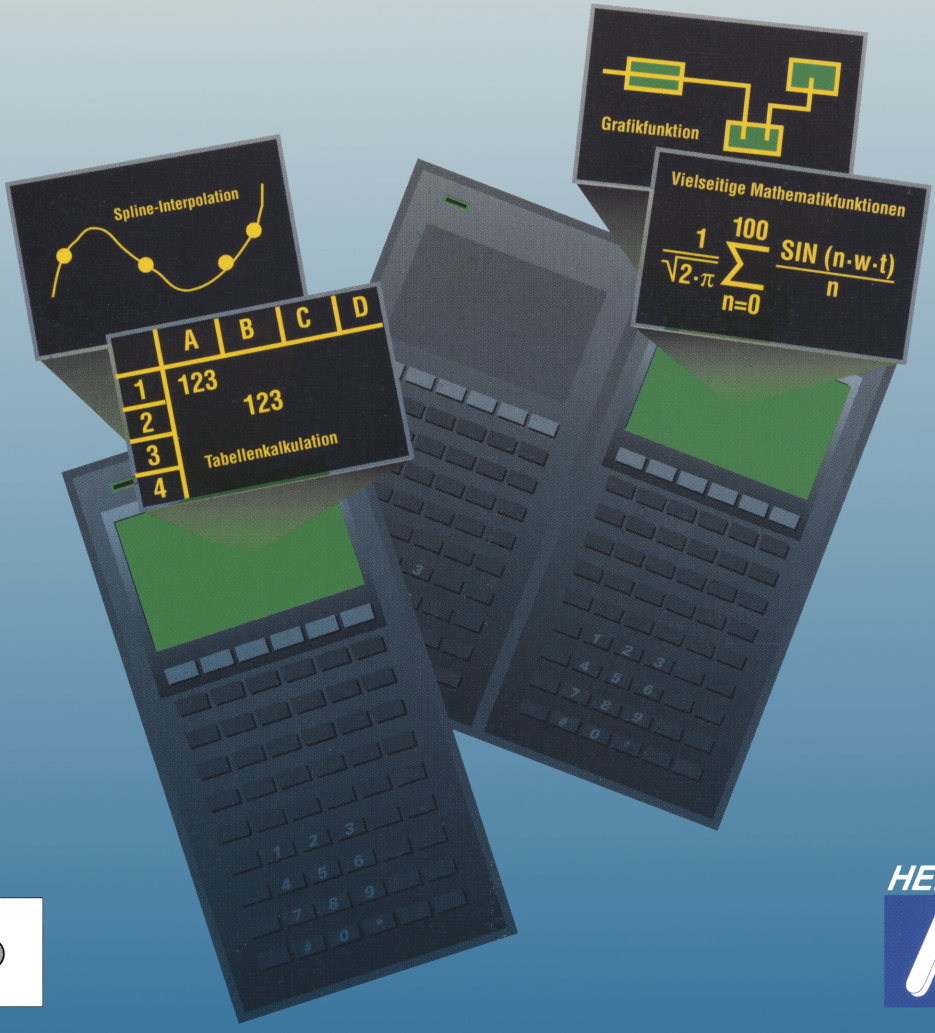
Stefan Rau / Christoph Gießelink

Programmsammlung HP 28S/48

Programme für Mathematik und Elektrotechnik



HEISE


HEISE

Stefan Rau / Christoph Gießelink

HP 28S/48

Progamme für Mathematik und Elektrotechnik

Verlag Heinz Heise

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Rau, Stefan:
HP 28S/48 : Progamme für Mathematik und Elektrotechnik /
Stefan Rau ; Christoph Giesselink. – Hannover : Heise, 1994
NE: Giesselink, Christoph:

ISBN 3-88229-031-5

Printed in Germany 5 4 3 2 1
1998 97 96 95 94

Umschlaggestaltung: MB-GRAFIK-DESIGN, Hannover
Reinzeichnungen: Olaf Schlierf, Celle
Druck: Paderborner Druck Centrum, Paderborn

ISBN 3-88229-031-5

Inhaltsverzeichnis

Vorwort

Jeder Besitzer eines programmierbaren Taschenrechners schreibt im Laufe der Zeit einige Programme, um gewisse Problemstellungen damit zu lösen. Es zeigt sich, daß viele Leute sehr viel Zeit damit verbringen, einen Algorithmus zu finden und zu implementieren, der ihr Problem löst. Deshalb ist es naheliegend, Programme von anderen zu übernehmen und diese nur noch den eigenen Bedürfnissen anzupassen. Die Arbeitsersparnis ist enorm. Das Buch verfolgt genau dieses Ziel. Zu einigen ausgewählten Themen bieten wir fertige Lösungen an, die in der Regel aus der Praxis stammen. Dieses Buch beschreibt die Anwendung der Programme, nicht aber die Algorithmen. Es ist also als eine reine Programmsammlung für den HP28 und den HP48 mit Dokumentation anzusehen. Wer Algorithmen sucht, sollte sich externer Literatur bedienen (eine Liste befindet sich im Anhang).

Trotz großer Gemeinsamkeiten bezüglich der Syntax ist in sehr vielen Fällen eine Programmversion für den HP28 und eine Version für den HP48 entstanden. Sie unterscheiden sich in der Regel im Bereich der Ein- und Ausgabe. Weiterhin gibt es HP28-Funktionen, die nicht auf auf den HP48 portiert wurden, da sie dort schon im Betriebssystem verankert sind. Viele dieser Funktionen sind aber ohne Anpassung auch auf dem HP48 lauffähig. Andere Funktionen sind wiederum nur auf dem HP48 funktionsfähig, da sie entweder HP48-spezifische Befehle oder die erweiterten Grafikfähigkeiten verwenden. Eine Anpassung an den HP28 dürfte hier sehr schwierig, wenn nicht gar unmöglich sein.

1 Einleitung

1.1 Aufbau

Die vorliegende Programmsammlung wendet sich an alle Anwender, die fertige Lösungen zu ihren Problemen suchen. Die Programmbeschreibungen dienen als Bedienungsanleitung. In einigen Fällen werden kurz Grundlagen vermittelt. Für den Einstieg in die Algorithmen ist externe Literatur erforderlich.

Als Zielplattform für die Programme ist entweder ein HP28S oder ein HP48 der Firma Hewlett Packard notwendig. Beide Rechner verwenden als Programmierbasis dieselbe Sprache. Beim HP48 sind vom Hersteller gegenüber dem HP28S einige Verbesserungen im Bereich der Sprachdefinition vorgenommen worden. Weiterhin wurden die Routinen der Ein- und Ausgabe, bedingt durch ein geändertes LCD und ein anderes Tastaturlayout, auf den HP48 angepaßt. Deshalb ist in vielen Fällen eine Programmversion für den HP28 und eine für den HP48 entstanden.

Da der HP48 über ein serielles Interface verfügt, lassen sich mit geeigneter Soft- und Hardware Programme sehr komfortabel auf den HP48 übertragen. Die hier vorgestellten HP48 Programme liegen auf der beigelegten Diskette vor. Folgende Themen werden der Reihe nach behandelt:

- Dokumentation von Softwarefehlern HP28S, HP48
- Aufbau des Grafikstrings HP28, Grafikobjekt HP48
- Übertragung von Programmen zum HP48
- Zeichenvereinbarungen
- Musterbeispiel einer Programmdokumentation
- Programmdokumentation inklusive Listings
- Literaturnachweis

1.2 Softwarefehler des HP28S

Hier werden bisher bekannt gewordene Softwarefehler des HP28S ROM-Version 2BB erläutert :

Die Befehle SAME und == arbeiten nicht immer zuverlässig. Bei einem Listenvergleich zweier identischer Listen, welche von zwei verschiedenen Programmen erzeugt wurden, wurde als Vergleichsergebnis 0 zurückgegeben. Wurde bei einer Liste in Ebene 1 der Befehl EDIT und dann wieder ENTER eingegeben, waren die Listen beim Vergleich identisch.

Der Programmeditor macht nach der Eingabe eines Programms bei folgender Konstruktion einen Fehler :

« IF THEN 1 [0] END »

Der Editor hängt nun hinter dem ']'-Zeichen ein '}'-Zeichen an. Zum Zeitpunkt des Programmablaufs hat dies keine Auswirkung, bei erneutem Editieren muß das Zeichen aber wieder entfernt werden. Der Fehler tritt immer dann auf, wenn ein ']'-Zeichen vor einem END und in der Struktur von THEN und END ein weiteres Objekt steht.

1.3 Softwarefehler des HP48

Beim HP48 gibt es eine Unzahl von Firmwareversionen. Dies mag auch der Grund sein, warum der Hersteller den SYSEVAL-Call zur Versionsabfrage in der mitgelieferten Dokumentation verschweigt. Im "HP48 Programmer's Reference Manual" wurde er veröffentlicht. Er lautet:

« # 30794h SYSEVAL ».

In diesem Zusammenhang wird darauf hingewiesen, daß ein falscher SYSEVAL-Aufruf mit großer Wahrscheinlichkeit zum Absturz des Rechners mit Speicherverlust führen wird. Das kleine 'h' hinter der Nummer bedeutet, daß dies eine Hexadezimalzahl ist. Er darf in keinem Fall beim HP28 angewendet werden!

Der SYSEVAL Befehl gibt als Antwort den String "HPHP48-x" zurück, wobei 'x' der Buchstabe der Softwareversion ist. Bei der Version 'E' konnten folgende Fehler festgestellt werden.

Der Befehl KEY arbeitet nicht zuverlässig. Der HP besitzt einen Tastaturpuffer, der mit KEY ausgelesen werden kann. Wird er jedoch zu schnell abgefragt, werden Tastenbetätigungen ignoriert.

Die häufig verwendete Konstruktion

DO UNTIL KEY END

sollte durch

DO .05 WAIT UNTIL KEY END

ersetzt werden.

Die Fehlerhäufigkeit sinkt dadurch rapide. Eine Beeinträchtigung der Bedienung ist, bei einer in diesem Punkt fehlerfreien ROM-Version 'J', nicht festzustellen.

Weiterhin gibt es einen Fehler im Bereich der Datenkonvertierung. So übersetzt der HP48 Strings im Umsetzungscode 0 in die anderen Umsetzungscodes falsch. Normalerweise sollte LF in CR LF umgewandelt werden. Steht jedoch schon ein CR vor einem LF, wird kein CR eingefügt. Dies ist aber nur dann von Bedeutung, wenn Strings als Umsetzungstabellen wie in der Funktion SKEY (Kap. 10.2) mißbraucht werden.

1.4 Der Grafikstring des HP28

Der HP28 besitzt ein LCD mit Punktmatrix. Das LCD hat eine Auflösung von 137 Pixeln in x-Richtung und 32 Pixeln in y-Richtung. Diese können mit dem PIXEL-Befehl einzeln gesetzt werden. Der HP28S hat die Möglichkeit das komplette LCD-Bild über den Befehl LCD → in einem Grafik-String abzuspeichern (oder über den Befehl → LCD wieder zurück zu laden).

Der Aufbau eines Grafikstrings wurde von der Herstellerfirma nicht dokumentiert und kann sich somit von Version zu Version ändern. Der nachfolgend beschriebene Aufbau bezieht sich auf die Rom-Version 2BB des HP28S.

1.4.1 Aufbau eines Grafikstrings

Das LCD bietet die Möglichkeit, vier Textzeilen auf einmal darzustellen. Eine Textzeile hat eine Höhe von acht Pixel. Jede Textzeile wird in Pixelblöcke aufgeteilt. Ein Pixelblock hat eine Breite von einem und eine Höhe von acht Pixel. Somit ergeben sich 137 Pixelblöcke pro Textzeile. Bei vier Textzeilen ergeben sich 548 Pixelblöcke. Dies entspricht der Länge des Grafik-Strings. Die Information eines Pixelblocks wird als Cha-

rakter abgelegt. Die Charakter werden nun nach der Zählweise in Abb. 1-1 zu einen String zusammengesetzt, wobei sich die Zahl immer auf den ersten bzw. den letzten Pixelblock bezieht.

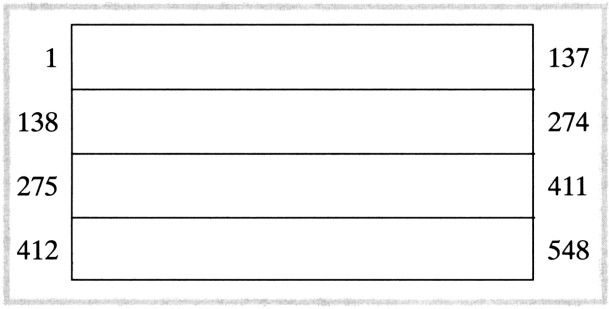


Abb. 1-1

1.4.2 Aufbau eines Pixelblocks

Ein Pixelblock hat, wie eingangs erwähnt, eine Breite von einem und eine Höhe von acht Pixel. Wandelt man nun ein Character des Grafik-Strings in eine Zahl (Befehl NUM) um, ergibt sich bei binärer Betrachtung der Zahl folgender Aufbau:

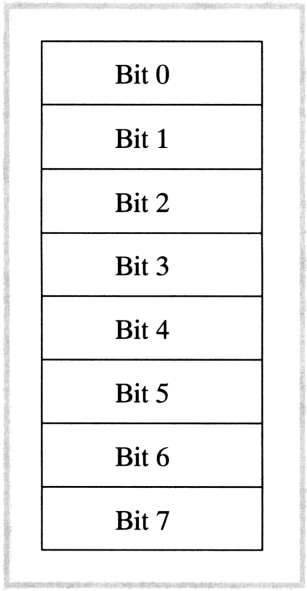


Abb. 1-2

Schreibt man die Dualzahl, beginnend mit Bit 0, untereinander, so hat man einen Pixelblock vor sich. Eine '1' bedeutet, daß das entsprechende Pixel gesetzt ist, eine '0', daß das entsprechende Pixel gelöscht ist. Durch Modifikation einzelner Bits läßt sich so ein mit dem Befehl LCD→ gespeichertes Bild verändern.

1.5 Das Grafikobjekt GROB des HP48

1.5.1 Aufbau eines Grafikobjekts

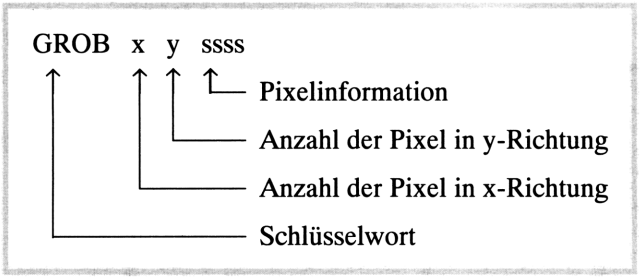


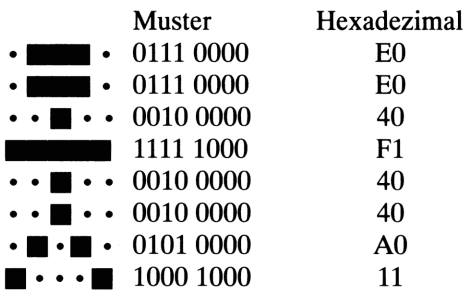
Abb. 1-3
Aufbau der Pixelinformation

Die Anzahl der Einträge errechnet sich mit folgender Formel:

anz = (2*INT((x-1)/8)+2)*y

Ein Eintrag besteht aus einer Hexzahl (0-F), welche die Information von 4 Pixel enthält. Der Aufbau erfolgt zeilenweise, wobei immer 4 Bit (4 Pixel) zu einem Eintrag zusammengefaßt werden. Das erste Pixel von links entspricht dem Bit 0. Ist die tatsächliche Anzahl der Pixel (x-Wert) pro Zeile kleiner als die reservierte Anzahl (bei 3 Pixel werden 8 Bit (8 Pixel) reserviert), müssen Sie die fehlenden Bits bzw. Pixel bis zum nächsten Eintrag auffüllen. Des Weiteren muß sich eine *gerade* Anzahl von Einträgen pro Zeile ergeben. Gegebenenfalls ist hier ein Dummy-Eintrag anzuhängen. Ein ausgeschaltetes Pixel wird mit 0, ein eingeschaltetes Pixel wird mit 1 gekennzeichnet. Anhand des Binärwertes von 4 Pixel wird die Hexzahl des Eintrages berechnet. Denken Sie bitte daran, daß das Bit 0 links steht! Zur Berechnung müssen daher die 4 Bit umgedreht werden. Der Aufbau der Einträge für die weiteren Zeilen entspricht dem der ersten Zeile.

