie könnten jetzt einwenden, daß die CHR$-Funktion doch unnötig sei, weil sich JH doch auch direkt eingeben läßt. Nun, Sie brauchen CHR$ aber für Zeichen, die sich nicht über Tasten eingeben lassen! So z.B. den nach links geneigten Trennstrich "\", den Sie vergeblich auf der Tastatur suchen werden! Dieses Zeichen hat den ASCII-Code 92. Wenn Sie dieses Zeichen benötigen, geben Sie CHR$(92) ein.

Während nun CHR$ das durch eine Zahl bezeichnete Zeichen ermittelt, wirkt die NUM-Funktion ("Nummer") genau umgekehrt: sie nimmt ein Zeichen und ermittelt dessen ASCII-Code-Nummer. So ergibt NUM("J") die Anzeige 74. Dabei werden Sie merken, daß sich große und kleine Buchstaben in der Code-Nummer unterscheiden. Die NUM-Funktion nimmt immer nur das erste Zeichen des bezeichneten Strings. So erhalten Sie auch auf NUM("Ja!") die Antwort 74. Alles was hinter dem ersten Zeichen steht, ist für NUM völlig uninteressant!

Mit CHR$ und NUM lassen sich viele Zahlenwerte sehr "raumsparend" speichern: Wenn Sie z.B. 100 Zahlen mit (ganzzahligen) Werten zwischen 0 und 255 abspeichern müssen, dann lassen sich diese Zahlen als ASCII-Zeichen in einem String unterbringen. Das benötigt viel weniger Platz im Speicher als ein Feld.

Beispiel (zuerst als Feld):
Um ein Feld mit Namen "A" für 100 ganzzahlige Werte anzulegen, müßten wir eingeben:

```
integer a(100)
```

Für das Besetzen des 17. Platzes dieses Feldes mit dem Wert 153 ist dann nötig:

```
a(17)=153
```

Der Aufruf des Wertes, der auf dem 17. Platz des Feldes A steht, geschieht dann mit:

```
a(17)
```

Gegen-Beispiel (jetzt als String):
Um einen String für 100 Zeichen anzulegen, ist nötig:

```
dim a$[100]
```

Das Besetzen der 17. Stelle des Strings mit einem Zeichen, dessen ASCII-Code 153 ist (denn das wollen wir speichern), ist einzugeben:

```
a$[17,17]=chr$(153)
```

Entsprechend erfolgt der Abruf des ASCII-Codes des an 17. Stelle stehenden Zeichens mit:

```
num(a$[17])
```

Hier brauchen Sie nicht "17,17" als Position anzugeben, weil die NUM-Funktion sowieso nur das erste Zeichen des Teilstrings "zur Kenntnis" nimmt!

Die Funktionen STR$ und VAL

Ist es Ihnen schon aufgefallen, daß der Rechner Zahlen stets mit vorangehendem und folgendem Leerzeichen anzeigt? Versuchen Sie mal folgendes: Sie geben x=153 und "An";x;". Stelle" ein! Was Sie sich erhoffen ist eine Anzeige, die so aussieht:

```
An 153. Stelle
```

Sie sind jedoch etwas enttäuscht, denn die Anzeige sieht so aus:

```
An 153 . Stelle
```

Es ist eine Leerstelle zwischen der Zahl und dem Punkt, mit dem wir dieser Zahl den Charakter einer Ordnungszahl geben wollten! Diesen Schönheitsfehler kann die STR$-Funktion beheben! Diese Funktion nimmt eine Zahl und wandelt sie in einen String um, der ebenso aussieht, wie die ursprüngliche Zahl, jedoch ohne "störende" Leerzeichen davor und dahinter! Sie können also (mit dem Befehlsstack hoffentlich!) ihre Anweisung so umstellen:

```
"An ";str$(x);". Stelle"
```

Sie haben es doch wohl bemerkt, daß Sie jetzt das vorangehende Leerzeichen in den führenden Text einbauen müssen!

Wenn Sie die STR$-Funktion nur zum Entfernen störender Leerzeichen heranziehen, dann wäre das etwa so, als wenn Sie den HP-71 grundsätzlich nur im CALC-Modus benutzen: Sie verzichten auf die meisten Möglichkeiten, die Ihnen der Rechner bieten kann! Der Hauptzweck der STR$-Funktion liegt darin, Zahlen als Strings behandeln zu können, weil eben manche Funktionen nur auf Strings angewendet werden dürfen. Mit wieviel Stellen eine Zahl dargestellt wird, läßt sich zwar durch Abzählen der Stellen in der Anzeige feststellen. Weit bequemer ist es aber, die Zahl mit der STR$-Funktion als String abzubilden und dessen Länge mit der LEN-Funktion zu bestimmen. Sie erhalten dann die Länge des Strings, der das Aussehen der Zahl hat! Es gilt für alle String-Funktionen, daß sie auch auf jede Zahl anwendbar sind, wenn diese zuvor mit STR$ in einen String gewandelt wurde!

Das Gegenstück zur STR$-Funktion ist die VAL-Funktion. VAL wandelt eine als String vorliegende Ziffernfolge (oder Rechen-Vorschrift!) in eine "echte" (d.h. algebraisch weiter verarbeitbare) Zahl. Geben Sie a$="153" ein. Wenn Sie jetzt die die Zahl 153 aus A$ wieder auslesen wollen, dann läßt sich die NUM-Funktion nicht dazu verwenden, sie würde nur den ASCII-Code für die 1 (49) liefern, der Sie gar nicht interessiert. Die Eingabe val(a$) bringt Ihnen dagegen sofort die 153 als Zahl! VAL kann aber noch viel mehr: es kann nämlich auch numerische Ausdrücke berechnen! Wenn Sie z.B. a$="1+2*3^4" eingegeben haben, liefert Ihnen val(a$) mit 163 sofort das Ergebnis, wie Sie es auch im CALC-Modus mit der Eingabe 1+2*3^4 erhalten hätten!

Und nun kommt die wirkliche Überraschung: Sie dürfen in die als String verpackten "numerischen Ausdrücke" auch Variable einbauen, wie z.B. a$="pi*r^2" (das ist die bekannte Formel für die Fläche eines Kreises mit Radius r). Für einen Kreis mit dem Radius 7 ist nur r=7 und anschließend val(a$) einzugeben, um 153.938040026 in der Anzeige zu sehen. Wenn die Formeln sehr lang werden, dann müssen Sie selbstverständlich zuvor die Strings entsprechend dimensionieren!

Diese erstaunliche Eigenschaft der VAL-Funktion wird in den raffinierteren Programmen für den HP-71 häufig benutzt.

Merken Sie etwas? Wir haben uns noch in keiner Weise über das Programmieren unterhalten und doch schon zwei Verfahren gefunden, mit denen der HP-71 bestimmte Berechnungen wiederholt (auch mit veränderten Werten) ausführen kann! Die erste Methode benutzte den CALC-Modus und den Befehlsstack dazu: Die Ausdrücke enthielten Variable, wurden im CALC-Modus entwickelt und konnten, nach möglicher Eingabe von neuen Werten für die Variablen und Aufruf des Befehlsstacks erneut ablaufen. Die zweite haben Sie eben erst kennengelernt: Die als String "getarnten" numerischen Ausdrücke wurden mit der VAL-Funktion "berechnet". Danach ließ sich die Berechnung (auch mit anderen Werten für die Variablen) erneut ausführen.

Von diesen beiden Methoden ist der CALC-Modus einfacher, wenn man nur ein oder zwei numerische Ausdrücke mehrmals mit verschiedenen Werten berechnen muß, denn der Befehlsstack ist nun mal in seiner Höhe begrenzt. Wenn Sie aber sehr viele Formeln parat haben wollen, werden Sie die "String-VAL-Methode" bevorzugen, denn hier begrenzt nur das Fassungsvermögen des Speichers die Zahl der Rechenvorschriften, weil Sie jede unter einer individuellen Stringbezeichnung ablegen können. Und noch ein Trick: Wenn Sie vergessen haben sollten, welche Formel sich unter einem Stringnamen verbirgt, reicht die Eingabe dieses Namens aus, um die damit angesprochene Formel in die Anzeige zu locken!

Die Funktionen POS, UPCR$ und VER$

Es sind jetzt noch drei weitere String-Funktionen vorzustellen, die jedoch zueinander keine Beziehung haben.

Als erstes die POS-Funktion (Position), die einen (längeren) String darauf untersucht, ob und an welcher Stelle ein anderer in diesem vorkommt.

Geben Sie folgendes ein:

n$="Ein einziges Faedchen kunn einen Mann fesseln"

Schauen Sie sich den Text an: es sollte "kann" heißen, nicht "kunn"! Wir kennen schon das Verfahren, wie sich mit eckigen Klammern das "u" gegen ein "a" austauschen läßt, dazu brauchen wir aber die Position, an der das verkehrte Zeichen steht! Das Abzählen ist uns zu mühsam, wozu ist denn der Rechner da? Wir geben ein:

pos(n$,"u")

und erhalten 24 in der Anzeige, weil das "u" an der 24. Stelle im String steht. POS hält Ausschau nach dem (oder den) angegebenen Zeichen und meldet die Stelle, an der diese Suche zum ersten Mal Erfolg hat. Um jetzt das "u" gegen ein "a" auszutauschen, ist folgende Eingabe nötig:

n$[RES,RES]="a" @ n$

Ich habe RES benutzt, weil das Gedächtnis von RES zuverlässiger ist, als mein eigenes! Dazu ein Tip: Gewöhnen Sie sich an, möglichst oft RES zu benutzen! Es ist eine sehr nützliche Funktion! Wenn die erfahrenen HP-41-Fan's LAST X so oft benutzen, dann sollten die HP-71-Fan's das mit RES ebenso halten! Ich kann mir vorstellen, daß es eines Tages ein HP-71-Programm gibt, in dem RES die wichtigste Funktion ist. Das tritt zweifellos dann ein, wenn die Anwender dieses Rechners erst etwas mehr Erfahrungen sammeln konnten.

Jetzt wollen wir einmal annehmen, daß Sie in dem String nicht das erste "a", sondern das zweite "a" suchen. Auch hierfür gibt es verschiedene Wege: Sie können einen Teilstring so definieren, daß das erste "a" in diesem nicht enthalten ist:

```
pos(n$[16],"a")
```

Der Rechner durchsucht nun den Teilstring ab Position 16 nach einem "a". Sie können den Beginn der Suche aber auch in der POS-Funktion selbst unterbringen:

```
pos(n$,"a",16)
```

Auch jetzt beginnt die Suche bei Position 16 und wird fortgesetzt, bis ein "a" gefunden ist. Dieses Verfahren läßt sich noch verfeinern, indem Sie den Rechner das erste "a" finden lassen und eine zweite Suche erst hinter dieser Stelle beginnen. Die Anweisung dazu sieht so aus:

```
pos(n$,"a",pos(n$,"a")+1)
```

Der Rechner bearbeitet zuerst die innere POS-Funktion, ermittelt dabei 15, addiert 1 und beginnt dadurch hinter dem ersten "a" die zweite Suche (äußere POS-Funktion). In der Anzeige muß nun eine 24 stehen, die Position des zweiten "a" in unserem String.

Sie dürfen nie vergessen, daß die Funktionen als Argumente nicht nur einfache Zahlen, sondern auch numerische Ausdrücke verarbeiten können, die weitaus komplizierter sein dürfen, als Sie es sich momentan vorstellen können! Mit zunehmender Erfahrung in BASIC werden auch Sie Gefallen an verschachtelten Anweisungen finden. Das ist wie mit unserer Umgangssprache: wir benutzen da häufig auch komplizierte Satzkonstruktionen, mit denen ein Ausländer seine Schwierigkeiten hat, da er von unserer Sprache nur die Grundregeln kennt. Es ist schon ein Segen, daß der HP-71 BASIC so perfekt versteht, wie wir es wohl nie werden sprechen können!

Eine schadenfrohe Randbemerkung gehört hierher: Der HP-41 konnte ASCII-Strings nur in kleinen Bissen und püriert verdauen. Wenn ein ASCII-(also TEXT-)File im Extended Memory in einem Record 253 mal das Zeichen "A" und danach einmal das Zeichen "B" enthielt und der Rechner mit dem POSFL-Befehl suchen sollte, an welcher Stelle auf 23 "A" ein "B" folgt, dann brauchte er dazu etwa 67 Sekunden! Der HP-71 frißt dagegen Strings grundsätzlich roh! Er sucht einen String gleicher Länge in weniger als 0,3 Sekunden ab! Wenn Sie also viel mit dem HP-41 gearbeitet haben und ihn auf diesem Gebiet als langsam empfanden, dann werden Sie sich wundern, wie flink der HP-71 mit String-Funktionen umgeht. Lahm kann das wohl niemand mehr nennen! Er ist dabei sogar flinker als die String-Funktionen im MicroSoft(R)-BASIC, das viele Heim-Computer (vom TRS-80(R) bis zum IBM(R)-PC) benutzen.

Eine weitere, sehr nützliche String-Funktion ist UPRC$ ("UPpeRCase String", "Groß-Buchstaben-String"), die Klein- in Großbuchstaben wandelt. Wenn Sie also die im String A$ steckenden Kleinbuchstaben als Großbuchstaben sehen wollen, müssen Sie eingeben:

uprc$(a$)

Beachten Sie aber, daß die UPRC$-Funktion den String A$ nicht verändert, sondern lediglich eine Darstellung in Großbuchstaben vornimmt! Wenn Sie im String A$ tatsächlich nur noch Großbuchstaben gebrauchen können, muß die Eingabe so lauten:

a$=uprc$(a$)

Auch das gilt für alle Stringfunktionen: sie verändern nicht das Argument, sondern geben eine bearbeitete Version des Arguments aus.

Angenommen, Sie suchen im String A$ nach einem bestimmten Buchstaben, wobei es Ihnen gleich ist, ob es sich um einen Groß- oder Kleinbuchstaben handelt (z.B. "r" oder "R"). Die Eingabe muß dann so aussehen:

pos(uprc$(a$),"R")

Diese Eingabe bildet eine durchweg mit Großbuchstaben geschriebene Version des Strings A$ und sucht dort nach dem Zeichen "R". Dieses Verfahren wird in den Programmen für den HP-71 oft benutzt. (Anmerkung des Übersetzers: Leider wirkt die UPRC$-Funktion nicht auf die Umlaute (ä, ö, ü) und auch nicht auf das ß, für das es ja keine Großschreibung mit einem einzelnen Zeichen gibt! Es war schon immer etwas schwieriger, wenn man Deutsch spricht!)

Nun zur letzten Stringfunktion: Um VER$ (Versions-String) kennenzulernen, brauchen Sie diese Anweisung nur einzugeben und abzuwarten, was passiert! Wenn in der Anzeige lediglich der Text HP-71:1BBBB erscheint, dann liegt ein nackter und leerer HP-71 vor Ihnen, dessen Ports alle unbesetzt sind. Wenn dagegen nach diesem Text noch weitere Anzeigen auftauchen, dann ist das ein Hinweis, daß sich Einschübe in den Ports befinden oder daß ein Kartenleser eingesetzt ist oder daß LEX-Files im Speicher des Rechners umherstreunen. Sie können dann mit VER$ genau ermitteln, was da alles existiert. Das "BBBB" charakterisiert die Version der vier eingebauten ROM's. Dabei ist "BBBB" die erste zum Verkauf gekommene Ausführung (wenn Sie "AAAA" sehen, dann brauchen Sie dieses Buch sicher nicht zu lesen!). Ein später gekaufter HP-71 kann durchaus andere Buchstaben zeigen. Zukünftige Modelle des HP-71 (z.B. ein HP-71C) werden bestimmt andere Anzeigen liefern. Der HP-41C (das erste Modell also) kam schon in mehreren Versionen auf den Markt, auch vom Nachfolger HP-41CV, der einen größeren RAM-Speicher fest eingebaut hat, ist gleichfalls mehr als eine Version bekannt. Für den HP-41CX gilt das entsprechende.

Mein eigener HP-41CX (mit dem erweiterten ROM) stammt aus einer der ersten Serien, die Kennbuchstaben für die drei eingebauten ROM's sind immerhin schon bei NFL!

Die VER$-Funktion wird also in Zukunft noch wichtig werden, zumal HP versprochen hat, die älteren HP-71, zu einem Festpreis zu modernisieren, wenn Sie das wünschen!

Haben Sie alles über die String-Funktionen verstanden? Das ist gut! Im Kapitel über BASIC-Programme werden Sie bei den Programm-Beispielen, Routinen und sonstigen Tricks dieses Wissen gut gebrauchen können.

Sie sollten jetzt noch etwas über einige wenige BASIC-Anweisungen erfahren, die ausgesprochen häufig vorkommen:

Die LET-Anweisung

Bisher haben wir die Variablen mit den Werten gleichgesetzt, die wir ihnen zuordnen wollten: wir sagten einfach x=pi, um die Variable X mit dem Wert von PI zu besetzen. Es gibt aber noch einen anderen Weg, um das gleiche auszudrücken, und dieser Weg verwirrt den Anfänger oft:

```
let x=pi
```

Der BASIC-Befehl "LET" bedeutet etwa "zulassen" oder "erlauben". (Der Berliner sagt: "Laß' dem Kind doch die Bulette!") Ich habe die sehr persönliche Ansicht (der Übersetzer auch!), daß die Abkürzung besser "SET" ("setze") heißen sollte! Es sollte auch (wenigstens hin und wieder) Bier regnen! Aber es passiert doch nie!

Wie würden Sie X um 1 vergrößern? Ganz einfach:

```
x=x+1
```

Bei dieser Anweisung bekommen Mathematiker (mit Recht!) Schüttelfrost! Wie, um alle Welt, kann X=X+1 sein? Dann kann man auch gleich sagen 0=1 und das Binär-System bricht zusammen! Deshalb der Trick mit dem "LET"! Als Purist können Sie also nur diese Eingabeform billigen:

```
let x=x+1
```

Gewiß, es sieht ordentlicher aus! In der Wirkung ist aber kein Unterschied. Die Anwendung von LET ist daher nicht zwingend notwendig. Es sollte Ihnen aber klar sein, daß in allen BASIC-Anweisungen das "Gleichheits"-Zeichen seine ursprüngliche Bedeutung eingebüßt hat und nur noch der Befehl ist, die Rechnung auf der rechten Seite auszuführen und das Ergebnis der Variablen auf der linken Seite zuzuordnen.

Die FOR / NEXT - Schleife

Was werden Sie tun, wenn eine Anweisung mehrere Male auszuführen ist? Der einfachste (und mühsamste) Weg führt über die mehrmalige Eingabe. Wenn der Rechner 10-mal BEEP ausführen soll, ist diese Taste und END LINE je 10-mal zu drücken. Mit dem Befehlsstack geht das auch nicht viel besser, aber es gibt einen bequemeren Weg: Dazu brauchen wir einen "Zähler", der von 1 bis 10 zählt und beim Zählen jedesmal BEEP ausführen läßt. Jede Variable kann ein solcher Zähler sein. Als Variable nehme ich X (das ist meine Lieblings-Variable) und schon geht es los: Ich sage dem HP-71, daß er X von 1 beginnend, jeweils um 1 erhöhen soll, bis der Wert 10 erreicht ist. Und außerdem soll der Rechner nach jeder Änderung von X BEEP ausführen. Diese (komplizierte) Anweisung sieht so aus:

```
FOR x=1 TO 10 @ BEEP @ NEXT x
```

Die Leerzeichen sind hier nicht vorgeschrieben, sie machen die Anweisung nur besser lesbar! Bitte, erinnern Sie sich an die Bedeutung von "@"! Das war der "Klammeraffe" oder (seriöser) der "Befehls-Verbinder", der mehrere kurze Befehle zu einer längeren Anweisung zusammenfaßt. Hier stehen drei Befehle: FOR X=1 TO 10 bedeutet: "benutze X als Zähler von 1 bis 10"; allein bewirkt dieser Befehl nur, daß X auf 1 gesetzt wird. Wenn aber in der gleichen Zeile ein NEXT X steht, dann durchläuft X tatsächlich alle Werte von 1 bis 10 und die zwischen FOR und NEXT eingeschlossenen Befehle werden während des Zählvorganges jedesmal (durchlaufen und) ausgeführt. Tauschen Sie die 10 gegen eine andere Zahl aus (unter Benutzung des Befehlsstacks)! Verändern Sie auch den BEEP-Befehl durch andere Spielarten (z.B. BEEP 50*x,1 oder etwas ähnliches).

Solche "Schleifen" sind auch bei Eingabe über das Tastenfeld oft eine große Hilfe, besonders dann, wenn man den momentanen Wert des Zähler selbst als Argument benutzen kann: Wenn Sie z.B. einen Teil der Plätze eines Feldes löschen wollen, läßt sich das recht einfach durchführen:

```
FOR x=10 TO 20 @ m(x)=0 @ NEXT x
```

Hierdurch werden im Feld M die Plätze 10 bis 20 mit Nullen besetzt.

Außer dem Start- und Endwert des Zählers können Sie auch noch festlegen, um wieviel jedesmal weitergezählt wird:

```
FOR x=200 TO 2000 step 100 @ BEEP x @ NEXT x
```

Jetzt gibt der Rechner eine etwas drollige "Melodie" von sich, mit der Sie ihre Katze ärgern können! Der Rechner sollte zuerst mit einer Frequenz von 200 Hz BEEP machen, danach mit 300 Hz, mit 400 Hz und so weiter, bis er bei 2000 Hz angelangt war.

"STEP 100" legt fest, daß bei jedem Durchlauf der Schleife der Zähler um 100 geändert wird. Wenn eine Angabe über den Änderungsschritt fehlt, nimmt der Rechner dafür den Ersatzwert 1.

Manchmal sind auch Schleifen nötig, die (von sich aus) kein Ende haben! Versuchen Sie mal dieses:

FOR x=1 TO inf @ time$ @ next x

Was ist das nun wieder? Eine laufende Digitaluhr, ohne daß wir ein Programm geschrieben haben! Als erstes einen Trost: Der Rechner läßt sich auch wieder ausschalten (oder auch anders "normal" weiterbenutzen), Sie müssen zuvor nur die ATTN-Taste drücken! Wenn Sie genau aufpassen, werden Sie bemerken, daß es anscheinend "lange" und "kurze" Sekunden gibt. Geben Sie deshalb ein:

delay 0,0

Jetzt ist der Spuk vorbei, weil der Rechner die Anzeige unverzüglich an den neuen Zustand anpassen darf und nicht immer erst 0,5 Sekunden warten muß, bis er Ihnen etwas Neues bieten kann. Da es für die weitere Handhabung besser ist, kehren wir mit der Eingabe delay .5 wieder zur normalen Verzögerung zurück!

Die FOR/NEXT-Schleife ist eine der besten Anweisungen, die uns BASIC bietet. Schleifen können praktisch ohne irgendwelche Einschränkungen verwendet werden. Angenommen, Sie wollen die Summe der ganzen Zahlen von 1 bis 100 ermitteln und "haben die Formel vegessen", mit der sich das berechnen läßt. Sie wollen die Formel nicht nachschlagen, aber auch nicht selbst diese stumpfsinnige Rechnung ausführen. Lassen Sie das doch den Rechner (mit einer Schleife) für Sie tun:

s=0 @ FOR x=1 TO 100 @ s=s+x @ NEXT x @ s

Das bedeutet im Einzelnen: "Lösche die Summenvariable S, addiere die Zahlen von 1 bis 100 in die Variable S und zeige dann dieses Ergebnis an!" In weniger als 2 Sekunden steht das Ergebnis (5050) in der Anzeige!

Sie können in einer Schleife auch rückwärts zählen. Dazu ist nur ein negativer Wert hinter STEP nötig! Geben Sie zuerst delay 1 ein und dann (nach END LINE) folgendes:

FOR X=10 TO 1 step-1 @ "T minus";x;"Sekunden" @ NEXT x @ "Start!"

Sehen Sie den "Schönheitsfehler"? es heißt "1 Sekunden", aber was wollen Sie lieber, schnell Anweisungen schreiben oder einwandfreie Grammatik? Hier kann man zwar ganz einfach das "störende" Wort mit Sek. abkürzen, das ist viel eleganter als der Eingriff in die Anweisung, der so erfolgen könnte:

```
FOR x=10 TO 2 step -1 @ "T minus";x;"Sekunden" @
NEXT x @ "T minus 1 Sekunde" @ "Start!"
```

Sie müssen diese Anweisung erst komplett eingeben und dürfen zwischendurch nicht END LINE drücken! Sie merken es deutlich: Die Anweisungen werden schon länger!

Nun dürfen Sie selbst verzwickte und verrückte FOR/NEXT-Schleifen schreiben. Die meisten Programmierer, die sich mit Variablen rumschlagen müssen, für die -zig verschiedene, regelmäßig gebildete Werte benötigt werden, kommen ohne FOR/NEXT-Schleifen nicht mehr aus. Mit diesen Schleifen läßt sich nicht nur eine Menge Zeit sparen, sondern sie sind einfach "elegant".

In diesem Kapitel haben Sie erfahren, wie man über die Tasten eingegebene Werte in Variablen ablegen kann. Das nächste Kapitel zeigt eine andere Methode, wie sich Daten noch sicherer im HP-71 speichern lassen!

Notizen:

Kapitel 6

Speichermöglichkeiten im HP-71

Bis zu diesem Zeitpunkt haben wir den HP-71 wirklich immer nur als Rechner benutzt. Im Rechner wurden (mit Ausnahme der Variablen und des Befehlsstacks) keinerlei Informationen auf längere Zeit gespeichert, wenn auch zugegeben werden muß, daß die numerischen Variablen (besonders in der Feld- oder Matrizen-Version) eine Menge Zahlenwerte und die String-Variablen (und String-Felder) viele komplette Formeln aufnehmen konnten.

Variable haben aber zwei schwere Nachteile: Erstens ist ihre Zahl begrenzt, wenn auch die meisten Benutzer die gebotenen Möglichkeiten nicht voll ausschöpfen. (Auch ich benutze am liebsten nur X, Y, Z, J und K, sowie A$, B$ und C$). Die in den Variablen gespeicherten Daten sind auch relativ "gefährdet", weil man sich selbst nie trauen kann! Wie oft benutzt man für eine neue Aufgabe Variable, deren Werte noch wichtig sind! Das ist so ähnlich, wie bei einer Tonband-Kassette, die wir zu einer Aufnahme benutzen und erst hinterher merken, daß diese Kassette schon eine (meistens "unwiederbringliche") Aufnahme enthielt! Variable haben nämlich keine "Löschsperre"! Warum auch! Sie sollen ja "variabel" sein!

Und nun der zweite Nachteil: Ohne ein gutes Gedächtnis werden Sie bald nicht mehr wissen, welche Variablen Ihren Rechner bevölkern und wofür die einzelnen bestimmt waren. Wenn Sie nun einige der Variablen löschen wollen, um wieder Platz im Speicher zu schaffen, gibt es keine Funktion, die Ihnen sagt, welche Variablen existieren. Wenn Sie gründlich im Speicher aufräumen wollen, geht das nur mit DESTROY ALL, und das löscht wirklich alle Variablen, wichtige und unwichtige!

Zum Lösen dieses Problems gibt uns der Rechner die Möglichkeit einen bestimmten Teil seines Speichers zum "Tresor" zu erklären und dort Daten sicher "einzulagern". Dieser Teil des Hauptspeichers ist dann gegen den DESTROY-Befehl geschützt. Dieser Tresor kann nun eine ganze Anzahl von "Schließfächern" enthalten, weil man ja recht verschiedene Dinge säuberlich voneinander getrennt dort verwahren möchte. Für das einzelne "Schließfach" ist das Wort "File" üblich, das wir in Zukunft häufig benutzen werden.

Bevor Sie bei einer Bank ein Schließfach benutzen können, müssen Sie es (meistens) mieten, nur für ganz dringende Fälle gibt es den "Nachttresor". Auch ein File muß vor dem Abspeichern von Daten (in passender Größe) eingerichtet werden. Dazu reserviert der Rechner nach einer entsprechenden Anweisung einen bestimmten Teil seines Speichers für diesen File. Da es recht unterschiedliche Arten von Schließfächern gibt, wird es Sie nicht wundern, daß der HP-71 auch verschiedene File-Arten mit unterschiedlichen "Miet"- bzw. Einrichtungs- und Benutzungsarten zur Verfügung stellen kann. Auch einen "Nachttresor" werden Sie im HP-71 finden!

Mit einer Aufzählung der verschiedenen File-Arten und den jeweiligen Anwendungsmöglichkeiten wollen wir beginnen:

File-Typ:	kann speichern:
SDATA	Zahlenwerte
TEXT	Strings
DATA	Zahlenwerte und Strings
KEY	Tastenzuordnungen
BASIC	Programme in BASIC
BIN	Programme in Maschinensprache
LEX	zusätzliche Schlüsselworte in Maschinensprache
FORTH	Programme in FORTH

Wir werden uns jetzt mit den einzelnen File-Arten nacheinander beschäftigen. Das erscheint mir um so notwendiger, als das Benutzerhandbuch für den HP-71 mit dem Abschnitt 14 (Speichern und Abrufen von Daten) hinsichtlich des Zwecks und der Benutzung der File-Arten große Verwirrung stiften kann! Nach der Lektüre dieses Teilabschnitts war ich sicher, daß der Autor bis zur Abfassung dieses Handbuch-Abschnittes noch nie einen HP-71 benutzt hat! Da wird die Art des Zugriffs ("sequentiell" oder "frei") zu sehr in den Vordergrund gestellt. Ein unvoreingenommener Leser kann dadurch zu der (irrigen) Ansicht kommen, daß für jeden File-Typ auch nur eine Zugriffsart zulässig ist. Aber das stimmt überhaupt nicht!

Der File-Typ SDATA

"SDATA" steht für "strömende Daten" (Stream Data). Dieser File-Typ soll angeblich nur für den Austausch von Daten mit dem HP-41 dienen. Lassen Sie sich nur nicht einreden, daß SDATA-Files nur für HP-41-Besitzer interessant sind! Ich benutze SDATA-Files öfter als die DATA-Files, aber ich habe noch niemals HP-41-Daten in SDATA-Files abgelegt! Auch Sie werden bald bemerken, daß der SDATA-Typ die nützlichste File-Art darstellt.

Wie man einen SDATA-File einrichtet

Wenn Sie ein numerisches Feld A() (mit z.B. 100 Plätzen) in einem File ablegen wollen, dann ist ein SDATA-File zweckmäßig. Zum Anlegen dieses Feldes ist folgende Eingabe nötig:

```
create sdata neudaten,100
```

Diese Eingabe erzeugt einen SDATA-File mit dem Namen "NEUDATEN", der 100 Zahlenwerte aufnehmen kann. Das entspricht einem Schließfach mit 100 kleinen Ablagefächern für z.B. Münzen. Jedes "Fach" eines Files wird "Record" genannt. Der File "NEUDATEN" enthält nun 100 (von 0 bis 99) numerierte Records, die nach dem Einrichten des Files alle noch leer sind, also Nullen enthalten.

Hinsichtlich der File-Namen haben Sie beträchtliche, aber nicht völlige Freiheit: Ein File-Namen darf höchstens 8 Zeichen lang sein, muß mit einem Buchstaben beginnen und darf nur die Zeichen A bis Z (Groß- und Kleinschreibung ist gestattet wird aber auch nicht unterschieden) und die Ziffern 0 bis 9 enthalten.