Beispiel: Zeichnen eines Strichmännchens. Die Pixel 5-7 werden nicht benutzt und können einen beliebigen Zustand haben, sie werden hier auf 0 gesetzt.



Muster	Hexadezimal
0111 0000	E0
0111 0000	E0
0010 0000	40
1111 1000	F1
0010 0000	40
0010 0000	40
0101 0000	A0
1000 1000	11

Befehl: GROB 5 8 E0E040F14040A011

Abb. 1-4
Ein Strichmännchen

1.6 Programmübertragung zum HP48

1.6.1 Hardware

HP48 4pol		SUBD 9pol	SUBD 25pol
1	→		1
2	→	2	3
3	→	3	2
4	→	5	7

Pin 1 ist bei Aufsicht der 4 poligen Buchse auf der rechten Seite.

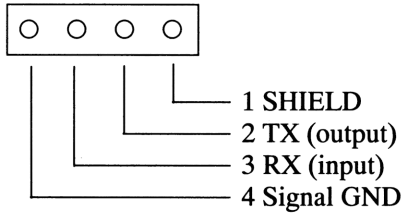


Abb. 1-5 *Pinbelegung für die Übertragung beim HP48*

Als Hardwarevoraussetzung wird ein serielles Kabel mit entsprechenden Steckern benötigt. Passende Leitungen sind im Handel erhältlich. Für Selbstbauer wird hier die Pinbelegung angegeben.

(Eine Bezugsquelle für die 4 polige HP-Buchse kann leider nicht angegeben werden.)

1.6.2 Software

Für die Übertragung von Daten vom und zum HP48 wird das KERMIT-File-Protokoll verwendet. Sie benötigen also ein Terminalprogramm, welches das KERMIT-Protokoll beherrscht. Auf der Diskette befindet sich ein Konfigurationsfile L48.INI für das Programm KERMIT der Trustees Columbia University in the City of New York.

Die HP48 Programme liegen zum einen in ASCII, zum anderen als Applikationsdateien auf der Diskette vor. Bei den ASCII Dateien wurde jedes Programm einzeln in einer Datei gespeichert. Der Name der Datei wurde so gewählt, daß er dem des HP zumindest ähnlich ist und keine Extension besitzt. Eine vollkommene Namensgleichheit ist nicht möglich, da bei DOS die Namenslänge auf acht Zeichen beschränkt ist und viele Namenssymbole des Taschenrechners verboten sind. Es ist immer der Programmname der Programmbeschreibung gültig und nicht der Dateiname. Er muß also unter Umständen auf dem HP48 manuell korrigiert werden. Beachten Sie bitte, daß der HP48 bei Dateinamen auf Groß- und Kleinschreibung achtet!

Die komfortabelste Möglichkeit Daten zu übertragen bietet das Programmpaket "Program Development Link", im weiteren kurz PDL genannt, der Herstellerfirma des Taschenrechners. Als Transferbasis dient auch hier das Programm KERMIT, dem eine grafische Benutzeroberfläche aufgesetzt wurde. Es beinhaltet einen Editor, eine Online HP48 Befehlsreferenz und verschiedene Dateitransfermöglichkeiten bis hin zum Backup. Unter PDL können mehrere Funktionen zu einer Applikationsdatei zusammengefaßt werden. Die Namensbeschränkungen fallen ebenfalls weg. So liegen alle im Buch veröffentlichten HP48 Programme auch in Form von Applikationsdateien mit der Dateiextension .APP vor.

1.7 Zeichenentsprechungen

Da die Zeichensätze der HP-Taschenrechner nicht zum erweiterten PC-Zeichensatz kompatibel sind, sind in den Programmlistings Ersatzzeichen verwendet worden. Diese müssen dann beim Eingeben durch das entsprechnde HP-Symbol ersetzt werden.

Ersatzzeichen	HP-Code	Bedeutung
f	132	Integral-Zeichen
d/dx	136	Differenzier-Symbol
->	141	Pfeil nach rechts

1.8 Aufbau einer Programmbeschreibung

Eine Programmbeschreibung besteht aus maximal acht Text- bzw. Grafikblöcken. Zu Beginn werden die einzelnen Blöcke und ihre Bedeutung erläutert, abschließend die Blöcke in einem grafischen Zusammenhang dargestellt.

Block1: Name des Programms (Rechnertypen)

Die Überschrift besteht aus dem Programmnamen unter dem das Programm auf dem Taschenrechner gespeichert werden sollte. In Klammer steht der Rechnertyp für den die Programmbeschreibung gilt, beziehungsweise auf dem das Programm implementiert ist. Falls nichts in Klammern steht, gilt die Beschreibung für beide Rechner. Die Implementierung kann jedoch unterschiedlich sein. In diesem Fall sind beide Listings abgedruckt.
Block2: Dieser Teil dient der generellen Beschreibung des Programms. Hier können Definitionen, Übergabeparameter oder Funktionsweise beschrieben werden.

Block3: Dieser Bereich ist optional und kann Beispiele enthalten.

Block4: Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
MUSTER	\MUSTER	X	

Hier werden die selbstdefinierten Unterprogramme, mit dem Verzeichnis unter dem sie abgespeichert sind, aufgelistet. Der Verzeichnisbaum entspricht der vorgeschlagenen Verzeichnisstruktur. Wird sie nicht eingehalten, sind Programmanpassungen im Quelltext notwendig. Die Felder '2' und '4' sind dann vorhanden, wenn sich die Implementation des HP28 und des HP48 unterscheiden. Die '2' steht dabei für den HP28, die '4' für den HP48. Wenn zum Beispiel das Unterprogramm für den HP28 benötigt wird, ist dies durch ein 'X' unter der '2' gekennzeichnet. Benötigt das vorgestellte Programm keine selbstdefinierten Unterprogramme, entfällt der gesamte Block.

Block5: Verwendete externe Variablen

Variable	Verzeichnis	Status	2	4
VMUSTER	\MUSTER	gelöscht		X

Hier werden die verwendeten globalen Variablen bezeichnet. Sie werden unter dem angegebenen Verzeichnis gespeichert. Vorhandene Variablen gleichen Namens werden ohne Warnung gelöscht oder überschrieben. Der Status kann entweder „gelöscht" oder „gesichert" sein. Steht er auf „gelöscht", wird die Variable vor Verlassen des Programms gelöscht, bei "gesichert" bleibt sie erhalten. Die Felder '2' und '4' haben dieselbe Bedeutung wie im Block zuvor. Werden keine globalen Variablen verwendet, entfällt der Block.

Block6: Stackdiagramm des Befehls

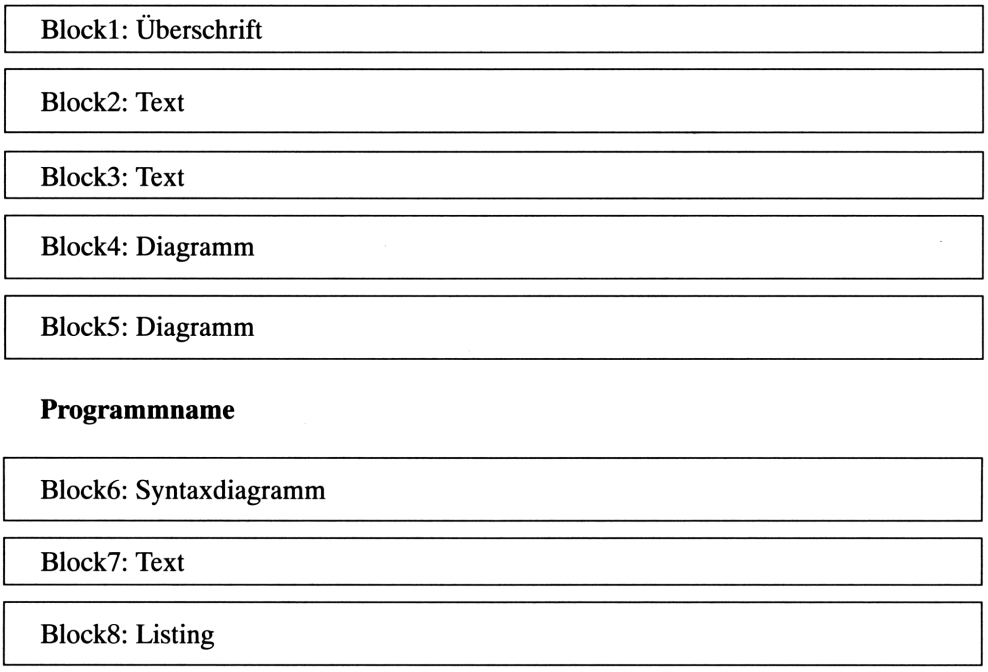
Ebene 1	Ebene 1
→	

Den Aufbau des Stackdiagrammes entnehmen Sie bitte dem HP28 Referenzhandbuch oder der Kurzreferenz des HP48.

Block7: Dieser Teil ist optional und beschreibt in Kurzform die Übergabeparameter des Stackdiagramms.

Block8: Listing für HP28/48 oder zwei Listings getrennt nach Taschenrechner.

Grafische Zusammenfassung der Blöcke



2 MATH

2.1 BRUCH

Die Funktion BRUCH zerlegt eine Realzahl in einen Bruch. Symbolische Konstanten wie *e* oder π werden in eine gerundete Realzahl umgewandelt. Eine Zerlegung von komplexen Zahlen in einen Bruch ist nicht möglich.

Die Funktion arbeitet nach dem Kettenbruchverfahren. In dem Programm führt die Funktion '1/*x*' zu rechnerbedingten Ungenauigkeiten. Deshalb sind Zahlen kleiner als 9.99E-3 unzulässig, andernfalls wird als Ergebnis 0 ausgegeben. Brüche werden immer gekürzt dargestellt.

Beim HP48 sollte die interne Funktion →Q oder →Qπ verwendet werden, da diese genauere Ergebnisse liefert.

BRUCH

Ebene 1		Ebene 2	Ebene 1
z	→	x	y

Die Realzahl z wird in einen Zähler x und in einen Nenner y zerlegt.

```
*****************************************************
* HP28/48 Funktion BRUCH,                           *
* wandelt eine Zahl in einen Bruch um               *
*                                                   *
* Eingabe  :                                        *
*  Ebene 1 : Zahl               : Real Number       *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 2 : Zähler             : Real Number       *
*  Ebene 1 : Nenner             : Real Number       *
*****************************************************

« ->NUM 1
  WHILE
    SWAP DUP IP ROT ROT FP OVER 18 < OVER ABS .0001 >
    AND
  REPEAT
    INV SWAP 1 +
  END
  DROP DUP 1 >
  « 1 2 ROT
    START
      ROT 3 PICK * + SWAP
    NEXT
  »
  IFT
»
```

2.2 SQUAD

Die Funktion SQUAD berechnet aus einer quadratischen Gleichung f die beiden Nullstellen x_1 und x_2. Als abhängige Variable wird 'X' verwendet. Sollte es bei der Berechnung zu einem Fehler kommen, bricht das Programm ab. Diese Funktion stellt im wesentlichen eine vereinfachte Form des QUAD-Befehls dar.

SQUAD

Ebene 1		Ebene 2	Ebene 1
'Symbol'	→	x	y

SQUAD löst den algebraischen Ausdruck 'Symbol' nach der Variable 'X' auf und gibt die Nullstellen von 'Symbol' auf dem Stack aus.

```
*******************************************************
* HP28 Funktion SQUAD,                                *
* löst quadratische Gleichungen                       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Funktion           : Algebraic           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 2 : Lösung 1           : Real Number         *
*  Ebene 1 : Lösung 2           : Real Number         *
*******************************************************

« 1
  -> s1
  « 'X' QUAD ->STR STR->
    IFERR DUP EVAL THEN
      DROP2 ERRM 1 DISP ABORT
    ELSE
      -1 's1' STO SWAP EVAL
    END
  »
»


*******************************************************
* HP48 Funktion SQUAD,                                *
* löst quadratische Gleichungen                       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Funktion           : Algebraic           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 2 : Lösung 1           : Real Number         *
*  Ebene 1 : Lösung 2           : Real Number         *
*******************************************************

« 1
  -> s1
  « 'X' QUAD EQ-> SWAP DROP ->STR STR-> DUP EVAL -1
    's1' STO SWAP EVAL
  »
»
```

2.3 F→L

Die Funktion F→L wandelt einen algebraischen Ausdruck in eine Listenschreibweise um. Die abhängige Variable wird in Ebene 1 definiert. Es können nur Polynome umgewandelt werden. Die Listenschreibweise wird z.B. von dem Programm PDIV (Kap. 2.5) verwendet. Der folgende algebraische Ausdruck

'2*x^4+3*x^2-4*x-5'

soll in die Listenschreibweise umgewandelt werden.

Das Programm sucht sich die Variable mit der höchsten Potenz und legt in absteigender Reihenfolge bis zu Potenz Null die Koeffizienten hintereinander in einer Liste ab.

Nach der vorangegangenen Definition ergibt sich folgende Liste für das obige Beispiel:

{ 2 0 3 -4 -5 }

Die Funktion sollte selten angewendet werden, da mit zunehmender Potenz die Laufzeit des Programms ansteigt. Falls keine automatische Konvertierung erforderlich ist, wäre es günstiger sie von Hand vorzunehmen.

Verwendete externe Variablen

Variable	Verzeichnis	Status
'Name'	\USER\MATH	gelöscht
'CHR(239)'	\USER\MATH	gelöscht

F→L

Ebene 2	**Ebene 1**		**Ebene 1**
'Symbol'	'Name'	→	{Liste}

F→L wandelt den algebraischen Ausdruck 'Symbol' mit der abhängigen Variablen 'Name' in die Listenschreibweise {Liste} um.

```
*******************************************************
* HP28/48 Funktion F->L,                              *
* wandelt Polynom in Listenschreibweise um            *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Polynom            : Algebraic           *
*  Ebene 1 : Variable           : Name                *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Listenschreibweise : List                *
*******************************************************

« 239 CHR STR-> DUP
  -> v
  « OVER STO { } ROT EVAL ROT PURGE
    DO
      0 v STO DUP EVAL 3 PICK SIZE FACT / ROT + SWAP v
      DUP PURGE d/dx
    UNTIL
      DUP 0 SAME
    END
  »
  DROP
»
```

2.4 L→F

Die Funktion L→F wandelt eine Liste in Listenschreibweise in einen algebraischen Ausdruck um und ist die Umkehrfunktion zu F→L. Es können nur Polynome umgewandelt werden. Für den Listenaufbau siehe Funktion F→L (Kap. 2.3).

L→F

Ebene 2	Ebene 1		Ebene 1
{Liste}	'Name'	→	'Symbol'

L→F wandelt die Listenschreibweise {Liste} in den algebraischen Ausdruck 'Symbol' mit der abhängigen Variablen 'Name' um.

```
*****************************************************
* HP28/48 Funktion L->F,                            *
* wandelt Listenschreibweise in ein Polynom um      *
*                                                   *
* Eingabe  :                                        *
*  Ebene 2 : Listenschreibweise  : List             *
*  Ebene 1 : Variable            : Name             *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Polynom             : Algebraic        *
*****************************************************

« OVER SIZE
  -> l x s
  « 0 1 s
    FOR i
      l i GET x s i - ^ * +
    NEXT
  »
»
```

2.5 PDIV

Die Funktion PDIV führt eine Polynomdivision zweier Polynome aus, die in Listenschreibweise vorliegen müssen. Den Aufbau einer Liste in Listenschreibweise entnehmen Sie bitte der Beschreibung der Funktion F→L (Kap. 2.3). Das Ergebnis ist ein algebraischer Ausdruck, der als abhängige Variable 'X' verwendet.

PDIV

Ebene 2	Ebene 1		Ebene 1
{Liste$_1$}	{Liste$_2$}	→	'Symbol'

PDIV dividiert das Polynom$_1$ in Listenschreibweise {Liste$_1$} durch das Polynom$_2$ in Listenschreibweise {Liste$_2$}. Das Ergebnis wird als algebraischer Ausdruck 'Symbol' mit der abhängigen Variablen 'X' in Ebene 1 des Stacks ausgegeben.

```
*******************************************************
* HP28/48 Funktion PDIV,                              *
* Polynomdivision von Zähler und Nenner in Listen-    *
* schreibweise                                        *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Zähler              : List               *
*  Ebene 1 : Nenner              : List               *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Polynom             : Algebraic          *
*******************************************************

« -> n
  « { } SWAP 0 OVER SIZE n SIZE -
    FOR i
      DUP i 1 + GET n 1 GET / SWAP 2 n SIZE
      FOR j
        j i + DUP2 GET n j GET 5 PICK * - PUT
      NEXT
      ROT ROT + SWAP
    NEXT
    OVER SIZE 1 + OVER SIZE SUB n
  »
  1 3
  START
    ROT DUP SIZE SWAP 0 1 4 PICK
    FOR i
      OVER i GET 'X' 5 PICK i - ^ * +
    NEXT
    ROT ROT DROP2
  NEXT
  / +
»
```

2.6 NEWTON

Das Programm NEWTON interpoliert aus Stützwerten ein Interpolationspolynom. Zur Berechnung des Polynoms wird das Newtonverfahren angewendet. Die Stützwerteingabe ist anwendergeführt. Beim Aufnehmen einer Kurve seien folgende vier Meßwerte ermittelt worden :

x_i	1	3	4	5
y_i	9	19	30	53

NEWTON ermittelt eine Funktion f(x), bei der die Bedingung $y_i = f(x_i)$ an jeder Stelle i erfüllt ist. Der Grad des Polynoms beträgt maximal die Anzahl der Stützstellen –1. Die Reihenfolge bei der Eingabe der Stützpunkte ist beliebig. Es müssen mindestens drei Stützpunkte eingegeben werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

NEWTON

Ebene 1		**Ebene 1**
	→	'Symbol'

NEWTON generiert aus den interaktiv eingegebenen Stützwerten die Funktion 'Symbol'.

```
*******************************************************
* HP28 Programm NEWTON,                               *
* interpoliert Stützwerte nach Newton                 *
*******************************************************

« CLLCD "  Newtoninterpolation" 1 DISP
  "Wieviele Stuetzpunkte " 58 CHR + 2 DISP PROGRAM
  WHILE
    "? " 3 SGET STR-> DUP 3 <
  REPEAT
    DROP
  END
  DUP
  -> z
  « "" 2 DISP 1 OVER
```

```
    FOR n
      n ->STR DUP ". x-Wert= " + 3 SGET STR-> SWAP
      ". y=Wert+ " + 3 SGET STR->
    NEXT
    z 2 DUP ->LIST ->ARRY MATH DUP
    -> w
    « CLLCD "Bitte warten ..." 1 DISP [ 0 1 ] * 2 ROT
      FOR i
        i 1 - GETI ROT ROT z
        FOR j
          j DUP2 GET DUP 5 ROLLD 4 ROLL - w j DUP + 1 -
          GETI ROT ROT 1 + i DUP + - GET - / PUT
        NEXT
        SWAP DROP
      NEXT
      z 1 - 1
      FOR i
        z DUP i - 1 +
        FOR j
          j 1 - DUP2 GETI ROT ROT GET w i j + z - 1 2
          ->LIST GET * - PUT
        -1 STEP
      -1 STEP
    »
    'X' z 1
  »
  FOR i
    OVER i GET 'X' i 1 - ^ * +
  -1 STEP
  SWAP DROP ->STR 3 OVER SIZE SUB "'" SWAP + STR-> CLMF
»


*******************************************************
* HP48 Programm NEWTON,                               *
* interpoliert Stützwerte nach Newton                 *
*******************************************************

« CLLCD " Newtoninterpolation" 1 DISP
  "Wieviele Stützpunkte :" 3 DISP PROGRAM
  WHILE
    "? " 4 SGET STR-> DUP 3 <
  REPEAT
    DROP
  END
  DUP
  -> z
  « 1 OVER
    FOR n
```

```
      "" 4 DISP n ". x-Wert: " + 3 SGET STR-> n
      ". y-Wert: " + 4 SGET STR->
    NEXT
    z 2 DUP ->LIST ->ARRY MATH DUP
    -> w
    « CLLCD "Bitte warten ..." 1 DISP [ 0 1 ] * 2 ROT
      FOR i
        i 1 - GETI ROT ROT z
        FOR j
          j DUP2 GET DUP 5 ROLLD 4 ROLL - w j DUP + 1 -
          GETI ROT ROT 1 + i DUP + - GET - / PUT
        NEXT
        SWAP DROP
      NEXT
      z 1 - 1
      FOR i
        z DUP i - 1 +
        FOR j
          j 1 - DUP2 GETI ROT ROT GET w i j + z - 1 2
          ->LIST GET * - PUT
        -1 STEP
      -1 STEP
    »
    'X' z 1
  »
  FOR i
    OVER i GET 'X' i 1 - ^ * +
  -1 STEP
  SWAP DROP ->STR 3 OVER SIZE SUB "'" SWAP + STR->
»
```

2.7 INTER

Die Funktion INTER berechnet mit Hilfe der VANDERMONDschen Determinante für die Stützstellen $x_0,x_1,..,x_n$ das Interpolationspolynom.

Die Stützwerte müssen in einer Matrix der Dimension 2 * n eingegeben werden, wobei links der x-Wert und rechts der f(x)-Wert stehen muß.

Mit den Zahlenwerten des Beispiels Newtoninterpolation (Kap. 2.6) ergibt sich folgende Matrix:

```
[[ 1  9 ]
 [ 3 19 ]
 [ 4 30 ]
 [ 5 53 ]]
```

Diese kann beim HP48 sehr einfach mit dem Matrix-Writer erstellt werden.

Grundlagen:

Gesucht wird ein Polynom

$$pn(x) = a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_nx_0^n$$

Für n+1 Stützstellen gibt es n+1 Gleichungen:

$$a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_nx_0^n = y_0$$
$$a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_nx_1^n = y_1$$
$$\vdots$$
$$a_0 + a_1x_n + a_2x_n^2 + \ldots + a_nx_n^n = y_n$$

Daraus folgt

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{vmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Aus dieser Gleichung lassen sich durch Umformung die a_n bestimmen.

Bei der derzeitigen Softwareversion ist jedoch zu beachten, daß bei der Lösung des linearen Gleichungssystems das Ergebnis durch Rundungsfehler beeinträchtigt werden kann. Eine Verbesserung kann durch die Verwendung des Befehls RSD erzielt werden (siehe hierzu HP-Benutzerhandbuch).

INTER

Ebene 1		Ebene 1
[Matrix]	→	'Symbol'

INTER generiert aus den Stützwerten [Matrix] die Funktion 'Symbol' durch Interpolation.

```
*******************************************************
* HP28/48 Funktion INTER,                             *
* interpoliert aus den Stützpunkten ein Polynom       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Stützpunkte         : Matrix             *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Polynom             : Algebraic          *
*******************************************************

« DUP SIZE LIST-> DROP2 DUP 1 ->LIST 0 CON
  -> p n v
  « 1 n
    FOR i
      1 p i DUP + 1 - GETI ROT ROT GET 'v' i ROT PUT 2
      n
      START
        DUP2 * SWAP
      NEXT
      DROP
    NEXT
    n DUP 2 ->LIST ->ARRY v SWAP / 0 n 1
    FOR i
      OVER i GET 'X' i 1 - ^ * +
    -1 STEP
  »
  SWAP DROP
»
```

2.8 Splineinterpolation

Die Splineinterpolation ist eine weit verbreitete Möglichkeit zur Interpolation von Stützstellen. Sie umgeht dabei einen großen Nachteil der NEWTON-Interpolation. Dieser Vorteil wird aber durch große, zu verarbeitende Datenmengen erkauft. Bei der Newtoninterpolation wird ein Polynom interpoliert, dabei kann je nach Stützstellenanzahl der Grad des Polynoms sehr groß werden. Die Folge ist ein stark oszillierendes Polynom.

Somit kann es zwischen zwei Stützpunkten zu sehr großen Wertedifferenzen zwischen dem Interpolationspolynom und einem wahrscheinlichen Funktionswert kommen.

Die Splineinterpolation umgeht dieses Problem, indem sie den Grad des Polynoms beschränkt. Es wird im wesentlichen auf die kubische Splineinterpolation zurückgegriffen, bei der der Grad des Polynoms drei beträgt. Hierbei wird der zu interpolierende Bereich in mehrere Teilintervalle zerlegt. Als Intervallgrenzen werden die Stützwerte der Funktion verwendet. Sollte eine Funktion mit vier Stützstellen interpoliert werden, so ergeben sich drei Teilintervalle. Für jedes Teilintervall wird ein kubischer Spline (Polynom dritten Grades) erzeugt. Dabei muß die Funktion an allen Stellen mindestens einmal stetig und auch stetig differenzierbar sein.

Das hier vorgestellte Programmpaket ermöglicht die Berechnung natürlicher kubischer Splines, die Berechnung allgemeiner kubischer Splines mit Vorgabe der ersten Ableitung an den Randstellen der Funktion und die Berechnung periodischer kubischer Splines. Da dieses Paket für den Studienbetrieb ausgelegt wurde, werden die Berechnungen von verschiedenen Programmodulen ausgeführt (zur Ermittlung vom Zwischenergebnissen) und müssen manuell nacheinander aufgerufen werden.

Das Programmpaket besteht aus den folgenden Programmen:

Shell	Programm SPLINE
Modul 1	Programm ASPLINE, NSPLINE oder PSPLINE Eingabe der Stützstellen und Berechnung der y''_i
Modul 2	Programm PARA Berechnung der Koeffizienten a_i - d_i
Modul 3	Programm HORNER Berechnung der Koeffizienten a'_i - d'_i
Modul 4	Programm FUNC Generierung der Polynome aus den Koeffizienten a'_i - d'_i
Modul 5	Programm SDRAW Zeichnen der Splineinterpolierenden

Berechnung der Koeffizienten in Modul 2

$$a_i = \frac{1}{6h_i}(y''_{i+1} - y_i'')$$

$$b_i = \frac{1}{2} y_i''$$

$$c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{6} h_i (y''_{i+1} + 2y_i'')$$

$$d_i = y_i$$

wobei y_i''die zweite Ableitung der Funktion und h_i der Abstand der x-Werte der Stützstellen im Intervall i ist.

Umrechnung der Koeffizienten im Modul 3

Polynom mit den Koeffizienten $a_i - d_i$:

$$s_i(x) = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i$$

Polynom mit den Koeffizienten $a'_i - d'_i$:

$$s_i(x) = a_i' x^3 + b_i' x^2 + c_i' x + d_i'$$

2.8.1 SPLINE

Die Funktion SPLINE dient der automatischen Splineinterpolation und ruft die Module 1-4 auf.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
ASPLINE	\USER\MATH\SPLINE
NSPLINE	\USER\MATH\SPLINE
PSPLINE	\USER\MATH\SPLINE
PARA	\USER\MATH\SPLINE
HORNER	\USER\MATH\SPLINE
FUNC	\USER\MATH\SPLINE
SDRAW	\USER\MATH\SPLINE

SPLINE

Ebene 1	Ebene 1
→	

SPLINE ist die Shell der Splineinterpolation.

```
**********************************************************
* HP28 Programm SPLINE,                                  *
* komplette allgemeine, natürliche oder periodische      *
* kubische Splineinterpolation                           *
*                                                        *
* Ausgabe  :                                             *
*  'PPAR'    Abbildungsparameter : List                  *
*  'ΣPAR'    Zuweisungsparameter : List                  *
**********************************************************

« CLLCD "(n)atuerliche," 1 DISP "(a)llgemeine oder" 2
  DISP "(p)eriodische kubische" 3 DISP
  "Splineinterpolation ?" 4 DISP
  WHILE
    DO
    UNTIL
      KEY
    END
    DUP "A" ≠ OVER "N" ≠ AND OVER "P" ≠ AND
  REPEAT
    DROP
  END
  IF DUP "N" SAME THEN
    DROP NSPLINE
  ELSE
    "A" SAME
    'ASPLINE'
    'PSPLINE'
    IFTE
  END
  "PARA laeuft..." 2 DISP PARA "HORNER laeuft..." 3
  DISP HORNER "FUNC laeuft..." 4 DISP FUNC CLLCD
  "  Splineinterpolation" 1 DISP 134 CHR
  " Ergebnis in FCL" + 2 DISP "Spline zeichnen (J/N) ?"
  4 DISP
  DO
  UNTIL
    KEY
  END
```

```
  IF "J" SAME THEN
    CLLCD SCLΣ DRWΣ SDRAW DGTIZ
  ELSE
    CLMF
  END
»


*******************************************************
* HP48 Programm SPLINE,                               *
* komplette allgemeine, natürliche oder periodische   *
* kubische Splineinterpolation                        *
*                                                     *
* Ausgabe  :                                          *
*  'PPAR'    Abbildungsparameter : List               *
*  'ΣPAR'    Zuweisungsparameter : List               *
*******************************************************

« CLLCD "(n)atürliche,
  (a)llgemeine oder
  (p)eriodische kubische
  Splineinterpolation ?" 2 DISP
  WHILE
    DO
      .05 WAIT
    UNTIL
      KEY
    END
    DUP 11 + OVER 32 + AND OVER 34 + AND
  REPEAT
    DROP
  END
  CASE
    DUP 32 SAME THEN
      DROP NSPLINE
    END
    DUP 11 SAME THEN
      DROP ASPLINE
    END
    34 SAME THEN
      PSPLINE
    END
  END
```

```
  "PARA läuft..." 2 DISP PARA "HORNER läuft..." 3 DISP
  HORNER "FUNC läuft..." 4 DISP FUNC 134 CHR
  " Ergebnis in FCL" + 6 DISP "Spline zeichnen (J/N)?"
  7 DISP
  DO
    .05 WAIT
  UNTIL
    KEY
  END
  24 SAME
  « ERASE SCLΣ DRAX SCATTER DRAW SDRAW GRAPH »
  IFT
»
```

2.8.2 ASPLINE

Die Funktion ASPLINE dient der Berechnung eines allgemeinen kubischen Spline mit Vorgabe der ersten Ableitung an den Rändern. Aus den eingegebenen Stützwerten und den ersten Ableitungen werden die y_i'' der Funktionen berechnet. Die Eingabe der Stützwerte und der Ableitungen ist anwendergeführt. Dabei ist jedoch zu beachten, daß die x-Werte in aufsteigender Reihenfolge eingegeben werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\USER\MATH\SPLINE	gesichert
XW	\USER\MATH\SPLINE	gesichert

ASPLINE

Ebene 1		**Ebene 1**
	→	[R-Feld]