Zweck und Anwendung der Kanalnummern

Das Ablegen von Daten in dem neu geschaffenen File setzt das "Öffnen" des Files voraus (wie man auch ein Schließfach öffnen muß). Um die Ansprache des Files zu erleichtern, verwendet der HP-71 als "Spitznamen" für die geöffneten Files einfache Zahlen. Dem entspricht im täglichen Leben die Bezeichnung der einzelnen Fernseh-Stationen durch (individuell) festgelegte "Kanalnummern". Auch der HP-71 benutzt diesen Ausdruck. Mit dem Öffnen des Files wird diesem auch eine Kanalnummer zugeteilt:

```
assign #1 TO neudaten
```

Von nun an dürfen Sie immer #1 sagen, wenn Sie "NEUDATEN" meinen! (In anderen BASIC-Dialekten gibt es einen "OPEN"-Befehl, der wirkt ebenso!) Sie können zur gleichen Zeit mehr als eine Kanalnummer zuordnen und damit mehrere Files offenhalten, für jeden File ist aber immer nur eine Kanalnummer gültig!

Ablegen von Daten in einem Record

Die 100 Records des Files "NEUDATEN" (Verzeihung! ich meinte "#1") entsprechen etwa 100 "Registern", wie Sie sie vielleicht vom HP-41 kennen. Diese Records sind nun von 0 bis 99 durchnumeriert, was sich intern offensichtlich einfacher durchführen läßt, als die gewohnte Zählweise von 1 bis 100. Bei Rechnern ist das nun mal so!

Um jetzt einen Wert in einem bestimmten Record zu speichern, müssen Sie drei Dinge angeben: die Kanalnummer, die Record-Nummer und den zu speichernden Wert:

```
PRINT #1,10;x
```

Diese Anweisung "schreibt" den momentanen Wert der Variablen X in den Record 10 des Kanals #1 (hinter dem sich der File "NEUDATEN" verbirgt). Beachten Sie, daß diese Anweisung nicht etwa X in den Record 10 einfügt (wie eine Karte in einen Kartenfächer), sondern das X den bisher im Record 10 gespeicherten Wert ersetzt. Später werden Sie noch erfahren, wie man Werte in einen File "einschieben" oder auch aus einem File (ohne bleibende Lücke) "verschwinden" lassen kann.

An die Stelle der Variablen X kann ebensogut ein numerischer Wert oder Ausdruck treten.

Lesen von Daten aus einem Record

Das Lesen eines Wertes aus einem Record ist dem Ablegevorgang sehr ähnlich: es muß die Kanalnummer, die Record-Nummer und eine Variable in der Anweisung stehen, die der Rechner mit dem gelesenen Wert besetzt.:

```
read #1,10;y
```

Diese Anweisung holt den Wert aus dem Record 10 des Kanals #1 und besetzt damit die Variable Y, die Sie anschließend beliebig weiter "verarbeiten" können. Verstehen Sie das bitte richtig: Y wird nicht dem Record 10 "zugeordnet"! Eine Änderung von Y beeinflußt nicht den im Record 10 gespeicherten Wert!

Ermittlung der Anzahl von Records eines SDATA-Files

Wenn Sie nun mit PRINT- oder READ-Anweisungen Daten in einen SDATA-File einbringen oder von dort abrufen, können Sie jederzeit jede existierende Record-Nummer verwenden. Das erscheint zwar selbstverständlich, gilt aber trotzdem nicht für die DATA-Files! (Mit denen ist der Umgang eben viel mühevoller! Jetzt merken Sie, warum ich die SDATA-Files so bevorzuge!)

Sollten Sie mal vergessen haben, wieviel Records ein bestimmter SDATA-File enthält, läßt sich das sehr einfach ermitteln, indem Sie (für unser Beispiel) **cat neudaten** eingeben (hier reicht die Kanal-Nummer nicht aus!), worauf der Rechner so antwortet:

```
NEUDATEN    SDATA    800
```

Das sagt Ihnen, daß unter dem Namen NEUDATEN ein SDATA-File mit 800 Bytes Länge existiert. Wenn Sie diese Zahl durch 8 teilen, erhalten Sie die Zahl der Records im File NEUDATEN. Ich empfinde es als praktisch, in jedem SDATA-File im Record 0 die Zahl der (noch freien) Records (hier 99) einzutragen, weil die Abfrage dann noch bequemer ist.

Vereinfachtes Ablegen und Lesen von mehreren Daten

Es folgen jetzt einige Tricks, die ich Ihnen bisher nicht verraten habe! Auch mehrere Werte lassen sich durch eine einzige Anweisung speichern oder lesen. Hier ein Beispiel:

```
PRINT #1,12;x,y,z,t
```

Durch diese Anweisung wird der Wert der Variablen X im Record 12, der Wert von Y im Record 13, der Wert von Z im Record 14 und der von T im Record 15 gespeichert. Dieses Verfahren findet nur dann seine Grenze, wenn die Anweisung länger als 96 Zeichen wird!

Der Lesevorgang läßt sich genau so einfach einleiten:

```
read #1,12;m,n,o,p
```

Der im Record 12 stehende Wert wird der Variablen M, der von Record 13 der Variablen N, der von Record 14 der Variablen O und der von Record 15 der Variablen P zugewiesen.

Noch verblüffender ist die Tatsache, daß sich auch ganze Felder mit einer einzigen (kurzen) Anweisung speichern oder lesen lassen. Falls Sie ein Feld mit **dim a(100)** definiert haben, lassen sich die Werte aus unserem SDATA-File so auf die Plätze des Feldes A() übertragen:

```
read #1,0;a()
```

Durch das Klammerpaar können Sie andeuten, daß es sich bei A um ein Feld handelt. Sie dürfen die Klammern ebensogut auch weglassen, der Rechner hat sich "gemerkt", daß unter A ein Feld angelegt wurde! Bei dieser Anweisung müssen Sie nur dafür sorgen, daß die Zahl der auszulesenden Records nicht kleiner ist, als die Zahl der Plätze des Feldes, weil der Rechner Sie sonst mit der Fehlermeldung "ERR:End of File" erschrecken wird. Bei dieser Gelegenheit möchte ich Sie noch darauf hinweisen, daß in der Variablen-Liste einfache Variable mit Feldern und Matrizen munter gemixt sein dürfen, achten Sie nur darauf, daß die Zahl der Werte im File zum Besetzen aller Variablen ausreicht!

Das Ablegen von Feldern (und Matrizen) ist genau so einfach:

```
PRINT #1,0;a()
```

Mit dieser Anweisung werden alle Werte des Feldes A, beginnend mit Record 0, abgelegt. Auch hier sind die Klammern wieder zugelassen, aber nicht erforderlich! Sie können, wenn Sie den Inhalt einer Matrix ablegen oder lesen wollen, in die Klammern noch ein Komma setzen, um anzudeuten, daß es sich um eine Matrix handelt. Dieses Komma ist, genau wie die Klammer-Symbole, nur eine "Lesehilfe" zum Verstehen der Anweisung durch den Menschen, der Rechner kann auf (beides) verzichten! Eine "Abstimmung" von Datenmenge und Recordanzahl ist bei PRINT-Anweisungen für SDATA-Files nicht nötig! Den Grund erfahren Sie umgehend!

Weil ein SDATA-File eine ganz einfache Aneinanderreihung von Werten darstellt, ist den abgelegten Werten nicht anzumerken, ob sie vorher einer Matrix oder einem Feld angehörten oder ob es sich um die Werte von gewöhnlichen Variablen handelt! Sie können daher die "en bloc" eingegebenen Werte eines Feldes oder einer Matrix auch als Einzelwerte auslesen, oder umgekehrt! Nur Sie allein wissen, wozu die gespeicherten Werte gehör(t)en und was sie bedeuten! Der SDATA-File selber kennt nur die Werte, aber nicht deren Beziehung untereinander.

Der sequentielle Zugriff

Bis jetzt haben wir bei PRINT- oder READ-Anweisungen, die ein SDATA-File betrafen, immer die Nummer des gewünschten Records angegeben. Das wird meist als "freier" oder auch "direkter" Zugriff (englisch: "random access" --> "zufälliger Zugriff") bezeichnet, weil die vom Benutzer frei getroffene Entscheidung über die Record-Nummer für den Rechner "zufällig", d.h. ohne innere Gesetzmäßigkeit erfolgt. Aber überlegen Sie mal, was wirklich geschah:

Als wir die Anweisung **read #1,12;m,n,o,p** eingaben, haben wir dem HP-71 nicht gesagt, den Inhalt vom Record 13 in die Variable N zu setzen, das tat der Rechner von sich aus! Wir hatten nur den Record 12 bezeichnet und dem Rechner dafür die Variable M (und drei weitere!) gegeben. Diese Anweisung setzte einen "File-Zeiger" zuerst auf 12 und dort begann auch der Lesevorgang. Nach Lesen des Records 12 (und der Zuordnung des Wertes an die Variable M) hat der Rechner automatisch den File-Zeiger auf 13 erhöht, dann bemerkt, daß eine weitere Variable auf einen Wert wartet, den File-Zeiger auf 14 erhöht...usw., bis auch die letzte Variable (hier P) besetzt war! Abschließend wurde der File-Zeiger nochmals erhöht, er müßte danach auf 16 stehen. Was wird wohl geschehen, wenn wir die folgende Anweisung (ohne Record-Nummer) eingeben?

```
read #1;q
```

Weil der File-Zeiger bereits auf 16 steht, wird der dort abgelegte Wert der Variablen Q zugeordnet und dann wird der File-Zeiger auf 17 erhöht! Das bezeichnet man nun als "sequentiellen" Zugriff, weil wir es dem Rechner überlassen, die Records einen nach dem andern in ihrer natürlichen Reihenfolge zu bearbeiten. Beachten Sie bitte, daß es keine "sequentiellen" oder "freien" bzw. "direkten" File-Arten gibt! Auf jeden File kann man sequentiell oder frei bzw. direkt zugreifen!

Einwirkung auf den File-Zeiger

Um den File-Zeiger auf einen bestimmten Record zu setzen, müssen Sie die RESTORE-Anweisung benutzen (hier soll der Zeiger auf den Record 57 gesetzt werden):

```
restore #1,57
```

Nun können Sie nach Herzenslust ab Record 57 den File sequentiell auslesen! Sie dürfen aber den File-Zeiger nicht über das Ende des Files hinaus zu setzen versuchen! Genau wie sich "nicht-existente" Records nicht lesen ließen, können Sie den File-Zeiger auch nicht auf einen "nicht-existenten" Record setzen. Entsprechende Anweisungen lösen stets die Fehlermeldung "ERR: End of File" aus. Wie Sie den tatsächlichen Umfang eines SDATA-Files ermitteln können, wissen Sie sicher noch?

Wie sich ein SDATA-File nachträglich verlängern läßt

Nun kommen die "magischen" Fähigkeiten des sequentiellen Zugriffs: Für PRINT-Anweisungen ist das File-Ende keine unabänderliche Barriere! Wir wollen als erstes den letzten Record unseres Files lesen:

```
read #1,99;x
```

Record 99 wird gelesen und danach setzt der Rechner den File-Zeiger auf 100 (und diesen Record gibt es noch nicht!). Die nächste Anweisung formulieren wir listigerweise ohne File-Zeiger:

```
PRINT #1;pi
```

Und nun vollbringt der HP-71 ein "Wunder"! Er gibt keine Fehlermeldung aus, sondern "erweitert" den SDATA-File um einen Record, legt dort den Näherungswert für PI ab und setzt den File-Zeiger auf 101. Wenn Sie jetzt **cat neudaten** ausführen, werden Sie sehen, daß der File nun 808 Bytes lang ist!. Den dafür nötigen Platz im Speicher hat der Rechner also automatisch durch eine Verschiebung der Inhalte geschaffen (was für Sie sehr mühselig gewesen wäre!). Auch wenn Sie mehr als einen Wert ablegen müssen (Einzelwerte, aber auch Felder und Matrizen), wird dabei das File-Ende automatisch genügend weit verschoben!

Merken Sie sich aber gut, daß mit der RESTORE-Anweisung dieser Trick nicht gelingt, weil diese den File-Zeiger nur auf existierende Records setzen kann! Nur mit einer READ- oder PRINT-Anweisung, die den letzten existierenden Record betrifft, können Sie den File-Zeiger über das File-Ende "hinausschieben". Während aber die READ-Anweisung nur bis zum letzten existierenden Record gehen darf (weil ein "Überlesen" des File-Endes zu einer Fehlermeldung führt), verhält sich die PRINT-Anweisung viel "praktischer": für die noch abzulegenden Werte schafft sie automatisch Platz! Deshalb brauchen Sie sich bei PRINT-Anweisungen für SDATA-Files nie Gedanken darüber zu machen, ob der vorhandene File genügend Platz bietet!

Einfügen und Entfernen von Werten

Wenn man die Möglichkeit zur Erweiterung eines bestehenden SDATA-Files mit dem einfachen Lesen und Abspeichern ganzer Felder geschickt kombiniert, ist es auch möglich, neue Records (mit den gewünschten Werten) in den File an beliebiger Stelle einzufügen. Nehmen wir an, daß unser File bisher 100 Records enthält und wir nachträglich zwischen den Records 52 und 53 einen zusätzlichen Record (für die Variable X) einfügen wollen: Der alte Record "53" heißt dann "54" usw.

Das hört sich schwieriger an, als es wirklich ist:

```
dim t(100-53)
read #1,53;t
PRINT #1,54;t
PRINT #1,53;x
```

Das alles dauert (ohne die Zeit für das Eintippen!) in der Ausführung etwa 0,7 Sekunden! Wir brauchen keine spezielle INSERT-Anweisung! Durch die Übernahme der Daten in ein passendes Feld und ein beliebig verschobenes Ablegen der Daten können Sie nicht nur Einfügungen erreichen, sondern auch Werte rotieren lassen, löschen und mehr!

Löschen der Zuordnung der Kanalnummer

Da jede Kanalnummer immer nur einem File zugeordnet sein kann, reicht es zum Schließen eines Files aus, die betreffende Kanalnummer anderweitig zuzuordnen. Das kann ein anderer File sein oder man ordnet die Kanalnummer einem "Nichts" oder einem verabredeten Symbol zu, hinter dem sich nie ein File verbergen kann (* oder "*"). Am einfachsten ist folgende Anweisung zu verstehen:

```
assign #1 TO ""
```

Damit wird die Kanalnummer 1 einem File zugeordnet, dessen Name ein "Nullstring" ist. Der bisher so oft strapazierte File NEUDATEN ist dann geschlossen und somit für PRINT- und READ-Anweisungen unerreichbar, bis er wieder erneut geöffnet wird. Beachten Sie, daß auch eine END- oder SUBEND-Anweisung alle offenen Files schließt. Wenn Sie einen offenen File mit einer PURGE-Anweisung löschen, dann verschwindet dieses File aus dem Speicher, aber auch die ihm zugeordnete Kanalnummer wird gelöscht.

Bedenken Sie bitte, daß auch bei jedem "Neuordnen" des Speichers alle offenen Files geschlossen werden. Das geht aus dem Benutzer-Handbuch nicht klar genug hervor! Der Speicher wird bei drei Gelegenheiten neu geordnet: beim Abtrennen eines unabhängigen Speicherbereichs durch die FREE PORT-Anweisung, beim Zurückgeben eines abgetrennten Speicherbereichs durch die CLAIM PORT-Anweisung und beim Ändern der Bestückung der Ports (der Kartenleser ist hiervon ausgenommen, weil er eine feste Adresse hat). Vermeiden Sie also diese Tätigkeiten bei offenen Files, da diese sonst geschlossen werden. Das kann einen Abbruch des laufenden Programms auslösen, der sich nicht mit der CONT-Taste rückgängig machen läßt!

Weitere Eigenheiten der SDATA-Files

Im Benutzerhandbuch sind gar fürchterliche Dinge angekündigt, die passieren sollen, falls Sie die "End-of-File-Markierung" nicht beachten. "Datenverlust" ist noch das geringste der Übel! Vergessen Sie diese Drohungen, solange Sie SDATA-Files benutzen! Diese schrecklichen Dinge können nur bei DATA-Files auftreten! Ich sagte es Ihnen bereits mehrfach: SDATA-Files sind eine herrliche Sache!

Eine Tatsache über die SDATA-Files kam bisher nicht zur Sprache: In dieser File-Art lassen sich (unter "normalen" Bedingungen) nur Zahlenwerte speichern, keine Strings! Nur wenn Sie einen HP-41 und das dazu passende HP-IL-Interface haben, können Sie Zahlen und ALPHA-Strings in den SDATA-Files des HP-71 ablegen!

Es mag den Anschein erwecken, als ob der Umgang mit Files nicht ganz leicht ist. Dabei muß man aber berücksichtigen, daß diese Dinge meistens von einem Programm erledigt werden. Das sichere Speichern der Werte eines Feldes ist aber eine fast ebenso häufige Tätigkeit, wie der Umgang mit einer FOR/NEXT-Schleife. Um die Reihenfolge der Werte in einem Feld mit 100 Plätzen umzudrehen genügen folgende Anweisungen:

```
PRINT #1,0;a()
FOR x=0 TO 99 @ read #1,x;a(99-x) @ NEXT x
```

Für die Mühe, die der Umgang mit Files erfordert, wird man reichlich dadurch entschädigt, daß die Files ein sehr sicherer Ort für Daten sind. Die Fußangeln und Gemeinheiten von so brutalen Funktionen wie DESTROY sind gegen Files nämlich wirkungslos.

Allgemeines über Files

Nachdem Sie jetzt einige Werte in einem SDATA-File abgelegt haben, könnten Sie nun ganz wundervolle Sachen damit anstellen. Es gibt eine Menge BASIC-Anweisungen, die sich auf Files anwenden lassen: Wenn Sie **cat all** eingeben, sehen Sie die Katalog-Eintragungen der im Speicher stehenden Files. Mit den AUF- und AB-Tasten können Sie den Katalog durchstöbern und mit den RECHTS- und LINKS-Tasten auch die gesamte Eintragung für jeden File anschauen. In der Anzeige erscheinen (als wichtigste Dinge) der Namen des Files, die (eventuelle) Sicherungsart, der File-Typ und die Länge in Bytes. Weiter rechts (durch die RECHTS-Taste erreichbar) ist mit Datum (als MM/TT/JJ) und Uhrzeit der Zeitpunkt des Anlegens des Files vermerkt.

Die File-Namen bestehen, wie schon früher gesagt, aus 1 bis 8 Zeichen, von denen das erste unbedingt ein Buchstabe (von A bis Z bzw. a bis z) sein muß. Danach können Buchstaben und Ziffern (von 0 bis 9) in beliebiger Anordnung auftreten. Umlaute sind jedoch hier nicht gestattet!

Wenn Sie jetzt den Namen eines Files ändern wollen, können Sie z.B. folgendes eingeben:

rename neudaten TO altdaten

Dadurch heißt das SDATA-File nicht mehr NEUDATEN sondern ALTDATEN. Interessant ist dabei, daß die Umbenennung den File nicht schließt und daß die Kanalnummer auch für den neuen Namen gültig bleibt. Auch davon steht nichts im Benutzerhandbuch!

Wenn Sie einen File nicht mehr benötigen, können Sie ihn löschen. Bevor Sie aber nach der folgenden Eingabe END LINE drücken, sollten Sie sich ganz sicher sein, daß die Werte im File ALTDATEN wirklich nicht mehr gebraucht werden!

purge altdaten

Nach END LINE ist der File ALTDATEN gelöscht, die Zuordnung zur Kanalnummer aufgehoben und die Daten sind kaum wieder aus dem Rechner hervorzuzaubern! Weg ist weg!

Was kann man nun tun, um Daten auch vor der PURGE-Anweisung zu schützen? Nun, wir können den so kostbaren File sichern:

secure altdaten

Wenn Sie im vorigen Beispiel wirklich den File ALTDATEN gelöscht haben sollten, erscheint jetzt die Fehlermeldung "ERR:File Not Found". Spätestens beim Versuch den File zu sichern, bemerken Sie, daß dieser gar nicht mehr existiert! Ich hoffe mit Ihnen, daß Sie diesen Fehler nicht zu oft machen!

Wenn ein File gesichert ist, erscheint im Katalog zwischen dem Namen und dem File-Typ ein S (secured). Das bedeutet, daß dieser File weder gelöscht noch verändert werden kann. Ein Lesen der gespeicherten Daten ist aber möglich. Wenn Sie die Eintragungen im File ändern wollen, muß zuerst die Sicherung aufgehoben werden:

unsecure altdaten

Nun ist der File wieder freigegeben! Ein sehr guter Einfall ist es, nach der Freigabe des Files die Kanalnummer zuzuordnen und dann den File sofort wieder zu sichern. Dann können Sie zwar mit PRINT-Anweisungen die Daten im File ändern (weil bei der Kanalnummern-Zuordnung der File ungesichert war), ein Löschen ist aber nicht möglich (weil der File sofort wieder gesichert wurde)! Auch das steht nicht im Benutzerhandbuch!

Mitunter ist es auch praktisch, von einem File eine Kopie anzufertigen, mit der sich unbeschwert experimentieren läßt, weil für den "Notfall" immer noch das Original vorhanden ist:

COPY altdaten TO versuch

Diese Anweisung erzeugt einen (ungesicherten) File namens VERSUCH, der sich nur durch den Namen vom (gesicherten) File ALTDATEN unterscheidet.

Wenn Sie über einen Kartenleser, ein Kassetten- oder Disketten-Laufwerk oder einen anderen Massenspeicher verfügen, können Sie jeden File auch auf das Medium dieser Massenspeicher kopieren. Da lassen sich ganze Wagenladungen von Daten und Files ablegen, die im Moment nicht benötigt werden und nur den Speicher des Rechners belegen. Sie können diese Daten und Files bei Bedarf leicht wieder in den Speicher des Rechners zurückholen. Die geeigneten BASIC-Anweisungen für diese Operationen werden in diesem Buch im Kapitel 10 ausführlich erklärt.

Der File-Typ TEXT

Im Gegensatz zu den SDATA-Files, die nur Zahlenwerte speichern konnten, sind die TEXT-Files nur für Strings geeignet, die sich in ihnen aber auch nur sequentiell speichern lassen. Außer diesem "zweckbedingten" Unterschied haben SDATA- und TEXT-Files zwar einige Gemeinsamkeiten, aber auch viele Unterschiede. Zum Anlegen eines TEXT-Files ist folgende Anweisung nötig:

```
create text neuworte
```

Diese Eingabe erzeugt ein TEXT-File mit dem Namen "NEUWORTE", dessen Umfang beim Anlegen in keiner Weise festgelegt werden kann. Um diesen File zu öffnen ist, wie üblich, die Zuordnung einer Kanalnummer erforderlich:

```
assign #2 TO neuworte
```

Nun können Sie im TEXT-File Strings ablegen, wie z.B. diesen:

```
PRINT #2;"Dies ist ein Versuch!"
```

Das fortlaufende Ablegen von Strings aus einer oder mehreren PRINT-Anweisungen erfolgt sequentiell. Um den File-Zeiger brauchen Sie sich deshalb nicht zu kümmern, er bleibt immer hinter dem zuletzt abgelegten String stehen!

Das Lesen von Strings aus TEXT-Files kann sequentiell oder im freien Zugriff erfolgen: Dazu können Sie den Record, an dem das Lesen beginnen soll, in der RESTORE- oder READ-Anweisung angeben. Wieviel Records gelesen werden, hängt von der Zahl der Variablen ab, die in der READ-Anweisung stehen. Nach Abschluß des Lesens steht der File-Zeiger wieder hinter dem zuletzt gelesenen Record. Wiederholtes Lesen setzt die Neupositionierung des File-Zeigers voraus!

Etwas tückisch sind die TEXT-Files bei der Ausführung erneuter PRINT-Anweisungen: Der abzulegende String wird in den Record gesetzt, auf den der File-Zeiger weist, die dort eventuell schon abgelegten Daten werden überschrieben und gehen verloren! Der eventuell schon früher besetzte Teil des TEXT-Files, der hinter den neuen Daten liegt, ist ebenfalls verloren!

Wenn Sie PRINT-Anweisungen nach einer Unterbrechung, die den File-Zeiger verstellt haben könnte, weiter fortsetzen wollen, müssen Sie sicherstellen, daß der File-Zeiger hinter dem letzten Record steht, der erhalten bleiben soll!

Sie müßten eigentlich erst alle Records mit READ-Anweisungen lesen, bis Ihnen der Rechner mit der Fehlermeldung "ERR:End of File" signalisiert, daß ab dieser Stelle neue Daten abgelegt werden können. Es geht aber auch einfacher!

Die TEXT-Files haben eine Eigenschaft, die sowohl den SDATA- als auch den DATA-Files abgeht: Sie können in einem TEXT-File den File-Pointer mit der RESTORE-Anweisung hinter den letzten Record setzen. Dazu ist es nicht nötig, daß Sie die genaue Zahl der besetzten Records kennen! Nehmen Sie nur einen Wert, der mit Sicherheit groß genug ist. Geben Sie bitte nacheinander folgendes ein:

```
b$="Eintragung in Record "
create text versuch @ assign #3 TO versuch
FOR i=0 TO 10 @ PRINT #3;b$&str$(i) @ NEXT i
```

In den Records 0 bis 10 sind jetzt (numerierte) Texte abgelegt, die mit der folgenden Anweisung (zur Abwechslung mal rückwärts) gelesen und angezeigt werden:

```
FOR i=10 TO 0 step-1 @ read #3,i;a$ @ a$ @ NEXT i
```

Wenn Sie jetzt hinter dem Record 10 weitere Daten ablegen wollen, ist die nächste Eingabe eine Möglichkeit, den File-Zeiger hinter den letzten Record zustellen (den Record 25 gibt es mit Sicherheit noch nicht!) und dann die Eintragung vorzunehmen:

```
restore #3,25
PRINT #3;"Neuer Text!"
```

Jetzt können Sie kontrollieren, ob die alten Records noch existieren und wo die neue Eintragung steht:

```
restore #3 @ FOR i=0 TO 11 @ read #3;a$ @ a$ @ NEXT i
```

Sie werden alle Eintragungen in den Records 0 bis 10 unverändert wiederfinden und im Record 11 den neuen Text!

Nach dieser etwas harten Kost noch etwas Erfreuliches: TEXT-Files sollen zwar nur Strings speichern können, und Sie müßten eigentlich alle numerischen Ausdrücke zuvor der STR$-Funktion unterwerfen, wenn Sie auch einmal eine Zahl im TEXT-File unterbringen wollen. **Aber** es gibt immer eine Ausnahme: Ein String, aus numerischen Zeichen (Ziffern 0 bis 9, Dezimalpunkt, Vorzeichen und e oder E) ist, der eine Zahl ergibt, kann (mit oder ohne Stringabgrenzungszeichen " oder ') in einem TEXT-File abgelegt werden, wie dieses Beispiel zeigt:

```
PRINT #3;1.2345e-235
```

Der Rechner führt diese Anweisung ohne Fehlermeldung aus, und Sie können später durch die Art der gewählten Variablen entscheiden, ob dieser Record als Zahl oder als String gelesen wird. Der Unterschied in der Anzeige ist gering und nur dann zu erkennen, wenn kein führendes Minuszeichen vorhanden ist, wie das folgende Beispiel zeigen soll!

Dieser "Zwitter"-Record wird als Zahl gelesen, wenn die Anweisung so aussieht:

```
read #3,12;x @ x        ergibt eine Anzeige als
 1.2345E-235

read #3,12;x$ @ x$      ergibt dagegen als Anzeige
1.2345E-235
```

Sie sehen, ein führendes Minuszeichen würde den Unterschied völlig verwischen! Das ist aber unwichtig, da ja die gelesenen Daten meistens weiterverarbeitet werden und der Rechner an der Art der Variablen sehr schnell merkt, womit er es zu tun hat. Diese Fähigkeit der TEXT-Files erspart Ihnen das Benutzen der STR$-Funktion vor dem Ablegen von Zahlenwerten und das Benutzen der VAL-Funktion nach dem Lesen von "Text"-Daten, die "eigentlich" Zahlen sind!

Trotzdem sollten Sie sich beim Ablegen von numerischen Werten in TEXT-Files etwas zurückhalten, weil der Platzbedarf für Zahlen dabei unnötig groß ist. Das obige Beispiel beansprucht in einem TEXT-File 11 Bytes, in einem SDATA- oder DATA-File wäre der gleiche Effekt (auch bei maximaler Stellenzahl) mit 8 Bytes zu erreichen.

Der File-Typ DATA

Wenn Sie nun sowohl Zahlenwerte als auch Text speichern wollen, dann sind SDATA-Files gar nicht und TEXT-Files nur bedingt brauchbar. Die Lösung bringen dann die DATA-Files, die beide Datenarten (auch gemischt) bequem aufnehmen können. Wie Sie sich erinnern werden, konnten SDATA-Files immer nur eine Zahl in einem Record speichern. Das ist bei DATA-Files ganz anders! Dieser File-Typ kann in einem Record soviel Zahlen und Strings aufnehmen, wie Sie wünschen! Die Sache hat nur einen Haken: Sie müssen vorher wissen, was alles in einen Record hinein soll! Das ist aber nicht schwierig, wie ein Beispiel zeigen soll:

Angenommen, Sie wollen sich ein Verzeichnis der Telefon-Nummern Ihrer Bekannten als DATA-File anlegen: Weil DATA-Records mehrere Dinge enthalten können, erscheint es vernünftig, für jeden Namen und die dazugehörige Telefon-Nummer einen Record zu verwenden.
Damit kommen wir zur wichtigsten Frage, wieviel Platz ein Record bieten muß! Wenn sich diese Frage nicht eindeutig beantworten läßt, sollten Sie keinen DATA-, sondern einen TEXT-File verwenden, bei dem dieses Problem nicht auftritt.

Wenn Sie sich bei den Telefon-Nummern auf Orts-Anschlüsse beschränken, dann dürfen die "Nummern" ausschließlich aus (bis zu 12) Ziffern aufgebaut sein und auf keinen Fall mit einer Null beginnen. Nur dann ist eine Speicherung als Zahl möglich!

Fern- oder Auslands-Ruf-Nummern (oder auch solche, die intern Leerzeichen, Bindestriche, Klammern o.ä. enthalten oder mit einer Null beginnen) können dagegen nur als String abgelegt werden. (Sie merken daran, von welchen "Kleinigkeiten" solche Entscheidungen abhängen!) Nun zum Namen: Es dürfte unzweckmäßig (und unötig) sein, für den Namen mehr als 21 Zeichen vorzusehen, weil der Rechner sowieso nur 22 Zeichen gleichzeitig anzeigen kann. Weiter dürfte es günstig sein, zuerst den Namen und dahinter die Telefon-Nummer anzuzeigen.

Nun zur "Berechnung" der Record-Größe:

Maximale Zeichen-Anzahl des Strings plus
3 Bytes für jeden String plus
8 Bytes für jede Zahl

Für unser "Orts"-Verzeichnis kommen wir damit auf 21+3+8=32 Bytes für jeden Record. Ein "Welt"-Verzeichnis benötigt dagegen 22+3+22+3=50 Bytes pro Record. Das ist zwar mehr Platz, aber dafür können Sie sich dann auch mit Klammern, Bindestrichen und ähnlichen "Lese"-Hilfen in der Rufnummer austoben (weltweite Telefon-Nummern sollen mit maximal 13 Stellen auskommen, wobei eingeschobene Zeichen, die keine Ziffern sind, nicht mitgezählt werden)!

Wie man einen DATA-File einrichtet

Beim Einrichten eines DATA-Files müssen Sie festlegen, wieviel Records der File enthalten soll und wieviel Bytes jeder Record umfassen soll. Wenn wir ein Rufnummern-Verzeichnis für den "Orts"-Bereich mit 50 Eintragungen anlegen wollen, muß die Anweisung zum Anlegen des DATA-Files so aussehen:

```
create data ort1,50,32
```

Der Rechner legt dadurch einen DATA-File mit dem Namen ORT1 an, in dem Sie 50 Namen und Rufnummern ablegen können. Sie müssen sich hier nur merken, daß in der Anweisung auf die Zahl der Records die Zahl der Bytes pro Record folgen muß! Das ist auch logisch, denn bei den SDATA-Files war nur die erste Angabe (Zahl der Records) nötig, weil die Record-Länge bei den SDATA-Files grundsätzlich auf 8 Bytes (ausreichend für eine Zahl) festgelegt ist.