ASPLINE generiert aus den interaktiv eingegebenen Werten den reellen Vektor [R-Feld] mit den y_i''.

```
*******************************************************
* HP28 Programm ASPLINE,                              *
* allgemeine kubische Splineinterpolation mit Vorgabe *
* der ersten Ableitung an den Rändern                 *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Y''k                : Matrix             *
*  '∑DAT'    Stützstellen        : Matrix             *
*  'XW'      Wertedifferenzen    : Matrix             *
*******************************************************

« CLLCD "  Splineinterpolation" 1 DISP
  "Wieviele Stuetzpunkte " 58 CHR + 2 DISP PROGRAM
  WHILE
    "? " 3 SGET STR-> DUP 4 <
  REPEAT
    DROP
  END
  -> z
  « "" 2 DISP 1 z
    FOR n
      n ->STR DUP ". x-Wert= " + 3 SGET STR-> SWAP
      ". y-Wert= " + 3 SGET STR->
    NEXT
    z 2 DUP ->LIST ->ARRY CLLCD "Vorgabe Ableitungen "
    58 CHR + 1 DISP "y0' = " 3 SGET STR-> "y" z 1 -
    ->STR + "' = " + 4 SGET STR->
    -> a b
    « MATH SPLINE STO∑ CLLCD "Bitte warten ..." 1 DISP
      z 1 - 2 DUP ->LIST 0 CON 'XW' STO 1 z 1 - 2 *
      FOR i
        'XW' i RCL∑ OVER GETI ROT ROT GETI ROT ROT GETI
        ROT ROT GET ROT - 5 ROLLD SWAP - PUTI ROT PUT
      2 STEP
      z 1 2 ->LIST 0 CON 2 z 1 -
      FOR i
        i 'XW' OVER 1 - 1 2 ->LIST GETI ROT ROT GETI
        ROT ROT GETI ROT ROT GET SWAP / SWAP ROT / -
        PUT
      NEXT
      6 * 1 XW OVER GETI ROT ROT GET SWAP / a - 6 * PUT
      z XW OVER 1 - 1 2 ->LIST GETI ROT ROT GET SWAP /
      b - -6 * PUT
    »
    z IDN 2 z 1 -
    FOR i
      i DUP 2 ->LIST 'XW' i 2 * 3 - DUP2 2 + GET ROT
      ROT GET + 2 * PUT
```

```
    NEXT
    2 z
    FOR i
      i DUP 1 - DUP2 SWAP 2 ->LIST ROT ROT 2 ->LIST ROT
      'XW' i 1 - 1 2 ->LIST GET SWAP ROT 3 PICK PUT ROT
      ROT PUT
    NEXT
    1 XW OVER GET 2 * PUT z DUP 1 - 2 ->LIST GETI 2 *
    PUT DUP2 / ROT 3 DUPN DROP RSD ROT / +
  »
  CLMF
»


*******************************************************
* HP48 Programm ASPLINE,                              *
* allgemeine kubische Splineinterpolation mit Vorgabe *
* der ersten Ableitung an den Rändern                 *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Y''k              : Matrix               *
*  '∑DAT'    Stützstellen      : Matrix               *
*  'XW'      Wertedifferenzen  : Matrix               *
*******************************************************

« CLLCD " Splineinterpolation" 1 DISP
  "Wieviele Stützpunkte :" 3 DISP PROGRAM
  WHILE
    "? " 4 SGET STR-> DUP 4 <
  REPEAT
    DROP
  END
  -> z
  « 1 z
    FOR n
      "" 4 DISP n ". x-Wert: " + 3 SGET STR-> n
      ". y-Wert: " + 4 SGET STR->
    NEXT
    z 2 DUP ->LIST ->ARRY "Vorgabe Ableitungen :" 3
    DISP "" 4 DISP "y0' = " 5 SGET STR-> "y" z 1 -
    ->STR + "' = " + 6 SGET STR->
    -> a b
    « MATH SPLINE STO∑ CLLCD "Bitte warten ..." 1 DISP
      z 1 - 2 DUP ->LIST 0 CON 'XW' STO 1 z 1 - 2 *
      FOR i
        'XW' i RCL∑ OVER GETI ROT ROT GETI ROT ROT GETI
        ROT ROT GET ROT - 5 ROLLD SWAP - PUTI ROT PUT
      2 STEP
```

```
    z 1 2 ->LIST 0 CON 2 z 1 -
    FOR i
      i 'XW' OVER 1 - 1 2 ->LIST GETI ROT ROT GETI
      ROT ROT GETI ROT ROT GET SWAP / SWAP ROT / -
      PUT
    NEXT
    6 * 1 XW OVER GETI ROT ROT GET SWAP / a - 6 * PUT
    z XW OVER 1 - 1 2 ->LIST GETI ROT ROT GET SWAP /
    b - -6 * PUT
  »
  z IDN 2 z 1 -
  FOR i
    i DUP 2 ->LIST 'XW' i 2 * 3 - DUP2 2 + GET ROT
    ROT GET + 2 * PUT
  NEXT
  2 z
  FOR i
    i DUP 1 - DUP2 SWAP 2 ->LIST ROT ROT 2 ->LIST ROT
    'XW' i 1 - 1 2 ->LIST GET SWAP ROT 3 PICK PUT ROT
    ROT PUT
  NEXT
  1 XW OVER GET 2 * PUT z DUP 1 - 2 ->LIST GETI 2 *
  PUT DUP2 / ROT 3 DUPN DROP RSD ROT / +
 »
»
```

2.8.3 NSPLINE

Die Funktion NSPLINE dient der Berechnung eines natürlichen kubischen Spline. Die zweiten Ableitungen an den Rändern werden auf Null gesetzt. Aus den eingegebenen Stützwerten werden auch hier die y''_i der Funktionen berechnet. Die Eingabe der Stützwerte und der Ableitungen ist anwendergeführt. Dabei ist jedoch zu beachten, daß die x-Werte in aufsteigender Reihenfolge eingegeben werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\USER\MATH\SPLINE	gesichert
XW	\USER\MATH\SPLINE	gesichert

NSPLINE

Ebene 1		Ebene 1
	→	[R-Feld]

NSPLINE generiert aus den interaktiv eingegebenen Werten den reellen Vektor [R-Feld] mit den y_i''.

```
*********************************************************
* HP28 Programm NSPLINE,                                *
* natürliche kubische Splineinterpolation               *
*                                                       *
* Ausgabe  :                                            *
*  Ebene 1 : Y''k                : Matrix               *
*  'ΣDAT'    Stützstellen        : Matrix               *
*  'XW'      Wertedifferenzen    : Matrix               *
*********************************************************

« CLLCD "  Splineinterpolation" 1 DISP
  "Wieviele Stuetzpunkte " 58 CHR + 2 DISP 0 PROGRAM
  WHILE
    "? " 3 SGET STR-> DUP 4 <
  REPEAT
    DROP
  END
  "" 2 DISP DUP 1 OVER
  FOR n
    n ->STR DUP ". x-Wert= " + 3 SGET STR-> SWAP
    ". y-Wert= " + 3 SGET STR-> ROT
  NEXT
  2 DUP ->LIST ->ARRY MATH SPLINE STOΣ CLLCD
  "Bitte warten ..." 1 DISP DUP 1 - 2 DUP ->LIST 0 CON
  'XW' STO 1 OVER 1 - 2 *
  FOR i
    'XW' i RCLΣ OVER GETI ROT ROT GETI ROT ROT GETI ROT
    ROT GET ROT - 5 ROLLD SWAP - PUTI ROT PUT
  2 STEP
  2 - DUP 1 2 ->LIST 0 CON 1 3 PICK
  FOR i
```

```
    i 'XW' OVER 1 2 ->LIST GETI ROT ROT GETI ROT ROT
    GETI ROT ROT GET SWAP / SWAP ROT / - PUT
  NEXT
  6 * SWAP DUP IDN 1 3 PICK
  FOR i
    i DUP 2 ->LIST 'XW' i 2 * 1 - DUP2 2 + GET ROT ROT
    GET + 2 * PUT
  NEXT
  2 ROT
  FOR i
    i DUP 1 - DUP2 SWAP 2 ->LIST ROT ROT 2 ->LIST ROT
    'XW' i 1 2 ->LIST GET SWAP ROT 3 PICK PUT ROT ROT
    PUT
  NEXT
  DUP2 / ROT 3 DUPN DROP RSD ROT / + ARRY-> 0 SWAP 1
  GETI + 1 SWAP PUT ->ARRY CLMF
»


*******************************************************
* HP48 Programm NSPLINE,                              *
* natürliche kubische Splineinterpolation             *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Y''k                : Matrix             *
*  '∑DAT'    Stützstellen        : Matrix             *
*  'XW'      Wertedifferenzen    : Matrix             *
*******************************************************

« CLLCD " Splineinterpolation" 1 DISP
  "Wieviele Stützpunkte :" 3 DISP 0 PROGRAM
  WHILE
    "? " 4 SGET STR-> DUP 4 <
  REPEAT
    DROP
  END
  DUP 1 OVER
  FOR n
    "" 4 DISP n ". x-Wert: " + 3 SGET STR-> n
    ". y-Wert: " + 4 SGET STR-> ROT
  NEXT
  2 DUP ->LIST ->ARRY MATH SPLINE STO∑ CLLCD
  "Bitte warten ..." 1 DISP DUP 1 - 2 DUP ->LIST 0 CON
  'XW' STO 1 OVER 1 - 2 *
  FOR i
    'XW' i RCL∑ OVER GETI ROT ROT GETI ROT ROT GETI ROT
    ROT GET ROT - 5 ROLLD SWAP - PUTI ROT PUT
  2 STEP
```

```
  2 - DUP 1 2 ->LIST 0 CON 1 3 PICK
  FOR i
    i 'XW' OVER 1 2 ->LIST GETI ROT ROT GETI ROT ROT
    GETI ROT ROT GET SWAP / SWAP ROT / - PUT
  NEXT
  6 * SWAP DUP IDN 1 3 PICK
  FOR i
    i DUP 2 ->LIST 'XW' i 2 * 1 - DUP2 2 + GET ROT ROT
    GET + 2 * PUT
  NEXT
  2 ROT
  FOR i
    i DUP 1 - DUP2 SWAP 2 ->LIST ROT ROT 2 ->LIST ROT
    'XW' i 1 2 ->LIST GET SWAP ROT 3 PICK PUT ROT ROT
    PUT
  NEXT
  DUP2 / ROT 3 DUPN DROP RSD ROT / + ARRY-> 0 SWAP 1
  GETI + 1 SWAP PUT ->ARRY
»
```

2.8.4 PSPLINE

Die Funktion PSPLINE dient der Berechnung eines periodischen kubischen Spline. Die zweiten Ableitungen an den Rändern sind hier gleich. Aus den eingegebenen Stützwerten werden die y_i'' der Funktionen berechnet. Die Eingabe der Stützwerte und der Ableitungen ist anwendergeführt. Dabei ist jedoch zu beachten, daß die x-Werte in aufsteigender Reihenfolge eingegeben werden und der y-Wert des ersten und letzten Stützpunktes identisch ist.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGE3T	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\USER\MATH\SPLINE	gesichert
XW	\USER\MATH\SPLINE	gesichert

PSPLINE

Ebene 1		Ebene 1
	→	[R-Feld]

PSPLINE generiert aus den interaktiv eingegebenen Werten den reellen Vektor [R-Feld] mit den y_i''.

```
*******************************************************
* HP28 Programm PSPLINE,                              *
* periodische kubische Splineinterpolation            *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Y''k                : Matrix             *
*  '∑DAT'    Stützstellen        : Matrix             *
*  'XW'      Wertedifferenzen    : Matrix             *
*******************************************************

« CLLCD "  Splineinterpolation" 1 DISP
  "Wieviele Stuetzpunkte " 58 CHR + 2 DISP PROGRAM
  WHILE
    "? " 3 SGET STR-> DUP 4 <
  REPEAT
    DROP
  END
  -> z
  « "" 2 DISP 1 z
    FOR n
      n ->STR DUP ". x-Wert≠ " + 3 SGET STR-> SWAP
      ". y-Wert≠ " + 3 SGET STR->
    NEXT
    z 2 DUP ->LIST ->ARRY MATH SPLINE STO∑ CLLCD
    "Bitte warten ..." 1 DISP z 1 - DUP 'z' STO 2 DUP
    ->LIST 0 CON 'XW' STO 1 z 2 *
    FOR i
      'XW' i RCL∑ OVER GETI ROT ROT GETI ROT ROT GETI
      ROT ROT GET ROT - 5 ROLLD SWAP - PUTI ROT PUT
    2 STEP
    z DUP 1 2 ->LIST 0 CON 1 ROT
    FOR i
      i z MOD 1 + 'XW' i 1 2 ->LIST GETI ROT ROT GETI
      ROT ROT GETI ROT ROT GET SWAP / SWAP ROT / - PUT
    NEXT
    6 * z DUP IDN 1 ROT
    FOR i
      i z MOD 1 + DUP 2 ->LIST XW [ 1 0 ] * i GETI ROT
      ROT GET + DUP + PUT
```

```
    NEXT
    2 z 1 +
    FOR i
      IF i z ≤ THEN
        i DUP 1 -
      ELSE
        z 1
      END
      DUP2 SWAP 2 ->LIST ROT ROT 2 ->LIST ROT 'XW' i 1
      - 1 2 ->LIST GET SWAP ROT 3 PICK PUT ROT ROT PUT
    NEXT
    DUP2 / ROT 3 DUPN DROP RSD ROT / + ARRY-> LIST->
    DROP + DUP PICK SWAP 1 2 ->LIST ->ARRY
  »
  CLMF
»


*******************************************************
* HP48 Programm PSPLINE,                              *
* periodische kubische Splineinterpolation            *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Y''k              : Matrix               *
*  '∑DAT'    Stützstellen      : Matrix               *
*  'XW'      Wertedifferenzen  : Matrix               *
*******************************************************

« CLLCD " Splineinterpolation" 1 DISP
  "Wieviele Stützpunkte :" 3 DISP PROGRAM
  WHILE
    "? " 4 SGET STR-> DUP 4 <
  REPEAT
    DROP
  END
  -> z
  « 1 z
    FOR n
      "" 4 DISP n ". x-Wert: " + 3 SGET STR-> n
      ". y-Wert: " + 4 SGET STR->
    NEXT
    z 2 DUP ->LIST ->ARRY MATH SPLINE STO∑ CLLCD
    "Bitte warten ..." 1 DISP z 1 - DUP 'z' STO 2 DUP
    ->LIST 0 CON 'XW' STO 1 z 2 *
    FOR i
      'XW' i RCL∑ OVER GETI ROT ROT GETI ROT ROT GETI
      ROT ROT GET ROT - 5 ROLLD SWAP - PUTI ROT PUT
    2 STEP
```

```
    z DUP 1 2 ->LIST 0 CON 1 ROT
    FOR i
      i z MOD 1 + 'XW' i 1 2 ->LIST GETI ROT ROT GETI
      ROT ROT GETI ROT ROT GET SWAP / SWAP ROT / - PUT
    NEXT
    6 * z DUP IDN 1 ROT
    FOR i
      i z MOD 1 + DUP 2 ->LIST XW [ 1 0 ] * i GETI ROT
      ROT GET + DUP + PUT
    NEXT
    2 z 1 +
    FOR i
      IF i z ≤ THEN
        i DUP 1 -
      ELSE
        z 1
      END
      DUP2 SWAP 2 ->LIST ROT ROT 2 ->LIST ROT 'XW' i 1
      - 1 2 ->LIST GET SWAP ROT 3 PICK PUT ROT ROT PUT
    NEXT
    DUP2 / ROT 3 DUPN DROP RSD ROT / + ARRY-> LIST->
    DROP + DUP PICK SWAP 1 2 ->LIST ->ARRY
  »
»
```

2.8.5 PARA

Die Funktion PARA berechnet zu den Ableitungen y''_i die entsprechenden Koeffizienten a_i - d_i.

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\USER\MATH\SPLINE	gesichert
XW	\USER\MATH\SPLINE	gesichert

PARA

Ebene 1		Ebene 1
[R-Feld]	→	{Liste}

PARA generiert aus dem Vektor [R-Feld] und den Variablen ΣDAT und XW die Koeffizientenliste {Liste}.

```
*******************************************************
* HP28/48 Funktion PARA,                              *
* errechnet aus Y''k die Koeffizienten ai - di        *
*                                                     *
* Die Koeffizienten pro Funktion werden in einer      *
* Liste abgelegt, die Koeffizientenlisten werden dann *
* in einer Liste zusammengefaßt.                      *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Y''k               : Matrix              *
*  'ΣDAT'    Stützstellen       : Matrix              *
*  'XW'      Wertedifferenzen   : Matrix              *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Koeffizientenliste : List                *
*******************************************************

« DUP
  -> a
  « SIZE LIST-> DROP - DUP 1 2 ->LIST 0 CON ARRY->
    LIST-> DROP2 ->LIST 1 ROT
    FOR i
      i { } 'XW' i 1 2 ->LIST GET a i GETI ROT ROT GET
      -> h z m
      « m z - h 6 * / + z 2 / + 'XW' i 2 DUP ->LIST GET
        h / m z 2 * + h * 6 / - + RCLΣ i 2 DUP ->LIST
        GET + PUT
      »
    NEXT
  »
»
```

2.8.6 HORNER

Die Funktion HORNER wandelt die Koeffizientenliste a_i - d_i in der Form

$s_i(x) = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i$

in die Koeffizientenliste a_i' - d_i' der Form

$s_i(x) = a_i' x^3 + b_i' x^2 + c_i' x + d_i'$

über ein Hornerschema um.