Die Zuordnung einer Kanalnummer geschieht auf die gleiche Weise, wie bei den bisher behandelten File-Arten:

```
assign #3 TO ort1
```

Von nun an brauchen Sie nicht mehr ORT1 zu sagen, sondern müssen #3 als Namen verwenden, wenn Sie in dem Orts-Rufnummern-Verzeichnis neue Daten ablegen oder lesen wollen.

Freier (direkter) Zugriff beim Ablegen

Um Daten in einem bestimmten Record eines DATA-Files abzulegen, müssen Sie die Kanalnummer, die Record-Nummer und die abzulegenden Daten zu einer Anweisung zusammenfassen:

```
PRINT #3,5;"Karlchen Einfalt",1234567
```

Damit wird Karlchen's Name und seine Rufnummer im Record 5 abgelegt. Merken müssen Sie sich, daß als erstes ein String und danach eine Zahl abgelegt wurden, damit Sie in einer späteren READ-Anweisung auch geeignete Variable in der richtigen Reihenfolge angeben! Anders als beim TEXT-File erwartet ein DATA-File für eine Zahl eine numerische Variable und für einen String eine String-Variable. Großzügigkeiten sind hier nicht gestattet!

Sequentieller Zugriff beim Ablegen

Diese Art des Zugriffs ist im Prinzip möglich, ist aber durch die Notwendigkeit erschwert, den von den Daten beanspruchten Speicherplatz laufend mit dem Umfang des Records zu vergleichen. Wenn Sie vorher nicht genau wissen, was Sie speichern müssen, sollten Sie lieber die Finger davon lassen!

Freier (direkter) Zugriff beim Lesen

Die Anweisung zum Lesen der Daten eines bestimmten Records muß außer der Kanalnummer und der Record-Nummer geeignete Variable in der richtigen Reihenfolge enthalten (sonst Fehlermeldung "ERR:Data Type"), die die gelesenen Daten aufnehmen können:

```
read #3,5;n$,p @ n$;p
```

Diese Anweisung liest aus dem Record 5 zuerst einen String und dann eine Zahl und zeigt danach die beiden damit besetzten Variablen an. Wenn Sie die bisherigen Beispiele korrekt ausgeführt haben, sollten Sie jetzt folgendes sehen:

```
Karlchen Einfalt 1234567
```

Anders als bei den SDATA-Files können Sie DATA-Files erst lesen, wenn zuvor etwas dort abgelegt wurde. Aus diesem Grund legen viele Programmierer in neu angelegten DATA-Files Nullen oder Leerzeichen ab. Ich bevorzuge dagegen eine Methode, bei der ich im Record 0 notiere, welche Records belegt sind und welche nicht: Ein String wird aus "0" (für "frei") und "1" (für "belegt") so zusammengesetzt, daß das n-te Zeichen dem Zustand des Record "n" entspricht und dann im Record 0 abgelegt. Lesen des Records 0 und Abfrage der entsprechenden Zeichen verrät dann die Zustände der interessierenden Records im File.

Sequentieller Zugriff beim Lesen

Wie bei den SDATA-Files führt auch bei den DATA-Files das Weglassen der Record-Nummer in der READ-Anweisung grundsätzlich zum "Weiterlesen". Aber Vorsicht ist geboten, weil DATA-Files in jedem Record viele Daten enthalten können (die SDATA-Files enthielten ja pro Record immer nur eine einzige Zahl)! Schauen wir uns das mal an:

```
read #3;n$,p
```

Diese Anweisung muß (in unserem Beispiel) den Record 6 lesen, weil der File-Zeiger, von der letzten Anweisung her, am Ende des Records 5 stehen blieb.

```
read #3;n$
```

Diese Anweisung (anschließend eingegeben), hätte zur Folge, daß aus dem Record 7 nur der Name gelesen wird! Der File-Zeiger steht nun mitten im Record 7, hinter dem Namen, aber vor der Zahl. Ein Zustand, den wir bisher noch nicht kannten! Wenn wir den File-Zeiger da nicht wegnehmen, kann als nächstes nur die im Record 7 stehende Zahl gelesen werden. Die Variablen-Liste der nächsten READ-Anweisung ohne Record-Nummer muß unbedingt mit einer numerischen Variablen beginnen, wenn wir die Fehlermeldung "ERR:Data Type" vermeiden wollen!

Ablegen und Lesen in mehreren Records mit einer Anweisung

Es tut mir leid, aber das geht nicht! Erinnern Sie sich noch, wie sich mit einer READ-Anweisung ein ganzes Bündel von Daten aus SDATA-Files lesen ließ? Das dürfen Sie mit DATA-Files nicht machen! Wenn Sie es versuchen, erhalten Sie die Fehlermeldung "ERR:Record Ovfl", sobald das Ende des Records erreicht ist.

Gewiß, ich muß jetzt zugeben, nicht die volle Wahrheit gesagt zu haben! Es gibt ein Verfahren, daß gut funktioniert, wenn es sich ausschließlich um Strings oder ausschließlich um Zahlen handelt. Dann können Sie ein Feld oder eine Matrix so einfach lesen, wie es schon bei den SDATA-Files möglich war. Denken Sie aber daran, daß das Ablegen eines Feldes oder einer Matrix ein sequentieller Zugriff ist, und damit ist ein DATA-File einfach überfordert (wie schon weiter vorn erwähnt). Mir ist es einmal gelungen, auf die erste Hälfte eines DATA-Files frei und auf die zweite Hälfte sequentiell zuzugreifen. Wenn ich mir aber nach einer Pause von einigen Wochen überlegte, wie das alles funktionieren sollte, bekam ich meistens üble Kopfschmerzen! Machen auch Sie es sich zum Grundsatz, nur solche Verfahren anzuwenden, die Ihnen auch nach längeren Pausen noch einigermaßen plausibel und übersichtlich erscheinen. Der Rechner soll uns ja die Arbeit erleichtern und nicht unser Wohlbefinden mindern!

Versetzen des File-Zeigers

Die RESTORE-Anweisung wirkt auf DATA-Files in der gleichen Weise, die wir schon bei den SDATA-Files kennen lernten: Der File-Zeiger wird an den Anfang des bezeichneten Records gesetzt (bei fehlender Record-Nummer auf Record 0).

```
restore #3,5
```

Diese Anweisung setzt den File-Zeiger auf den Anfang von Record 5.

Unglücklicherweise gibt es keine Möglichkeit, den File-Zeiger gezielt mitten in einen Record zu setzen. Wenn Sie das wollen, müssen Sie den File-Zeiger auf den Anfang des Records setzen und ihn dann, durch teilweises Lesen der gespeicherten Daten auf den gewünschten Punkt bringen. Der File-Zeiger bleibt hinter der zuletzt gelesenen Eintragung stehen. Wenn diese Möglichkeit auch besteht, ist es doch von zweifelhaftem Nutzen, den File-Zeiger innerhalb des Record herumgeistern zu lassen, weil Sie während dieser Bewegungen zu leicht vergessen können, was eigentlich bezweckt war.

Weitere Eigenheiten der DATA-Files

Eine üble "Angewohnheit" des HP-71 macht eine weitere Warnung vor dem sequentiellen Abspeichern nötig: Eine "End-of-File"-Marke wird nach jeder sequentiell formulierten PRINT-Anweisung in den zuletzt beschriebenen Record gesetzt. Dadurch ist es nicht möglich, über diesen Punkt sequentiell hinwegzulesen. Glücklicherweise werden die freien Zugriffsmöglichkeiten davon grundsätzlich nicht betroffen! Das Benutzerhandbuch schildert aber die Wirkung der End-of-File-Marke zu schrecklich: Die dahinter vorhandenen Daten werden nicht gelöscht! Sie können also (auch sequentiell) sowohl vor als auch hinter der End-of-File-Marke alles lesen, nur nicht in einem Zuge! Tatsächlich verloren sind nur der Teil der Daten, die zwischen der Marke und dem Ende des "markierten" Records vorhanden waren.

Wenn Sie zuviel in einem Record ablegen wollen, erhalten Sie die Fehlermeldung "ERR:Record Ovfl". Das wird z.B. passieren, wenn Sie im ORT1-File einen Namen, der aus mehr als 21 Zeichen besteht, und eine Zahl ablegen wollen. Wenn Ihnen solche Fehlermeldungen das Leben schwer machen, ist es an der Zeit, mehr Platz zu schaffen. Dummerweise kann man aber einen bereits bestehenden DATA-File nicht nachträglich vergrößern. Sie müssen vielmehr einen neuen DATA-File (mit mehr Platz) anlegen, auch diesem eine besondere Kanalnummer zuordnen, um dann die bereits gespeicherten Daten Record für Record aus dem alten in den neuen übernehmen. Das läßt sich mit einer Schleife sogar relativ mühelos machen. Den alten File können Sie dann löschen und dem neuen File dann wieder den altgewohnten Namen geben.

Als Beispiele für den Umgang mit DATA-Files können im Kapitel 9 die Programme "GLOBETEL" und "PHONE" mit ihren Kommentaren dienen.

Der File-Typ KEY

Als eine der besten Eigenschaften des HP-71 kann es wohl gelten, fast jeder Taste des Rechners Spezialaufgaben zuordnen zu können. Wie habe ich mir bis jetzt auf die Zunge beißen müssen, um Ihnen diesen Trick nicht vorzeitig zu verraten!

Wie Ihnen inzwischen schon geläufig sein dürfte, hat fast jede Taste des HP-71 drei verschiedene Wirkungen, je nachdem ob sie allein oder erst nach Betätigen der f- bzw. g-Taste gedrückt wird. Nun gibt es aber im "USER"-Modus (der Ihnen vielleicht schon von anderen HP-Rechnern bekannt ist) die Möglichkeit, wiederum fast jeder Taste drei weitere Funktionen zuzuordnen.

Damit ist die individuelle Ausgestaltung einer noch gut übersehbaren Tastatur zur Anpassung an jedes gewünschte Spezial-Gebiet möglich, denn es wird wohl unter uns kaum ein "Genie" geben, das alle Rechner-Funktionen immer braucht und ein entsprechend "universales" Tastenfeld wirklich "beherrscht".

Der HP-71 läßt sich mit der Taste "0" des Ziffernblocks zwischen Normal- und USER-Modus hin- und herschalten, wenn zuvor die f- oder g-Taste gedrückt wurde. Dabei bewirkt die f-Taste (goldgelb: USER) den Übergang in den anderen Modus bis zur erneuten Betätigung von f-Taste und "0". Die g-Taste (blau: 1 USER) ruft dagegen den anderen Modus nur für eine Tastenfolge auf, danach fällt der Rechner wieder in den vorher gültigen Modus zurück. Sie können sich damit also für eine einzelne Eingabe im anderen Modus das Zurückschalten ersparen! (Auch das verschweigt das Handbuch halb!)

Angenommen, Sie benutzen häufig den DESTROY-Befehl (so wie ich!); und angenommen, Sie brauchen (fast) nie den RETURN-Befehl (ich brauch ihn auch selten!): Das Tastenfeld enthält also einerseits eine für Sie "überflüssige" Tastenfunktion, dafür dürfen Sie aber jedesmal Zeichen für Zeichen DESTROY eintippen, wenn Sie diesen Befehl benötigen. Das sollte man doch ändern! Versuchen Sie es mal, indem Sie folgendes eingeben:

```
key"fD","DESTROY ";
```

Nachdem Sie dann noch END LINE gedrückt haben, sollten Sie den USER-Modus einschalten und die Tasten f und D drücken: Es erscheint nicht (wie bisher) RETURN in der Anzeige, sondern DESTROY und das auch noch mit einem korrekt stehenden Cursor, der zu weiteren Eingaben einlädt!

Diese Tastenzuordnung gilt nur im USER-Modus, auf den die kleine Anzeige (links unten) hinweist. Wenn Sie den USER-Modus verlassen, wird auch diese Anzeige wieder verschwinden. Auch die Tastenzuordnung selbst ist sehr einfach zu löschen:

```
key"fD"
```
(und natürlich!) END LINE

Diese Aufhebung bewirkt, daß die Taste auch im USER-Modus nur wieder ihre Grundfunktionen ausführt.

Eine Tastenzuordnung erfordert grundsätzlich die Eingabe des Befehls KEY, dem der Tasten-Name (in Anführungszeichen) folgen muß. Nach einem Komma folgt dann (wieder in Anführungszeichen) die eigentliche Zuordnung. Als Tasten-Name und als Zuordnung können auch Stringvariable an die Stellen des Klartextes treten. Beachten Sie in der bisherigen Zuordnung das Semikolon. Es hat eine wichtige Aufgabe: Das Semikolon setzt den gewünschten Text in die Anzeige und positioniert den Cursor, für eine weitere Eingabe passend, dahinter. Es gibt aber noch weitere Varianten für die Wirkung von Tastenzuordnungen.

Eine Zuordnung, hinter der kein Semikolon steht, wirkt verblüffend: es erscheint (kurz) der zugeordnete Text in der Anzeige und dann tut der Rechner so, als wenn auch schon END LINE betätigt wurde! Weil ich sehr oft die Befehle CAT ALL und DESTROY ALL benutze, habe ich ALL der FACT-Taste (über "=") zugeordnet. Weil auf ALL immer END LINE folgen muß, lasse ich das den Rechner für mich tun:

```
key"f=","ALL"
```

Nun ist es im USER-Modus möglich, DESTROY ALL und CAT ALL mit jeweils vier Tastenbetätigungen (f D f = bzw. f X f =) einzugeben!

Die dritte Art der Tastenzuordnung ist noch geheimnisvoller und sollte daher (vorläufig) nur mit Vorsicht benutzt werden! Dazu ist die Zuordnung mit einem Doppelpunkt abzuschließen. Die Anweisung wird dann sofort ausgeführt, d.h. in der Anzeige erscheint sofort das Ergebnis der Anweisung, sie selbst bleibt (auch im Befehls-Stack) unsichtbar! Versuchen Sie mal diese Eingabe:

```
key"f(","time$":
```

Wenn Sie jetzt im USER-Modus die AUTO-Funktion (über "Klammer auf") aufrufen, wird sofort die Zeit angezeigt, ohne daß zuvor "time$" angezeigt wurde. Weil diese Art der Zuordnung gewissermaßen "verdeckt" arbeitet, sollten Sie sie nur benutzen, solange Sie noch genau wissen, was sich hinter den einzelnen Tasten verbirgt. Ein Ergebnis zu sehen, ohne zu wissen, wie es entstand, kann sehr verwirrend sein!

Wenn Sie aber vergessen haben sollten, welche Zuordnungen Sie getroffen haben, können Sie sich leicht orientieren: Sie geben zuerst die VIEW-Anweisung (über dem Punkt) ein und drücken danach nicht END LINE, sondern drücken und halten jetzt die zu untersuchende Taste! In der Anzeige erscheint dann entweder "Unassigned" oder, nach einem Semikolon, Leerzeichen oder Doppelpunkt, die getroffene Zuordnung. Das vorangehende Symbol beschreibt dabei die Arbeitsweise der Zuordnung.

Die bequemste Methode, die Zuordnung einer Taste zu verändern, führt über die FETCH KEY-Anweisung. Wenn Sie es versuchen wollen?

FETCH key"f("

Diese Anweisung bringt Ihnen die Tasten-Zuordnung in ihrer Originalform in die Anzeige:

DEF KEY 'f(','time$':

Wie Sie sehen, hat der Rechner von sich aus nur jedes Anführungszeichen in einen Apostroph verwandelt, aber der Cursor steht schon an der richtigen Stelle, falls Sie die Zuordnung ändern wollen! Das Wort DEF ist bei der Tastenzuordnung eine "freiwillige" Maßnahme (wie LET bei der Wertzuweisung für Variable).

Wenn Sie sich über alle momentan gültigen Tastenzuordnungen orientieren wollen (oder müssen!), Dann geben Sie bitte **delay 8** ein, damit jede Anzeige beliebig lange stehen bleibt, und anschließend:

LIST keys

Jetzt erhalten Sie eine übersichtliche Liste aller Zuordnungen, die Sie durch Druck auf eine (fast) beliebige Taste von oben nach unten durchwandern können. Einen "Rückweg" (wie beim Stöbern im Befehls-Stack) gibt es hier nicht. Die Shift-Tasten f und g sind zum Weiterschalten ungeeignet, RECHTS- und LINKS-Taste behalten ihre (notwendigen) Funktionen. Wenn Sie in der Liste zu weit nach "unten" fortschreiten, fallen Sie aus der Liste heraus und der Rechner kann plötzlich irgendeine Zeile des zuletzt abgelaufenen Programms anzeigen. (Verzeihung, Sie wissen ja noch nicht, was ein Programm ist!) Das ist eine offensichtliche Schwäche des Betriebs-Systems der momentan gelieferten Version des HP-71! Dieser Schönheitsfehler läßt sich mildern, wenn Sie vorher dem Rechner eine EDIT-Anweisung geben, mit der der (hoffentlich leere) "workfile" aufgerufen wird. Nachdem Sie sich ausgiebig an den Tastenzuordnungen ergötzt haben, vergessen Sie nicht, mit **delay .5** den Rechner wieder auf seine Standard-Verzögerung bei der Anzeige zurückzuschalten!

Wenn Sie jetzt mit CAT ALL mal nachschauen, was sich alles im Speicher des Rechners tut, entdecken Sie einen File, der den Namen "keys" trägt und vom Typ "KEY" ist: In diesem File sind die von Ihnen vorgenommenen Tastenzuordnungen abgelegt. Diesen File können Sie mit COPY-, PURGE- und SECURE-Anweisungen bearbeiten, aber auch mit allen anderen Anweisungen, die auf Files wirken. Diesem File gehört die besondere Aufmerksamkeit des Rechners: Sobald Sie diesem File den Namen "keys" wegnehmen, erkennt der Rechner ihn nicht mehr als Zuordnungs-File an, und alle Tasten-Zuordnungen sind aufgehoben! Das ist nun kein Fehler, sondern eine herrliche Erweiterung der Möglichkeiten, wie ich Ihnen gleich zeigen werde!

Wahlweise Benutzung mehrerer KEY-Files

Nach einiger Zeit der Benutzung werden auch Sie erkennen, daß es zweckmäßig ist, dem HP-71 für unterschiedliche Anwendungszwecke spezielle Tasten-Zuordnungen zu geben. Da der Rechner im Haupt-Speicher einerseits für jeden File einen speziellen Namen benötigt, andererseits aber nur den KEY-File mit dem Namen "keys" für die Tastenzuordnung als verbindlich ansieht, müssen Sie ausgearbeitete Tasten-Zuordnungen für spätere Verwendung mit der RENAME-Anweisung umbenennen. Es ist logisch, dafür einen kennzeichnenden Namen zu wählen.

Auf diese Weise werden Sie bald einige sehr persönliche Tasten-Belegungen im Speicher haben, auf die Sie bei Bedarf zurückgreifen können. Dazu müssen Sie nur den momentanen (und nicht mehr benötigten) KEY-File "keys" mit der PURGE-Anweisung löschen (vorausgesetzt, es existiert irgendwo bereits eine Kopie unter passendem Namen) und einen der umbenannten KEY-Files mit dem Namen "keys" in den Hauptspeicher kopieren, wodurch der Rechner die in diesem File abgelegten Tastenzuordnungen sofort aktiviert.

Der File-Typ BASIC

Endlich! Wir haben es geschafft und sind bei den BASIC-Files angekommen! Darauf warten Sie doch schon, seit Sie die Lektüre dieses Buches begonnen haben!

DATA-Files sind Sammlungen von Daten. TEXT-Files sind Sammlungen von Texten. KEY-Files sind Sammlungen von Tasten-Zuordnungen. Also werden BASIC-Files wohl Sammlungen von BASIC-Anweisungen sein.

Der Sinn des Erhaltens einer Sammlung von BASIC-Anweisungen in einem File liegt nicht in der Möglichkeit, die einzelnen BASIC-Anweisung zu betrachten und diese wieder und wieder einzeln ausführen zu lassen. Der Sinn liegt vielmehr darin, den Rechner anzuweisen, die ganze Sammlung von Anweisungen abzuarbeiten!

Nun gibt es für eine Sammlung von BASIC-Befehlen, die der Rechner in geschlossener Form ausführt, einen besonderen Namen: Man spricht von einem "Programm". Die Tätigkeit, einzelne Befehle und Anweisungen zusammenzustellen nennt man "programmieren" und jeder, der so etwas kann, ist ein "Programmierer". Wenn die verwendete Programmsprache BASIC ist, sprechen wir von einem BASIC-Programm, das von einem BASIC-Programmierer geschrieben wurde. Der HP-71 ist vorzüglich zum Schreiben von BASIC-Programmen. Weil er BASIC beherrscht, kann er BASIC-Anweisungen zu einem BASIC-File zusammenstellen und den gesamten, mit BASIC-Anweisungen gefüllten File in einem Anlauf abarbeiten. Und das als Rechner, der wirklich noch in die Hand paßt!

Als Sie vorhin CAT ALL ablaufen ließen, haben Sie sicher einen File mit dem Namen "workfile" vom Typ BASIC bemerkt. Mit dem wollen wir jetzt etwas spielen!

Um zum "workfile" zu kommen, ist nur der EDIT-Befehl (über dem Z) nötig. Um sicher zu gehen, daß "workfile" leer ist, führen wir eine PURGE-Anweisung (über dem V) durch. Sie haben jetzt ein leeres "workfile" vor sich, mit dem Sie nun spielen dürfen.

Obwohl ich Ihnen das Programmieren erst in einem späteren Kapitel erklären will, sollen Sie schon hier eine Kostprobe bekommen! Geben Sie bitte die folgenden Zeilen ein, indem Sie am Ende jeder Zeile END LINE betätigen:

```
10 "Ich bin Programmierer!"
20 BEEP 800
30 "Das ist aber einfach!"
40 BEEP 650
50 GOTO 10
```

Dies ist ein Programm, also eine Liste von fünf Anweisungen. Haben Sie es bemerkt? Keine der Anweisungen wurde ausgeführt!

Der Grund für dieses bisher nicht beobachtete Verhalten liegt in der Tatsache, daß jede Zeile mit einer Nummer beginnt, die man folgerichtig "Zeilennummer" nennt: Immer, wenn vor einer Anweisung eine Nummer steht, führt der HP-71 diese nicht aus, sondern versteht sie als Teil eines Programmes und ordnet sie entsprechend der Zeilennummer dort ein. In unserem Fall wurden die Zeilen im "workfile" abgelegt. Mit den AUF- und AB-Tasten können Sie sich alle Zeilen des Programms ansehen. Starten Sie dann das Programm mit der RUN-Taste und ergötzen Sie sich an den Anzeigen! Zum Anhalten brauchen Sie nur die ATTN-Taste (ON) zu drücken.

Der "workfile" soll für den Programmierer eine Art Notizbuch sein. Lassen Sie darum niemals wichtige Programme im "workfile"! Fertige Programme sollten immer kennzeichnende Namen bekommen. Wenn für ein Programm der Name von Anfang an festgelegt werden kann, verläuft die "Taufe" nach folgendem Schema:

EDIT ulamconj

Diese Anweisung legt einen BASIC-File mit dem Namen ULAMCONJ an. Wenn Ihnen dieser Name nicht gefällt, nehmen Sie einen anderen! Die Vorschriften für einen File-Namen kennen Sie ja inzwischen: Er darf höchstens 8 Zeichen umfassen, muß mit einem Buchstaben beginnen und darf nur die Zeichen A bis Z (bzw. a bis z) und die Ziffern 0 bis 9 enthalten. Sie können auf jeden BASIC-File die Anweisungen RENAME, PURGE, SECURE und andere anwenden. Mehr darüber erfahren Sie etwas später!

Und nun kommt (wegen der restlichen drei File-Typen) die längst fällige Enthüllung: Der HP-71 "versteht" weder BASIC noch FORTH noch sonst irgendeine andere "höhere" Programmiersprache! Er "kann" BASIC nur so, wie Sie z.B. Griechisch "können", weil Sie von dieser Sprache ein Wörterbuch und eine Grammatik besitzen! Damit "können" Sie (mit viel Geduld!) einen Artikel aus dem Griechischen in ihre Muttersprache übersetzen. So macht es auch der HP-71, er übersetzt, Bit für Bit die BASIC-Anweisungen, -Funktionen und -Befehle in seine eigene Sprache, die er wirklich beherrscht.

Die "Mutter"-Sprache des HP-71 heißt (wie bei allen Rechnern) "Maschinensprache". Sie besteht aus einem fürchterlichen Kuddel-Muddel von Zahlen und ist für den Menschen vollkommen unübersichtlich. Der Rechner versteht das aber und zwar sehr schnell! Ein in Maschinensprache geschriebenes Programm läuft deshalb zig-mal schneller ab, als ein BASIC-Programm, das die gleiche Aufgabe erledigen soll (auch andere höhere Programmiersprachen sind da nicht viel besser!). Ärgerlich ist dabei nur eines: An der Programmierung einer bestimmten Aufgabe, die ein halbwegs erfahrener Programmierer in BASIC in 10 Stunden löst, kann in Maschinensprache ein Experte 10 Jahre arbeiten! Hier ist also Teamwork nötig! Programmieren in Maschinensprache ist also immer sehr mühevoll und zeitraubend. Die File-Typen BIN und LEX sind Beispiele für (meist kurze) Programme in Maschinensprache.

Der File-Typ BIN

Ein BIN-File (von BINary) enthält ein in Maschinensprache abgefaßtes (Hilfs-)Programm, mit dem für bestimmte Sonderanwendungen auch völlig neue Rechnerroutinen und Schlüsselwörter geschaffen werden. Mit einem BIN-File lassen sich (meist individuell und in kleinerem Umfang) ähnliche Erweiterungen erreichen, wie sie im Prinzip mit einem ROM möglich sind.

Der File-Typ LEX

Auch ein LEX-File (von Language EXtension) bietet die Möglichkeit, in Maschinensprache neue Funktionen und Schlüsselwörter zu schaffen. Dabei sind aber die Eingriffe in das Betriebssystem des Rechners nicht so weitreichend, wie bei den BIN-Files.

Für BIN- und LEX-Files gilt gemeinsam, daß sie in einer Sprache geschrieben werden, die für den Menschen zwar auch noch recht mühsam ist, bei der aber ganze Komplexe der Maschinensprache schon zusammengefaßt sind. Diese Sprache heißt "Assembler". Das Erstellen von BIN- und LEX-Files sollte Menschen vorbehalten bleiben, die dafür eine Begabung aufweisen und nichts Besseres zu tun haben! Wenn Sie das FORTH/ASSEMBLER-ROM besitzen, "können" Sie selbst Programme in der Assemblersprache schreiben, die der Rechner dann in Maschinensprache umsetzt, damit sie recht schnell ablaufen. Raten Sie mal, warum das Wort können auch hier wieder in Anführungszeichen steht! Wie gesagt, Sie müssen dafür begabt sein und nichts Dringenderes zu tun haben!

Der File-Typ FORTH

Mit dem Einsetzen des FORTH/ASSEBLER-ROM's geben Sie dem HP-71 "Grammatik und Wörterbuch" für die Programmiersprache FORTH. Sie "können" nun Programme in FORTH schreiben und ausführen lassen. Es gilt als erwiesen, daß FORTH-Programme schneller ablaufen als BASIC-Programme. Ich weiß nur, daß sich BASIC-Programme schneller als FORTH-Programme schreiben lassen, sobald die Probleme etwas komplizierter sind. Mag FORTH im Ablauf schneller sein, dafür ist (mir) BASIC beim Programmieren angenehmer!

Obwohl dieses Buch eigentlich nur BASIC behandeln soll, werde ich Ihnen am Schluß doch einen Weg zeigen, wie Sie auch ohne das FORTH/ASSEMBLER-ROM BIN- und LEX-Files in den Speicher bringen können! Einige wunderhübsche LEX-Files werden dann in den stillen Winkeln des Speichers herumgeistern, ohne daß Ihr HP-71 jemals ein FORTH/ASSEMBLER-ROM gesehen hat!

Kapitel 7

Kalender und Uhr des HP-71

Von allen Fähigkeiten, die der HP-71 hat, macht die Uhr/Kalender-Funktion wahrscheinlich den geringsten technischen Aufwand, obwohl diese Funktion gar nicht so einfach zu beschreiben ist! Das Benutzerhandbuch war mir nur eine sehr geringe Hilfe dabei. Ich hoffe, daß Sie meine Erklärungen besser verstehen.

Zuerst sollten Sie das Datum auf den richtigen Wert bringen:

setdate "86/04/16"

Diese Anweisung setzt das Datum auf den 16. 4. 1986. Beachten Sie die Reihenfolge "Jahr/Monat/Tag", die zwar nicht der üblichen Gewohnheit, wohl aber der sonst bei einfachen Zahlen benutzten Logik (und einer schon lange bestehenden Empfehlung) entspricht. Die wichtigste Angabe steht ganz links! Auch hier wieder etwas Lästerhaftes: Wissen Sie, warum in der wissenschaftlichen bzw. technischen Darstellung von Werten die Größenordnung ganz rechts steht? Ich weiß es auch nicht!

Zur Abfrage des Datums ist einzugeben:

date$

Diese Anweisung ("DATE-String" genannt) zeigt Ihnen das Datum im eingegebenen Format an, also 86/04/16. Wenn Sie das Datum morgen abfragen (nachdem mindestens 24 Stunden vergangen sind), werden Sie 86/04/17 als Anzeige erhalten, auch wenn die Uhr des HP-71 noch nicht richtig eingestellt sein sollte.

Sie können das Jahr auch vierstellig als 1986 eingegeben, das ist aber ("vorläufig") nicht notwendig und hat auch auf die Anzeige keinen Einfluß. Nur wenn Sie dieses Buch tatsächlich erst nach dem Jahr 2059 lesen sollten oder schon vor dem Jahr 1960 gelesen hätten (was beides etwa gleich unwahrscheinlich ist), müßten Sie die Jahreszahl mit allen vier Stellen eingeben. Auch dann werden nur die letzten beiden Stellen des Jahres in der Anzeige erscheinen.

Bevor Sie nun die Uhr des HP-71 stellen, sollten Sie an zwei wichtige Tatsachen erinnert werden:

Es gibt erstens keine (unabhängige) Uhr, die absolut genau geht! Jede Uhr wird, bei genügend langer Beobachtungszeit, gegenüber einer "Normal"-Uhr etwas vor- oder nachgehen. Diese Abweichung wird "Gang" oder "Drift" genannt. Bei minderwertigen Uhren kann die Drift dem Betrag nach sehr groß sein und von vielen Einflüssen, wie Temperatur, Feuchtigkeit, Lage und anderen unvermuteten Dingen, abhängen.

Gute Uhren haben dagegen eine sehr geringe und konstante Drift. Die mit der Zeit anwachsende Abweichung kann dann durch eine simple Rechnung korrigiert werden, wenn die Drift bekannt ist.

Die zweite Tatsache ist, daß die im HP-71 eingebaute Uhr von erstaunlicher Qualität ist und noch die Möglichkeit bietet, die Drift automatisch zu ermitteln und deren Einfluß auf die Zeitangaben auszuschalten, wodurch Sie einen praktisch perfekten Zeitmesser erhalten! Es wird dafür aber stillschweigend ein durchschnittliches Raumklima vorausgesetzt. Eine Aufbewahrung des HP-71 in einem Tiefkühlfach oder im Handschuhfach eines Autos bei der Fahrt durch die Sahara wird der Uhr sicher nicht bekommen und evt. zu einem Totalausfall führen! Der HP-71 kennt übrigens für die Uhrzeit nur das 24-Stunden-Format: Der Tag beginnt um Mitternacht mit 00:00:00, die letzte Sekunde davor (also auch vor dem Datumswechsel) heißt 23:59:59.

Der Abgleichfaktor, der die Drift kompensiert, wird in den deutschen HP-Veröffentlichungen meist (richtig) als "Anpassungsfaktor", in den englischen Texten jedoch häufig (sinnwidrig) als "Accuracy Factor" bezeichnet. Wenn Sie den HP-71 von einem Vorbenutzer erworben haben, sollten Sie dem Abgleichfaktor mißtrauen, es sei denn, der Vorbenutzer schwört einen heiligen Eid, daß der Abgleichfaktor stimmt!

Und jetzt sollen Sie endlich erfahren, wie Sie Uhr und Abgleichfaktor einstellen können! Wenn der Abgleichfaktor angezweifelt werden muß, ist folgende Eingabe angebracht:

reset clock

Diese Eingabe hat keinen Einfluß auf die Zeiteinstellung, sondern löscht nur den (angezweifelten) Abgleichfaktor und die (eventuell) inzwischen aufsummierte Abweichung. Als nächstes könnten Sie dann die Uhr stellen:

settime "20:00:00"

Das wäre z.B. eine geeignete Eingabe, um den Rechner nach dem Beginn der "Tagesschau" (um 20.00 Uhr) zu stellen. Aber auch die Rundfunksender bringen zu (fast) jeder vollen Stunde ein kurzes Zeitzeichen (das oft auch Musikaufnahmen stört!). Beim Stellen der Uhr müssen Sie auch berücksichtigen, daß die Übernahme des eingegebenen Wertes erst mit END LINE erfolgt! Eine Eingabe dieser Art dürfte deshalb oft zweckmäßiger sein:

settime "20:00:01"

Dann können Sie im Fernsehen oder im Rundfunk das Ende der vorangehenden Minute abwarten und unmittelbar danach END LINE betätigen. Das ist für die meisten Zwecke bereits genau genug und überfordert nicht Ihr Reaktionsvermögen!

Wenn Sie jetzt (oder irgendwann) wissen wollen, wie spät es ist, hilft Ihnen folgende Eingabe:

```
time$
```

Die "TIME-String"-Funktion bringt dann die Zeit im Augenblick der Abfrage in die Anzeige. Nach dem Stellen der Uhr sollten Sie dem Rechner möglichst bald mitteilen, daß diese Zeit tatsächlich stimmt:

```
exact
```

Auch die EXACT-Anweisung hat keinen direkten Einfluß auf die Funktion der Uhr, es wird nur der momentane Zeitpunkt (mit Datum) gespeichert um daraus beim nächsten Zeitvergleich Schlüsse ziehen zu können. Nun lassen Sie ein oder zwei Wochen vergehen, ohne sich besonders um die Uhr zu kümmern. Auch die kleinen Abweichungen, die Sie bei der Abfrage der Zeit feststellen, dürfen Sie nicht beunruhigen! Dann aber können Sie die Uhr mit der SETTIME-Anweisung berichtigen, den neuen Zustand nochmals kontrollieren und den Rechner mit der EXACT-Anweisung auf die Korrektur hinweisen: Der Rechner ermittelt jetzt die bisher aufgelaufene Abweichung, berechnet daraus den Abgleichfaktor und korrigiert von nun an alle Zeitangaben vor der Ausgabe. Jetzt dürfte die Uhr schon recht genau gehen. Eine störende Abweichung werden Sie erst wieder nach einigen Monaten bemerken können. Erneute Anwendung der SETTIME- und EXACT-Anweisung bringt eine weitere Verbesserung.

Nun kann es auch vorkommen, daß Sie die Uhr des HP-71 korrekt abgeglichen haben und dann entdecken, daß die Vergleichsuhr zwar keine Drift hat, dafür aber um einen bestimmten Betrag falsch geht! Ein ähnliches Problem tritt auf, wenn Sie auf einer Reise in eine andere Zeitzone gelangen oder es bei Ihnen üblich ist, zwischen Winter- und Sommerzeit zu unterscheiden! Mit SETTIME und EXACT würden Sie den langwierigen Abgleich zunichte machen, es gibt aber einen anderen Weg mit der ADJABS-Anweisung (ADJust ABSolute), die den Abgleichfaktor der Uhr nicht beeinflußt. Damit läßt sich die Uhr um die gewünschte Zahl von Stunden, Minuten und Sekunden vor- oder zurückstellen:

```
adjabs "01:00:00"
```

Diese Anweisung stellt die Uhr um eine Stunde weiter, was z.B. beim Übergang von der Winter- zur Sommerzeit nötig ist. Die Rückkehr zur Winterzeit müßte dann so eingeleitet werden:

```
adjabs "-01:00:00"
```

Wenn Sie nun ein Genauigkeitsfanatiker sind, ist es denkbar, daß Sie die Uhr auch um Bruchteile einer Sekunde verstellen wollen. Dazu ist nur das Argument von ADJABS in anderer Form, nämlich als numerischer Ausdruck in Sekunden, einzugeben!

Wenn es Sie stört, daß die Uhr um 0,3 Sekunden vorgeht, bringt dies die Uhr wieder auf den Pfad der Korrektheit zurück:

adjabs -.3

Diese Anweisung stellt die Uhr um 0,3 Sekunden zurück (also nach). Auf diese Weise läßt sich auch die Verzögerung ausgleichen, die beim Stellen der Uhr nach Rundfunk- oder Fernseh-Uhr durch den Menschen verursacht wird (etwa 0,3 Sekunden). Aber auch der HP-71 ist bei der Ausführung von SETTIME nicht sehr schnell, er braucht immerhin etwa 0,5 Sekunden für diese Anweisung. Es ist also angebracht, die Uhr nach dem Stellen mit der SETTIME-Anweisung um 0,8 Sekunden vorzustellen:

adjabs .8

Womit dann die Stellprozedur beendet wäre!

Sobald die Uhr mit ausreichender Genauigkeit läuft, sollten Sie sich für den nun geltenden Abgleichfaktor interessieren, der sich sehr einfach abfragen läßt:

af

Die AF-Anweisung bringt den Abgleichfaktor in die Anzeige und es ist zweckmäßig, wenn Sie sich diesen Wert vorsichtshalber notieren. Wenn Sie dann mit dem Rechner ein "memory lost" fabrizieren (oder provozieren), dauert es nicht wieder Wochen und Monate, um die Uhr erneut abzugleichen! Der bereits gefundene Abgleichfaktor (z.B. 1035) wird einfach so gesetzt:

af(1035)

Die AF-Anweisung ermittelt also nicht nur den momentanen Abgleichfaktor, sondern setzt auch den angegebenen Wert in den Rechner. Denken Sie aber daran, vorher EXACT auszuführen!

Nun gibt es außer der recht klar wirkenden ADJABS-Anweisung noch eine weitere, ähnliche Anweisung, die aber mit Vorsicht zu genießen ist: ADJUST benötigt die gleichen Eingaben wie ADJABS, wirkt aber nur dann in gleicher Art, wenn das Argument ein ganzzahliges Vielfaches von 30 Minuten ist! Bei Abweichungen von diesem Halbstunden-Raster wird außer dem Stellen der Uhr die Differenz zum nächsten Rasterpunkt als Abweichung gewertet, einer eventuell bereits ermittelten Abweichung zugerechnet und bei der nächsten Ausführung der EXACT-Anweisung zum Verändern des Abgleichfaktors ausgewertet. Die ADJUST-Anweisung sollten Sie vorerst nur benutzen, wenn Sie genau wissen, um wieviel die Uhr verkehrt geht (es ersetzt dann das umständlichere SETTIME), zum Stellen der Uhr ist aber ADJABS gefahrloser! Einige ausführlich kommentierte Beispiele mit typischen Argumenten und deren Auswirkungen sollen das verdeutlichen:

Zuerst der "harmlose" Fall, bei dem die Uhr nur um den gewünschten Betrag vor- oder zurückgestellt wird:

adjust "01:00:00" oder **adjust 3600**

Beide Anweisungen stellen die Uhr um 1 Stunde (3600 Sekunden) weiter (also vor), ohne für den Abgleichfaktor wirksam zu werden.

Nun der zweite Effekt in "Reinkultur": Wenn Sie festgestellt haben, daß die Uhr um 15 Sekunden vorgeht, ist eine dieser Anweisungen sinnvoll:

adjust "-00:00:15" oder **adjust -15**

Hierdurch wird die Uhr um 15 Sekunden zurückgestellt, außerdem wird aber eine Abweichung von +15 Sekunden seit der letzten Ausführung von EXACT bzw. ADJUST festgestellt und bei der nächsten Berechnung des Abgleichfaktors mit berücksichtigt (die nächste EXACT-Anweisung macht die Uhr dann "langsamer").

Und nun, zum Abschluß, die Kombination beider Möglichkeiten, die zwar nicht schwierig ist, aber etwas Aufmerksamkeit erfordert! Angenommen, Sie wollen beim Übergang von der Sommer- zur Winterzeit die Uhr im Prinzip um eine Stunde zurückstellen. Außerdem haben Sie festgestellt, daß die Uhr um 9 Sekunden nachgeht. Ihre Eingabe müßte dann eine der beiden folgenden Formen haben:

adjust "-00:59:51" oder **adjust -3591**

Die Uhr wird dann um 59 Minuten und 51 Sekunden (3591 Sekunden) zurückgestellt, außerdem wird aber die Abweichung gegenüber 2 mal 30 Minuten (9 Sekunden) festgestellt, die bei der nächsten Berechnung des Abgleichfaktors mit zu berücksichtigen ist (die nächste EXACT-Anweisung macht die Uhr also "schneller").

Es gibt nun noch zwei weitere Zeit- bzw. Datum-Funktionen von zweifelhaftem Nutzen: Mit der Anweisung TIME können Sie (mit einer Auflösung von 1/100 Sekunde) feststellen, wieviel Sekunden seit Mitternacht vergangen sind. Damit läßt sich durch zweimalige Abfrage bei Beginn und Ende eines Vorganges und Differenzbildung die Dauer des Vorganges ermitteln. Das funktioniert so einfach aber nur, wenn der Vorgang nicht vor Mitternacht beginnt und erst danach endet!

Die DATE-Anweisung gibt das laufende Jahr aus und daran angehängt, im wievielten Tag des Jahres wir uns gerade befinden. Wenn DATE die Anzeige 85231 ergibt, dann handelt es sich um den 231. Tag des Jahres 1985 (an diesem Tag wurde der amerikanische Autor dieses Buches 30 Jahre alt!). Warum diese Abfrage nicht immer richtige Ergebnisse liefert, ist ein trauriges Kapitel des HP-71, aber wer ist schon völlig ohne Fehler?

Leider hat der HP-71 in der Version 1BBBB (zu erfahren durch die VER$-Anweisung) einen Fehler in der DATE-Routine, der diese Funktion ziemlich wertlos macht: Wenn die vorletzte Ziffer der Jahreszahl ungerade, und die letzte Ziffer eine 2 oder 6 ist, wird die laufende Nummer des Tages (vom 1. März an) um 1 zu niedrig angegeben (da hat wohl einer die Schaltjahre verschlafen!). Hoffen wir gemeinsam, daß neuere Versionen des HP-71-ROM's diesen Fehler nicht mehr haben!

Bei einigen kaufmännischen Dingen wäre es schon nützlich, die laufende Nummer des Tages (zuverlässig) zu erfahren, was sich mit mod(date,1000) recht einfach bewerkstelligen ließe.

Zum Kapitel-Abschluß aber nun etwas Positives: Die Uhr des HP-71 ist wesentlich glaubwürdiger als die des HP-75, die für ihre manchmal rätselhaften Ausgaben berüchtigt ist! Die Ursache dieses Fehlers ist sehr simpel: Die jeweilige Zeit steht in einem Speicher, aus dem der Rechner den Wert bei Bedarf Zeichen für Zeichen abruft. Das beansprucht eine gewisse Zeit, in der es passieren kann, daß die Zeit von der Uhr auf den neuen Wert aktualisiert wird. Die Anzeige besteht dann zum Teil aus dem alten und zum Teil aus dem neuen Wert. Dieser Fehler ist bei der Entwicklung des HP-71 dadurch vermieden worden, daß die Zeit erst dann angezeigt wird, wenn zwei aufeinanderfolgende Abrufe den gleichen Wert ergaben, der dann zwangsläugig zuverlässig sein muß.

Notizen:

Kapitel 8

PEEK$ und POKE

PEEK$ und POKE sind die beiden BASIC-Schlüsselworte, denen sowohl das Benutzerhandbuch, als auch das Referenzhandbuch für den HP-71 den wenigsten Platz einräumen. Das ist an sich schon "verständlich", bieten doch diese beiden Schlüsselworte einen praktisch unbegrenzten Zugriff auf den Speicher des Rechners! Dabei könnten Sie womöglich Dinge sehen, die HP Ihnen lieber vorenthalten möchte! Die Wandlung, die ein Programmierer durchmacht, wenn ihm plötzlich PEEK$ und POKE zur Verfügung stehen, gleicht etwa der Entwicklung vom Bader des Mittelalters zum Gehirn-Chirurgen der Gegenwart!

Genug der Vorschuß-Lorbeeren. Wie lassen sich PEEK$ und POKE praktisch anwenden?

Als erstes können Sie mit PEEK$ Daten aufrufen, die sonst unerreichbar sind: Angenommen, ein Programm benutzt sehr intensiv alle Flags (von 0 bis 63) und Sie wollen den (zuvor) bestehenden Zustand aller Flags für später festhalten. Dann brauchten Sie dazu 64 (INTEGER-)Variable und entsprechend viel Speicherplatz. Hier ist PEEK$ die Patentlösung, weil die 64 Flags in 16 Nibbels des Speichers stehen, beginnend mit der (hexadezimalen) Adresse 2F6E9 (Was Nibbels und Adressen sind, erfahren Sie umgehend!). Mit der Anweisung **f$=peek$("2f6e9",16)** wird der Zustand aller Flags in einem Zug in der String-Variablen F$ abgelegt. Wenn Sie dann mit dem "Flag-intensiven" Programm fertig sind, können Sie den ursprünglichen Zustand mit **poke "2f6e9",f$** wieder herstellen.

POKE wird in erster Linie zu Steuerzwecken verwendet, weil es viele Dinge gibt, zu denen der HP-71 nicht bereit ist, wenn man ihm nicht mit POKE kräftig auf die Zehen tritt. Angenommen, Sie wollen sicherstellen, daß ein begonnenes Programm auf keinen Fall (von außen) unterbrochen werden soll. Auch nicht mit der ATTN-Taste! Das läßt sich nur mit **poke "2f441","F"** erreichen, wodurch die ATTN-Taste unwirksam wird! (Natürlich müssen Sie diese Taste mit **poke "2f441","0"** wieder aktivieren, weil sonst "nichts mehr geht"). Derartige Einwirkungen auf das Verhalten des Rechners sind auf andere Weise nicht möglich.

Neben den Programmbeispielen, mit denen dieses Buch die Anwendung von PEEK$- und POKE-Anweisungen beschreibt, gibt eine Tabelle mit den (hexadezimalen) primären Adressen weitere Informationen.

Aber Vorsicht!! Mit der POKE-Anweisung werden Sie ein "Zauber-Lehrling"! Unüberlegtes "Herum-POKEn" ist ausgesprochen leichtsinnig, weil dadurch eine Störung des Rechners ausgelöst werden kann, begonnen mit einem harmlosen (!) "memory lost" bis zu einem Total-Absturz! Wenn Sie dann die POKE-Anweisungen dahin wünschen,

wo der Pfeffer wächst, sind Sie eigentlich selbst schuld daran! Mit genügender Vorsicht und etwas gesundem Menschenverstand sollte es Ihnen gelingen, derartige Pannen zu vermeiden.

Wie funktionieren nun die PEEK$- und POKE-Anweisungen? Wenn Sie eine Zahl, z.B. 1234, in der Variablen X ablegen wollen, dann besetzt jede Stelle einen kleinen Bereich des Speichers, den wir als "Nibbel" bezeichnen. Im Gegensatz zu den meisten Rechnern, die ihre Bits (als kleinste Speicher-Stelle) in Bytes (mit je 8 Bits) organisiert haben, ist der HP-71 ein Rechner, der seine Bits in Nibbels (mit je 4 Bits) verwaltet. Jedes Nibbel ist im Speicher unter einer ganz bestimmten Adresse (einer Zahl zwischen 0 und 1048575) zu erreichen. PEEK$- und POKE-Anweisung benutzen aber nicht so lange Adress-Nummern, weil die Adressen in hexadezimaler Form angegeben werden. Machen Sie sich mit den Funktionen HTD und DTH$ vertraut, um einen Begriff davon zu bekommen, wie bestimmte Zahlenwerte in den beiden Zahlensystemen aussehen.

Wie Sie bereits wissen, läßt sich die Anzeige mit der CONTRAST-Anweisung an die Lichtverhältnisse und den Blickwinkel durch Eingeben einer Zahl zwischen 0 und 15 beeinflussen. Hexadezimal entspricht das dem Bereich von 0 bis F. Das läßt vermuten, daß die Kontrast-Einstellung in einem einzelnen Nibbel steht. Und das ist tatsächlich so! Die Kontrast-Einstellung steht im Nibbel mit der dezimalen Adresse 189438, die sich hexadezimal als 2E3FE darstellen läßt. Überzeugen Sie sich mit der DTH$-Funktion davon! Wir sollten jetzt in der Lage sein, mit PEEK$- und POKE-Anweisungen, den momentanen Zustand der Kontrast-Einstellung abzufragen und auch zu verändern.

Versuchen wir's! Mit **contrast 7** setzen wir den Kontrast auf 7. Dann geben wir **peek$("2e3fe",1)** ein und werden nach END LINE eine 7 zu sehen bekommen, den momentanen Wert des Kontrast-Nibbels! Jetzt setzen wir mit **contrast 10** einen neuen Kontrast fest und führen die obige PEEK$-Anweisung (mit dem Befehls-Stack!) nochmals aus und werden als Anzeige A erhalten. Warum A? Weil A das hexadezimale Symbol für die dezimale 10 ist! Prüfen Sie es mit der Anweisung **htd(peek$("2e3fe",1))** nach, die direkt eine 10 in die Anzeige bringt.

Nun können wir mit der POKE-Anweisung auch das Kontrast-Nibbel ändern. Das ist zwar eine brotlose Kunst, da wir dazu ja die CONTRAST-Anweisung haben, es ist aber sehr anschaulich und sofort sichtbar! Versuchen Sie's: **poke "2e3fe","4"** wird nach END LINE sofort den Kontrast auf 4 umstellen. Das geschieht, weil wir den HP-71 angewiesen haben, in das Nibbel mit der Adresse "2F3FE" (das ist das Kontrast-Nibbel) den Wert "4" zu setzen.

Aber Vorsicht! Benutzen Sie keine andere Adresse und auch nicht mehr als eine Stelle in der POKE-Anweisung! Sie könnten sonst den Speicher durcheinanderbringen oder auch einen ernstlichen Absturz des Rechners provozieren!

Hier folgt nun eine Auswahl von interessanten Adressen in hexadezimaler Darstellung. Einige davon sollte man nur mit der PEEK$-Anweisung abfragen, andere kann man nur mit der POKE-Anweisung verändern und mit vielen darf man beides tun. Aber bitte, mit Vorsicht!

Hex.-Adresse:	**Nibbel-Anzahl:**	**Bedeutung des Inhaltes:**
2C014	**(1)**	Verfügbarkeit des Kartenlesers (0, falls nicht vorhanden)
2E100	**(2)**	Zusatzanzeigen links in der Anzeige
2E34C	**(2)**	Zusatzanzeigen rechts in der Anzeige
2E350	**(16)**	Reihen-Treiber der Anzeige (1 nach INIT)
2E3FE	**(1)**	Kontrasteinstellung der Anzeige
2E3FF	**(1)**	Anzeige-Art (1 dauernd, 2 blinkend)
2F438	**(4)**	Kalt-Start-Konstante (memory lost bei Änderung!)
2F43C	**(5)**	Interrupt-Meldung (böser Absturz bei Änderung!)
2F441	**(1)**	ATTN-Taste (0 wirksam, F unwirksam)
2F443	**(1)**	Zeiger für Tasten-Buffer
2F444	**(15*2)**	Tasten-Buffer (2 Nibbels pro Taste)
2F471	**(2)**	Zahl der Zeichen, links vom WINDOW (Fenster)
2F473	**(2)**	um 1 verminderte WINDOW-(Fenster-)Länge
2F47C	**(2)**	Buffer-Zeiger für das 1. dargestellte Zeichen
2F47E	**(2)**	Buffer-Zeiger für Kursor
2F480	**(96*2)**	Anzeige-Buffer (2 Nibbels pro Zeichen)
2F540	**(96/4)**	Lesbarkeitsmaske für Anzeige (1 Bit pro Zeichen)
2F576	**(5)**	Adresse des oberen Endes vom Befehls-Stack
2F58A	**(5)**	Adresse des unteren Endes vom Befehls-Stack
2F59E	**(5)**	Adresse des FOR-NEXT-Stacks
2F5A3	**(5)**	Adresse des GOSUB-Stacks
2F5A8	**(5)**	Startadresse der aktivierten Variablen
2F5AD	**(5)**	Adresse des CALL-Stacks
2F5B2	**(5)**	Adresse des Speicher-Endes
2F5B7	**(5)**	Adresse der zu übernehmenden Parameter bei CHAIN-Anweisungen)
2F5BE	**(26*7)**	CHAIN-Variable (7 Nibbels für jeden Variablen-Namen)
2F6C6	**(5)**	Adresse des INPUT-Buffers
2F6CB	**(4)**	Inkrement der Programmzeilen (AUTO-Funktion)
2F6D9	**(64/4)**	System-Flags (-1 bis -64, 1 Bit pro Flag)
2F6E9	**(64/4)**	Anwender-Flags (0 bis 63, 1 Bit pro Flag)
2F6F9	**(5)**	IEEE-Fehlerbehandlungs-Vereinbarung (INX, UNF, OVF, DVZ und IVL)
2F6FE	**(15)**	Ausgangszahl für nächste RND-Funktion
2F761	**(2)**	Verzeichnis noch ausstehender Alarme
2F763	**(12)**	Aufsummierte Abweichung der Uhr
2F76F	**(12)**	Zeitpunkt der letzten Uhr-Einstellung
2F77B	**(12)**	Zeitpunkt der letzten Berechnung des Abgleichfaktors
2F787	**(6)**	Aktueller Wert des Abgleichfaktors
2F7AD	**(3)**	Name der Variablen für die STAT-Funktion

Hex.-Adresse:	Nibbel-Anzahl:	Bedeutung des Inhaltes:
2F7B0	(1)	TRACE-Modus (0 OFF, 2 FLOW, 4 VARS und 6 beides)
2F7B2	(8*2)	LOCK-(Sicherungs-)Passwort (2 Nibbels pro Zeichen)
2F7C2	(34)	RES (speichert jedes numerische Ergebnis, auch komplexe Zahlen)
2F7E4	(4)	ERRN (Nummer des zuletzt aufgetretenen Fehlers)
2F7E8	(4)	Momentane Zeilen-Nummer (bei nicht laufendem Programm)
2F7EC	(4)	Zeilen-Nummer, die letzten Fehler auslöste
2F870	(1)	Flag für Mehr-Zeilen-Funktion (0 bei Einzeilen-Funktion FN)
2F946	(2)	Wander-Geschwindigkeit (2. Argument bei DELAY)
2F948	(2)	Sprung-Geschwindigkeit (1.Argument bei DELAY)
2F94F	(2)	WIDTH-(Zeilen-)Länge bei DISP- und LIST-Anw.
2F956	(2)	Momentane PRINT-Position innerhalb der Zeile
2F958	(2)	Momentan definierte Zeilenlänge für PRINT-Anw.
2F95A	(1)	Zeichenzahl (0 bis 3) der vereinbarten EOL-Sequenz (Zeilenabschluß)
2F95B	(3*2)	EOL-Sequenz (2 Nibbel pro Zeichen)
2F967	(2)	Stringlänge der Tasten-Definition in Nibbels (2 Nibbel pro Byte)
2F969	(1)	Tasten-Art
2F96A	(5)	Adresse des Strings zur Tasten-Definition
2F96F	(2)	Nummer des Kanals, auf den zugegriffen wird
2F976	(1)	um 1 verringerte Anzahl der Eingaben im Befehls-Stack
2F977	(5)	Interne Taktdauer/16
2F986	(1)	Flag für komplexe IMAGE-Anweisungen (0, falls nicht komplex)

Damit dürften Sie für einige Zeit "beschäftigt" sein!

Denken Sie daran, daß ein Programm den vorgefundenen Zustand aller Anwender-Flags mit **u$=peek$("2F6E9",16)** konservieren kann, um auch nach beliebiger Veränderung aller Flags schließlich mit der Anweisung **poke "2F9E6",u$** wieder zum alten Zustand zurückzukehren. Entsprechendes gilt auch für andere Festlegungen, wie TRAPS, WIDTH usw.

Normalerweise ist es auch unmöglich, den Wert der momentan im Rechner gespeicherten Zufallszahl zu erfahren, die irgendwann einmal berechnet wurde, weil bei jeder Ausführung der RND-Funktion diese Zahl zuerst verändert und danach erst angezeigt wird! Den alten Wert kann man nur so erkunden:

```
poke "2f7c2","999"&peek$("2f701",12)&"0" @ r=RES @ r
```

Diese Anweisung holt mit dem PEEK$-Teil die alte Zufallszahl (ohne Exponent!) aus dem Speicher. Dieser "Torso" wird mit "0" und "999" komplettiert und mit dem POKE-Teil dem Rechner als RES "untergejubelt". Die Zuordnung zur Variablen R ist dann Routine!

Haben Sie schon bemerkt, daß im HP-71 alle Zahlenwerte gewissermaßen "rückwärts" im Speicher stehen? Überzeugen Sie sich, indem Sie **pi** eingeben und sich anschließend den Inhalt von RES anschauen:

peek$("2f7c2",16) ergibt Anzeige:

0009535629514130

Die drei führenden "000" verkörpern (auch wieder rückwärts) den Exponenten, dann folgen (wieder rückwärts) 12 Stellen (wobei das Dezimalzeichen zwischen "1" und "3" weggelassen wird, da alle im Rechner behandelten Zahlen immer nur eine Stelle vor dem Dezimalzeichen haben) und schließlich noch eine "0", als Zeichen, daß der Wert positiv ist (eine "9" kennzeichnet einen negativen Wert). Als Erkenntnis aus dieser Betrachtung sollten wir uns merken, daß auch bei allen POKE-Anweisungen die Ziffern in umgekehrter Reihenfolge als String vorliegen müssen.

Das kann manche Dinge kompliziert machen. Nehmen wir an, Sie wollen die interne Taktdauer abfragen, deren Wert halbieren und dann wieder in den Speicher setzen: Dazu brauchen Sie eine PEEK$-Anweisung, dann müssen Sie den erhaltenen String umdrehen, in eine Dezimalzahl wandeln, diese durch 2 teilen, das Ergebnis wieder in eine Hexadezimalzahl wandeln, diesen String wiederum umdrehen um ihn dann mit einer POKE-Anweisung wieder an den richtigen Platz zu bringen. Nur wenn Sie (durch einen LEX-File) eine Funktion zur Verfügung haben, die Ihnen einen String reversiert, können Sie diese lange Prozedur in einer Zeile erledigen:

```
poke"2f977",rev$(dth$(htd(rev$(peek$("2f977",5)))/2))
```

Durch diese recht komplizierte Anweisung (vorausgesetzt, Sie verfügen über die REV$-Funktion) haben Sie erreicht, daß alle BEEP-Vorgänge nur noch die halbe Zeit brauchen und ihre Frequenz verdoppelt haben.

Besonders erstaunlich erschien mir beim Umgang mit der PEEK$- und der POKE-Anweisung die Möglichkeit, LEX-Files über das Tastenfeld in den Rechner zu bringen! HP sagt nämlich im Benutzerhandbuch, daß dies nicht möglich ist. In einigen anderen Publikationen wurde lediglich eingeräumt, daß es vielleicht gelingen könnte. Schauen Sie sich das Programm MAKELEX an, mit dem Sie die REV$-Funktion und einige weitere verblüffende Funktionen in Maschinensprache in den HP-71 bringen können!

Kapitel 9

BASIC-Programme

Jetzt kommen Sie endlich zu dem Gebiet, auf das Sie schon so lange gewartet haben! Es tut mir leid, daß ich Sie so lange hinhalten mußte, aber Sie haben auf den eben durchgelesenen (107 !) Seiten so viel gelernt, daß nun wirklich "kaum noch was" übrigbleibt. Sie haben sogar schon ein Programm eingegeben und es ablaufen lassen! Was wollen Sie eigentlich noch?

Die wirkliche Meisterschaft im Programmieren setzt ganz einfach Erfahrung voraus, und die kann man nicht durch Lesen gewinnen. Man kann aber den eigenen Erfahrungsschatz sehr schnell durch eingehendes Durchdenken fremder Programme vergrößern. Lassen Sie mich hier nur noch einige wichtige Kleinigkeiten sagen, bevor ich Ihnen 13 (sind Sie etwa abergläubisch?) Programme präsentiere, durch die Sie sich regelrecht durchbeißen müssen, wenn Sie verstehen wollen, wie alles funktioniert.

Wenn Sie ein Programm in den Rechner eingeben wollen, das Ihnen bereits gedruckt vorliegt, geben Sie es bitte nicht ins "workfile" ein! Besser ist es, dem Programm gleich einen richtigen Namen zu geben:

```
EDIT PROGRAMM
```

Jetzt heißt das Programm "PROGRAMM", Sie können aber auch jeden anderen Namen (mit bis zu 8 Zeichen, das erste unbedingt ein Buchstabe!) nehmen. Vergessen Sie nicht, das "workfile" ist eigentlich nur für Versuche bestimmt!

Wenn ein Programm nach dem Eingeben nicht richtig laufen will, schauen Sie sich mit den AUF- und AB-Tasten und den Cursor-Tasten alle Zeilen nochmals sehr kritisch auf eventuelle Eingabe-Fehler an und korrigieren Sie diese! Die Anweisungen TRACE VARS, TRACE FLOW und TRACE OFF sind beim Aufspüren von Fehlern in Programmen ausgesprochen hilfreich.

Kleine Eingabehilfen, wie z.B. die AUTO-Anweisung werden in diesem Buch nicht besonders erwähnt, weil diese Dinge so einfach und im Benutzerhandbuch auch sehr gut beschrieben sind. Jetzt, wo Sie über BASIC schon einiges wissen, um es im HP-71 anwenden zu können, sollten Sie sich Zeit und Ruhe nehmen, um die beiden zum Rechner gehörenden Handbücher aufmerksam zu lesen. Einige Dinge, die Ihnen zuvor noch rätselhaft erschienen, werden Sie nun als simpel beurteilen. Die wirklich schwierigen Gebiete (wie IMAGE- und PRINT USING-Anweisungen) verstehen Sie am besten durch Ausprobieren der Beispiele, die in den HP-Büchern zu finden sind.

Geben Sie jedes Programm zuerst genau so ein, wie es abgedruckt ist, und lassen Sie es ablaufen und wühlen Sie sich dann durch alle

Programmzeilen, um zu entdecken, wie das Programm funktioniert. Versuchen Sie dann, ob das Programm auch nach kleinen Änderungen durch Sie noch vernünftig läuft. Oder setzen Sie zur Großtat an! Sie haben nichts zu verlieren, Sie können nur gewinnen: nämlich Erfahrung im Programmieren in BASIC!