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\USER\MATH\SPLINE	gesichert

HORNER

Ebene 1		Ebene 1
{Liste$_1$}	→	{Liste$_2$}

HORNER wandelt die Koeffizienten a_i - d_i aus {Liste$_1$} in die Koeffizienten a'_i - d'_i in {Liste$_2$} um.

```
*******************************************************
* HP28/48 Funktion HORNER,                            *
* umrechnen von ai-di in die Koeffizienten a'i - d'i  *
*                                                     *
* Vorgabe der Koeffizienten in der Form :             *
*   s(x) = ai*(x-xi)^3 + bi*(x-xi)^2 + ci*(x-xi) + di *
*                                                     *
* Ausgabe der Koeffizienten in der Form :             *
*   s(x) = a'i*x^3 + b'i*x^2 + c'i*x + di             *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Koeffizientenliste  : List               *
*  'ΣDAT'    Stützstellen        : Matrix             *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Koeffizientenliste  : List               *
*******************************************************

« 1 OVER SIZE
  FOR i
    i DUP2 GET RCLΣ i 1 2 ->LIST GET NEG
    -> x
    « 1 3
      FOR j
        1 4 j -
        FOR k
          k GETI x * 3 DUPN DROP GET + PUT
        NEXT
      NEXT
    »
    PUT
  NEXT
»
```

2.8.7 FUNC

Die Funktion FUNC generiert aus den Koeffizienten a_i'- d_i' Polynome dritten Grades. Die Funktionen der Teilintervalle werden in einer Liste zusammengefaßt, wobei zum ersten Teilintervall auch der erste Listeneintrag gehört. Das Ergebnis wird in der externen Variable FCL gespeichert.

Verwendete externe Variablen

Variable	Verzeichnis	Status
FCL	\USER\MATH\SPLINE	gesichert

FUNC

Ebene 1		**Ebene 1**
{Liste}	→	

FUNC wandelt die Koeffizientenliste a_i'- d_i' {Liste} in eine Funktionsliste um, die in FCL gespeichert wird.

```
*******************************************************
* HP28/48 Funktion FUNC,                              *
* generiert Polynome 3-ten Grades aus a'i - d'i       *
*                                                     *
* Die Funktionen werden in einer Liste zusammengefaßt *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Koeffizientenliste  : List               *
*                                                     *
* Ausgabe  :                                          *
*  'FCL'     Funktionenliste     : List               *
*******************************************************

« 1 OVER SIZE
  FOR i
    i DUP2 GET 'X' 1 4
    FOR j
      OVER j GET 'X' 4 j - ^ * +
    NEXT
    SWAP DROP ->STR 3 OVER SIZE SUB "'" SWAP + STR->
    PUT
  NEXT
  'FCL' STO
»
```

2.8.8 SDRAW

Die Funktion SDRAW zeichnet die Spline-Interpolierende. Die Abbildungsparameter stehen in der Variablen PPAR. Sie können am einfachsten mit der Funktion SCL∑ bestimmt werden. Weiterhin werden zur Darstellung die Variablen ∑DAT und FCL, die durch die Splineinterpolation erzeugt werden, benötigt. Im Gegensatz zur Funktion DRAW wird bei direktem Aufruf das LCD nicht gelöscht. Dies muß manuell mit dem Befehl CLLCD geschehen.

Verwendete externe Variablen

Variable	Verzeichnis	Status
PPAR	\USER\MATH\SPLINE	gesichert
∑DAT	\USER\MATH\SPLINE	gesichert
FCL	\USER\MATH\SPLINE	gesichert
X	\USER\MATH\SPLINE	gelöscht

SDRAW

Ebene 1	**Ebene 1**
→	

SDRAW zeichnet die Spline-Interpolierende.

```
*******************************************************
* HP28 Prozedur SDRAW,                                 *
* zeichnet SPLINE - Interpolierende                    *
*                                                      *
* Eingabe  :                                           *
*  '∑DAT'    Stützstellen        : Matrix              *
*  'FCL'     Funktionenliste     : List                *
*                                                      *
* Ausgabe  :                                           *
*           Funktion im LCD                            *
*******************************************************

« DRAX PPAR LIST-> 4 DROPN SWAP - C->R DROP 137 / 4 FIX
  RND STD 1 FCL SIZE
  FOR i
    FCL i GET RCL∑ i DUP + 1 - GETI ROT ROT 1 + GET
    FOR j
      j DUP 'X' STO OVER EVAL R->C PIXEL
    OVER STEP
    DROP
```

```
  NEXT
  DROP 'X' PURGE
»

*******************************************************
* HP48 Prozedur SDRAW,                                *
* zeichnet SPLINE - Interpolierende                   *
*                                                     *
* Eingabe  :                                          *
*  'ΣDAT'    Stützstellen        : Matrix             *
*  'FCL'     Funktionenliste     : List               *
*                                                     *
* Ausgabe  :                                          *
*           Funktion im LCD                           *
*******************************************************

« DRAX FUNCTION 1 FCL SIZE
  FOR i
    FCL i GET STEQ RCLΣ i DUP + 1 - GETI ROT ROT 1 +
    GET INDEP DRAW
  NEXT
  'EQ' PURGE
»
```

2.9 APPROXIMATION

Das Programm APPROXIMATION approximiert aus einer Funktion f_a im Intervall $[x_1,x_2]$ ein Polynom f_p. Der Grad des Polynoms ist beliebig. Je höher der Grad des Polynoms, desto länger jedoch die Rechenzeit. Zur Berechnung des Polynoms wird eine Approximation im quadratischen Mittel angewendet.

Von einer Approximation spricht man, wenn eine beliebige Funktion f_a durch eine einfachere Funktion im Intervall $[x_1,x_2]$ angenähert werden soll. Als einfache Funktionen werden in der Regel Polynome verwendet. Bei der Approximation im quadratischen Mittel (Gaußsche Approximation) ist die Summe der Fehlerquadrate aus $f_a(i) - f_p(i)$ minimal.

Die Funktion f_a = ATAN(x) soll durch ein Polynom dritten Grades im Intervall [-1,1] approximiert werden :

1. Eingabe der Funktion 'ATAN(X)' in Ebene 2
2. Eingabe der abhängigen Variablen 'X' in Ebene 1
3. Aufruf der Funktion APPROXIMATION

Bedienung HP28

Das Programm fordert Sie nun nacheinander zur Eingabe des Grades des Polynoms f_p, der unteren Intervallgrenze x_1, der oberen Intervallgrenze x_2 und der Genauigkeit der internen Berechnung auf. Die letzte Angabe dient der Berechnung von Integralen im Programm. Je höher die geforderte Genauigkeit ist, desto länger wird die Rechenzeit. Für dieses Beispiel hat sich eine Genauigkeitsangabe im Bereich von 1E-8 bis 1E-12 bewährt.

Bedienung HP48

Sie können nun den Grad des Polynoms f_p, die obere und untere Intervallgrenze eingeben. Die Genauigkeit der Integralen und der somit berechneten Funktion hängt von der Anzahl der angezeigten Stellen im LCD ab. Die Genauigkeit muß vor Aufruf der Funktion APPROXIMATION eingestellt werden.

Verwendete selbstdefinierte Unterprogramme

Name	*Verzeichnis*	*2*	*4*
SGET	*\USER\PROGRAM*	*X*	

APPROXIMATION

Ebene 2	***Ebene 1***		***Ebene 1***
'Symbol$_1$'	*'Name'*	→	*'Symbol$_2$'*

APPROXIMATION generiert aus der eingegebenen Funktion 'Symbol$_1$' mit der abhängigen Variablen 'Name' das Polynom 'Symbol$_2$'.

```
*******************************************************
* HP28 Funktion APPROXIMATION,                        *
* approximiert Funktion im quadratischen Mittel       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Funktion           : Algebraic           *
*  Ebene 1 : Variable           : Algebraic           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Funktion           : Algebraic           *
*******************************************************

« -> f a
  « CLLCD PROGRAM "Grad des Polynoms + " 1 SGET STR->
    "untere Grenze + " 2 SGET STR-> "obere  Grenze + "
    3 SGET STR-> "Genauigkeit + " 4 SGET STR-> MATH
    CLLCD "Bitte warten ..." 1 DISP
    -> n u o g
    « 0 n
      FOR i
        a i ^ f * a u o 3 ->LIST g ƒ DROP
      NEXT
      n 1 + ->ARRY 1 n 1 +
      FOR i
        o i ^ u i ^ - i /
      NEXT
      n 2 + n 2 * 1 +
      FOR i
        n DUPN o i ^ u i ^ - i /
      NEXT
      n 1 + DUP 2 ->LIST ->ARRY / ARRY-> 1 GET 1 - a
      SWAP 0
      FOR i
        SWAP a i ^ * +
      -1 STEP
    »
  »
  ->STR 3 OVER SIZE SUB "'" SWAP + STR-> CLMF
»
```

```
*******************************************************
* HP48 Funktion APPROXIMATION,                        *
* approximiert Funktion im quadratischen Mittel       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Funktion            : Algebraic          *
*  Ebene 1 : Variable            : Algebraic          *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Funktion            : Algebraic          *
*******************************************************

« -> f a
  « "Approximation" { ":Grad des Polynoms:
    :untere Grenze:
    :obere  Grenze: " { 1 0 } V } INPUT CLLCD
    "Bitte warten ..." 1 DISP OBJ-> DTAG SWAP DTAG ROT
    DTAG
    -> o u n
    « 0 n
      FOR i
        u o a i ^ f * a ƒ ->NUM
      NEXT
      n 1 + ->ARRY 'IERR' PURGE 1 n 1 +
      FOR i
        o i ^ u i ^ - i /
      NEXT
      n 2 + n 2 * 1 +
      FOR i
        n DUPN o i ^ u i ^ - i /
      NEXT
      n 1 + DUP 2 ->LIST ->ARRY / ARRY-> 1 GET 1 - a
      SWAP 0
      FOR i
        SWAP a i ^ * +
      -1 STEP
    »
  »
  ->STR 3 OVER SIZE SUB "'" SWAP + STR->
»
```

2.10 TPOL

Die Funktion TPOL generiert Tschebyscheff-Polynome der ersten Art. Tschebyscheff-Polynome spielen eine wichtige Rolle bei der Approximation von Funktionen. Die Funktion dient als Hilfe für die Entwicklung einer Tschebyscheff-Approximation. Definition der Tschebyscheff-Polynome erster Art

$T_n(x) = \cos(n\varphi)$

mit

$x = \cos(\varphi)$

Aus der Identität von

$\cos((n+1)\varphi) + \cos((n-1)\varphi) = 2\cos(\varphi)\cos(n\varphi)$

folgt die Rekursionsformel für T_n

$T_{n+1}(x) = 2 x T_n(x) - T_{n-1}(x)$

TPOL

Ebene 1		Ebene 1
n	→	'Symbol'

TPOL berechnet das T-Polynom $T_n(x)$ der ersten Art, wobei n das zu ermittelnde T-Polynom festlegt. Das Ergebnis 'Symbol' wird in Ebene 1 auf dem Stack abgelegt.

```
*******************************************************
* HP28/48 Funktion TPOL,                              *
* generiert Tschebyscheff Polynom                     *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Grad des Polynoms   : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Polynom             : Algebraic          *
*******************************************************

« DUP
  -> n
  « DUP NOT + DUP 1 + 1 ->LIST 0 CON DUP ROT 1 PUTI ROT
    SWAP 1 PUT SWAP 2 n ≤
    « 2 n
```

```
      START
        DUP ARRY-> 0 SWAP ->ARRY SWAP DROP 2 * ROT -
      NEXT
    »
    IFT
    SWAP DROP 0 0 n
    FOR i
      OVER i 1 + GET 'X' n i - ^ * +
    2 STEP
  »
  SWAP DROP
»
```

2.11 PASCAL

Die Funktion PASCAL erzeugt für eine Potenz die entsprechende Reihe des Pascalschen Dreiecks. Die Funktion arbeitet rekursiv, deshalb sollte der Programmname beibehalten werden. Sonst ist eine Anpassung des Quelltextes notwendig, da bei einem rekursiven Algorithmus die Funktion bis zu einer Abbruchbedingung immer wieder selbst aufgerufen wird.

PASCAL

Ebene 1		Ebene 1
n	→	{Liste}

PASCAL erzeugt für die Potenz n die Reihe des Pascalschen Dreiecks und legt die Reihe in der Liste {Liste} ab.

```
*******************************************************
* HP28/48 Funktion PASCAL,                            *
* berechnet Reihe des Pascalschen Dreiecks            *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Potenz der Reihe    : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Koeffizentenreihe   : List               *
*******************************************************

« IF DUP 2 < THEN
    { 1 } +
  ELSE
    1 - PASCAL 1 + DUP SIZE 2 - 1
    FOR n
      n GETI OVER 4 PICK SWAP GET + PUT
    -1 STEP
  END
»
```

2.12 PRIM

Die Funktion PRIM zerlegt eine ganze reelle Zahl in ihre Primfaktoren. Gleiche Primfaktoren werden auch mehrfach in der Ergebnisliste ausgegeben. Primfaktoren sind nur durch eins und durch sich selbst ohne Rest teilbar.

PRIM

Ebene 1		Ebene 1
n	→	{Liste}

PRIM zerlegt n in seine Primfaktoren und legt sie in der Liste {Liste} ab.

```
*******************************************************
* HP28/48 Funktion PRIM,                              *
* zerlegt eine Zahl in ihre Primfaktoren              *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Zahl                : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Primfaktoren        : List               *
*******************************************************

« 2
  -> z
  « { } SWAP
    DO
      WHILE
        DUP z MOD NOT OVER z > AND
      REPEAT
        z / SWAP z + SWAP
      END
      z DUP 2 ≠ + 1 + 'z' STO
    UNTIL
      DUP √ z <
    END
    +
  »
»
```

2.13 Fakultät

Mit diesen beiden Programmen kann man die Fakultät (das endliche Produkt natürlicher Zahlen) einer Zahl berechnen.

2.13.1 Fakultät iterativ

Es gibt Zahlen, deren Fakultät selbst der HP nicht mehr darstellen kann. Dieses Programm bietet die Möglichkeit, Fakultäten sehr großer Zahlen zu berechnen.

Beispiel: 100,FAK liefert: "9.33262152901E157"
1000,FAK liefert: "4.02387267366E2567"

FAK

Ebene 1		Ebene 1
n	→	"String"

FAK berechnet die Fakultät von n und legt das Ergebnis in einen String.

```
*******************************************************
* HP28/48 Funktion FAK,                               *
* berechnet die Fakultät einer Zahl größer 253        *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Zahl               : Real Number         *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Ergebnis           : String              *
*******************************************************

« 0 1 ROT
  FOR y
    y LOG +
  NEXT
  DUP FP ALOG ->STR "E" + SWAP IP ->STR +
»
```

2.13.2 Fakultät rekursiv

Die Fakultät rekursiv ist wie folgt definiert

n! = n * (n-1)!
0! = 1
1! = 1

Die Berechnung geht nicht über den Zahlenbereich des HPs hinaus.

Beispiel: 20,FAKU liefert: 2.43290200818E18

FAKU

Ebene 1		Ebene 1
n	→	x

```
*****************************************************
* HP28/48 Funktion FAKU,                            *
* Fakultät mal anders rekursiv berechnet            *
*                                                   *
* Eingabe  :                                        *
*  Ebene 1 : Zahl              : Real Number        *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Ergebnis          : Real Number        *
*****************************************************

« -> n
  'IFTE(n<2,1,FAKU(n-1)*n)'
»
```

2.14 GGT

Die Funktion GGT bestimmt den größten gemeinsamen Teiler zweier Zahlen.

GGT

Ebene 2	Ebene 1		Ebene 1
x	y	→	z

GGT bestimmt den größten gemeinsamen Teiler z der Zahlen x und y.

```
*******************************************************
* HP28/48 Funktion GGT,                               *
* bestimmt den größten gemeinsamen Teiler             *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : x                  : Real Number         *
*  Ebene 1 : y                  : Real Number         *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : z                  : Real Number         *
*******************************************************

« DO
    SWAP OVER MOD
  UNTIL
    DUP NOT
  END
  DROP
»
```

2.15 B→D

Die Funktion B→D wandelt eine Zahl zur Basis b in eine Dezimalzahl um. In Anlehnung an das Hexadezimalsystem werden für Ziffern größer als neun die Großbuchstaben A, B, C, usw. verwendet. Mit dieser Funktion können vorzeichenlose reelle Zahlen umgewandelt werden.