WALDLAUF eine Kalorien/Joule-Berechnung für's Joggen (ehemals Waldlauf genannt, als es noch viel Wald gab!)

```
WALDLAUF    BASIC   321
**********************
10 DIM G,E,T,T1,K @ INPUT "kg? ";G
20 IF G<50 OR G>110 THEN 120
30 INPUT "km? ";E
40 IF E<=0 THEN 120
50 INPUT "Minuten.Sekunden? ";T
60 IF T<=0 THEN 120
70 T1=IP(T)+FP(T)/.6
80 IF T1/E<3.47 THEN BEEP @ DISP "zu schnell!" @ END
90 IF T1/E>6.94 THEN BEEP @ DISP "zu langsam!" @ END
100 K=E*(2.237+G*(1.009-.0292*T1/E))
110 DISP "verbraucht:";IP(K+.5);"kcal =";IP(4.18*K+.5);"kJ" @ END
120 BEEP @ DISP "außer Bereich!"
```

Starten Sie dieses Programm mit der RUN-Taste, nachdem Sie einen heilsamen Lauf hinter sich gebracht haben: Das Programm fragt nach dem Körpergewicht, nach der zurückgelegten Strecke und nach der dafür benötigten Zeit. Es berechnet dann den Energieverbrauch in Kilo-Kalorien bzw. Kilo-Joule.

Es ist ein sehr einfaches Programm, das Ihnen Anwendung und Wirkung der IF/THEN-Anweisungen zeigen soll, die hier die Annahme sinnloser Eingabewerte verhindern.

CMDSTK ein Programm, um den Befehls-Stack zu vergrößern oder zu verkleinern.

```
CMDSTK      BASIC   265
**********************
10 SUB CMDSTK(X) @ POKE "2F441","1" @ STD
20 DEF FNR$(A$)=A$[5]&A$[4,4]&A$[3,3]&A$[2,2]&A$[1,1]
30 X=IP(MIN(MAX(X,1),16))
40 DIM S$[X*6] @ A=HTD(FNR$(PEEK$("2F576",5)))
50 FOR Y=1 TO X @ S$=S$&"000300" @ NEXT Y
60 E$=FNR$(DTH$(A+X*6)) @ POKE "2F580",E$&E$&E$
70 POKE DTH$(A),S$ @ POKE "2F976",DTH$(X-1)[5]
80 POKE "2F441","0" @ END SUB
```

Dieses Programm können Sie nicht mit der RUN-Taste starten, weil es als Unter-Programm deklariert ist! Wenn Sie es benutzen wollen, muß der Aufruf mit der CALL-Anweisung geschehen:

CALL CMDSTK(5)

Durch diese Eingabe erhält der Befehls-Stack Raum für fünf Anweisungen (das ist der normale Umfang). Wenn Ihnen das nicht reicht, setzen Sie eine 10 in die Klammer und schon passen 10 Anweisungen in den Befehls-Stack! Sie können den Umfang des Befehls-Stacks so auf 1 bis 16 Anweisungen einrichten.

TOKENIZE ein Programm, daß sich mit sich selbst beschäftigt?!

```
TOKENIZE   BASIC   153
**********************
1 ! Hierher beliebige Zeile!
10 SUB A @ N=256 @ DIM A$[N] @ A$=ADDR$(CAT$(0))
20 A$=PEEK$(DTH$(HTD(A$)+55),N)
30 A$=A$[1,POS(A$,"0F0100")-1]
40 PLIST 1,9 @ DISP A$;" (";STR$(LEN(A$));")"
```

Dieses Programm interessiert sich nur für sich selber! Und zwar wird die Zeile 1 des Programms analysiert und so angezeigt, wie der Rechner diese Zeile intern gespeichert hat: Setzen Sie in die Zeile 1 die folgende Anweisung:

1 BEEP 153

Wenn Sie nun (mit zwei Tastenfolgen) CALL A eingeben, sehen Sie zuerst **1 BEEP 153** und danach **8EB0351 (7)**. Dabei stellt 8E die BEEP-Anweisung, B0351 die 153 und (7) die Zahl der von dieser Anweisung besetzten Nibbels dar. Zwei Anwendungen hat dieses Programm: Es mag manchmal interessant sein, den Rechner bei der Zerlegung einer Zeile in sogenannte "Token" (also Grund-Routinen) zu beobachten. Immer ist es aber nützlich, von zwei gleich wirksamen Anweisungen herauszubekommen, welche weniger Platz im Speicher benötigt. Ein Beispiel erklärt das besser, als viele Worte:

1 DIM M(3) @ M(1)=1 @ M(2)=2 @ M(3)=3
und
1 DIM M(3) @ READ M @ DATA 1,2,3

sind verschiedene Formulierungen, die genau die gleiche Wirkung haben. Welche Anweisung braucht nun weniger Platz im Speicher? Geben Sie (nacheinander!) jede der beiden Formen als Zeile 1 in das obige Programm und starten Sie es! Die angezeigte Zahl der benötigten Nibbels (im 1. Fall 53, im 2. nur 34) wird Ihnen schnell sagen, was günstiger ist!

Nach diesen doch recht anspruchsvollen Kostproben nun wieder ein recht einfach zu überblickendes Programm, das wohl keinem Rechner erspart bleibt, obwohl die Idee dazu bereits in der Antike gefunden wurde:

PRIMSIEB das Sieb, mit dem Eratosthenes die Primzahlen fing!

```
PRIMSIEB    BASIC    180
**********************
10 REAL K,Q,T,X @ DESTROY P @ OPTION BASE 1
20 INPUT "Primzahlen bis? ";Q @ T=(Q-1) DIV 2 @ INTEGER P(T)
30 DISP "2" @ FOR K=1 TO (SQR(Q)-1)/2 @ IF P(K) THEN 50
40 DISP STR$(K+K+1) @ FOR X=3*K+1 TO T STEP X-K @ P(X)=1 @ NEXT X
50 NEXT K @ FOR K=K TO T @ IF P(K)=0 THEN DISP STR$(K+K+1)
60 NEXT K @ END
```

Starten Sie das Programm und geben Sie bei der Abfrage eine Zahl (z.B. 100 oder 1000) ein, bis zu der Sie die Untersuchung fortgesetzt wissen wollen. Das Programm wird jetzt bis zur angegebenen Grenze mit erstaunlicher Geschwindigkeit alle Primzahlen anzeigen. Nur zuerst geht es etwas stockend, aber dann purzeln die Primzahlen nur so heraus! Wenn das nicht der Fall ist, ist der Rechner nicht auf DELAY 0 eingestellt.

Dieses Programmbeispiel zeigt das Einrichten und Benutzen eines Datenfeldes und die Verwendung der Index-Zahl der Feldvariablen zur Festlegung der STEP-Weite in der FOR/NEXT-Schleife.

FEHLER Anzeige der festgelegten Fehler-Meldungen.

```
FEHLER      BASIC    274
**********************
10 DIM X @ DELAY 0,0 @ POKE "2F7E6","00"
20 POKE "2F443","0" @ INPUT "Fehler-Nr.? ";X
30 X=MOD(X,249)
40 IF X>97 AND X<228 THEN X=229 ELSE IF X=228 THEN X=97
50 POKE "2F7E4",DTH$(MOD(X,16))[5]&DTH$(X DIV 16)[5]
60 DISP STR$(X);'"';ERRM$;'"'
70 IF NOT KEYDOWN THEN 70
80 IF KEYDOWN("#51") THEN X=X+1 @ GOTO 30
90 IF KEYDOWN("#50") THEN X=X-1 @ GOTO 30
100 IF KEYDOWN THEN 20 ELSE 70
```

Wenn Sie dieses Programm starten, will es von Ihnen eine Fehler-Nr. wissen. Nach einer Eingabe (z.B. 10 oder 20) erscheint die mit dieser Nummer verknüpfte Fehler-Meldung. Nun können Sie mit den AUF- und AB-Tasten den ganzen "Fehler-Katalog" durchstöbern! Sobald Sie aber irgendeine andere Taste betätigen, fragt der Rechner erneut nach einer Fehler-Nr. Beachten Sie auch, wie das Programm auf das Festhalten der AUF- und AB-Tasten reagiert!

Die POKE-Anweisung in Zeile 10 ist nötig, um vor Ablauf des Programms eine eventuell noch anstehende Fehlermeldung zu löschen, die von einem LEX-File oder einem Zusatz-ROM ausgelöst wurde und den Ablauf des Programms stören könnte.

GLOBETEL ein komfortables Telefon-Verzeichnis!

```
GLOBETEL    BASIC  1101
**********************
10 DIM N$[22],S$[22],P$[22],N,R,M
20 ON ERROR GOTO 370
30 UNSECURE GLOBEDAT @ ASSIGN #1 TO GLOBEDAT @ READ #1;N,M
40 DISP "Bereit" @ ON ERROR GOTO 50
50 GOTO KEY$
60 'Q': PRINT #1,0;N,M @ SECURE GLOBEDAT @ DISP "Ende!" @ END
70 'P': IF R=0 THEN 90
80 DISP P$ @ GOTO 50
90 DISP "kein Eintrag!" @ GOTO 50
100 'N': IF R=0 THEN 90
110 DISP N$ @ GOTO 50
120 'D': IF R=0 THEN 90
130 IF R=N THEN 150
140 READ #1,N;N$,P$ @ PRINT #1,R;N$,P$
150 R=0 @ N=N-1 @ DISP "gelöscht" @ GOTO 50
160 'A': IF N=M THEN DISP "Voll!" @ GOTO 50
170 N$="" @ S$="" @ N=N+1 @ R=N
180 ON ERROR GOTO 190
190 LINPUT "Name: ",N$;N$ @ IF N$="" THEN 190
200 ON ERROR GOTO 210
210 INPUT "TEL.: ",S$;S$ @ P$=S$
220 PRINT #1,R;N$,P$ @ GOTO 40
230 'S': IF N=0 THEN 90 ELSE LINPUT "Suche: ";S$ @ S$=UPRC$(S$)
240 FOR R=1 TO N @ READ #1,R;N$,P$
250 IF POS(UPRC$(N$),UPRC$(S$))=0 THEN 270 ELSE DISP N$;"?"
260 ON POS("YN",KEY$)+1 GOTO 260,80,270
270 NEXT R @ R=0 @ GOTO 90
280 'Z': DISP "Alle";N;"Namen?"
290 ON POS("YN",KEY$)+1 GOTO 290,300,40
300 PURGE GLOBEDAT @ GOTO 370
310 'F': IF N=0 THEN 90 ELSE R=R+1 @ IF R>N THEN R=1
320 READ #1,R;N$,S$ @ GOTO 110
330 'B': IF N=0 THEN 90 ELSE R=R-1 @ IF R<1 THEN R=N
340 GOTO 320
350 'U': IF R=0 THEN 90 ELSE S$=P$ @ GOTO 180
360 'M': DISP "A,B,D,F,M,N,P,Q,S,U,Z?" @ GOTO 50
370 OFF ERROR @ INPUT "Welcher Umfang? ";M @ N=0
380 CREATE DATA GLOBEDAT,M+1,50 @ ASSIGN #1 TO GLOBEDAT
390 PRINT #1;N,M @ GOTO 40
```

Dieses Programm ist eine auf europäische Verhältnisse abgestimmte Version des vom Verfasser angegebenen Originals, das nur für amerikanische Telefon-Nummern zweckmäßig war. Es leistet trotz seiner Kürze Erstaunliches: Namen und Telefon-Nummern werden in einem DATA-File gespeichert und auf Anforderung schnellstens herausgesucht. Beim ersten Start des Programms wird abgefragt, wieviel Eintragungen von Ihnen beabsichtigt sind. Geben Sie hier einen passenden Wert (z.B. 30) ein.

Wenn Sie nun einen Namen und eine Rufnummer abspeichern wollen, drücken Sie "A". Bei der Eingabe des Namens ist lediglich darauf zu achten, daß der Text höchstens 22 Zeichen (Abstände mitgezählt!) haben darf. Die Eingabe ist mit END LINE abzuschließen, worauf der Rechner die Rufnummer erfragt. Auch für diese gilt die obige Begrenzung auf insgesamt 22 Zeichen! Da eine weltweite Fernwahl mit höchstens 13 Ziffern auskommt, bleiben Ihnen noch 9 Zeichen, um diese lange Zahl durch Klammern, Binde- und Schrägstriche aber auch Abkürzungen wie "App." landesüblich zu gliedern. Auch die Rufnummern-Eingabe wird mit END LINE abgeschlossen, worauf die "Bereit"-Anzeige wieder erscheint.

Nach Beendigung der Eingaben können Sie mit "F" bzw. "B" in den abgelegten Namen vorwärts und rückwärts herumstöbern und sich bei jedem Namen mit "P" die Rufnummer anzeigen lassen. Der Name erscheint nach "N" wieder..

Bei dieser Durchsicht läßt sich mit "U" die angezeigte Eintragung ändern, wobei alle Editiermöglichkeiten wirksam sind. Mit "D" kann jede angezeigte Eintragung gelöscht werden.

Wenn der Rechner einen Namen suchen soll, müssen Sie nach der Eingabe von "S", einen oder mehrere aufeinander folgende, für den Namen typische Buchstaben eingeben, z.B. "NG" oder "ng", wenn Sie nach Angermann suchen. Der passende Name erscheint mit einem Fragezeichen in der Anzeige, worauf Sie mit "Y" (YES) oder "N" (NO) reagieren können. Es erscheint dann entweder die zum (richtigen) Namen gehörende Rufnummer, oder der Rechner sucht weiter nach Namen, die auch diese Kombination enthalten.

Wenn Sie das gesamte Verzeichnis löschen wollen, reicht dazu "Z" aus. Sie werden danach aber nochmals gefragt, ob der Wunsch nach Löschung ernst gemeint war. Auch hier können Sie mit "Y" oder "N" antworten.

Wenn Sie das Verzeichnis nicht mehr benötigen drücken Sie "Q" (und nicht die f-Taste und ATTN!)

Sollten Sie vergessen haben, welche Tasten in diesem Programm wirksam sind, geben Sie "M" ein und es erscheinen alle besetzten Zeichen!

Dieses Programm demonstriert einige Tricks. So werden z.B. die Labels direkt durch die Tasten angesprochen (siehe Zeile 50). Solche Menue-gesteuerten Programme sind immer angenehmer als solche, bei denen Sie eine Menge auswendig lernen müssen und etliche Fragen beantworten sollen, bevor überhaupt etwas geschieht. Fehler-Fallen scheiden (viele) ungeeignete Eingaben aus. Es ist in Programmen dieser Art auch selten, daß beim Löschen eines Eintrages der im DATA-File besetzte Raum wieder freigegeben wird: Hier wird die entstehende Lücke einfach mit der letzten Eintragung gefüllt und die Zahl der Eintragungen berichtigt.

Bei der Bearbeitung wurden die Namen für Programm und DATA-File geändert, das Menue-Schema jedoch übernommen, um beide Programm-Versionen in gleicher Weise benutzen zu können. Wer andere Abkürzungen wünscht, kann dies durch Ändern der Label leicht erreichen. Zum Abschluß noch (für Reisende nach USA!) die Original-Version, die als Rufnummern nur Zahlen ohne führende Nullen und Namen mit höchstens 21 Zeichen verträgt:

PHONE ein komfortables Telefon-Verzeichnis für die USA!

```
PHONE       BASIC  1076
**********************
10 DIM N$[22],S$[22],P,N,R,M
20 ON ERROR GOTO 370
30 UNSECURE PHDAT @ ASSIGN #1 TO PHDAT @ READ #1;N,M
40 DISP "Ready" @ ON ERROR GOTO 50
50 GOTO KEY$
60 'Q': PRINT #1,0;N,M @ SECURE PHDAT @ DISP "Done" @ END
70 'P': IF R=0 THEN 90
80 DISP USING "'('3*') '3*'-''4*";P @ GOTO 50
90 DISP "No such record" @ GOTO 50
100 'N': IF R=0 THEN 90
110 DISP N$ @ GOTO 50
120 'D': IF R=0 THEN 90
130 IF R=N THEN 150
140 READ #1,N;N$,P @ PRINT #1,R;N$,P
150 R=0 @ N=N-1 @ DISP "Deleted" @ GOTO 50
160 'A': IF N=M THEN DISP "No room" @ GOTO 50
170 N$="" @ S$="" @ N=N+1 @ R=N
180 ON ERROR GOTO 190
190 LINPUT "Name: ",N$;N$ @ IF N$="" THEN 190
200 ON ERROR GOTO 210
210 INPUT "Phone #:",S$;S$ @ P=VAL(S$)
220 PRINT #1,R;N$,P @ GOTO 40
230 'S': IF N=0 THEN 90 ELSE LINPUT "Search: ";S$ @ S$=UPRC$(S$)
240 FOR R=1 TO N @ READ #1,R;N$,P
250 IF POS(UPRC$(N$),UPRC$(S$))=0 THEN 270 ELSE DISP N$;"?"
260 ON POS("YN",KEY$)+1 GOTO 260,80,270
270 NEXT R @ R=0 @ GOTO 90
280 'Z': DISP "Zap All";N;"names?"
290 ON POS("YN",KEY$)+1 GOTO 290,300,40
300 PURGE PHDAT @ GOTO 370
310 'F': IF N=0 THEN 90 ELSE R=R+1 @ IF R>N THEN R=1
320 READ #1,R;N$,S$ @ GOTO 110
330 'B': IF N=0 THEN 90 ELSE R=R-1 @ IF R<1 THEN R=N
340 GOTO 320
350 'U': IF R=0 THEN 90 ELSE S$=STR$(P) @ GOTO 180
360 'M': DISP "A,B,D,F,M,N,P,Q,S,U,Z?" @ GOTO 50
370 OFF ERROR @ INPUT "How many records? ";M @ N=0
380 CREATE DATA PHDAT,M+1,32 @ ASSIGN #1 TO PHDAT
390 PRINT #1;N,M @ GOTO 40
```

MAKELEX ein Programm zur direkten Eingabe von LEX- und BIN-Files.

```
MAKELEX     BASIC    660
**********************
10 SFLAG -1 @ ON ERROR GOTO 30 @ DESTROY ALL
20 PURGE DUMMY
30 ON ERROR GOTO 220
40 INPUT 'Byte-Anzahl: ';N
50 CREATE TEXT DUMMY,N
60 A=HTD(ADDR$('DUMMY')) @ A1=A @ P$="----------------"
70 Q=1 @ X=0 @ INPUT '000: ',P$;A$ @ C$=A$ @ GOSUB 200
80 Q=2 @ X=1 @ GOSUB 190
90 A$=A$&C$ @ A=A+37 @ N=N*2+31 @ Q=3 @ SFLAG 5
100 FOR X=2 TO N DIV 16-1
110 GOSUB 190
120 IF FLAG(5) THEN C$=C$[6]
130 POKE DTH$(A),C$ @ A=A+16-5*FLAG(5,0) @ NEXT X @ Q=4
140 DISP DTH$(X)[3]; @ INPUT ': ',P$[1,MOD(N,16)];C$
150 GOSUB 200
160 L=LEN(C$) @ IF C$[L,L]='-' THEN C$=C$[1,L-1] @ GOTO 160
170 POKE DTH$(A),C$ @ POKE DTH$(A1),A$
180 OFF ERROR @ CFLAG -1 @ END
190 DISP DTH$(X)[3]; @ INPUT ': ',P$;C$
200 DISP DTH$(X)[3]; @ INPUT ' ok ','--';C1$
210 S=0 @ FOR Z=1 TO LEN(C$) @ S=IP(NUM(C$[Z,Z])*Z+S) @ NEXT Z
220 IF C1$=DTH$(MOD(S,256))[4] THEN RETURN
230 DISP 'Eingabefehler!' @ BEEP @ POP @ ON Q GOTO 70,80,110,140
240 DISP 'Fehler: '&ERRM$ @ BEEP @ GOTO 180
```

Dieses Programm wurde von Stephan Tobiasson und Thomas Fange entwickelt. Es ist zweifellos ein BASIC-Edelstein! Durch dieses Programm können Sie als Listings vorliegende LEX- oder BIN-Files direkt in den Rechner eingeben, ohne dazu einen Kartenleser, einen Lesestift oder eine HP-IL-Massenspeicher-Einheit zu verwenden. Sie sparen damit eine beträchtliche Summe!

Starten Sie das Programm und geben Sie zunächst die Byte-Anzahl ein, die bei den folgenden Listings von LEX-Files in der Überschrift steht. Danach können Sie den File, beginnend mit der Zeile 000, als Hex-Code eingeben. Die Eingabe darf keine Leerzeichen enthalten, diese sollen im Listing nur die Lesbarkeit verbessern.

Nach Eingabe jeder Zeile fordert der Rechner (nach END LINE) die Angabe des Prüf-Bytes zur Kontrolle der Eingabe. Danach kann die nächste Zeile eingegeben werden. Das Programm endet automatisch nach Eingabe aller Zeilen.

Jetzt müssen Sie den HP-71 nur noch aus- und wieder einschalten, wodurch sich dieser neue LEX-File in eine womöglich schon bestehende LEX-File-Kette einfügt. Die durch den neuen LEX-File gebotenen Funktionen sind nun benutzbar.

Als erstes Beispiel sollten Sie jetzt das Listing eines LEX-Files eingeben, der Ihnen die REV$-Funktion beschert:

```
REVLEX      LEX       36 Bytes    ID#5D
**************************************

     0123 4567 89AB CDEF     ck (Prüf-Byte)

000: 2554 65C4 5485 0202     B1
001: 802E 0052 0010 1000     2D
002: 8400 0D51 0100 0000     3D
003: F710 0000 0000 0000     A7
004: 0710 00F7 2554 6542     FE
005: 101F F411 048F E83B     C7
006: 18DC 32F0               0E
```

Sobald sich dieser LEX-File im Speicher des Rechners befindet, können Sie die REV$-Funktion über die Tastatur oder im Programm aufrufen. Geben Sie ein:

REV$("Dies ist ein STRING-Umkehr-Test!")

In der Anzeige erscheint nach END LINE (erstaunlich schnell!):

!tseT-rhekmU-GNIRTS enie tsi seiD

REV$ kann einen String mit 14 000 Zeichen in einer einzigen Sekunde umdrehen! Das ist so schnell, daß sich BASIC eigentlich schämen müßte. Andererseits war aber BASIC immerhin gut genug, um den LEX-File REVLEX in den Rechner aufzunehmen!

Als nächstes das Listing eines von HP geschriebenen LEX-Files, durch das die KEYWAIT$-Funktion ermöglicht wird:

```
KEYWAIT     LEX       58 Bytes    ID#52
**************************************

     0123 4567 89AB CDEF     ck (Prüf-Byte)

000: B454 9575 1494 4502     BB
001: 802E 0053 9080 2058     AB
002: 3700 0251 0100 0000     D2
003: F710 0000 0000 0000     A7
004: 0E10 00FF B454 9575     9C
005: 1494 4542 101F F001     5C
006: 3610 8137 1098 F2C6     90
007: 0045 1119 1351 1813     20
008: 48D8 ACA1 8F12 7006     38
009: BDF                     9C
```

KEYWAIT$ ermittelt (genau wie die KEY$-Funktion) den Code der zuerst betätigten Taste, also die älteste Eintragung im Tastenbuffer. Der Unterschied zwischen beiden Funktionen zeigt sich bei leerem Tastenbuffer: Während KEY$ dann einen Leer-String zurückgibt und der Rechner voll aktiv bleibt, versetzt KEYWAIT$ bei diesem Tatbestand den Rechner in eine Art Schlummer und wartet (daher "WAIT") auf die nächste Tastenbetätigung. Und darin liegt der Hauptnutzen dieser Funktion: im Schlummer verbraucht der Rechner wesentlich weniger Energie als etwa in einer Warteschleife. Ersetzen Sie deshalb KEY$ in allen Programmen (in GLOBETEL bzw. PHONE je 3 mal) durch KEYWAIT$, die Batterien danken es Ihnen!

SMATTER ein kaum zu "knackendes" Codier- u. Decodier-Programm.

```
SMATTER     BASIC   435
**********************
10 ! S.M.A.T.T.E.R. ! (C) 1984 Joseph Horn
20 DIM A$[96],K,X @ DELAY 9,0
DISP "Code Decode Mehr Ende"
40 A$=KEYWAIT$ @ IF A$="E" THEN 130 ELSE IF A$="M" THEN 70
50 IF A$="C" THEN K=1 ELSE IF A$="D" THEN K=-1 ELSE 40
60 OPTION ROUND NEG @ INPUT "Schlüssel: ";X @ RANDOMIZE X
70 LINPUT "Text: ";A$ @ IF A$="" THEN 30 ELSE PRNT "{";
80 FOR X=1 TO LEN(A$) @ IF A$[X,X]<" " OR A$[X,X]>"z" THEN 120
90 RANDOMIZE 10^(RND*100)
100 POKE "2F6FE",STR$(FLAG(-12,RND<RND)+1)
110 PRINT CHR$(MOD(NUM(A$[X,X])+IP(RND*92)*K-32,91)+32);
120 NEXT X @ PRINT "}" @ GOTO 30
130 OPTION ROUND NEAR @ RANDOMIZE @ DELAY 0,0 @ DISP "fertig"
    @ END
```

"SMATTER" ist die Abkürzung von "Self Modifying Algorithm Transcryptor That's Essentially Random" (das heißt etwa: Sich selbst veränderndes Rechenverfahren für eine Geheimschrift unter Benutzung von Zufallszahlen").

Hier wird der LEX-File KEYWAIT benutzt, den Sie eben kennen lernten. Wenn Sie das Menue sehen ("Code Decode Mehr Ende"), brauchen Sie nur den ersten Buchstaben des passenden Wortes einzugeben! Der Rechner fragt bei "Code" bzw. "Decode" nach dem "Schlüssel", den nur der Partner und Sie kennen! Als Schlüssel ist jeder numerische Ausdruck zugelassen, der einen von 0 verschiedenen Wert ergibt (auch SQR(150+PI), wenn Sie das wollen!).
Dann können Sie den Text eingeben. Um führende oder folgende Leerzeichen zu erkennen, steht der codierte Text in geschweiften Klammern. END LINE führt Sie wieder zum Menue zurück.

Ich glaube, daß diese Codierung nicht zu entschlüsseln ist. Es ist ein Codierungsverfahren, über das mir keine Veröffentlichungen bekannt sind. Versuchen Sie doch selbst, den Code zu brechen!

Hier folgen noch weitere Listings von LEX-Files, die für Sie vielleicht interessant oder nützlich sein könnten:

```
LOCKOFF    LEX       36 Bytes   ID#5C
************************************

     0123 4567 89AB CDEF    ck (Prüf-Byte)

000: C4F4 34B4 F464 6402    B8
001: 802E 0014 1010 1000    2A
002: 6400 0C50 0000 0000    23
003: FE00 0000 0800 001F    7F
004: F200 431D F961 4000    09
005: 342B 7F21 37AF 0151    23
006: 7137 00                1E
```

LOCKOFF erzeugt zwar kein neues BASIC-Schlüsselwort, hat aber eine andere überraschende Eigenschaft: Das Passwort, das den Rechner für fremde Benutzer sperren soll, wird beim Einschalten gelöscht! Besonders unter "netten" Kollegen ist es ein beliebter Scherz, den Rechner eines anderen mit einem möglichst exotischen Passwort zu versehen und sich dann diebisch zu freuen, wenn der Besitzer den Rechner nicht mehr in Gang bekommt, ohne seine Programme zu verlieren! Setzen Sie sich diesen LEX-File in den Rechner, und er ist gegen diesen Schabernack "immun"!

```
CURLEX     LEX      112 Bytes   ID#5E
************************************

     0123 4567 89AB CDEF    ck (Prüf-Byte)

000: 3455 25C4 5485 0202    A0
001: 802E 0083 1010 1000    53
002: 0E00 0E50 0000 0000    4B
003: FE00 0000 0800 001F    7F
004: F31B 1961 4000 DB10    87
005: B317 F8FC 4631 4901    D6
006: 1BD7 0011 BD71 188F    24
007: B14B 1DA3 1F99 62D0    3E
008: 310A 962C 0003 1446    C5
009: 7003 134A F0AE AAF2    00
00A: 3141 8FB3 4B1A DA10    87
00B: 3078 F534 B107 8F53    90
00C: 4B10 7108 11B8 F1C3    C6
00D: 2011 BA5E 4801 0B59    6B
00E: E118 068F B14B 1068    5D
00F: FB14 B106 8408 2101    65
```

Auch CURLEX erzeugt kein neues BASIC-Schlüsselwort, es macht aber das Programmieren etwas bequemer! Wenn Sie sich eine lange Zeile ansehen wollen, geht das mit der RECHTS- und LINKS-Taste doch recht langsam und die (blau) geshifteten Funktionen setzen den Cursor nur an das Ende bzw. den Anfang der Zeile. Das ist für Zeilen mit mehr als 44 Zeichen auch nicht das Wahre! Wäre es nicht praktisch, mit den geshifteten Tasten den Cursor um genau 21 Schritte springen zu lassen? Hier ist die Lösung: Sobald CURLEX im Speicher ist und sich die Tastatur im USER-Modus befindet, lassen die geshifteten Tasten den Cursor in der gewünschten Weise springen! Versuchen Sie es mal (aber bitte nicht im CALC-Modus! Da könnten Sie sehr drollige Ergebnisse bekommen)! Dieser LEX-File stammt von John Baker, einem Weisen, wenn es um LEX-Files für den HP-71 geht!

```
CLOCKDSP   LEX      330 Bytes   ID#52
************************************

     0123 4567 89AB CDEF    ck (Prüf-Byte)

000: 34C4 F434 B444 3505    72
001: 802E 0082 0010 1000    42
002: 4920 0259 2920 0000    95
003: F710 0000 0E10 0000    84
004: 0BD1 00D9 34C4 F434    BF
005: B492 1FF9 6950 5031    6D
006: 1B13 5112 1CB1 378B    A4
007: 6A11 3510 B3B1 4A3B    E4
008: 4C43 4021 5DB0 01FC    2F
009: 79F2 1530 A044 4000    44
00A: D48F 3E32 09F6 10CF    DC
00B: B11D F241 BFD3 1000    10
00C: 01B1 09F2 1101 5071    75
00D: 6F11 1150 7191 4DB1    09
00E: 4420 313D 8FC4 6314    48
00F: 606A C08F 2B52 1BF6    EB
010: BF68 1E8F 9223 18F2    E1
011: 5231 20AE 6F23 0AF2    63
012: F2AE 9BF2 30AB F2BF    BA
013: 2AEB 1B11 9F21 5271    D0
014: 0118 F152 7100 1612    25
015: 7154 0303 1601 5401    AC
016: 600D 5CE1 9102 0310    A8
017: 214C 2EAF 2328 0DAF    A8
018: 7201 368F CB91 01B1    A4
019: 74F2 1423 300C 015C    A4
01A: 3239 1271 1B87 4F21    FD
01B: 460B 8410 B154 31B1    85
01C: 49F2 146D 7001 B149    D6
```

```
01D: F2DB 1443 13D8 F106    5E
01E: 318F 2B52 11B1 49F2    9F
01F: 146D 7001 B149 F2DB    E0
020: 1443 13D8 FAF5 3167    21
021: EEA7 000C 7000 14A3    59
022: 10E9 6224 7350 3070    60
023: E061 5101 F174 F233    B6
024: 0051 15D3 1F87 4F21    81
025: 470B 8410 B155 37A6    28
026: F8D8 4A80 7310 3080    BB
027: E0E1 5107 C7F6 5EF1    1B
028: FC79 F215 3001 8DCF    C2
029: 2508 FA2C 200E 1101    5F
02A: EC00 008D B2E2 08DA    F4
02B: E720                   09
```

CLOCKDSP ist ein sehr langer LEX-File, ich möchte ihn aber nicht mehr missen! Sobald Sie ihn in Aktion erlebt haben, werden Sie verstehen, warum! CLOCK ON und CLOCK OFF heißen die beiden neuen BASIC-Schlüsselwörter.

Mit CLOCK ON erhalten Sie eine dauernd sichtbare Zeitanzeige am rechten Ende des Displays, obwohl Sie gleichzeitig, links davon, in der Anzeige machen können, was Sie bisher gewohnt waren. Das gilt auch für den CALC-Modus! CLOCK OFF läßt die Zeitanzeige verschwinden und stellt wieder das ganze Display zur Verfügung.

Weil der linke Teil des Displays für die Zeitanzeige geschützt bleiben muß, wird bei jeder Aktualisierung der Zeit (also jede Sekunde) automatisch eine WINDOW 13-Anweisung ausgeführt, weshalb die normale WINDOW-Anweisung während der Zeitanzeige gesperrt bleiben muß.

CLOCK OFF bringt wieder normale Verhältnisse und aktiviert auch wieder die WINDOW-Anweisung.

Dieser LEX-File wurde von HP ausgearbeitet.

Kapitel 10

Der Umgang mit der COPY-Anweisung

Die COPY-Anweisung wird zum Kopieren von kompletten Files benutzt. Diese Anweisung tut (unter anderem) die Dinge, für die bei manchen Rechnern STORE- bzw. LOAD-Anweisungen üblich sind. Die COPY-Anweisung ist jedoch universeller anwendbar und im Prinzip eine sehr klare und einfache Angelegenheit. Die vielen Ausnahmen machen es jedoch praktisch unmöglich, ein für den Anfänger gut verständliches Syntax-Diagramm aufzustellen. Den meisten Nutzen wird Ihnen noch die Lektüre der Seiten 112 bis 114 des Benutzerhandbuches für den HP-71 bringen. Im Gegensatz zur dortigen Darstellungsart soll Ihnen hier die COPY-Anweisung zuerst in ihrer vollständigen Form begründet und erklärt werden. Die "Vereinfachungen" durch Weglassen bestimmter Teile der COPY-Anweisung prägen sich dann leichter ein.

Untersuchen wir als erstes, von wo nach wohin das Kopieren von Files nötig (und möglich) ist:

Wenn Sie nur den 'nackten' HP-71 besitzen, können Sie Files nur innerhalb des Rechners kopieren. Dabei ist es von großem Vorteil, daß Sie den vorhandenen Schreib-Lese-Speicher in mehrere, voneinander unabhängige Bereiche unterteilen können: Der Rechner achtet bei diesen Kopiervorgängen darauf, daß in jedem Teilbereich jeder File-Namen nur einmal vorkommt.

Sobald Sie sich einen Kartenleser gekauft haben, werden zwei weitere Anwendungsarten möglich, mit denen Files aus dem Rechner auf Magnetkarten geschrieben oder von diesen in den Rechner eingelesen werden. Entsprechendes gilt auch, wenn über ein HP-IL-Modul ein einzelner externer Massenspeicher (Kassetten- oder Disketten-Laufwerk) erreicht werden kann.

Die vierte Anwendungsart ist erst möglich, wenn sich in der HP-IL-Schleife mindestens zwei Massenspeicher befinden: Es lassen sich dann Files von einem zum anderen Massenspeicher übertragen.

Nun sollte uns auch klar werden, daß der Rechner zum Ausführen einer COPY-Anweisung grundsätzlich vier Angaben braucht:

Wie heißt der Original-File?	**Wie soll die File-Kopie heißen?**
Wo ist das Original zu finden?	**Wohin ist zu kopieren?**

Von diesen Angaben beschäftigen sich die beiden oberen mit den Namen, die unteren mit den Ortsangaben. Links stehen die Angaben über das Original, rechts die über die Kopie.

Für den Aufbau einer kompletten COPY-Anweisung gilt eine entsprechende Einteilung: zwischen COPY und TO stehen die Angaben über das Original, rechts davon die über die Kopie.

In der COPY-Anweisung sind nun noch (an zwei Stellen) Markierungen nötig, damit der Rechner File-Namen von Ortsangaben unterscheiden kann. Es werden hierzu zwei Satzzeichen benutzt, denen verschiedene Bedeutungen unterlegt werden:

Auf einen Doppelpunkt (:) folgt immer die Bezeichnung für ein (externes) Gerät oder einen bestimmten Teil des Speichers im Rechner. Auf einen einfachen Punkt (.) folgt stets der Name (Volume-Label) einer Kassette (oder Diskette). Der Rechner muß dann den (oder die) Speicher-Geräte abfragen, ob eine Kassette (oder Diskette) dieses Namens zugänglich ist. Das geht zwar etwas langsamer, hat aber manchmal seine Vorteile.

Und jetzt kommt das dicke Ende: Unter bestimmten Voraussetzungen können von den oben aufgezählten Angaben eine, mehrere oder auch alle wegfallen. Sogar das **TO** darf in der Anweisung fehlen! Schauen Sie auf die Seiten 113/14 des Benutzerhandbuches: Alle 16 möglichen Kombinationen sind vertreten! Dort wird auch nach 15 Ausnahmen als letztes der (gut verständliche) Regelfall aufgeführt!

Bei den Ausnahmen wollen wir zuerst die behandeln, die leicht zu begreifen sind:

Eine Kopie wird immer im Hauptspeicher des Rechners angelegt, wenn für die Kopie kein Ort angegeben ist. Das **:MAIN** kann man sich also ersparen.

Lassen Sie den Namen für die Kopie weg, dann erhält diese den gleichen Namen wie das Original. Das erscheint konsequent!

Beim Weglassen der Ortsangabe für das Original muß der Rechner den File im eigenen Speicher suchen. Auch das merkt sich leicht!

Kompliziert wird es erst, wenn der Namen des Originals fehlt:

Fehlen sämtliche Angaben (d.h. die "gesamte" Anweisung besteht nur aus dem Wort **COPY**) dann wird der momentan aufgerufene (Programm-) File (aus einem der Ports) in den Hauptspeicher kopiert.

Ist nur der Name für die Kopie genannt, wird im Hauptspeicher eine Kopie des momentan aufgerufenen (Programm-) Files mit dem neuen Namen angelegt.

Ist nur der Ort für die Kopie angegeben, wird der momentan aufgerufenene (Programm-) File unter Beibehaltung des Namens an die gegebene Stelle (innerhalb oder außerhalb des Rechners) kopiert.

Eine COPY-Anweisung, die nur den Ort des Originals nennt, ist nur mit **:CARD** bzw. **:PCRD** zulässig und leitet das Einlesen des Files von Magnetkarte(n) in den Hauptspeicher ein, wobei der bestehende Namen für die Kopie übernommen wird.

Da Magnetkarten (pro Spur) immer nur einen File enthalten können, ist es logisch, daß hier der Name des Originals unnötig ist.

Fehlt der Name des Originals, ist aber gleichzeitig der Ort (Kassette oder Diskette oder ein entsprechendes Laufwerks) und der Name der Kopie festgelegt, dann sucht der Rechner an der angegebenen Stelle nach einem Original gleichen Namens. Ist das nicht ausgesprochen freundlich?

Als Einschränkung ist zu beachten, daß sich Files von Magnetkarten immer nur in den Hauptspeicher kopieren lassen. In umgekehrter Richtung kann aber jeder irgendwo im Rechner befindliche (ungeschützte) File auf Magnetkarten gespeichert werden. Kopiervorgänge zwischen Magnetkarten und Kassette (oder Diskette) sind in beiden Richtungen nur mit dem Speicher des Rechners als "Zwischen-Station" möglich.

Wenn ein gesicherter File kopiert wird, ist die Kopie ungesichert. Geschützte (PRIVATE) Files lassen sich nur in den Rechner kopieren, eine weitere Kopie auf einen anderen Datenträger außerhalb des Rechners ist nicht möglich.

Eine gewisse Unlogik soll Ihnen den Abschied von diesem Kapitel erleichtern: Der im Rechner eingesetzte Kartenleser ist sicher kein externes Gerät, hat aber doch "einen gewissen Abstand" vom eigentlichen Betriebssystem des Rechners. Sie dürfen den Kartenleser deshalb mit **:CARD** bzw. **:PCRD** aber auch einfach mit **CARD** bzw. **PCRD**, also mit oder ohne Doppelpunkt ansprechen.

Wahrscheinlich werden Sie jetzt die Ausführungen der Referenzhandbücher etwas besser verstehen und Sie werden erkennen, daß ein "Filespezifikator" nichts anderes ist, als eine ausreichende Angabe über Namen und Ort des Files ist, die aber von Fall zu Fall sehr verschieden umfangreich sein kann.

Für die Fälle, in denen Sie sich nicht ganz sicher fühlen, können Sie immer zur ausführlichen COPY-Anweisung mit vier Angaben Zuflucht nehmen. Lieber zehnmal umständlich, als einmal gar nicht oder verkehrt. Die Routine kommt von ganz allein! Überflüssig zu sagen, daß an die Stelle der Strings auch Stringausdrücke treten können.

Notizen:

Postscriptum

Der HP-71-Antwort-Dienst

Haben Sie noch eine Frage, die sich auf den HP-71 bezieht? Dann wird Sie das folgende Angebot interessieren, das für Sie praktisch kostenlos ist, während es für mich verheerende Folgen haben kann! (Das Recht auf diesen "Kundendienst" haben Sie mit dem Kauf dieses Buches erworben, aber darüber wollte ich gar nicht reden). Immer wenn Sie eine individuelle Antwort auf eine Frage wünschen, die sich auf Ihren eigenen HP-71 bezieht, können Sie folgendes tun:

1. Bringen Sie die Frage, so klar wie möglich, zu Papier.

2. Schreiben Sie die eigene Adresse auf eine Postkarte.

3. Stecken Sie diese Postkarte, das Blatt mit der Frage und einen "Internationalen Antwort-Schein" in einen Umschlag.

4. Adressieren Sie diesen Umschlag an

   Joseph K. Horn

   1042 Star Rt

   Orange, CA 92667

   U S A

5. Frankieren Sie diesen Brief ausreichend für seine Reise in die USA.

Sobald es mir möglich ist, werde ich die Antwort auf die Postkarte schreiben und diese an Sie absenden. Wenn die obigen Bedingungen nicht erfüllt sind, kann ich eine Antwort nicht garantieren! Dieser "Kundendienst" kann immer wieder in Anspruch genommen werden.

Erwarten Sie aber nur solche Antworten, die auf einer Postkarte Platz haben! Es kommt dabei auch nicht darauf an, wie komplex oder wie einfach Ihre Frage zu beantworten ist! Die schwierigen Fragen sind für mich eine Herausforderung, die einfachen oft nur Routine, weil ich sie meist mit einem Hinweis auf eine Passage dieses Buches beantworten kann. Andererseits dürfen Sie auch nur eine Antwort erwarten, die sich auf einer Postkarte unterbringen läßt. Ich werde unter gar keinen Umständen auf die Postkarte ein 100 Zeilen langes BASIC-Programm schreiben (99 Zeilen eventuell, 100 aber nie!). Verlangen Sie auch nie, daß ich Ihnen weitere Unterlagen, wie z.B. Magnetkarten zusende, weil es so schwierig ist, diese Dinge in einer Postkarte zu "verpacken"!

Einige den Antwort-Dienst "eingrenzende" Vereinbarungen:

Der Antwort-Dienst wird ausschließlich vom Autor angeboten und geleistet, nicht vom Verlag oder anderen Personen (auch nicht vom Übersetzer!). Anfragen, die diesen Dienst betreffen, dürfen nur an die auf der vorigen Seite angegebene Adresse des Autors gerichtet werden. Das gilt auch für Hinweise auf Fehler in der amerikanischen Originalfassung.

Die Inanspruchnahme des Antwort-Dienstes setzt stillschweigend voraus, daß der Autor die Frage (ohne Nennung von Namen und Adresse des Fragestellers, aber auch ohne Entschädigung für diesen) in späteren Veröffentlichungen verwenden darf.

Eine Beantwortung der Fragen darf nicht mehr erwartet werden, wenn der HP-71 seit mehr als fünf Jahren in der offiziellen Preisliste von Hewlett-Packard nicht mehr erscheint.

Obwohl der Autor die Antworten nach bestem Wissen und bester Erfahrung ausarbeitet, kann er weder stillschweigend noch ausdrücklich Fehler in seinen Antworten ausschließen, noch kann er für direkt oder indirekt aus der Befolgung der gegebenen Antwort erwachsende Schäden verantwortlich oder haftbar gemacht werden.

Dieser Hick-Hack ist aus juristischen Gründen unentbehrlich, weil es immer einige besonders "liebe" Mitmenschen gibt, die hilfreich gemeinte Angebote nicht als solche hinnehmen und sich darüber freuen, sondern aus diesen rücksichtslos Rechtsansprüche ableiten möchten. Für die vielen netten Benutzer und Fans des HP-71 sind diese etwas kühlen Sätze sowieso nicht wichtig, denn sie kämen überhaupt nicht auf solche Ideen!

Für Hinweise auf Fehler in der deutschen Bearbeitung ist der Übersetzer dankbar. Diese sollten direkt an die im Vorspann genannte Adresse gerichtet werden, wobei nicht in jedem Fall mit einer Antwort zu rechnen ist! Auch hierfür gelten die oben formulierten Einschränkungen und Haftungsausschlüsse sinngemäß.

HP-71 Schlüsselwort- und Anweisungs-Verzeichnis

In dieser Aufstellung sind die Schlüsselwörter und Anweisungen aufgeführt, die vom HP-71, vom Mathematik-Paket (HP 82480A), vom HP-IL-Interface (HP 82401) und von einigen LEX-Files der User-Library benutzt werden. Vom FORTH/Assembler-ROM (HP 82441) sind nur die Wörter aufgeführt, die in BASIC-Programmen auftreten können. Bei allen Schlüsselwörtern und Anweisungen finden Sie einen Hinweis, in welchem ROM oder LEX-File diese verwendet werden. Dahinter steht meist ein Hinweis auf die Seite des entsprechenden Handbuches. Wenn kein ROM angegeben ist, handelt es sich um ein Schlüsselwort oder eine Anweisung des Hauptgerätes, die Seitenangabe bezieht sich dann auf das Referenzhandbuch, das Sie zusammen mit dem Rechner erhalten haben.

Auf Schlüsselwörter oder Anweisungen, deren Wirkung oder Bedeutung nicht klar ersichtlich ist, folgt eine kurze Erklärung.

Syntax-Richtlinien.

Die benutzte Schreibweise stimmt (bis auf eine Ausnahme) mit der auch sonst in den Kurzanleitungen verwendeten überein: \ wird in Verbindung mit { . . . } und [. . .] benutzt, wenn mehrere Möglichkeiten der Eingabe bestehen, von denen eine auszuwählen ist (Beispiel):

ADJUST {*Sekunden* \ *Zeit-String*}

Hier müssen Sie das Wort ADJUST eingeben und danach die Korrekturangabe entweder als numerischen Ausdruck oder als String-Ausdruck.

Eckige Klammern besagen (wie üblich), daß die darin eingeschlossene Eingabe möglich, aber nicht zwingend vorgeschrieben ist (Beispiel):

CONT [*Zeilennummer* \ *Label*]

Hier können Sie nach dem Wort CONT die Anweisung enden lassen, Sie dürfen aber auch noch eine Zeilennummer oder ein Label folgen lassen.

Alle fett gedruckten Teile der Anweisung (runde und eckige Klammern, Schlüsselwörter und Satzzeichen) müssen genau in der vorgeschriebenen Weise eingegeben werden.

ABS (*num. Ausdruck*). Seite 10 auch Mathe-ROM, Seite 41
Ermittelt Betrag des Ausdrucks.

ACOS (*num. Ausdruck*). Seite 11
Ermittelt Arcus Cosinus des Ausdrucks.

ACOSH (*num. Ausdruck*). Mathe-ROM, Seite 28
Ermittelt Area Hyperbel-Cosinus des Ausdrucks.

ACS (*num. Ausdruck*). Seite 11
Ermittelt Arcus Cosinus des Ausdrucks.

ADD [*Liste num. Ausdrücke*]. Seite 12
Schlägt Werte als Koordinaten dem momentanen Statistik-Feld zu.

ADDR$ (*String-Ausdruck als File-Namen*). Seite 13
Ermittelt die Hex-Adresse des angegebenen Files.

ADJABS {*Sekunden* \ *Zeit-String*}. Seite 14
Stellt Uhr um 0 bis +/-360000 Sekunden bzw. 00:00:00 bis +/-99:59:59 Stunden:Minuten:Sekunden vor oder nach.

ADJUST {*Sekunden* \ *Zeit-String*}. Seite 15
Stellt Uhr um 0 bis +/-360000 Sekunden bzw. 00:00:00 bis +/-99:59:59 Stunden:Minuten:Sekunden vor oder nach. Beeinflußt zusätzlich Abgleich-Faktor.

AF [(*num. Ausdruck*)]. Seite 17
Ermittelt momentanen Abgleich-Faktor und setzt diesen auf neuen Wert (falls vorhanden und Betrag der Eingabe mindestens 10 und höchstens 8388607 ist).

ALL <Postfix>
Verwendet hinter CAT, CFLAG, DELETE, DESTROY, END, PURGE, SFLAG, siehe dort!

& <String-Verbinder> Anwendung: *String* **&** *String* Seite 307
Verbindet angrenzende String-Ausdrücke miteinander. Oft auch "Et" bzw. "Ampersand" genannt.

AND <Operator> Anwendung: *num. Ausdr.* **AND** *num. Ausdr.*. Seite 19
Logische UND-Verknüpfung, nur wahr (1) falls beide Ausdrücke von Null verschieden, sonst falsch (0).

ANGLE (*num. Ausdruck für X, num. Ausdruck für Y*). Seite 20
Ermittelt im Koordinatensystem Winkel zwischen pos. X-Achse und Zeiger vom Punkt (0,0) zum Punkt(X,Y) im momentanen Winkel-Modus.

ANGLE <als Postfix>
Verwendet hinter OPTION, siehe dort!

ARG (*reeller oder komplexer num. Ausdr.*). Mathe-ROM, Seite 42
Ermittelt im Koordinatensystem Winkel zwischen pos. X-Achse und Zeiger vom Punkt(0+i0) zum Punkt(a+ib) im momentanen Winkel-Modus.

ASIN (*num. Ausdruck*). Seite 22
Ermittelt Arcus Sinus des Ausdrucks.

ASINH (*num. Ausdruck*). Mathe-ROM, Seite 28
Ermittelt Area Hyperbel-Sinus des Ausdrucks.

ASN (*num. Ausdruck*). Seite 22
Ermittelt Arcus Sinus des Ausdrucks.

ASSIGN# *Kanalnummer* **TO** {*File-Namen* \ * \ "" \ "*" }.
Seite 23 auch HP-IL-ROM, Seite 77
Öffnet bzw. schließt Kanal der genannten Nummer.

ASSIGN IO {*Zuweisungscodeliste* \ * \ "" \ "*" }.
HP-IL-ROM, Seite 80
Ordnet HP-IL-Geräte zu bzw. hebt Zuordnungen auf.

@ <Anweisungs-Verketter>: Anwendung: *Anweisung* **@** *Anweisung*
Seite 306
Erlaubt Zusammenfassung mehrerer Anweisungen zu einer Zeile. Meist "At" (Jargon: "Klammeraffe") genannt.

ATAN (*num. Ausdruck*). Seite 25
Ermittelt den Arcus Tangens des Ausdrucks.

ATANH (*num. Ausdruck*). Mathe-ROM, Seite 28
Ermittelt den Area Hyperbel-Tangens des Ausdrucks.

ATN (*num. Ausdruck*). Seite 25
Ermittelt den Arcus Tangens des Ausdrucks.

!. Seite 242, gleiche Wirkung wie REM, siehe dort!

AUTO [*Nummer der Anfangszeile* [, *Zeilenraster*]]. Seite 26
Numeriert Zeilen automatisch. Nur über Tastatur!

BASE. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

BASIC. <Postfix> Verwendet zusammen mit TRANSFORM, siehe dort!

BEEP [*Frequenz in Hz* [, *Dauer in Sekunden*]\ **ON** \ **OFF**]. Seite 28
Beeinflußt Ton-Höhe und Dauer, schaltet akustische Signale ein und aus.

BINAND (*num. Ausdruck , num. Ausdruck*). HP-IL-ROM, Seite 82
Bit für Bit ausgeführte UND-Verknüpfung.

BINCMP (*num. Ausdruck*). HP-IL-ROM, Seite 84
Bit für Bit ausgeführte Negation.

BINEOR (*num. Ausdruck , num. Ausdruck*). HP-IL-ROM, Seite 86
Bit für Bit ausgeführte Exklusiv-ODER-Verknüpfung.

BINIOR (*num. Ausdruck , num. Ausdruck*). HP-IL-ROM, Seite 88
Bit für Bit ausgeführte Inklusiv-ODER-Verknüpfung.

BIT (*num. Ausdruck , Bit-Nummer*). HP-IL-ROM, Seite 90
Ermittelt Zustand des bezeichneten Bits.

BSTR$ (*num. Ausdruck , Basiszahl*). Mathe-ROM, Seite 16
Ermittelt Darstellung des dezimalen Ausdrucks mit 2, 8 oder 16 als Basis.

BVAL (*String-Ausdruck , Basis*). Mathe-ROM, Seite 15
Ermittelt dezimale Darstellung des binär (oktal, hexadezimal) aufgefaßten Strings.

BYE. Seite 30, auch FORTH/Ass.-ROM
Schaltet Rechner bzw. FORTH-Modus ab.

CALL [*Unterprgm-Name* [(*Parameter-Liste*)][**IN** *File-Namen*]]. Seite 31
Ruft Unterprogramm zur Ausführung.

CARD. <Postfix>
Verwendet zusammen mit CAT und COPY, siehe dort!

CAT [**ALL** \ **KEYS** \ **CARD** \ *File-Namen* \ **:MAIN** \ **:PORT** \ *Gerät*].
Seite 35 auch HP-IL-ROM, Seite 92
Ruft bezeichneten Katalog auf.

CAT$ (*File-Position* [, *File-Lagerstelle*]).
Seite 38, auch HP-IL-ROM, Seite 94
Ruft Katalog des bezeichneten Files auf.

CEIL (*num. Ausdruck*). Seite 40
Ermittelt kleinste ganze Zahl, gleich oder größer als der Ausdruck.

CFLAG { **ALL** \ **MATH** \ *Liste der Flags von -32 bis 64*}. Seite 41
Löscht angegebene Flags

CHAIN *String-Ausdruck als File-Namen.*
Seite 42, auch HP-IL-ROM, Seite 96
Das angegebene Programm ersetzt das momentan im Haupt-Speicher befindliche.

CHARSET *String-Ausdruck.* Seite 43
Legt den alternativen Zeichensatz gemäß String fest.

CHARSET$. Seite 46
Ermittelt momentan gültigen alternativen Zeichensatz.

CHR$ (num. Ausdruck). Seite 47
Erzeugt dasjenige ASCII-Zeichen, dessen Code sich aus dem Ausdruck ergibt.

CLAIM [:]**PORT** (*Portnummer p.dd*). Seite 48
Gibt abgeteilten Bereich des Speichers an Hauptspeicher zurück. Nur über Tastatur!

CLASS (*num. Ausdruck*). Seite 49
Ermittelt Klasse des Wertes.

CLEAR [*Adresse* \ **LOOP**]. HP-IL-ROM, Seite 98
Löscht einzelne oder alle HP-IL-Geräte.

CLOCK. <Postfix>
Verwendet zusammen mit RESET, siehe dort!

CLSTAT. Seite 51
Besetzt alle Elemente des momentanen statistischen Feldes mit Nullen.

CNORM (*Feld-Variable*). Mathe-ROM, Seite 70
Summiert die Beträge der Elemente derjenigen Spalte, die das höchste Ergebnis liefert.

COMPLEX *Liste der Variablen.* Mathe-ROM, Seite 19
Deklariert komplexe Variable gemäß Liste.

CON [(*num. Ausdr.*[, *num. Ausdr.*])]. Mathe-ROM, Seite 52
Dimensioniert Matrix gemäß den Ausdrücken um und besetzt alle Elemente der neuen Matrix mit Einsen.

CONJ (*komplexer num. Ausdruck*). Mathe-ROM, Seite 42
Ermittelt den konjugiert komplexen Wert des Ausdrucks.

CONT [*Zeilenummer* \ *Label*]. Seite 52
Startet Programm an bezeichneter Stelle. Nur über Tastatur!

CONTRAST *num. Ausdruck.* Seite 54
Beeinflußt Kontrast-Einstellung der Anzeige.

CONTROL {**ON** \ **OFF**}. HP-IL-ROM, Seite 100
Schaltet den Controller-Status des HP-71 ein bzw. aus.

CONTROL. <Postfix> Verwendet zusammen mit PASS, siehe dort!

COPY [*Quellenangabe*][**TO** *Zielangabe*]. Seite 55, HP-IL-ROM, S. 96
aber auch Benutzerhandbuch, Seiten 112-114.
Kopiert den bezeichneten File (auch mit geändertem Namen) an die gewünschte Stelle.

CORR (*Variablennummer* , *Variablennummer*). Seite 58
Ermittelt Korrelation für das bezeichnete Variablenpaar des momentanen statistischen Feldes.

COS (*num. Ausdruck*). Seite 59, auch Mathe-ROM, Seite 38
Ermittelt den Kosinus des Argumentes.

COSH (*num. Ausdruck*). Mathe-ROM, Seiten 27 u. 39
Ermittelt den Hyperbel-Kosinus des Argumentes.

CREATE {**TEXT****LIF1****DATA****SDATA**} *File-Namen*
[,*Recordanzahl*[,*Recordlänge*]].
Seite 60, auch HP-IL-ROM, Seite 106
Richtet File mit den definierten Eigenschaften ein.

CREATE ALL Theologie-ROM (Glaubensangelegenheit!)

DATA [*Datenliste*]. Seite 62
Stellt Daten bereit. Nur im Programm-Modus ausführbar!

DATA. <Postfix> Verwendet zusammen mit CREATE, siehe dort!

DATE. Seite 62
Ermittelt das momentane Datum der Systemuhr als Zahl im Format JJTTT.(Vorsicht bei Schaltjahren mit Endziffern 2 oder 6 ab Anfang März!)

DATE$. Seite 66
Ermittelt das momentane Datum der Systemuhr als String der Form "JJ/MM/TT".

DEBUG. LEX-File #2 aus User's Library, Hex-ID=53
Nur in Verbindung mit DEBUGGER-ROM anwendbar!

DEF. <Postfix> Verwendet zusammen mit END, siehe dort!

DEF FN*num. Var.*[(*Parameter-Liste*)][=*Ausdruck*]. Seite 67
Nur im Programm-Modus ausführbar! Definition einer num. Funktion.
Beispiel: DEF FNA(X) bedeutet:
"A ist eine Funktion von X"

DEF FN*String Var.* [[*Stringlänge*]][(*Parameter-Liste*)][=*Ausdruck*].
Seite 67
Nur im Programm-Modus ausführbar! Definition einer String-Funktion.
Beispiel: DEF FNA$(X$) bedeutet:
"String A$ ist eine Funktion des Strings X$"

[**DEF**] **KEY** *Tasten-Name* [, *zugeordneter String*[; \ :]]. Seite 69
Legt Tastenzuordnung im USER-Modus fest.

DEFAULT {**EXTEND** \ **ON** \ **OFF**}. Seite 72
Setzt bestimmte Ersatzwerte für mathematische Ausnahmen fest.

DEG (*num. Ausdruck*). Seite 73
Rechnet das als Bogenmaß gewertete Argument in Altgrad um.

DEGREES. Seite 74
Schaltet den Winkelmodus auf Altgrad

DEGREES. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

DELAY *Zeilen-Zeit*[, *Zeichen-Zeit*]. Seite 75
Legt Anzeigedauer und Durchlaufzeit für Anzeigen fest.

DELETE {**ALL** \ *Startzeile* [, *Zielzeile*]}. Seite 77
Löscht (einschließlich) Start- bis Zielzeile. Nur über Tastatur ausführbar!

DELETE # *Kanal-Nummer* , *Record-Nummer*. FORTH- bzw. TEXT-ROM

DESTROY {**ALL** \ *Variablen-Liste*}. Seite 78
Löscht alle bzw. nur die angegebenen Variablen.

DET bzw. **DETL**. Mathe-ROM, Seite 69
Ermittelt die Determinante der letzten reellen Matrix.

DET (*quadratische Matrix*). Mathe-ROM, Seite 69
Ermittelt die Determinante der angegebenen Matrix.

DEVADDR (*Geräte-Bezeichnung*). HP-IL-ROM, Seite 109
Ermittelt die IL-Adresse des angegebenen Gerätes.

DEVAID (*Geräte-Bezeichnung*). HP-IL-ROM, Seite 111
Ermittelt die Zubehör-Kennung des angegebenen Gerätes.

DEVID$ (*Geräte-Bezeichnung*). HP-IL-ROM, Seite 113
Ermittelt die Typen-Bezeichnung des angegebenen Gerätes.

DIM *num. Var.*[(*1. Dimension* [,*2. Dimension*])]. Seite 79
Dimensionierung eines numerischen Feldes bzw. einer Matrix.

DIM String Var.[(*Zeilenanzahl*)][[*Zeilenlänge*]]. Seite 79
Dimensionierung einer String-Variablen bzw. eines String-Feldes.

[**DISP**][*usdr.*\ **TAB**(*num. Ausdr.*)[{ ; \ , } *Ausdr.*]][; \ ,]...
Seite 82 und Mathe-ROM, Seite 54.
Tabelliert anzuzeigende Ausdrücke.

[**DISP**] **USING** {*Format-String* \ *Zeilen-Nummer*};[*Ausdr.*][; \ ,]...
Seite 84 und Mathe-ROM, Seite 54.
Formatiert anzuzeigende Ausdrücke.

DISP$. Seite 86
Übernimmt den Inhalt der Anzeige in einen String.

DISPLAY IS {*Geräte-Bezeichnung* \ * }. HP-IL-ROM, Seite 114
Festlegung des anzeigenden Gerätes.

DIV <Operator> Anwendung: *num. Ausdr.* **DIV** *num. Ausdr.* Seite 87
Ermittelt ganzzahligen Teil bei Division des linken Wertes durch den rechten.

\ <Operator> Anwendung: *num. Ausdr.* \ *num. Ausdr.* Seite 311
Gleiche Wirkung wie DIV, siehe dort!

DOT (*Vektor 1 , Vektor 2*). Mathe-ROM,Seite 71
Ermittelt das Skalar-Produkt der beiden Vektoren.

DROP [*Liste num. Ausdrücke*]. Seite 88
Löscht eine aus mehreren Werten bestehende Eingabe im momentanen statistischen Feld.

DTH$ (*dezimaler num. Ausdruck*). Seite 89
Ermittelt die Hexdarstellung des Argumentes als String.

DVZ. Seite 90
Ermittelt den Zustand des DVZ-Flags (-7), das beim Versuch einer Division durch Null gesetzt wird.

EDIT [*File-Bezeichnung*]. Seite 91
Nur über Tastatur ausführbar!
Macht bezeichnetes Programm zugänglich.

EDPARSE$ (*Befehls-String*). FORTH- bzw. TEXT-ROM

EDTEXT *File-Name.* FORTH- bzw. TEXT-ROM
Macht angegebenen Text-File zur Bearbeitung zugänglich.

ELSE <Postfix> Verwendet zusammen mit IF, siehe dort!

ENABLE INTR *num. Ausdruck.* HP-IL-ROM, Seite 117
Setzt Argument als Interrupt-Maske ein.

END [**ALL**]. Seite 93

END DEF Seite 94
Nur im Programm-Modus ausführbar!
Abschluß der Definition einer USER-Funktion.

END SUB. Seite 94
Abschluß eines Unterprogramms

ENDLINE [*String*]. Seite 96
Legt die nach dem Wagenrücklauf-Signal ausgegebene Zeichenfolge fest.

ENG *num. Ausdruck.* Seite 97
Schaltet auf Darstellung im technischen Format, rundet Argument und grenzt auf Werte von 0 bis 11 ein. Dadurch Zahl der signifikanten Stellen 1 bis 12.