B→D

Ebene 2	Ebene 1		Ebene 1
"String"	n	→	x

Die Zahl "String" zur Basis n wird in die dezimale Zahl x umgewandelt.

```
*******************************************************
* HP28/48 Funktion B->D,                              *
* wandelt eine positive reelle Zahl zur Basis n, in   *
* eine Dezimalzahl um, für Ziffern größer als 10 müs- *
* sen die Zeichen A,B,C,... verwendet werden          *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Zahl zur Basis n    : String             *
*  Ebene 1 : Basis n             : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Dezimalzahl         : Real Number        *
*******************************************************

« -> b
  « ->STR b OVER "." POS 2 -
    IF DUP -2 SAME THEN
      DROP OVER SIZE 1 -
    END
    ^ 0
    WHILE
      ROT DUP SIZE
    REPEAT
      DUP 2 OVER SIZE SUB 4 ROLLD NUM
      IF DUP 46 ≠ THEN
        48 - DUP 17 ≥ 7 * - ROT DUP b / 4 ROLLD * +
      ELSE
        DROP
      END
    END
    ROT DROP2
  »
»
```

2.16 C→PT

Die Funktion C→PT wandelt einen Temperaturwert im Bereich von -200°C bis 800°C in einen Pt100 Widerstandswert um. Ein Pt100 ist ein temperaturabhängiger Widerstand der in Temperaturmeßgeräten eingesetzt wird. Die Einheit der Temperatur ist °C, die des Widerstandes ist Ω (Ohm).

C→PT

Ebene 1		Ebene 1
x	→	y

Die Temperatur x wird in einen Pt100 Widerstandswert y umgerechnet.

```
*******************************************************
* HP28/48 Funktion C->PT,                             *
* wandelt Temperaturwert (Bereich -200°C - 850°C) in  *
* PT100 Widerstandswert um                            *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Temperatur          : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Widerstandswert     : Real Number        *
*******************************************************

« -> t
  « 1 .00390802 t * + .000000580195 t SQ * -
    t 0 <
    « 4.2735E-12 t 100 - * t 3 ^ * - »
    IFT
  »
  100 *
»
```

3 Elektrotechnik-Programme

3.1 P→S

P→S wandelt die Parallelschaltung eines reellen und eines komplexen Widerstands in eine äquivalente Reihenschaltung um.

Die Parallelschaltung von R=25 Ω mit X=40 Ω kap. soll in eine äquivalente Reihenschaltung umgewandelt werden. Man gibt ein, (25,-40), P→S. Als Ergebnis bekommt man (16,-8) also R'=16 Ω und X'=8 Ω kap.

P→S

Ebene 1		Ebene 1
z_1	→	z_2

Die Parallelschaltung z_1 wird in eine äquivalente Serienschaltung z_2 umgewandelt.

```
*******************************************************
* HP28/48 Funktion P->S,                              *
* Parallelschaltung von R und X nach Serienschaltung  *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Zpar             : Complex Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Zser             : Complex Number        *
*******************************************************

« DUP IM INV NEG SWAP RE INV SWAP R->C INV »
```

3.2 S→P

S→P ist die Umkehrfunktion zu P→S.

S→P

Ebene 1		Ebene 1
z_1	→	z_2

Die Parallelschaltung z_1 wird in eine äquivalente Serienschaltung z_2 umgewandelt.

```
*******************************************************
* HP28/48 Funktion S->P,                              *
* Serienschaltung von R und X nach Parallelschaltung  *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Zser             : Complex Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Zpar             : Complex Number        *
*******************************************************

« INV DUP IM INV NEG SWAP RE INV SWAP R->C »
```

3.3 Übertragungsfunktionen

3.3.1 RLCN

RLCN ist ein Programm zur numerischen Auswertung komplexer Ausdrücke. Die Auswertung geschieht entweder graphisch auf dem LCD oder von Hand durch Eingabe von Einzelwerten. Man kann sowohl die Phase als auch den Pegel eines Ausdruckes berechnen lassen.

Der komplexe Ausdruck ist in der Regel eine Übertragungsfunktion eines passiven oder aktiven Netzwerkes und steht in der Variablen XFN. Um eine optimale Darstellung zu erhalten, sollte man die Funktion XFN um die Resonanzfrequenz herum normieren,

so daß die Resonanzfrequenz immer in der Mitte der Anzeige ist:

f'=f/f(res)

und daraus

XFN=XFN(f').

Das Programm selbst ist dialoggeführt. Nach dem Start erscheint auf dem LCD:

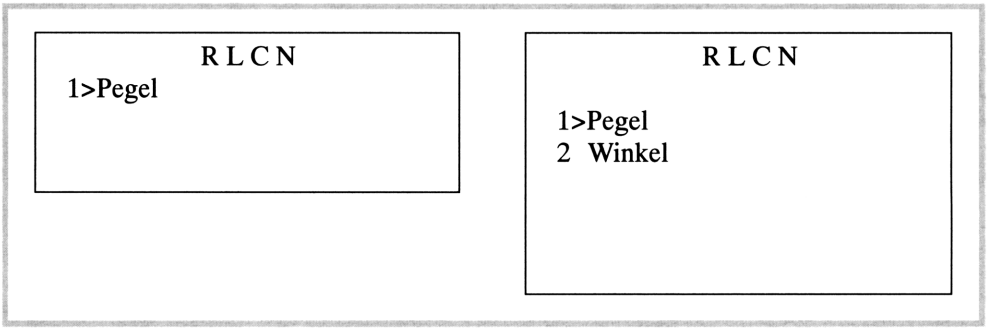


Abb. 3-1 *Abfrage nach der dargestellten Berechnung*

Es wird eine Auswahl zwischen Pegel und Winkel dargestellt.

Als nächstes wird die Darstellungsart abgefragt. Man hat hier die Möglichkeit zwischen Diagrammdarstellung und Einzelwertberechnung zu wählen:

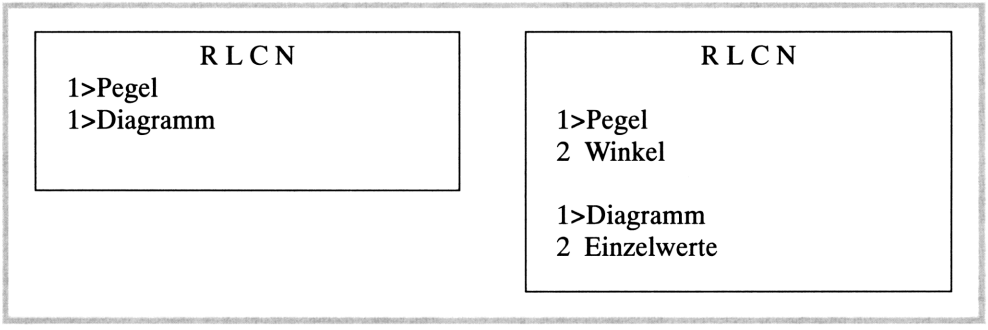


Abb. 3-2 *Abfrage des Ausgabetyps*

Wählt man den Diagrammodus, so erscheint ein Koordinatenkreuz. Bei der Phasenberechnung gilt

PMIN=(10^-4,-90°)

und

PMAX=(10^4,90°).

Will man den Pegel darstellen, so gilt hier

PMIN=(10^-4,-80 dB)

und

PMAX=(10^4,80 dB).

HP28

Um den Diagrammodus zu beenden, drückt man einfach eine beliebige Taste.

Weitere Funktionen beim HP48

Der HP48 befindet sich in der Grafikumgebung. Folglich sind alle in der Menüleiste befindlichen Operationen möglich. Mit der <ATTN>-Taste beendet man den Diagrammodus. Danach wird man gefragt, ob man das Programm beenden möchte:

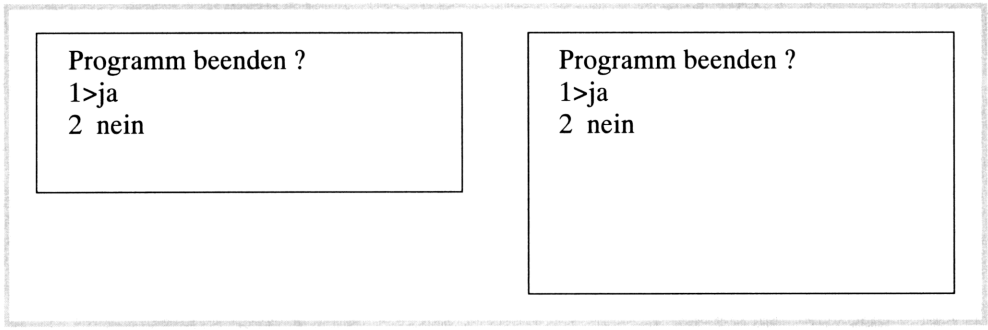


Abb. 3-3 *Programmende*

Mit einem 'nein' gelangt man an den Anfang des Programms.

Hat man die Einzelwertdarstellung gewählt, kann man die Werte im logarithmischen Maßstab direkt eingeben. Es erscheint 'Wert:' in der Anzeige. Geben Sie dahinter den zu berechnenden Wert ein.

In die nächste Zeile wird das Ergebnis geschrieben.

In der unteren Bildhälfte wird 'Ende' angezeigt, gefolgt von einem Menü, in dem man die Auswahl zwischen 'nein' und 'ja' hat. 'nein' wiederholt die Einzelwerteingabe, mit 'ja' gelangt man zur Abfrage 'Programm beenden ?'.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
XFN	\USER\ETEC
SGET	\USER\PROGRAM
ILIST	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
n	\USER\ETEC	gelöscht

RLCN

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm RLCN,                                 *
* wertet komplexe Übertragungsfunktion aus            *
*******************************************************

« 60 FS?C
  DO
    CLLCD "       R L C N" 1 DISP { "Pegel" "Winkel" }
    2 1 PROGRAM ILIST ETEC
    IF 2 SAME THEN
      'ARG(XFN(ALOG(n)))'
      (-4,-90) (4,90)
    ELSE
      '20*LOG(ABS(XFN(ALOG(n))))'
      (-4,-80) (4,80)
    END
    PMAX PMIN 'n' INDEP STEQ { "Diagramm" "Einzelwerte"
    } 3 1 PROGRAM ILIST ETEC
    -> d
    « DO
        CLLCD
        IF d 1 SAME THEN
          DRAW
          DO
          UNTIL
            KEY
          END
          DROP 2
```

```
      ELSE
        CLLCD PROGRAM "Wert=" 1 SGET ETEC STR-> 'n'
        STO RCEQ ->NUM ->STR 2 DISP "Ende" 3 DISP {
        "nein" "ja" } 4 1 PROGRAM ILIST ETEC
      END
    UNTIL
      2 SAME
    END
  »
  CLLCD "Programm beenden ?" 1 DISP { "ja" "nein" } 2
  2 PROGRAM ILIST ETEC
UNTIL
  1 SAME
END
'n' PURGE CLMF
IF THEN
  60 SF
END
»


*******************************************************
* HP48 Programm RLCN,                                 *
* wertet komplexe Übertragungsfunktion aus            *
*******************************************************

« # 83h # 40h PDIM RCLF -18 CF -17 CF
  DO
    CLLCD "       R L C N" 1 DISP { "Pegel" "Winkel" }
    3 2 PROGRAM ILIST ETEC
    IF 2 SAME THEN
      'ARG(XFN(ALOG(n)))'
      (-4,-90) (4,90)
    ELSE
      '20*LOG(ABS(XFN(ALOG(n))))'
      (-4,-80) (4,80)
    END
    PMAX PMIN 'n' INDEP STEQ { "Diagramm" "Einzelwerte"
    } 6 2 PROGRAM ILIST ETEC
    -> d
    « DO
        CLLCD
        IF d 1 SAME THEN
          ERASE DRAX DRAW LABEL GRAPH TEXT 2
        ELSE
          CLLCD "RLCN Einzelwerte" 1 DISP PROGRAM
          "Wert:" 2 SGET ETEC STR-> 'n' STO RCEQ ->NUM
          ->STR 3 DISP "Ende?" 5 DISP { "nein" "ja" } 6
```

```
        2 PROGRAM ILIST ETEC
      END
    UNTIL
      2 SAME
    END
  »
  CLLCD "Programm beenden ?" 1 DISP { "ja" "nein" } 2
  2 PROGRAM ILIST ETEC
UNTIL
  1 SAME
END
'n' PURGE STOF
»
```

3.3.2 XFN

In XFN steht die komplexe Übertragungsfunktion für das Programm RLCN. Die Funktion XFN selbst ist von der Kreisfrequenz ω abhängig. Nach Eingabe eines reellen Wertes erhält man einen komplexen Wert zurück. Dieser Wert kann als Punkt auf der Ortskurve gedeutet werden. Ein Tiefpass hat die Funktion:

XFN(ω) = i*:w

Ein Bandpass mit der Bandbreite B hat die Funktion:

$$XFN(\omega) = \frac{1}{B} \cdot \left(i \cdot \omega + \frac{1}{i \cdot \omega}\right)$$

XFN

Ebene 1		Ebene 1
x	→	z

XFN berechnet eine komplexe Funktion mit der Kreisfrequenz aus x. Das Ergebnis steht in z.

```
**********************************************************
* HP28/48 Funktion XFN,                                  *
* komplexe Uebertragungsfunktion                         *
*                                                        *
* Eingabe  :                                             *
*  Ebene 1 : Zahl             : Complex Number           *
*                                                        *
* Ausgabe  :                                             *
*  Ebene 1 : Funktionswert    : Complex Number           *
**********************************************************

« -> w
  '(i*w+INV(i*w))*.2'
»
```

3.4 Kettenbruchentwicklung

Bei der Kettenbruchentwicklung handelt es sich um ein Verfahren zur Synthese von RC- bzw. RL-Netzwerken, falls nur Nullstellen und Polstellen der Übertragungsfunktion bekannt sind. Man dividiert, ähnlich wie bei einer Partialbruchzerlegung, die Polynome M(s) durch N(s) und erhält mehrere einfache Ausdrücke. Man hat allgemein

$$\phi(s) = \frac{M(s)}{N(s)} = \frac{a_n \cdot s^n + a_{n-1} \cdot s^{n-1} + \ldots + a_0}{b_m \cdot s^m + b_{m-1} \cdot s^{m-1} + \ldots + b_0}$$

und erhält mit dem Kettenbruchverfahren

$$\phi(s) = \alpha_1 \cdot s + \cfrac{1}{\alpha_2 \cdot s + \cfrac{1}{\ldots + \cfrac{1}{\alpha_r \cdot s}}}$$

3.4.1 KETB

Die Eingabe wird anhand eines Beispieles erläutert:

$$\phi(s) = \frac{s^3 + 2 \cdot s}{2 \cdot s^2 + 1}$$

$a_3s = s^3$

$a_2s = 0$

$a_1s = 2s$

$a_0s = 0$

$b_2s = 2s^2$

$b_1s = 0$

$b_0s = 1$

Der Aufruf von KETB lautet allgemein:

$a_n \cdot s^n + a_{n-1} \cdot s^{n-1} + \ldots + a_0$

$b_n \cdot s^n + b_{n-1} \cdot s^{n-1} + \ldots + b_0$

KETB

In unserem Beispiel lautet der Aufruf:

{ 'S^3' '2*S' }

{ '2*S^2' 1 }

KETB

Das Resultat steht in Q und lautet:

{ '1.5*S' '1.333333333*S' '.5*S' }

$\alpha_3 = 1.5$

$\alpha_2 = 1.333333333$

$\alpha_1 = 0.5$

In obige Kettenbruch-Formel eingesetzt lautet das Ergebnis:

$$\phi(s) = \frac{1}{2} \cdot s + \cfrac{1}{\frac{4}{3} \cdot s + \cfrac{1}{\frac{3}{2} \cdot s}}$$

Anstatt obiger Eingabe kann man sich das 'S' auch sparen, falls man sicher ist, daß die Entwicklung nicht schon frühzeitig abbricht (z.B. bei modifizierten Hurwitzpolynomen).

Eingabe:

{ 1 2 }

{ 2 1 }

KETB

Jetzt lautet das Ergebnis: { 1.5 1.333333333 .5 }, und das reicht für einen Zulässigkeitstest in der Regel aus. Falls der HP nicht fertig werden sollte, muß man Zähler und Nenner vertauschen, da nicht alle Brüche zulässig sind.

(Die Grundlagen, sowie weitere Beispiele und Anwendungen findet man in: „Lineare Systeme und Netzwerke" von H.Wolf)

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
OPM	\USER\ETEC
OPL	\USER\ETEC
PG0	\USER\ETEC

Verwendete externe Variablen

Variable	Verzeichnis	Status
Q	\USER\ETEC	gesichert
Z	\USER\ETEC	gelöscht
N	\USER\ETEC	gelöscht
Z1	\USER\ETEC	gelöscht
S	\USER\ETEC	gelöscht

KETB

Ebene 2	**Ebene 1**		**Ebene 1**
{Liste$_1$}	{Liste$_2$}	→	{Liste$_3$}

Das Polynom in {Liste$_1$} wird durch das Polynom in {Liste$_2$} dividiert und in einen Kettenbruch umgewandelt, der nach {Liste$_3$} geschrieben wird.