ENTER {*Geräte-Bezeichnung* \ **LOOP**}
[**USING** {*Formatstring* \ *Zeilen-Nummer*}][; *Variablen-Liste*]
HP-IL-ROM, Seite 120
Verteilt eingehende Daten (nach Formatierung) auf die Variablen.

EPS. Seite 99
Setzt den kleinsten von Null verschiedenen Wert ein, der mit voller Stellenzahl darstellbar ist.

ERRL. Seite 100
Ermittelt Zeile im Programm, bei deren Bearbeitung der letzte Fehler auftrat.

ERRM$. Seite 101
Ermittelt den Text der letzten Fehlermeldung.

ERRN. Seite 102
Ermittelt die Nummer der letzten Warnung oder Fehlermeldung.

ERROR <Postfix> Verwendet zusammen mit OFF und ON, siehe dort!

ESCAPE *String , Tasten-Code.* FORTH-ROM

ESCAPE. <Postfix> Verwendet zusammen mit RESET, siehe dort!

EXACT. Seite 103
Beeinflußt den Abgleichfaktor der eingebauten Uhr.

EXOR. <Operator> Anwendung: *num. Ausdr.* **EXOR** *num. Ausdr.*
Seite 105
Logische Exklusiv-Oder-Verknüpfung, nur wahr (1), wenn lediglich einer der beiden Ausdrücke von Null verschieden ist, sonst falsch (0).

EXP (*num.Ausdruck*). Seite 106, und Mathe-Rom, Seite 37
Ermittelt den Wert von e^X.

EXPM1 (*num. Ausdruck*). Seite 107
Ermittelt den Wert von e^X-1.
Genauer als EXP(X)-1 für x<0.

EXPONENT (num. Ausdruck). Seite 108
Ermittelt die Größenordnung des Arguments.
Genauer als IP(LOG10(X)).

EXTEND. <Postfix> Verwendet zusammen mit DEFAULT, siehe dort!

FACT (*0<=Ganzzahl<=253*). Seite 109
Ermittelt die Fakultät (X!) des Argumentes.

FETCH (*Zeilen-Nummer* \ *Label*). Seite 110
Nur über Tastatur ausführbar!
Zeigt die bezeichnete Programm-Zeile zum Bearbeiten an.

FETCH KEY *Tastenname* Seite 111
Nur über Tastatur ausführbar!
Zeigt die Zuordnung der bezeichneten Taste zum Bearbeiten an.

FGUESS. Mathe-ROM, Seite 90
Zeigt letzten Näherungswert für Nullstelle an.

FILESZR (*File-Namen*). FORTH- bzw. TEXT-ROM
Ermittelt die File-Größe in Records.

FIX *num. Ausdruck.* Seite 112
Schaltet auf Darstellung mit Festkomma, rundet Argument und grenzt auf Werte von 0 bis 11 ein. Dadurch Zahl der signifikanten Stellen 1 bis 12.

FLAG (*num. Ausdruck 1* [, *num. Ausdruck 2*]) Seite 114
Ermittelt Zustand des mit Argument 1 bezeichneten Flags. Argument 2 bestimmt neuen Zustand des Flags.

FLOOR (*num. Ausdruck*). Seite 115
Ermittelt größte ganze Zahl, deren Wert den des Argumentes nicht überschreitet.

FLOW. <Postfix> Verwendet zusammen mit TRACE, siehe dort!

FNORM (*Feldbezeichnung*). Mathe-ROM, Seite 70
Ermittelt die Summe aller Quadrate aller Feldelemente des bezeichneten Feldes (Frobenius-Norm).

FNROOT (*1. Näherungswert , 2. Näherungswert , Funktion von FVAR*)
Mathe-ROM, Seite 89
Ermittelt eine reelle Nullstelle, ausgehend von den beiden Näherungen.

FNVariable = *Funktionsgleichung* Seite 116
Nur in mehrzeiligen FN-Festlegungen zulässig!
Beispiel: FNA=Q bedeutet: "A ist immer gleich Q".

FN*Variable*[(*Parameter-Liste*)]. Seite 116
Übernimmt die nötigen Parameter für die Funktion.

FN*Variable*. <Postfix> Verwendet zus. mit DEF und LET, siehe dort!

FOR *num. Var.* = *Startwert* **TO** *Endwert* [**STEP** *Schrittweite*]. Seite 118
Legt eine (mehrmals) zu durchlaufende Schleife fest.

FORTH FORTH-ROM
Nur über Tastatur ausführbar!
Schaltet Rechner in den FORTH-Modus.

FORTH$ FORTH-ROM
Holt einen String aus dem PAD-Bereich (bezeichnet durch die Daten in oberster Position des FORTH-Daten-Stacks) in eine BASIC-String-Variable.

FORTHF. FORTH-ROM
Holt einen numerischen Wert aus dem im X-Register des FORTH-Fließkomma-Stacks in eine BASIC-Variable.

FORTHI FORTH-ROM
Holt einen numerischen Wert aus der obersten Position des FORTH-Daten-Stacks in eine BASIC-Variable.

FORTHX *Befehls-String* [, *Parameter-Liste*]. FORTH-ROM
Übergibt Befehle und Daten an das FORTH-System.

FOUR. Mathe-ROM. Seite 133
Führt Fourier-Transformation aus.

FP (*num. Ausdruck*). Seite 121
Ermittelt den nicht-ganzzahligen Teil des Argumentes.

FREE [:] PORT(*p.dd* **).** Seite 122
Nur über Tastatur ausführbar!
Trennt einen besonders bezeichneten Bereich vom Hauptspeicher ab.

FVALUE. Mathe-ROM, Seite 90
Gibt den zuletzt mit FNROOT berechneten Funktionswert zurück.

FVAR. Mathe-ROM, Seite 90
Gibt den Wert zurück, für den zuletzt mit FNROOT der Funktionswert ermittelt wurde.

GAMMA (*num. Ausdruck* **).** Mathe-ROM, Seite 28
Ermittelt den Wert der Euler'schen Gamma-Funktion für das Argument.

GDISP *Stringausdruck.* Seite 124
Erzeugt ein graphisches Muster unter Auswertung des Stringausdrucks.

GOSUB { *Zeilen-Nummer* \ *Label* }. Seite 129
Sprungbefehl zu einer Subroutine.

GOSUB. <Postfix> Verwendet zusammen mit ON, siehe dort!

GOTO { *Zeilen-Nummer* \ *Label* }. Seite 131
Programm-Verzweigung.

GOTO. <Postfix> Verwendet zusammen mit ON, siehe dort!

HPIL. <Postfix> Verwendet zusammen mit RESET, siehe dort!
Hewlett-Packard Interface-Loop ("Pill" im HP-Jargon).

HTD (*Stringausdruck für Hex-Zahl zwischen 0 und FFFFF***).** Seite 133
Wandelt den Wert der durch den String verkörperten Hex-Zahl in einen Dezimalwert um.

IBOUND. Mathe-ROM, Seite 103
Ermittelt, ob die zuletzt durchgeführte Integration die angegebene Toleranz einhält.

IDN. <Postfix> Verwendet zusammen mit MAT, siehe dort!

IF *Ausdr.* **THEN** {*Anweisung* \ *Zeilen-Nummer* \ *Label*}
[**ELSE** {*Anweisung* \ *Zeilen-Nummer* \ *Label*}]. Seite 134
Prüft und leitet bedingte Verzweigung(en) ein.

IMAGE [#{ *Vorschubsteuerzeichen* \ , }] *Format-String* Seite 136
Formatierungs-Anweisung für vielseitige Anwendungen.

IMPT (*komplexer num. Ausdruck*). Mathe-ROM, Seite 21
Ermittelt den Imaginär-Teil des komplexen Argumentes.

IN. <Postfix> Verwendet zusammen mit CALL, siehe dort!

INF. Seite 15
Ruft die rechnerspezifische Darstellung des Begriffs "Unendlich" auf.

INIT:{ **1** \ **2** \ **3** } Seite 13 des Benutzerhandbuches!
Initialisiert den Rechner auf einen definierten Zustand.

INPUT [*Eingabeaufforderung* [, *Vorgabestring*] ;] *Variablenliste*
Seite 151, Mathe-ROM, S. 53 und HP-IL-ROM, S. 129
Fordert zur Eingabe (oder Bestätigung) auf.

INSERT # *Kanal-Nummer* , *Record-Nummer* ; *Stringausdruck*.
FORTH- bzw. TEXT-ROM

INT (*num. Ausdruck*). Seite 154
Ermittelt die größte ganze Zahl, deren Wert den des Argumentes nicht überschreitet.

INTEGER *num. Var.*[(*Dimension*)], *num. Var.*[(*Dimension*)]... Seite 155
Dimensioniert INTEGER-Variable bzw.INTEGER-Felder.

INTEGRAL (*untere* , *obere* , *Fehler-Grenze* , *Funktion von* IVAR)
Mathe-ROM, Seite 101
Ermittelt das Integral der gegebenen Funktion innerhalb der Grenzen mit der festgelegten Unsicherheit.

INTO. <Postfix> Verwendet zusammen mit TRANSFORM, siehe dort!

INV. <Postfix> Verwendet zusammen mit MAT, siehe dort!

INX. Seite 157
Ermittelt den Zustand des INX-Flags (-4), das bei Operationen gesetzt wird, die nur mit verringerter Genauigkeit durchgeführt werden konnten.

IO. <Postfix> Verwendet zusammen mit ASSIGN, LIST, OFF und RESTORE, siehe dort!

IP (*num. Ausdruck*). Seite 158
Ermittelt den ganzzahligen Anteil des Argumentes.

IROUND (*num. Ausdruck*). Mathe-ROM, Seite 30
Rundet das Argument auf ganzzahligen Wert, gemäß der geltenden OPTION ROUND-Festlegung.

IS. <Postfix> Verwendet zusammen mit DISPLAY, KEYBOARD und PRINTER, siehe dort!

IVALUE. Mathe-ROM, Seite 102
Gibt die zuletzt mit INTEGRAL berechnete Näherung zurück.

IVAR. Mathe-ROM, Seite 102
Gibt den zuletzt benutzten Wert der Integrations-Variablen zurück.

IVL. Seite 159
Ermittelt den Zustand des IVL-Flags (-8), das beim Versuch der Ausführung einer unzulässigen Operation gesetzt wird.

KEY *Tasten-Name* [, *zugeordneter String*[; \ :]]. Seite 69
Legt Tastenzuordnung im USER-Modus fest.

KEY. <Postfix> Verwendet zusammen mit DEF und FETCH, siehe dort!

KEY$. Seite 160
Ermittelt (und löscht) die älteste im Tasten-Buffer stehende Eintragung.

KEYBOARD IS {*Geräte-Bezeichnung* \ * }. FORTH-ROM

KEYDEF$ (*Tasten-Name*). Seite 162
Ermittelt die für den USER-Modus festgelegte Tasten-zuordnung.

KEYDOWN [(*Tasten-Name*)]. Seite 164
Ermittelt, ob eine (bzw. die bezeichnete) Taste gedrückt wurde.

KEYS. <Postfix> Verwendet zusammen mit CAT, COPY, LIST, PURGE, RENAME, SECURE und UNSECURE, siehe dort!

KEYWAIT$. LEX-File #1 aus User's Library, Hex-ID=52
Liest (und löscht) die älteste im Tasten-Buffer stehende Eintragung. Siehe auch Seite 123 dieses Buches!)

LBND (*Feldbezeichnung* , { 1 \ 2 }. Mathe-ROM, Seite 72
Ermittelt Untergrenze und Gestalt von Matrizen und Feldern

LBOUND siehe LBND

LC [**ON** \ **OFF**]. Seite 166
Schaltet zwischen Groß- und Kleinschreibung um.

LEN (*Stringausdruck*). Seite 167
Ermittelt die Anzahl der Zeichen des Argumentes.

[**LET**] *Variable* = *Ausdruck*. Seite 168
Ordnet der Variablen den Wert des Ausdrucks zu.

[**LET**] **FN***Variable* = *Ausdruck*. Seite 168
Nur innerhalb mehrzeiliger Definitionen für Anwender-Funktionen zulässig!
Ordnet der Variablen den Wert des Ausdrucks zu.

LGT (*num. Ausdruck*). Seite 178
Ermittelt den dekadischen Logarithmus des Argumentes.

LIF1. <Postfix> Verwendet zusammen mit CREATE und TRANSFORM, siehe dort!

LINPUT [*Eingabeaufforderung* [, *Vorgabestring*] ;] *Stringvariable*. Seite 171
Weist den Inhalt der Anzeige (ab Vorgabestring!) der Stringvariablen zu.

LIST [*Anfangszeile* [, *Endzeile*]]. Seite 173
Listet das momentan aufgerufene Programm ganz oder im gewünschten Umfang.

LIST *Programm-Name*[, *Anfangszeile* [, *Endzeile*]]. Seite 173
Listet das bezeichnete Programm ganz oder im gewünschten Umfang.

LIST *Keyfile-Name*[, *Tastennummer* [, *Tastennummer*]]. Seite 173
Listet den bezeichneten Zuordnungsfile ganz oder im gewünschten Umfang.

LIST KEYS [*Tastennummer* [, *Tastennummer*]]. Seite 173
Listet den momentan gültigen KEYS-File ganz oder im gewünschten Umfang.

LIST IO. HP-IL-ROM, Seite 131
Gibt erteilte Zuweisungs-Codes und die HP-IL-Adressen aus.

LN (*num. Ausdruck*). Seite 176
Ermittelt den natürlichen Logarithmus des Argumentes.

LOCAL [*Gerätebezeichnung* \ **LOOP**]. HP-IL-ROM, Seite 132
Schaltet bezeichnetes Gerät (oder alle Geräte der Schleife) von der Fernsteuerung frei.

LOCAL LOCKOUT. HP-IL-ROM, Seite 134
Macht in fernsteuerungsfähigen Geräten die lokalen Einstellorgane wirkungslos.

LOCK *Schlüsselwort*. Seite 175
Erfordert für die Benutzung des Rechners Eingabe des (vereinbarten) Schlüsselwortes.

LOG (*num. Ausdruck*). Seite 176 und Mathe-ROM, Seite 37
Ermittelt für das Argument den natürlichen Logarithmus (mit Mathe-ROM auch für komplexes Argument).

LOG10 (*num. Ausdruck*). Seite 178
Ermittelt für das Argument den dekadischen Logarithmus.

LOG2 (*num. Ausdruck*). Mathe-ROM, Seite 29
Ermittelt für das Argument den Logarithmus zur Basis 2.

LOGP1 (*num. Ausdruck*). Seite 177
Ermittelt für das um 1 vergrößerte Argument den natürlichen Logarithmus.

LR *unabh. Var., abh. Var., 1. num. Var., 2. num. Var.*. Seite 179
Führt für die angegebenen Variablen des momentanen statistischen Feldes eine lineare Regression durch. Der Schnittpunkt mit der Y-Achse wird der 1. Variablen, die Steigung der 2. Variablen zugeordnet.

MAIN <Postfix> Verwendet zusammen mit CAT, siehe dort!

MAT *Feld-\Matrix-Name* = *Feld-\Matrix-Name*. Mathe-ROM, Seite 51
Die Dimension(en) und die Werte des Feldes (bzw. der Matrix) rechts werden auf das Feld (bzw. die Matrix) links übertragen.

MAT *Feld-\Matrix-Name* = (*num. Ausdruck*) Mathe-ROM, Seite 52
Besetzt alle Elemente des Feldes (bzw. der Matrix) mit dem Wert des numerischen Ausdrucks.

MAT *Feld-\Matrix-Name* = **CON**[(*num. Ausdruck*[, *num. Ausdruck*])].
Mathe-ROM, Seite 52
Besetzt alle Elemente des Feldes (bzw. der Matrix) mit dem Wert 1 und dimensioniert das Feld (bzw. die Matrix) um, wenn Argument(e) angegeben.

MAT *Feld-\Matrix-Name* = **IDN**[(*num. Ausdruck*, *num. Ausdruck*)]
Mathe-ROM, Seite 52
Formt (gemäß Argumenten, die Rundungen auf gleiche Ganzzahlen ergeben müssen) das Feld (bzw. die Matrix in quadratische Matrix um und besetzt alle Elemente der Hauptdiagonalen mit dem Wert 1, alle übrigen mit dem Wert 0 -->Einheitsmatrix.

MAT *Feld-\Matrix-Name* = **ZER**[**O**] [(num. Ausdruck[, *num. Ausdruck*])].
Mathe-ROM, Seite 53
Besetzt alle Elemente des Feldes (bzw. der Matrix) mit dem Wert 0 und dimensioniert das Feld (bzw. die Matrix) um, wenn Argument(e) angegeben.

MAT INPUT *Feld-\Matrix-Namens-Liste*.
Mathe-ROM, Seite 53
Fragt die einzugebenden Werte für die einzelnen Felder (bzw. Matrizen) nacheinander ab.

MAT DISP [**USING** {*Format-String\Zeilen-Nummer*};] *Feld-\Matrix-Name*
[{ , \ ; }[*Feld-\Matrix-Name*]]...[].
Mathe-ROM, Seiten 54/55
Zeigt nacheinander alle Elemente der benannten Felder (bzw. Matrizen) an.

MAT PRINT [**USING** {*Format-String\Zeilen-Nummer*};]
Feld-\Matrix-Name[{ , \ ; }[*Feld-\Matrix-Name*]]...[].
Mathe-ROM, Seiten 55/56
Druckt nacheinander alle Elemente der benannten Felder (bzw. Matrizen) aus.

MAT *Feld-\Matrix-Name* = - *Feld-\Matrix-Name*.
Mathe-ROM, Seite 63
Das Feld (bzw. die Matrix) links wird gemäß den Dimensionen des Feldes (bzw. der Matrix) rechts dimensioniert. Die Elemente des Feldes (bzw. der Matrix) rechts werden unter Umkehr der Vorzeichen den Elementen links zugeordnet.

MAT *Feld-\Matrix-Name* = *Feld-\Matrix-Name* + *Feld-\Matrix-Name*.
Mathe-ROM, Seite 64
Addiert entsprechende Elemente der Felder (bzw. Matrizen) rechts und überträgt die Summen an die Elemente des Feldes (bzw. der Matrix) links.

MAT *Feld-\Matrix-Name* = *Feld-\Matrix-Name* - *Feld-\Matrix-Name*.
Mathe-ROM, Seite 64
Subtrahiert entsprechende Elemente des Feldes (bzw. der Matrix) ganz rechts von Elementen des Feldes (bzw. der Matrix) in der Mitte und überträgt Differenzen an die Elemente des Feldes (bzw. der Matrix) ganz links.

MAT *Feld-\Matrix-Name* = (*num. Variable*) * *Feld-\Matrix-Name*.
Mathe-ROM, Seite 65
Multipliziert das Feld (bzw. die Matrix) rechts mit dem Argument, bringt das Feld (bzw. die Matrix) links auf gleiche Dimension(en) und besetzt die Elemente des Feldes (bzw. der Matrix) links mit den Produkten.

MAT *Feld-\Matrix-Name* = *Feld-\Matrix-Name* * *Feld-\Matrix-Name*.
Mathe-ROM, Seite 65
Multipliziert entsprechende Elemente der Felder (bzw. Matrizen) und überträgt Produkte an die Elemente des Feldes (bzw. der Matrix) links.

MAT *Feld-\Matrix-Name* = **TRN**(*Matrix-Name*) * *Feld-\Matrix-Name*.
Mathe-ROM, Seite 66
Formt Feld (bzw. Matrix) links den Forderungen der rechts stehenden Matrix-Feld- oder Matrix-Matrix-Konstellation um, bildet die Transponierte der mittleren Matrix, multipliziert diese mit dem Feld (bzw. der Matrix) rechts und besetzt mit den Ergebnissen das Feld (bzw. die Matrix) links.

MAT *Matrix-Name* = **INV**(*Matrix-Name*).
Mathe-ROM, Seite 77
Formt linke Matrix entsprechend der rechten (quadratischen) Matrix um, bildet die Invertierte der rechten Matrix und besetzt mit den Werten die Elemente der linken Matrix.

MAT *Matrix-Name* = **TRN**(*Matrix-Name*).
Mathe-ROM, Seite 77
Formt linke Matrix entsprechend der rechten Matrix um, bildet die Transponierte der rechten Matrix und besetzt mit den Werten die Elemente der linken Matrix.

MAT *Feld-\Matrix-Name* = **SYS**(*Matrix-Name*, *Feld-\Matrix-Name*).
Mathe-ROM, Seite 79
Formt Feld (bzw. Matrix) links dem Feld (bzw. der Matrix) rechts um, berechnet aus der mittleren Matrix und dem Feld (bzw. der Matrix) rechts die Werte für die Elemente des Feldes (bzw. der Matrix) links so, daß folgende Matrizen-Gleichung erfüllt ist:

Matrix * Feld\Matrix = Feld\Matrix
(rechts) (Mitte) (links)

MAT *Komplex-Feld-\-Matrix-Name* = **PROOT** (*Feld-\Matrix-Name*).
Mathe-ROM, Seite 120
Formt komplexes Feld (bzw. komplexe Matrix) links passend für reelles Feld (bzw. reelle Matrix) rechts und errechnet alle (auch komplexe) Nullstellen des Polynoms, dessen Koeffizienten in dem Feld (bzw. der Matrix) rechts abgelegt sind.

MAT *Komplex-Feld-\-Matrix-Name* = **FOUR** (*Komplex-Feld-\-Matrix-Name*).
Mathe-ROM, Seite 135
Formt komplexes Feld (bzw. komplexe Matrix) links passend für komplexes Feld (bzw. komplexe Matrix) rechts und errechnet die komplexen Koeffizienten der finiten Fourier-Transformierten, deren Punkt-Koordinaten in dem komplexen Feld (bzw. der komplexen Matrix) rechts abgelegt sind.

MATH. <Postfix> Verwendet mit CFLAG und SFLAG, siehe dort!

MAX (*num. Ausdruck*, *num. Ausdruck*). Seite 181
Ermittelt das Argument mit dem größeren Wert.

MAXREAL. Seite 182
Ruft größte, durch Rechner darstellbare Zahl auf.

MEAN [(*Variablenbezeichnung*)]. Seite 183
Bestimmt den Mittelwert der angegebenen Variablen des momentanen statistischen Feldes.

MEM [(*Port-Nummer p.dd*)]. Seite 184
Ermittelt die Zahl der freien Bytes im Hauptspeicher, bzw. im angegebenen Port.

MERGE *Programm-Name* [, *Anfangszeile* [, *Endzeile*]]. Seite 186
Fügt das bezeichnete Programm im gewünschten Umfang in das momentan aufgerufene Programm ein.

MERGE *Keyfile-Name* [, *Tastennummer* [, *Tastennummer*]]. Seite 186
Übernimmt den bezeichneten Key-File im gewünschten Umfang in den momentan gültigen System-File "keys".

MIN (*num. Ausdruck*, *num. Ausdruck*). Seite 188
Ermittelt das Argument mit dem kleineren Wert.

MINREAL. Seite 189
Ruft die kleinste, durch Rechner darstellbare Zahl auf.

-. <Operator> Anwendung: *num. Ausdr.* - *num. Ausdr.*. Seite 309
Subtrahiert rechtes Argument vom linken.

MOD (*num. Ausdruck*, *num. Ausdruck*). Seite 190
Teilt linkes Argument durch rechtes, gibt Rest zurück.

MSG$ (*Meldungs-Nummer*). LEX-File #1 aus User's Library, Hex-ID=52

NAME *File-Name*. Seite 191
Weist dem "workfile" den angegebenen Namen zu.

NAN. Seiten 192 und 343
Erzeugt ein "aktives" NaN ("aktiv" im Sinne von "vorsorglich"). NaN oder NAN bezeichnet eine nicht zu definierende Zahl, die z.B. nach einer unerlaubten Rechenoperation auftreten kann.

NAN$ (*num. Ausdruck*). Mathe-ROM, Seite 30
Gibt Nullstring zurück, falls sich das Argument im normalen Zahlenbereich bewegt, sonst String mit Fehlernummer (wenn das Argument nur noch mit NaN bezeichnet werden kann).

NEAR. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

NEG. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

\- {*num. Ausdruck***NaN*}. <Operator> Seite 309
Kehrt Vorzeichen des folgenden Argumentes um.

NEIGHBOR (*num. Ausdruck, num. Ausdruck*). Mathe-ROM, Seite 30
Ermittelt für das linke Argument den unmittelbar benachbarten, vom Rechner darstellbaren Wert in Richtung auf das rechte Argument.

NEXT *num. Variable*. Seite 118
Übergibt den Programmablauf an die davor nächstliegende FOR-Anweisung, die sich auf die gleiche Variable bezieht, sofern der Schleifenzähler noch nicht den Endwert überschritten hat.

NOT *num. Ausdruck* <Operator> Seite 193
Wendet logische NICHT-Operation auf das Argument an. Gibt 1 zurück, falls Argument gleich Null ist. Bei allen anderen Werten des Argumentes wird 0 zurückgegeben.

NUM (*String-Ausdruck*). Seite 194
Ermittelt den ASCII-Code des am weitesten links stehenden Zeichens des Argumentes.

OFF. <Postfix> Verwendet zusammen mit BEEP, CONTROL, DEFAULT, LC, STANDBY, TRACE und USER, siehe dort!

OFF {**ERROR** \ **TIMER #** *Timer-Nummer*}. Seite 195
Hebt vorausgegangene ON ERROR- bzw. ON TIMER-Anweisungen auf.

OFF. Schaltet den HP-71 aus. Seite 195

OFF {**INTR** \ **IO**}. HP-IL-ROM, Seite 136/37
Hebt vorausgegangene ON INTR-Vereinbarung auf bzw. unterbricht den I/O-Betrieb des HP-71.

ON *num. Ausdr.*{**GOSUB** \ **GOTO** \ **RESTORE**} *Liste von Zeilen-Nummern oder Labels*. Seite 199
Verzweigt das Programm entsprechend dem ganzzahligen Teil des Argumentes.

ON ERROR {**GOSUB** \ **GOTO**}{*Zeilen-Nummer* \ *Label*}. Seite 197
Vorbereitung zum Verzweigen zur bezeichneten Stelle des Programms bei Auftreten eines Fehlers.

ON INTR {**GOSUB** \ **GOTO**}{*Zeilen-Nummer* \ *Label*}.
HP-IL-ROM, Seite 139
Vorbereitung zum Verzweigen zur bezeichneten Stelle des Programms bei Auftreten eines Interrupts.

ON TIMER # *Timer-Nummer, num. Ausdr.* {**GOSUB** \ **GOTO**} {*Zeilen-Nummer* \ *Label*}. Seite 201
Vorbereitung zum Verzweigen zur bezeichneten Stelle des Programms nach Ablauf der angegebenen Zeit.

ON. <Postfix> Verwendet zusammen mit BEEP, CONTROL, DEFAULT, LC, STANDBY, und USER, siehe dort!

OPTION ANGLE {**DEGREES** \ **RADIANS**}. Seite 204
Wählt Maßeinheit (Altgrad\Bogenmaß) für Winkeldarstellung.

OPTION BASE { **0** \ **1** }. Seite 204
Legt die Untergrenze der Indizes auf 0 bzw. 1 fest.

OPTION ROUND {**NEAR****NEG****POS****ZERO**}. Seite 204
Legt die Art der Rundung von numerischen Ausdrücken fest.

OR <Operator> Anwendung: *num. Ausdr.* **OR** *num. Ausdr.*. Seite 206
Logische ODER-Verknüpfung, nur wahr (1), wenn mindestens eines der Argumente von Null verschieden ist, sonst falsch (0).

OUTPUT {*Gerätebezeichnung***LOOP**}[**USING** {*Formatstring**Zeilennummer*}] [; *Ausgabeliste*]. HP-IL-ROM, Seite 142
Gibt numerische Werte und Strings gemäß der Ausgabeliste an das bezeichnete Gerät oder an die Schleife.

OVF. Seite 207
Ermittelt den Zustand des OVF-Flags (-6), das bei Bereichsüberschreitungen gesetzt wird.

PACK *Speicherbezeichnung*. HP-IL-ROM, Seite 150
Entfernt Leer-Files aus dem Katalog und den Eintragungen des bezeichneten Mediums.

PACKDIR *Speicherbezeichnung*. HP-IL-ROM, Seite 152
Entfernt lediglich aus dem Katalog des bezeichneten Mediums die Leer-Files.

PASS CONTROL [*Gerätebezeichnung***LOOP**]. HP-IL-ROM, Seite 154
Gibt die Controller-Funktion des Rechners an das bezeichnete Gerät bzw. an die Schleife zurück.

PAUSE. Seite 208
Nur im Programm-Modus ausführbar!
Hält die Ausführung des laufenden Programms an.

PCRD. Seite 292 des Benutzerhandbuches
Nur mit Kartenleser möglich!
Deklariert die so beschriebene Magnetkarte als "privat" (Kopierschutz!).

PEEK$ (*Hex-Adress-String, Zahl der Nibbels*). Seite 209
Liest die angegebene Zahl Nibbels ab der genannten Adresse aus dem Speicher.

PI. Ruft die 12-stellige Näherung der Zahl Pi auf. Seite 210

PLIST [*Anfangszeile* [, *Endzeile*]]. Seite 211
Läßt das momentan aufgerufene Programm durch die als Drucker deklarierte Einheit ganz oder im gewünschten Umfang ausgeben.

PLIST *Programm-Name* [*Anfangszeile* [, *Endzeile*]]. Seite 211
Läßt das bezeichnete Programm durch die als Drucker deklarierte Einheit ganz oder im gewünschten Umfang ausgeben.

PLIST *Key-File-Name*[, *Tastennummer* [, *Tastennummer*]]. Seite 211
Läßt den bezeichneten Zuordnungsfile durch die als Drucker deklarierte Einheit ganz oder im gewünschten Umfang ausgeben.

PLIST KEYS [*Tastennummer* [, *Tastennummer*]]. Seite 211
Läßt den momentan gültigen KEYS-File durch die als Drucker deklarierte Einheit ganz oder im gewünschten Umfang ausgeben.

POKE *Hex-Adress-String, Datenstring.* Seite 213
Legt die Daten im Speicher ab, beginnend an der genannten Adresse.

POLAR (*komplexer num.Ausdruck*). Mathe-ROM, Seite 40
Berechnet die Polar-Koordinaten des (kartesischen) Argumentes.

POP. Seite 215
Verhindert den Rücksprung aus einem Unterprogramm in das aufrufende Programm, macht vorangehendes GOSUB unwirksam.

PORT. <Postix> Verwendet zusammen mit CAT, CLAIM, FREE und SHOW, siehe dort!

POS (*Stringausdr. 1, Stringausdr. 2* [, *num.Ausdr.*]). Seite 216
Ermittelt die Position des Strings 2 im String 1, beginnend an der durch das num. Argument gegebenen Stelle.

POS. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

PREDV (*num. Ausdruck*). Seite 218
Nur nach vorangegangenem LR möglich!
Berechnet den Funktionswert des Argumentes unter Verwendung der mit LR gewonnenen Koeffizienten.

PRINT [*Ausdr.*\ **TAB** (*num.Ausdr.*)[{ ; \ , } *Ausdr.*]][; \ ,]...
Seite 220 und Mathe-ROM, Seite 55
Tabelliert Ausdrücke, die auf dem als Drucker deklarierten Gerät ausgegeben werden sollen.

PRINT USING {*Formatstring**Zeilennummer*}[{;\,}*Ausdr.*]][;\,]...
Formatiert Ausdrücke, die auf dem als Drucker deklarierten Gerät ausgegeben werden sollen.

PRINT # *Kanalnummer*[, *Recordnummer*]; *Datenliste.*
Schreibt die in der Liste stehenden Daten in den der Kanalnummer zugeordneten File.

PRINTER IS {*Gerätebezeichnung* \ * }. HP-IL-ROM, Seite 156
Legt das auf PRINT-Anweisungen reagierende Gerät fest.