```
*******************************************************
********** Kettenbruchentwicklungs-Paket *************
*******************************************************
* HP28/48 Funktion KETB,                              *
* Kettenbruchentwicklung einer Funktion               *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Zaehler          : List                  *
*  Ebene 1 : Nenner           : List                  *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Ergebnis         : List                  *
*                                                     *
* Das Konstrukt „d/dx" im Listing entspricht dem      *
* Differentialzeichen in einem String                 *
*******************************************************

« { } 'Q' STO 'Z1' STO 'N' STO
  DO
    DO
      Z1 'Z' STO N 1 GET Z 1 GET / COLCT Q + 'Q' STO N
      Z DUP 'N' STO Q 1 GET "*" OPM "-" OPL 'Z1' STO
    UNTIL
      Z1 1 GET 1 'S' STO ->NUM 5 FIX RND STD 'S' PURGE
      0 SAME
    END
  UNTIL
    IF Z1 SIZE 1 ≤ THEN
      1
    ELSE
      Z 'S' "d/dx" OPM 'Z1' STO 0
    END
  END
  { Z N Z1 } PURGE
»
```

3.4.2 OPM

OPM verknüpft jedes Element einer Liste mit einer Zahl. Die angewandte Operation steht in einem String.

OPM

Ebene 3	Ebene 2	Ebene 1		Ebene 1
$\{Liste_1\}$	x	"String"	→	$\{Liste_2\}$

$\{Liste_1\}$ wird elementweise mit x verknüpft. Die Verknüpfungsoperation steht in "String". Das Resultat wird nach $\{Liste_2\}$ geschrieben.

```
**********************************************************
* HP28/48 Funktion OPM,                                  *
* Verknüpfung einer Liste mit einer Zahl                 *
*                                                        *
* Eingabe  :                                             *
*  Ebene 3 : Menge           : List                      *
*  Ebene 2 : Zahl            : Real Number               *
*  Ebene 1 : Operator        : String                    *
*                                                        *
* Ausgabe  :                                             *
*  Ebene 1 : Ergebnis        : List                      *
**********************************************************

« { } 1 5 PICK SIZE
  FOR f
    4 PICK f GET 4 PICK 4 PICK STR-> COLCT +
  NEXT
  4 ROLLD 3 DROPN
»
```

3.4.3 OPL

Das Hilfsprogramm OPL verknüpft zwei Listen elementweise miteinander und löscht das erste Element der Ergebnisliste. Die angewandte Verknüpfungsoperation steht in einem String.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
PG0	\USER\ETEC

OPL

Ebene 3	Ebene 2	Ebene 1		Ebene 1
$\{Liste_1\}$	$\{Liste_2\}$	"String"	→	$\{Liste_3\}$

$\{Liste_1\}$ wird elementweise mit $\{Liste_2\}$ verknüpft. Die Verknüpfungsoperation steht in "String". Das Resultat wird nach $\{Liste_3\}$ geschrieben. Das erste Element vom $\{Liste_3\}$ wird gelöscht.

```
*******************************************************
* HP28/48 Funktion OPL,                               *
* Verknüpft 2 Listen Elementweise miteinander und     *
* nimmt das erste Element weg                         *
*                                                     *
* Eingabe  :                                          *
*  Ebene 3 : Menge 1          : List                  *
*  Ebene 2 : Menge 2          : List                  *
*  Ebene 1 : Operator         : String                *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Ergebnis         : List                  *
*******************************************************

« 3 ROLLD DUP2 { } 3 ROLLD SIZE SWAP SIZE MAX 1 SWAP
  FOR f
    f PG0 f PG0 6 PICK STR-> COLCT +
  NEXT
  4 ROLLD 3 DROPN
  IF DUP SIZE 1 > THEN
    DUP SIZE 2 SWAP SUB
  END
»
```

3.4.4 PG0

PG0 holt ein Element aus einer Liste in Ebene 4, falls es vorhanden ist. Gibt es in der Liste kein Element, wird auf dem Stapel eine 0 zurückgegeben. Der Listen-Index steht vor dem Aufruf in Ebene 1.

PG0

Ebene 4	Ebene 3,2	Ebene 1		Ebene 4,3,2	Ebene 1
Liste	?	x	→	unverändert	Element

```
*******************************************************
* HP28/48 Funktion PG0,                               *
* holt Element f aus einer Liste auf Ebene 4, falls   *
* es vorhanden ist. Andernfalls legt es 0 auf den     *
* Stapel.                                             *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 : Menge          : List                    *
*  Ebene 3 : ?              : ?                       *
*  Ebene 2 : ?              : ?                       *
*  Ebene 1 : f              : Real Number             *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Ergebnis       : List                    *
*******************************************************

« -> f
  « 3 PICK
    IF DUP SIZE f < THEN
      DROP 0
    ELSE
      f GET
    END
  »
»
```

3.4.5 Unterprogramm-Beispiele

Das Unterprogramm OPM kann man auch anderweitig verwenden, nämlich um Listen, die Ausdrücke enthalten, zu verarbeiten. Hiermit hat man eine einfache Möglichkeit, Formel-Vektoren zu benutzen (wenn auch mit Einschränkungen).

Bei Eingabe von

{ 1 2 3 }, x, "*", OPM

wird jedes Element der Liste mit x multipliziert. Man erhält

{ x '2*x' '3*x' }

Gibt man

{ 'SQ(a)' 'a*2' '√a' }, a, "∂", OPM

ein, so lautet das Ergebnis hier

{ '2*a' 2 '.5/√a' }

Die Liste wurde nach a differenziert.

OPL verknüpft zwei Listen elementweise miteinander.

Geben Sie

{ x y z }, { a b c }, "^", OPL

ein und Sie erhalten

{ 'y^b' 'z^c' }

4 Reglersimulation

4.1 Grundlagen

Mit dem Regler-Programmpaket können einfache Regelkreise simuliert werden. Dazu müssen nur die Strecken- und die Reglerfunktion angegeben werden. Die Reaktion des Reglers auf einen Sprung kann z.B. auf dem Display qualitativ verfolgt werden. Eine quantitative Bestimmung des Zeitverhaltens ist nicht Sinn des Programms.

Dieses Programm bildet einen Regler und eine Strecke im Zeitbereich nach. Die Beschreibung des Streckenverhaltens XG ist etwas kompliziert, weil die Funktion nicht von der Zeit direkt, sondern von der Eingabe und meistens noch von einer Differenz zur Stellgröße abhängt.

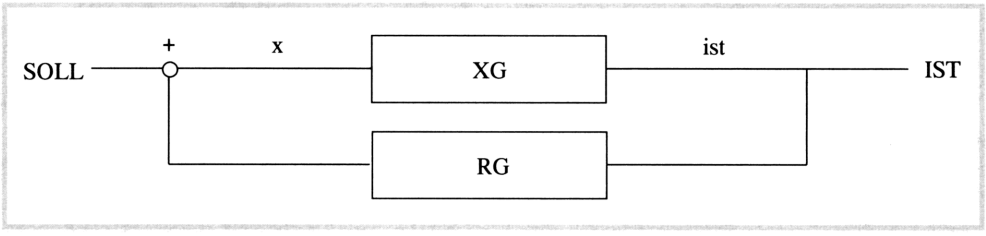


Abb. 4-1 *Regelstrecke*

Ob die Strecke das gewünschte Verhalten zeigt, testet man dadurch, daß anstatt der Stellgröße ein Sprung aufgeschaltet wird. Dazu ersetzt man die Reglerfunktion durch:

« DROP SOLL »

4.1.1 Beispiel

Eine e-Funktion

$$XG = \left(1 - e^{\frac{-t}{\tau}}\right) \cdot x$$

als Sprungantwort muß als eine vom Ist-Wert abhängige Differenzengleichung dargestellt werden. 'x' ist die über den Regler rückgekoppelte Größe IST, und 'ist' ist der momentane Ist-Wert der Strecke. Wenn 'x' eine Sprungfunktion ist, muß sich die Funktion 'ist(t)' letztendlich an x annähern.

Aus der DGL:

$$\left.\frac{dist}{dt}\right|_{t = t_{i-1}} \cdot (t_i - t_{i-1}) + ist(t_{i-1}) = ist(t_i)$$

mit der Abtast-Zeitkonstante

$$t_i - t_{i-1} = \tau_A$$

setzt man nun $ist(t_{i-1})$ ein und erhält dann:

$$ist(t_i) = x \cdot \left[e^{\frac{-t_{i-1}}{\tau}} \cdot \left(\frac{\tau_A}{\tau} - 1\right) + 1 \right]$$

Als letztes wird $XG(t_{i-1})$ in die Gleichung eingesetzt, τ_A zu eins normiert und die Gleichung umgeformt

$$ist(t_i) = (x - ist(t_{i-1})) \cdot \frac{1}{\tau} + ist(t_{i-1})$$

Wie man erkennt, haben wir eine Rekursionsgleichung gefunden. Als HP-Programm sieht die Funktion folgendermaßen aus:

« IST - k * IST + »

wobei

$k=1/\tau$

Mit

x(XG)=RG(XG)

folgt

XG=(RG(XG)-IST)*(1/τ)+IST.

Der Aufbau eines Druckes p in einem Kessel (p-Sprung) und die Spannung U am Kondensator eines RC-Gliedes (U-Sprung) sind Beispiele für Strecken mit obigem Verhalten.

Der Regler selber ist viel einfacher zusammenzustellen als die Strecke, die fertigen Funktionen PR, DR und IR existieren ja schon. Ein PI-Regler hat also das folgende Aussehen

RG(y)=PR(SOLL-y)+k*IR(SOLL-y)

wobei

y=XG.

Als HP-Programm (k ist eine beliebige Konstante)

« DUP PR SWAP IR k * + »

oder algebraisch formuliert

« → X 'PR(X)+k*IR(X)' »

4.2 Bedienung des Programms REG

Nach dem Start wird der Sollwert auf 0 gesetzt. Mit den Tasten Cursor hoch und Cursor runter kann der Sollwert um ±5 verstellt werden.

In P steht die Konstante für das Proportionalglied, in TD die Zeitkonstante des D-Gliedes. Alle anderen Variablen werden ausschließlich intern genutzt. Die Werte sind normiert und daher einheitenlos.

4.2.1 REG

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
RG	\USER\REGLER
XG	\USER\REGLER
PR	\USER\REGLER
IR	\USER\REGLER
DR	\USER\REGLER

Verwendete externe Variablen

Variable	Verzeichnis	Status
SOLL	\USER\REGLER	gesichert
IST	\USER\REGLER	gesichert
TD	\USER\REGLER	gesichert
P	\USER\REGLER	gesichert
I	\USER\REGLER	gesichert
D	\USER\REGLER	gesichert

REG

Ebene 1	Ebene 1
→	

```
*******************************************************
****************** Regelungs-Paket ********************
*******************************************************
* HP28 Programm REG,                                  *
* simuliert einen Regler ( Hauptprogramm )            *
*******************************************************

« 0 'SOLL' STO 0 'IST' STO 0 'I' STO 0 'D' STO
  CLLCD (0,-10) PMIN (138,10) PMAX 0 138
  FOR X
    IF KEY THEN
      IF DUP "UP" SAME THEN
        5 'SOLL' STO+
      END
      IF "DOWN" SAME THEN
        -5 'SOLL' STO+
      END
    END
    X SOLL IST - RG DUP2 XG DUP 'IST' STO R->C PIXEL
    R->C PIXEL
  NEXT
»
```

```
*****************************************************
****************** Regelungs-Paket ******************
*****************************************************
* HP48 Programm REG,                                 *
* simuliert einen Regler ( Hauptprogramm )           *
*****************************************************

« 0 'SOLL' STO 0 'IST' STO 0 'I' STO 0 'D' STO (0,0)
  'ISTA' STO (0,0) 'STA' STO ERASE {# 0d # 0d} PVIEW
  (0,-10) PMIN (131,10) PMAX 0 131
  FOR X
    IF KEY THEN
      CASE
        DUP 25 SAME THEN
          5 'SOLL' STO+ DROP
        END
        35 SAME THEN
          -5 'SOLL' STO+
        END
      END
    END
    X SOLL IST - RG DUP2 XG DUP 'IST' STO R->C ISTA
    OVER 'ISTA' STO LINE R->C STA OVER 'STA' STO LINE
  NEXT
  {ISTA STA} PURGE 7 FREEZE
»
```

4.2.2 RG

RG beinhaltet die Funktion des Reglers. Die Reglerfunktion ist ein HP-Programm. Die Funktion kann in algebraischer Schreibweise oder in UPN erstellt werden.

Es stehen einige Grundfunktionen für die Programmierung bereit, die entsprechend zusammengefügt werden müssen. Diese Funktionen können entweder algebraisch oder in UPN-Schreibweise aufgerufen werden.

Bei den folgenden Beispielen muß k durch eine reelle Zahl ersetzt werden.

Beispielfunktion 1: PI-Regler

Algebraische Schreibweise:	« → x 'PR(x)+k*IR(x)' »
UPN-Schreibweise:	« DUP PR SWAP IR k * + »

Beispielfunktion 2: PID-Regler

Algebraische Schreibweise:	« → x 'PR(x)+k*IR(x)+DR(x)' »
UPN-Schreibweise:	« DUP DUP PR SWAP IR k * + SWAP DR + »

Eventuell verwendete, selbstdefinierte Unterprogramme

Name	Verzeichnis
PR	\USER\REGLER
IR	\USER\REGLER
DR	\USER\REGLER

RG

Ebene 1		Ebene 1
x_1	→	x_2

RG ist die Reglerfunktion.

```
*******************************************************
* HP28/48 Funktion RG,                                *
* Dies ist die Reglerfunktion                         *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Wert             : Real Number           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wert             : Real Number           *
*******************************************************

Beispielfunktion 1:PI-Regler

« -> x 'PR(x) + .5 x IR(x)'»
```

4.2.3 XG

XG enthält die Funktion der Strecke.

Evtl. verwendete, selbstdefinierte Unterprogramme

Name	Verzeichnis
PR	\USER\REGLER
IR	\USER\REGLER
DR	\USER\REGLER

XG

Ebene 1		Ebene 1
x_1	→	x_2

```
*******************************************************
* HP28/48 Funktion XG,                                *
* Hier steht die Streckenfunktion drinnen             *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Wert             : Real Number           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wert             : Real Number           *
*******************************************************

Beispielfunktion 1: Sprungantwort ist eine e-Funktion

« IST - .01 * IST + »
```

4.2.4 DR

DR ist der Differential-Anteil des Reglers.

Verwendete externe Variablen

Variable	Verzeichnis	Status
TD	\USER\REGLER	gesichert
D	\USER\REGLER	gesichert

DR

Ebene 1		Ebene 1
x_1	→	x_2

```
*****************************************************
* HP28/48 Funktion DR,                              *
* Differenzierglied                                 *
*                                                   *
* Eingabe  :                                        *
*  Ebene 1 : Wert             : Real Number         *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Wert             : Real Number         *
*****************************************************

« -> X 'X' D OVER 'D' STO - TD / »
```

4.2.5 IR

IR ist der Integral-Anteil des Reglers.

Verwendete externe Variablen

Variable	Verzeichnis	Status
I	\USER\REGLER	gesichert

IR

Ebene 1		Ebene 1
x_1	→	x_2

```
*****************************************************
* HP28/48 Funktion IR,                              *
* Integrationsglied                                 *
*                                                   *
* Eingabe  :                                        *
*  Ebene 1 : Wert             : Real Number         *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Wert             : Real Number         *
*****************************************************

« -> X 'X' I + DUP 'I' STO »
```

4.2.6 PR

PR ist der Proportional-Anteil des Reglers.

Verwendete externe Variablen

Variable	Verzeichnis	Status
P	\USER\REGLER	gesichert

PR

Ebene 1		Ebene 1
x_1	→	x_2

```
*******************************************************
* HP28/48 Funktion PR,                                *
* Proportionalglied                                   *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Wert             : Real Number           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wert             : Real Number           *
*******************************************************