PRIVATE *File-Namen.* Seite 225 und HP-IL-ROM, Seite 159
Schützt den angegebenen Programm-File gegen Ändern und Auflisten.

PROJ (*komplexer num.Ausdr.*). Mathe-ROM, Seite 42
Ermittelt die Projektion des Argumentes auf sich selbst, sofern der Betrag des Argumentes den Wertebereich des Rechners nicht überschreitet.

PROOT. <Postfix> Verwendet zusammen mit MAT, siehe dort!

PROTECT. Seite 226
Nur mit Kartenleser möglich!
Schützt eine Spur der Magnetkarte gegen Überschreiben!

PURGE {*File-Angaben***KEYS****ALL**}. Seite 227
Löscht den bezeichneten File bzw. den KEYS-File bzw. alle (ungesicherten) Files des Hauptspeichers.

PUT *Tastenname*. Seite 229
Setzt die angegebene Taste an das Ende des Tastenbuffers.

PWIDTH *num. Ausdruck*. Seite 230
Legt die Zeilenlänge für die nachfolgenden PRINT-Anweisungen fest.

RAD (*num. Ausdruck*). Seite 231
Rechnet das als Altgrad gewertete Argument in das Bogenmaß um.

RADIANS. Seite 232
Schaltet den Winkelmodus auf Bogenmaß.

RADIANS. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

RANDOMIZE [*num. Ausdruck*]. Seite 233
Legt den Startwert des Generators für Zufallszahlen durch das Argument (#0) oder den Momentanwert des Zeitgenerators fest.

READ *Variablen-Liste*. Seite 234
Besetzt die aufgelisteten Variablen der Reihe nach mit den in DATA stehenden Ausdrücken.

READ # *Kanalnummer*[, *Recordnummer*]; *Variablen-Liste*. Seite 236
Besetzt die in der Liste stehenden Variablen mit den Daten des der Kanalnummer zugeordneten Files.

READDDC. HP-IL-ROM, Seite 165
Ermittelt die Nummer des zuletzt empfangenen gerätetypischen Befehls.

READINTR. HP-IL-ROM, Seite 165
Liest (und löscht) den Inhalt des Interrupt-Registers.

REAL *num. Var.*[*(Dimension(en))*], *num. Var.*[*(Dimension(en))*].
Seite 238
Dimensioniert REAL-Variable bzw. REAL-Felder.

RECT (*komplexer num. Ausdruck*). Mathe-ROM, Seite 40
Berechnet die kartesischen Koordinaten des als Polar-Koordinaten gewerteten Argumentes.

RED (*num. Ausdruck 1, num. Ausdruck 2*). Seite 240
Berechnet die kleinste Differenz zum nächsten ganzzahligen Quotienten aus den beiden Argumenten.

REM. Seite 242, (Durch ! als Kurzform ersetzbar.)
Kennzeichen für Bemerkungen innerhalb einer Programm-Zeile, die keinerlei Rechneraktivität auslösen sollen.

REMOTE {*Gerätebezeichnung***LOOP**}. HP-IL-ROM, Seite 168
Versetzt das bezeichnete Gerät (oder alle Geräte der Schleife) in den Fernsteuer-Modus.

RENAME [*File-Angaben***KEYS**] **TO** {*File-Namen***KEYS**}.
Seite 243 und HP-IL-ROM, Seite 171
Belegt den bezeichneten File mit dem neuen Namen.

RENUMBER [*neue Anfangszeilennummer*[, *neues Inkrement*
[, *alte Anfangszeilennummer*[, *alte Endzeilennummer*]]]].
Seite 245
Numeriert den durch die Alt-Angaben bezeichneten Teil des Programms nach den neuen Anweisungen.

RENUMBER 1,1,1,1.
Sonderfall: Es erfolgt zwar keine Umnumerierung, alle Verzweigungen werden aber kompiliert, d.h. für den Ablauf vorbereitet.

REPLACE # *Kanalnummer, Recordnummer*; *String*. FORTH- und TEXT-ROM
Setzt den String in den bezeichneten Record.

REPT (*komplexer num. Ausdruck*). Mathe-ROM, Seite 21
Ermittelt den Real-Teil des komplexen Argumentes.

REQUEST *num. Ausdruck*. HP-IL-ROM, Seite 173
Besetzt das (nur extern) mit SPOLL abzufragende Status-Register mit dem (ganzzahligen) Argument (Werte von 0 bis 255 zulässig).

RES. Seite 247
Ruft den zuletzt berechneten Wert auf.

RESET. Seite 248
Setzt Benutzer- und System-Flags auf den Einschalt-zustand zurück.

RESET CLOCK. Seite 248
Hebt die Wirkung einer EXACT-Anweisung auf.

RESET HPIL. HP-IL-ROM, Seite 177
Versetzt das HP-IL-Interface in einen vordefinierten Zustand.

RESET ESCAPE. FORTH-ROM

RESTORE [*Zeilennummer\Label*]. Seite 249
Setzt den Zeiger auf den Anfang der (durch Zeilennummer oder Label bezeichneten) DATA-Anweisung.

RESTORE IO. HP-IL-ROM
Gibt den Betrieb auf der HP-IL-Schleife frei.

RESTORE # *Kanalnummer*[, *Recordnummer*]. Seite 250
Setzt den File-Zeiger auf den Anfang (des bezeichneten Records).

RESTORE. <Postfix> Verwendet zusammen mit ON, siehe dort!

RETURN. Seite 251
Gibt den Programmablauf vom Unterprogramm an das aufrufende Programm an die auf das GOSUB folgende Anweisung zurück.

RMD (*num. Ausdruck 1, num. Ausdruck 2*). Seite 252
Ermittelt den Rest, der sich bei der Division des 1. Argumentes durch das 2. Argument ergibt.

RND. Seite 254
Gibt den nächsten, vom Zufalls-Zahlen-Generator bereitgestellten Wert aus.

RNORM (*Matrix- oder Feld-Name*). Mathe-ROM, Seite 70
Ermittelt die höchste Summe aus den Beträgen aller Elemente einer Zeile.

ROUND. <Postfix> Verwendet zusammen mit OPTION, siehe dort!

RUN [*Zeilennummer\File-Name* [, {*Zeilennummer\Label*}]].
Seite 255
Startet die Ausführung eines im Speicher des Rechners stehenden Programms an der bezeichneten Stelle.

RUN *File-Angabe.* HP-IL-ROM, Seite 181
Kopiert den angegebenen Programm-File in den Hauptspeicher des Rechners und startet dieses Programm.

SCALE10 (*num. Ausdr., ganzzahl. num. Ausdr.*). Mathe-ROM, Seite 29
Addiert das ganzzahlige (2.) Argument zum Exponenten des 1. Argumentes (Vermeidet damit Überlauf).

SCI *num. Ausdruck*. Seite 257
Schaltet auf Darstellung im wissenschaftlichen Format, rundet Argument und grenzt auf Werte von 0 bis 11 ein. Dadurch Zahl der signifikanten Stellen von 1 bis 12.

SCROLL *Positions-Nr.*. LEX-File #1 aus User's Library, Hex-ID=52

SDATA. <Postfix> Verwendet zusammen mit CREATE, siehe dort!

SDEV [(*Variablenbezeichnung*)]. Seite 259
Bestimmt die Standardabweichung der angegebenen Variablen des momentanen statistischen Feldes.

SEARCH (*Stringausdruck, Seitennummer., Startzeile, Schlußzeile, Kanalnummer*). FORTH- bzw. TEXT-ROM
Durchsucht Text-File im bezeichneten Bereich nach dem angegebenen String.

SECURE [*File-Angaben***KEYS**]. Seite 260 und HP-IL-ROM, Seite 183
Sichert momentanen (oder angegebenen) File gegen Löschen und Ändern.

SEND [*Liste von HP-IL-Meldungen*]. HP-IL-ROM, Seite 185
Sendet vorbereitete (oder aufgelistete Meldungen) aus.

SETDATE {*numerisches Datum**Datums-String*}. Seite 262
Stellt die Systemuhr auf das angegebene Datum.

SETTIME {*Sekunden seit Mitternacht*\"hh:mm:ss"}. Seite 264
Stellt die Systemuhr auf die angegebene Zeit.

SFLAG { **ALL** \ **MATH** \ *Liste der Flags von -32 bis 64*}. Seite 267
Setzt angegebene Flags.

SGN (*numerischer Ausdruck*). Seite 268 und Mathe-ROM, Seite 41
Ermittelt das Vorzeichen des Argumentes.
Mathe-ROM: Ermittelt Einheitsvektor des Argumentes.

SHORT *num. Var.*[(*Dimension(en)*)], *num. Var.*[(*Dimension(en)*)]...
Dimensioniert SHORT-Variable bzw. SHORT-Felder.

SHOW [:] **PORT**. Seite 271
Zeigt Besetzung der Ports und die Aufteilung der unabhängigen Speicher des Rechners an.
Nur über Tastatur!

SIN (*num. Ausdruck*). Seite 272 auch Mathe-ROM, Seite 38
Ermittelt den Sinus des Argumentes.

SINH (*num. Ausdruck*). Mathe-ROM, Seiten 27 u. 39
Ermittelt den Hyperbel-Sinus des Argumentes.

SPOLL (*Gerätebezeichnung*). HP-IL-ROM, Seite 192
Fragt Status des bezeichneten HP-IL-Gerätes ab.

SQR (*num. Ausdruck*). Seite 273 auch Mathe-ROM, Seite 40
Ermittelt den auf 12 Stellen gerundeten Wert der Quadratwurzel aus dem Argument.
Mathe-ROM: Ermittelt komplexen Wert der Quadratwurzel aus dem Argument.

SQRT (*num. Ausdruck*). siehe SRQ!

STANDBY { **ON** \ **OFF** \ *Gesamtwartezeit* [, *Prüfabstand*]}.
HP-IL-ROM, Seite 197
Prüft die HP-IL-Schleife wiederholt in gegebenen Abständen, bis die Gesamtwartezeit verstrichen ist.

STARTUP *Stringausdruck*. Seite 274
Legt Anweisung(en) fest, die der Rechner nach dem Einschalten (im BASIC-Modus) sofort ausführt.

STAT *Feldname* [(*Anzahl der Variablen*)]. Seite 275
Wählt bezeichnetes Feld zum momentanen statistischen Feld (oder legt neues Feld mit diesem Namen und der angegebenen Zahl von Variablen an).

STATUS. HP-IL-ROM, Seite 200
Ermittelt den momentanen Zustand (Status) des HP-IL-Interfaces.

STD. Seite 277
Setzt den Rechner auf das Standard-BASIC-Anzeige-Format (Kombination von FIX 11 und SCI 11).

STEP. <Postfix> Verwendet zusammen mit FOR, siehe dort!

STOP. Seite 279
Beendet die Programmausführung, und macht die Fortsetzung des Programms mit CONT unmöglich!

STR$ (*num. Ausdruck*). Seite 280
Formt einen String aus dem Argument.

SUB *Unterprogramm-Name* [(*Liste der Parameter*)]. Seite 282
Deklariert das so bezeichnete Programm als (aufrufbares) Unterprogramm.

SUB. <Postfix> Verwendet zusammen mit END, siehe dort!

SYS. <Postfix> Verwendet zusammen mit MAT, siehe dort!

TAB. <Postfix> Verwendet zusammen mit DISP und PRINT, siehe dort!

TAN (*num. Ausdruck*). Seite 284 auch Mathe-ROM, Seite 38
Ermittelt den Tangens des Argumentes.

TANH (*num. Ausdruck*). Mathe-ROM, Seiten 27 und 39
Ermittelt den Hyperbel-Tangens des Argumentes.

TEXT. <Postfix> Verwendet zusammen mit CREATE und TRANSFORM, siehe dort!

THEN. <Postfix> Verwendet zusammen mit IF, siehe dort!

TIME. Seite 285
Ermittelt die Zahl der seit Mitternacht vergangenen Sekunden.

TIME$. Seite 286
Ermittelt die Uhrzeit als String der Form hh:mm:ss.

TIMER. <Postfix> Verwendet zusammen mit OFF und ON, siehe dort!

TO. <Postfix> Verwendet zusammen mit ASSIGN#, COPY, FOR und RENAME, siehe dort!

TOTAL [(*Variablennummer*)]. Seite 287
Ermittelt im momentanen statistischen Feld die Summe aller unter dieser Variablen abgelegten Werte.

TRACE { **FLOW** \ **VARS** \ **OFF** }. Seite 288
Protokolliert Programmablauf (mit Angabe der Werte der Variablen).

TRANSFORM [*File-Angaben*] **INTO** *File-Typ* [*File-Name*].
Seite 289 auch HP-IL-ROM, Seite 203
Wandelt BASIC-Programm-Files in TEXT-Files (und umgekehrt) um.

TRAP (*Flagnummer*[, { **0** \ **1** }]). Seite 293
Ermittelt die momentane Vereinbarung des bezeichneten Flags (und setzt es auf neuen Wert).

TRIGGER [*Gerätebezeichnung* \ **LOOP**]. HP-IL-ROM, Seite 205
Startet im bezeichneten Gerät (oder allgemein in der Schleife) einen bestimmten, vorher festgelegten Vorgang.

TRN. <Postfix> Verwendet zusammen mit MAT, siehe dort!

TYPE (*num. Ausdruck*). Mathe-ROM, Seite 31
Ermittelt die Art und das Format des Argumentes.

UBND (*Feldbezeichnung* , { 1 \ 2 }. Mathe-ROM, Seite 71
Ermittelt Obergrenze und Gestalt von Feldern und Matrizen.

UBOUND siehe UBND

UNF. Seite 295
Ermittelt den Zustand des UNF-Flags (-5), das bei Bereichsunterschreitungen gesetzt wird.

UNPROTECT Seite 296
Nur mit Kartenleser möglich!
Hebt den Schutz für eine Spur der Magnetkarte auf!

UNSECURE [*File-Angaben***KEYS**]. Seite 297 und HP-IL-ROM, Seite 207
Hebt Schutz für momentanen (bzw. angegebenen) File auf.

UPRC$ (*Stringausdruck*). Seite 298
Liest alle im Argument stehenden Kleinbuchstaben (aber nicht die Umlaute!) als Großbuchstaben.

USER [**ON** \ **OFF**]. Seite 299
Wechselt zwischen Normal- und User-Tastenfeld (oder schaltet USER-Modus ein und aus).

USING. <Postfix> Verwendet zusammen mit DISP, ENTER, OUTPUT und PRINT, siehe dort!

VAL (*Stringausdruck*). Seite 300
Wandelt Ziffern-Zeichenfolge im Argument in ein numerisches Argument um.

VARS. <Postfix> Verwendet zusammen mit TRACE, siehe dort!

VER$. Seite 301
Ermittelt die Versionen vom Rechner-Betriebssystem, von allen ROM's und allen zugänglichen LEX-Files.

WAIT *numerischer Ausdruck*. Seite 302
Wertet das Argument als Sekunden und hält die Programmausführung für diese Zeit an.

WIDTH *numerischer Ausdruck*. Seite 303
Legt die Zeilenlänge für die nachfolgenden DISP- und LIST-Anweisungen fest.

WINDOW *num. Ausdruck 1* [, *num. Ausdruck 2*]. Seite 305
Legt durch die beiden Argumente Anfang und Ende des Anzeigefensters fest.

ZER. <Postfix> Verwendet zusammen mit MAT, siehe dort!

ZERO. <Postfix> Verwendet zusammen mit MAT und OPTION, siehe dort!

Anwender-Gruppen:

Wer für den HP-71 Ansprechpartner finden möchte, um sich über spezielle Probleme zu unterhalten oder Erfahrungen auszutauschen, soll hier noch einige Adressen erfahren:

Interessante Einzelheiten über die Rechner HP-41, HP-75, HP-71 und die HP-Rechner der 80er Serie sind im **PRISMA** zu finden. Das ist die Clubzeitschrift vom

CCD
Computerclub Deutschland e.V.
Postfach 2129
D-6242 Kronberg 2

Probehefte dieser Zeitschrift sind gegen Einsendung von 10,00 DM (in bar oder als Verrechnungsscheck) erhältlich von:

Detlev Bock
Petrikirchstr. 36
D-3400 Göttingen

In Österreich finden HP-Freunde Unterstützung durch den

* CCA *
HP-Club Austria
Postfach 50
A-1111 Wien

Wer auch über die Grenzen der deutschen Sprache hinweg Informationen sucht, soll hier noch zwei weitere Adressen finden, von denen sich die Mitgliedsbedingungen (gegen Einsendung von jeweils 1 Dollar) erfragen lassen:

Club for HP Handheld Users
2545 W. Camden Place
Santa Ana, CA 92704
U. S. A.

und

PPC
P. O. Box 9599
Fountain Valley, CA 92728
U. S. A.

COMPUTERBÜCHER IM HELDERMANN VERLAG

Albers, K.: HP-41 Barcodes mit dem HP-IL-System.
1986, 320 Seiten, ISBN 3-88538-804-9, DM 44.-
Der Barcodelesestift ist ein preiswertes, dabei jedoch sehr effizientes Zubehörteil für das Einlesen von Daten oder zum Programmieren. Barcodes (BCs) sind die kostengünstigste Methode der externen Datenspeicherung und deren Massenverbreitung. Über HP-41 BCs, ihre oft verblüffend ergiebige Anwendung, Herstellung und ihren Aufbau ist wenig dokumentiert. Mit diesem Buch schließt der Autor die Lücke.
Das Buch behandelt ausführlich Vor- und Nachteile von BCs und Lesestift, gibt eine Übersicht über alle BC-Typen des 41-er Systems und erläutert den grundsätzlichen Aufbau und das Kodierungsschema. Die Herstellung von BCs auf dem IL-Thermodrucker, dem ThinkJet-Drucker und dem Plotter 7470A mit vielen nützlichen Hinweisen für einwandfreie Ergebnisse, sowie die dafür notwendigen oder wünschenswerten Geräte werden eingehend beschrieben. 2-Byte-BCs der ASCII-Zeichen 0-127, 3-Byte-Ausführung 0-255, Alpha- und synthetische Textzeilen-BCs zum Programmieren, überraschend einfache und schnelle Verfahren für 'load bytes' ohne Hilfsprogramm und BCs von einfachen oder längeren 'BLDSPEC'-Sonderzeichen geben rationelle Arbeitshilfen. Die Drucker-Modi 8-Bit/ESCAPE werden genau erklärt. BCs von Zahlen, trickreiches Programmieren großer Zahlen, Kurzformexponenten, numerische und Alphafolgedaten für sequentielle Registereingabe werden ausführlich beschrieben. BCs von 1- und 2-Byte-Tastenfeldfunktionen und von nicht programmierbaren Rechnerfunktionen, INDirekt, besondere Funktionen und deren BCs werden erschöpfend behandelt. Aus einer speziellen Art BCs bewirkt 'SNAP 2', der BC-Byteschnapper die verblüffend problemlose Lesestifteingabe jedes synthetischen Befehls an beliebiger Programmstelle ohne Tastenbelegungen. XROM-Funktionen-BCs, Mehrbyte-BCs von Tastenfeldfunktionen mit und ohne Argument, alle nicht programmierbaren Rechner- und XROM-Funktionen mit Argument (SIZE, DEL, ASN, PRP usw.) sowie BCs von LBL/GTO/XEQ-"Alpha" und für ein simuliertes synthetisches Tastenfeld für Sofortausführung jeder Art synthetischer Befehle ohne Tastenzuweisung erweitern die Arbeitsmöglichkeiten ganz erheblich. Der Aufbau von Programm-BCs und deren Herstellung auf dem Thermodrucker ohne das Plottermodul, auf dem ThinkJet-Drucker und dem Plotter wird in allen Einzelheiten erklärt. Eine große Zahl von Tabellen der Funktions- und Alphazeichen 0-127, 0-255 (replace und append), für das synthetische Programmieren, für ein simuliertes, synthetisches Tastenfeld sowie der XROM-Befehl aller zur Zeit verfügbaren Systemmodule sind willkommene Arbeitserleichterungen auch für den Anwender, der keinen Drucker besitzt. Es werden eine Menge praktischer Anwendungen gezeigt für das synthetische Programmieren, zur Datenmasseneinlesung für Schriftfahnen oder Querdruck, zur Herstellung beliebiger Textzeilen und zum Umgang mit speziellen LBL, GTO bzw. XEQ-Befehlen.
Jedes Kapitel enthält komfortable Programme zur BC-Herstellung oder für Umrechnungen in Form von Listing und BCs, insgesamt über 90 Programme! Nach Möglichkeit sind alle Programme dreifach vorhanden: 1. zur Herstellung nur mit Rechner und Thermodrucker ohne Systemmodule; 2. zusätzlich mit dem XF-Modul und 3. außerdem mit dem Plottermodul. Durch 1. sind die Möglichkeiten der BC-Herstellung weitestgehend auch dem Benutzer erschlossen, der nicht über Zusatzmodule vefügt. In einem Anhang werden nicht HP-BCs wie der Europäische Artikelnummern-BC und einige anderen Typen vorgestellt. Weiterhin wird ein Ausblick auf die zukünftige Entwicklung gegeben.
Wer die Möglichkeiten seines Lesestiftes mit dem HP-41 System optimal nutzen und selbst BCs anfertigen will, braucht dieses leicht verständlich geschriebene Buch. Es wird ihm bald zur unentbehrlichen Arbeitshilfe werden.

Dearing, J. S.: Tricks, Tips und Routinen für Taschenrechner der Serie HP-41
1984, 220 Seiten, ISBN 3-88538-801-4, DM 36.-
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, Heinz Dalkowski, 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.

Horn, J.: HP-71 Basic - leicht gemacht
1986, 170 Seiten, ISBN 3-88538-807-3, DM 44.-
Der HP-71 bietet bei erstaunlich geringen Abmessungen Anwendungmöglichkeiten in einer Vielfalt, wie sie bisher bei Rechnern dieser Größe unbekannt war. Es ist klar, daß zu einem so komfortablen Rechner auch entsprechend umfangreiche Handbücher gehören. Deren Lektüre stellt aber, nicht nur für Computer-Neulinge, eine teils harte, teils zu gehaltvolle Kost dar, deren Genuß durchaus zu Schwierigkeiten führen kann. In diesem Vergleich spielt nun das Buch "HP-71 Basic - leicht gemacht" die Rolle des Kräuter-Likörs, der Völlegefühl und Verstimmungen beseitigt und damit weiteren Appetit auf das ganze Menue macht. Der betont lockere Stil des Originaltextes wurde auch in der deutschen Bearbeitung voll beibehalten, weil der Umgang mit einem Rechner vorwiegend Freude und möglichst wenig Streß erzeugen soll.

Jarett, K.: Erweiterte Funktionen des HP-41 - leicht gemacht
1986, 240 Seiten, ISBN 3-88538-803-0, DM 44.-
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. Die vorliegende deutsche Ausgabe von Heinz Dalkowski wurde darüber hinaus in vielerlei Hinsicht inhaltlich ergänzt. 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 speziellen Vorgänge beim Einsatz von X-Funktionen (Betriebssystemfehler). Sämtliche Programme sind als BCs abgedruckt, insgesamt 4181 Bytes! Wer aus seinem Kraftpaket herausholen will, was drin steckt, benötigt dieses Buch.

Jarett, K.: Synthetisches Programmieren auf dem HP-41 - leicht gemacht
1985, 170 Seiten, eine Quick Reference Card beiliegend, ISBN 3-88538-802-2, DM 40.-
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. 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 fachmännischer Weise Heinz Dalkowski.

Meschede, W.: Plotten und Drucken auf dem HP-41 Thermodrucker
1985, 176 Seiten, ISBN 3-88538-805-7, DM 36.-
Dieses Buch enthält 18 Programme zum Plotten auf den HP-41 Thermodruckern und zusätzlich 224 Zeichen in drei Darstellungsformen und alle benötigten Zahlencodes für BLDSPEC und die synthetische Programmierung der Zeichen. Jedem Programm ist ein Programm-Ablaufplan und eine ausführliche Bedienungsanleitung beigegeben, damit sowohl reine Programm-Anwender, als auch Selbst-Programmierer voll auf ihre Kosten kommen. Durch diese Programme stellen selbst logarithmische Skalierung, die Darstellung mehrerer Funktionen in einem Schaubild, hochauflösendes Plotten, sowie Histogramme (Balkendiagramme) keine Probleme mehr dar und durch kleine Änderungen ist die Anpassung an ganz spezielle Anforderungen leicht möglich. Hat man ein Programm längere Zeit nicht mehr benutzt, ermöglichen die Kurzanleitungen im Anhang ein schnelles Rekapitulieren der sehr einfachen und komfortablen Bedienung. Zum Schluß werden dann in Kapitel 7 alle Wünsche nach ganz speziellen Zeichen für die selbst programmierte Ausgabe - einschließlich Querschrift - erfüllt.
Alle Programme sind als BCs abgedruckt, insgesamt 6755 Bytes! Wer graphische Ausgaben oder mehr als nur die einfachen 127 Zeichen benötigt, kann an diesem Buch nicht vorübergehen.

Stroinski, W.: Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I/O-ROM, IB- und IL-Interface der HP-Rechner der Serie 80
1986, 296 Seiten, ISBN 3-88538-806-5, DM 58.-
Handbücher in deutscher Sprache sind meist nur für den Computer und die wichtigsten Peripherie-Geräte (Drucker, Monitor, Massenspeicher) erhältlich, für die "seltenere" Peripherie, durch deren Anschluß der Computer erst seine volle Wirksamkeit erlangt, sind die Beschreibungen häufig nur in englischer Sprache lieferbar. Unabhängig von der Qualität der vorhandenen Sprachkenntnisse ist das Verstehen dieser neuen Materie sicherlich einfacher, wenn die Beschreibungen in deutscher Sprache vorliegen.
Für die Hewlett-Packard-Rechner der 80er Serie (HP-83/85 bzw. HP-86/87) wird mit diesem Buch über das I/O-ROM und die Interfaces für HP-IB (IEEE 488 bzw. IEC 625) und HP-IL (Interface-Loop) dieser Mangel behoben. Es enthält die Übersetzungen der nachfolgend genannten HP-Druckschriften in korrigierter Form:

I/O-ROM, Owner's Manual, 00087-90121, Jan. 83
HP-IB Interface, Owner's Manual, 82937-90017, Jan. 82
HP-IL Interface, Owner's Manual, 82938-90001, Jan. 82.

Der Vorläufer dieser Handbücher (I/O Programming Guide, 00085-90142) wurde ebenfalls berücksichtigt, wenn die dort gegebenen Erläuterungen umfangreicher waren, als in den neueren Beschreibungen. Auch für die GPIO-, BCD- und Serial Interfaces sind im Syntax-Anhang vollständige Angaben über die Auswirkungen der einzelnen Anweisungen zu finden.

Wickes, W. C.: Synthetische Programmierung auf dem HP-41C/CV, 2. erw. Aufl.
1984, 165 Seiten, ISBN 3-88538-800-6, DM 36.-
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 8000 Exemplare verkauft. Die deutsche Ausgabe von Heinz Dalkowski 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 ROM's; 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 eGoBEEP, Programme zur Herstellung vollständiger hexadezimaler Speicherauszüge, synthetischer Vorstoß ins X-Memory.
Obwohl dieses Buch manche Schwierigkeit enthält, was für den einen oder anderen Leser die vorausgehende Lektüre des Buches von K. Jarett, "Synthetisches Programmieren auf dem HP-41 - leicht gemacht" (ebenfalls im 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.

DIE HP-PALETTE DES HELDERMANN VERLAGES

Albers, K.: HP-41 Barcodes mit dem HP-IL-System. 320 Seiten, 44.00 DM, ISBN 3-88538-804-9 (1986)

Dalkowski, H., Fegert, S.: Eine Programmsammlung für den HP-41. ISBN 3-88538-809-X (1987)

Dearing, J. S.: Tricks, Tips und Routinen für Taschenrechner der Serie HP-41. Deutsche Ausgabe von H. Dalkowski. 220 Seiten, 36.00 DM, ISBN 3-88538-801-4 (1984).

Emery, K.: HP-41 Mikrocode-Programmierung für Anfänger. ISBN 3-88538-808-1 (1987)

Horn, J.: HP-71 Basic - leicht gemacht. Deutsche Ausgabe von W. Stroinski. Ca. 170 Seiten, 44.00 DM, ISBN 3-88538-807-3 (1986).

Jarett, K.: Erweiterte Funktionen des HP-41 - leicht gemacht. Deutsche Ausgabe von H. Dalkowski. 240 Seiten, 44.00 DM, ISBN 3-88538-803-0 (1986).

Jarett, K.: Synthetisches Programmieren auf dem HP-41 - leicht gemacht. Deutsche Ausgabe von H. Dalkowski. 170 Seiten, 40.00 DM, ISBN 3-88538-802-2 (1985). Dem Buch liegt eine Quick Reference Card bei.

Meschede, W.: Plotten und Drucken auf dem HP-41 Thermodrucker. 176 Seiten, 36.00 DM, ISBN 3-88538-805-7 (1985).

Stroinski, W. (Herausgeber): Zusammenfassung der Bedienungs- und Programmier-Anleitungen für I/O-ROM, IB- und IL-Interface der HP-Rechner der Serie 80. 296 Seiten, 58.00 DM, ISBN 3-88538-806-5 (1986).

Wickes, W. C.: Synthetische Programmierung auf dem HP-41C/CV. Deutsche Ausgabe von H. Dalkowski. 165 Seiten, 36.00 DM, ISBN 3-88538-800-6 (1983).

HP-41 Kombinierte Hex/Dezimale Byte Tabelle, 7x11.5 cm Plastikkarte, 6.00 DM.

HP-41 Quick Reference Card, 7x15 cm Plastikkarte, 8.00 DM. (Diese Karte liegt dem Buch "Synthetisches Programmieren auf dem HP-41 - leicht gemacht" von K. Jarett kostenlos bei.)

Alle Produkte sind zu den oben angegebenen Preisen ohne Versandkosten direkt vom Verlag erhältlich, die Plastikkarten nur auf diese Weise. Bitte richten Sie Ihre Bestellung an

Heldermann Verlag Berlin
Nassauische Str. 26
D-1000 Berlin 31

Nutzen Sie ihren HP-71 besser!

Der HP-71 setzt neue Maßstäbe für Rechner dieser "Größe". Dieses Buch **HP-71-BASIC, leicht gemacht** ergänzt auf mehreren wichtigen Teilgebieten die Benutzerhandbücher und zeigt eindrucksvoll, welche verblüffenden Bequemlichkeiten dieser Rechner bietet.

Befehls-Stack und CALC-Modus machen die Berechnungen übersichtlicher und schneller. Hier zeigt sich eine sehr bemerkenswerte Alternative zur "UPN-Logik", an die man sich beim HP-41 und anderen HP-Taschenrechnern erst gewöhnen mußte! Womöglich werden Sie dadurch zum "CALC-Modus-Fan"?

Neben allgemeinen Hinweisen zum "Tastenfeld-BASIC" erfahren Sie auch, wann und wie PEEK- und POKE-Anweisungen zu benutzen sind. Mehrere Programm-Beispiele in BASIC zeigen Ihnen die wichtigsten Programmiertechniken.

Mit einem dieser BASIC-Programme gelingt es Ihnen sogar, in Maschinensprache aufgelistete Programme in den Rechner zu bringen! Damit wird auch Ihnen deutlich, wie elegant und schnell derartige Programme arbeiten: Sie können neben der normalen Arbeit des Rechners dauernd die Uhrzeit angezeigt bekommen, Sie können aber auch eine aus 14 000 Zeichen bestehende Folge in einer Sekunde reversieren!

Ein 30 Seiten umfassendes alphabetisch geordnetes Verzeichnis erklärt Zweck und Syntax von mehreren hundert Schlüsselwörtern und Anweisungen.

Durch **HP-71-BASIC, leicht gemacht** werden Sie erst erkennen, was der HP-71 alles kann! Viel Spaß dabei!

3-88538-807-3