« -> X
  'X*P'
»
```

5 Mengenalgebra

5.1 Logik-Optimierung

CONS bildet aus einer Menge von Termen alle Primterme einer Schaltfunktion $f(x_1 ... x_n)$. Ein Term einer Schaltfunktion hat n Stellen und kann Einsen, Nullen oder '-' enthalten. Eine Null an der Stelle i heißt hierbei, daß x_i negiert in die Schaltfunktion eingeht.Eine Eins, daß x_i **nicht** negiert in die Schaltfunktion eingeht. Steht an einer Stelle i ein '-', kann die Variable x_i beliebige Werte annehmen. Man hat zum Beispiel die Funktion

f(d,c,b,a)=/dc/a + dcb/a + dc/b/a + d/cb +/dcba

und will diese vereinfachen, dann lautet die Eingabe

01-0 1110 1100 101- 0111

Nach Aufruf von CONS steht in CONL folgendes Ergebnis

101- 1-10 011- -1-0

Dieses Ergebnis stellt alle Primterme dar, die die Funktion hat. Anhand einer Überdeckungstabelle und mit dem Nelson-Verfahren kann man dann eventuell noch einige Primterme streichen.

5.1.1 LINP

Mit LINP werden die Daten eingegeben. Die einzelnen Elemente müssen gleich lang sein. Bestätigen Sie jedes Element mit <ENTER>. Durch Eingabe eines Leerstrings (nur <ENTER>-Taste betätigen) kann man das Programm beenden. Die Eingabe wird in der Variable CONL gespeichert.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
CONL	\USER\LOGS\CONS	gesichert

LINP

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm LINP,                                 *
* Listeneingabe; Liste unter CONS abspeichern         *
*******************************************************

« CLLCD STD { } 1 PROGRAM
  WHILE
    "Ausdruck " OVER ->STR 58 CHR + + OVER SGET DUP
    SIZE
  REPEAT
    ROT + SWAP 1 +
  END
  DROP2 LOGS CONS 'CONL' STO CLMF
»


*******************************************************
* HP48 Programm LINP,                                 *
* Listeneingabe; Liste unter CONS abspeichern         *
*******************************************************

« CLLCD STD { } 1 PROGRAM
  WHILE
    "Ausdruck " OVER ->STR 58 CHR + + OVER SGET DUP
    SIZE
  REPEAT
    ROT + SWAP 1 +
  END
  DROP2 LOGS CONS 'CONL' STO
»
```

5.1.2 CONS

CONS führt den Algorithmus des Consensusverfahrens durch. Das Ergebnis steht anschließend in der Variablen CONL.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
CONB	\USER\LOGS\CONS

Verwendete externe Variablen

Variable	Verzeichnis	Status
CONL	\USER\LOGS\CONS	gesichert
SCH	\USER\LOGS\CONS	gelöscht
I	\USER\LOGS\CONS	gelöscht
J	\USER\LOGS\CONS	gelöscht
K	\USER\LOGS\CONS	gelöscht
X	\USER\LOGS\CONS	gelöscht
Z	\USER\LOGS\CONS	gelöscht

CONS

Ebene 1	**Ebene 1**
→	

```
*******************************************************
************* Consensus-Optimierungspaket *************
*******************************************************
* HP28/48 Programm CONS,                               *
* bildet alle Primterme von Schaltfunktionen           *
*******************************************************

« CONL SIZE 'SCH' STO
  DO
    2 'I' STO 0 'Z' STO
    WHILE
      I SCH ≤
    REPEAT
      1 'J' STO
      WHILE
        J I <
      REPEAT
```

```
      'CONL' DUP I GET SWAP J GET CONB 'X' STO
      IF 1 SAME THEN
        1 'K' STO
        DO
          'CONL' K GET X DUP2 CONB SWAP DROP DUP
          IF ROT SAME THEN
            DROP2 "" 'X' STO 1 'K' STO+
          ELSE
            IF SAME THEN
              CONL DUP 1 K 1 - SUB SWAP K 1 + SCH SUB
              + 'CONL' STO -1 'SCH' STO+
              IF K J < THEN
                -1 DUP 'I' STO+ 'J' STO+
              ELSE
                IF K I < THEN
                  -1 'I' STO+
                END
              END
            ELSE
              1 'K' STO+
            END
          END
        UNTIL
          K SCH > X "" SAME OR
        END
        IF X "" ≠ THEN
          CONL X + 'CONL' STO 1 'SCH' STO+ 1 'Z' STO+
        END
      END
      1 'J' STO+
    END
    1 'I' STO+
  END
UNTIL
  Z 0 SAME
END
{ SCH I J K X Z } PURGE
»
```

5.1.3 CONB

CONB ist ein Unterprogramm, das von CONS aufgerufen wird. Es führt eine Consensusoperation zweier Primterme durch.

CONB

Ebene 2	Ebene 1		Ebene 1	Ebene 2
"String$_1$"	"String$_2$"	→	"String$_3$"	x

String$_1$ wird mit String$_2$ verknüpft und ergibt String$_3$ und die Anzahl der Operationen x.

```
*******************************************************
* HP28/48 Funktion CONB,                              *
* bildet den Consensus-Term zweier Strings            *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : String 1         : String                *
*  Ebene 1 : String 2         : String                *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 2 :     1=ok         : Real Number           *
*           sonst=Fehler      : Real Number           *
*  Ebene 1 : Ergebnis         : String                *
*******************************************************

« DUP SIZE
  -> a b l
  « IF l 0 ≠ a SIZE l SAME AND THEN
      0 "" 1 l
      FOR i
        a i i SUB b i i SUB
        -> ai bi
        « IF ai "-" SAME THEN
            bi
          ELSE
            IF bi "-" SAME THEN
              ai
            ELSE
              IF bi ai SAME THEN
                ai
              ELSE
                "-" ROT 1 + ROT ROT
              END
            END
          END
```

```
        »
        +
      NEXT
    ELSE
      0 ""
    END
  »
»
```

5.2 Logiksimulator

Mit dem Logiksimulator können einfache logische Schaltungen simuliert werden. Das Programm erkennt zwar keine Hazards, ist aber zur funktionalen Simulation von einfachen Schaltungen geeignet. Rückführungen sollte man daher vermeiden.

Das Programm braucht zwei Eingabelisten: eine Klemmenliste KL, in der alle vorhandenen Klemmen eingetragen sind, und eine Ausgabeliste AL, in der die Klemmen stehen, deren Wert im Display angezeigt werden soll. Weiter wird noch eine Datei mit den logischen Verknüpfungen gebraucht (Programm: LOGF).

5.2.1 GO

Nach dem Aufruf von GO wird in der obersten Zeile der Pegel der in AL definierten Klemmen angezeigt. Die Anzeige entspricht der in AL festgelegten Reihenfolge. Für fünf Klemmen zeigt das Display:

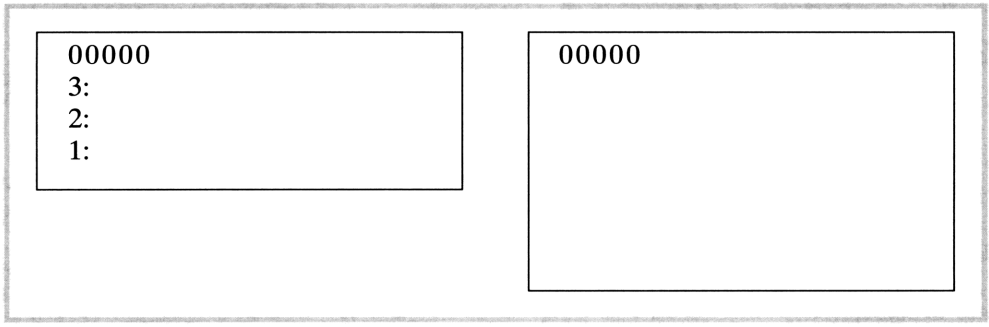


Abb. 5-1 *HP28/48 - Display nach Start von GO*

Wenn man während des Ablaufes eine Taste drückt, kann man einer Klemme einen Wert zuweisen. Man wird zuerst nach der Klemmennummer und dann nach dem Wert gefragt:

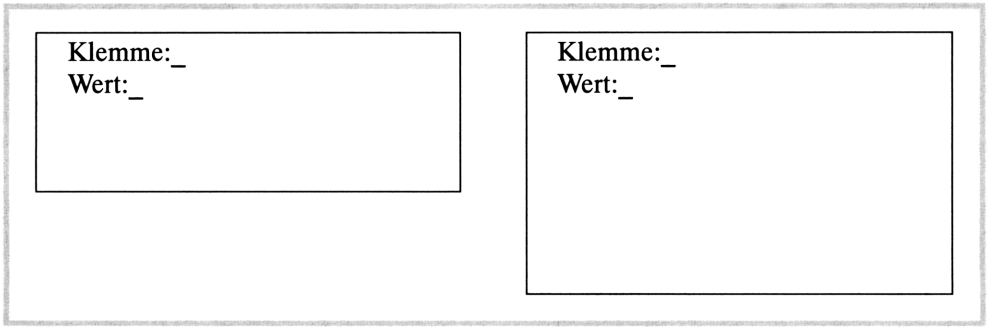


Abb. 5-2 *HP28/48 - Display zur Werteeingabe*

Eine Überprüfung auf unzulässige Eingaben wird nicht gemacht. Eine Klemmenbezeichnung wird als Variable deklariert. Kommt diese Variable nicht als Klemme vor, kann eine vorhandene Variable oder ein Programm gelöscht werden! Zum Beenden des Programms weist man Klemme 0 eine 1 zu.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
GETV	\USER\LOGS\SIM	X	
LOGF	\USER\LOGS\SIM	X	X
GETE	\USER\LOGS\SIM	X	

Verwendete externe Variablen

Variable	Verzeichnis	Status
KL	\USER\LOGS\SIM	gesichert
AL	\USER\LOGS\SIM	gesichert
Vx	\USER\LOGS\SIM	gelöscht

'x' sind die in KL spezifizierten Namen.

GO

Ebene 1	**Ebene 1**
→	

GO startet eine Simulation.

```
*******************************************************
****************** LOGSIM-Paket ***********************
*******************************************************
* HP28 Programm GO,                                    *
* startet eine Simulation                              *
*******************************************************

« 0 KL + 1 OVER SIZE
  FOR i
    DUP i GETE 0 SWAP STO
  NEXT
  DROP
  DO
    IF KEY THEN
      DROP GETV
    END
    LOGF "" AL 1 OVER SIZE
    FOR i
      DUP i GETE ->NUM ->STR ROT SWAP + SWAP
    NEXT
    DROP 1 DISP
  UNTIL
    V0 1 SAME
  END
  VARS 1 OVER SIZE
  FOR i
    DUP LOGS i GET DUP ->STR SIM
    IF 2 2 SUB "V" SAME THEN
      PURGE
    ELSE
      DROP
    END
  NEXT
  DROP CLMF
»


*******************************************************
****************** LOGSIM-Paket ***********************
*******************************************************
* HP48 Programm GO,                                    *
* startet eine Simulation                              *
*******************************************************

« CLLCD 0 KL LIST-> 0 SWAP
  START
    0 "'V" ROT + STR-> STO
  NEXT
```

```
  DO
    IF KEY THEN
      DROP CLLCD PROGRAM "V" "Klemme#" 1 SGET + STR->
      "Wert#" 2 SGET STR-> SWAP LOGS SIM STO CLLCD
    END
    LOGF AL LIST-> "" 1 ROT
    START
      "V" ROT + STR-> SWAP +
    NEXT
    1 DISP
  UNTIL
    V0 1 SAME
  END
  VARS LIST-> 1 SWAP
  START
    DUP ->STR
    IF 2 DUP SUB "V" SAME THEN
      PURGE
    ELSE
      DROP
    END
  NEXT
»
```

5.2.2 LOGF

In LOGF ist die zu simulierende Verknüpfung gespeichert.

Zur Erklärung ist hier als Beispiel ein UND-ODER- Schaltnetz dargestellt, das simuliert werden soll.

AL = { 7 } und KL = { 1 2 3 4 5 6 7 }.

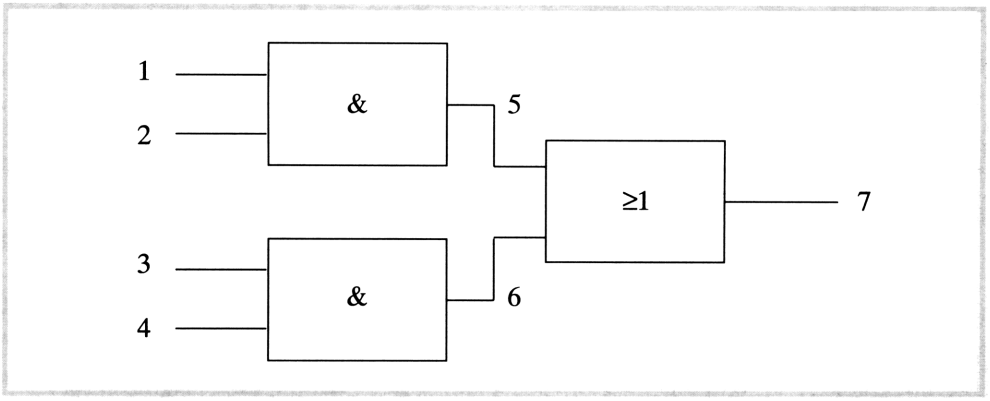


Abb. 5-3 *Beispielfunktion, die simuliert werden soll*

Das Programm, das in LOGF stehen muß:

```
« V1 V2 AND 'V5' STO
  V3 V4 AND 'V6' STO
  V5 V6 OR 'V7' STO
»
```

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
Macros	\USER\LOGS\SIM

Verwendete externe Variablen

Variable	Verzeichnis	Status
Vx	\USER\LOGS\SIM	gelöscht

'x' sind die in KL spezifizierten Namen.

LOGF

Ebene 1	**Ebene 1**
→	

LOGF führt die Verknüpfung aus, die simuliert werden soll.

```
*****************************************************
* HP28/48 Programm LOGF,                            *
* eigentliche logische Gleichungen                  *
*****************************************************

« V1 V2 AND V1 V4 AND OR 'V3' STO »
```

5.2.3 GETE (HP28)

Die Funktion GETE holt einen Element aus einer Liste und gibt es als Name zurück.

Verwendete externe Variablen

Variable	Verzeichnis	Status
Vx	\USER\LOGS\SIM	gelöscht

'x' sind die in KL spezifizierten Namen.

GETE

Ebene 2	**Ebene 1**		**Ebene 1**
{Liste}	x	→	Name

Holt Element x aus {Liste} und gibt Name zurück.

```
*****************************************************
* HP28 Funktion GETE,                               *
* nimmt Element aus Liste und macht Variable daraus *
*                                                   *
* Eingabe  :                                        *
*  Ebene 2 : Liste           : List                 *
*  Ebene 1 : Elementnummer   : Real Number          *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : globale Variable : Name                *
*****************************************************

« LOGS GET ->STR "V" SWAP + STR-> SIM »
```

5.2.4 GETV (HP28)

Die Funktion GETV fragt die Klemmennummer ab und weist der Klemme einen Wert zu (Funktion siehe unter GO).

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
Vx	\USER\LOGS\SIM	gelöscht

'x' sind die in KL spezifizierten Namen.

GETV

Ebene 1	**Ebene 1**
→	

Fragt Klemmennummer ab.

```
*******************************************************
* HP28 Programm GETV,                                 *
* frägt Klemmennummer ab und weist Klemme einen       *
* Wert zu                                             *
*******************************************************

« CLLCD PROGRAM "V" "Klemme+" 1 SGET + STR-> "Wert+" 2
  SGET STR-> SWAP LOGS SIM STO CLLCD
»
```

5.2.5 Makros

Ein Makro ist nichts weiter als ein Unterprogramm, das eine bestimmte Funktion ausführt. Die Anzahl und der Typ der einzelnen Parameter, die benötigt werden, sind nicht festgelegt sondern vom Anwender bestimmt. Ein Makro kann auch sich selbst oder andere Makros aufrufen. Der Übersichtlichkeit halber sollte man dies jedoch vermeiden.

Ein Makro greift auf keine globale Variable zu. Alle Parameter, die es benötigt, werden auf dem Stack übergeben. Allgemein gilt für Makros folgende Vereinbarung:

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
Makros	\USER\LOGS\SIM

Makro

Stack		Stack
Parameter	→	Parameter

5.2.5.1 M2M1

Hierbei handelt es sich um einen 2 zu 1 Multiplexer. Aus einer Liste wird der Wert herausgeholt, der mit dem als binär aufzufassenden Parameter x indiziert wird. Eine '0' wählt also das erste und eine '1' das zweite Element aus.

M2M1

Ebene 2	Ebene 1		Ebene 1
{Liste}	x	→	Objekt

```
*******************************************************
* HP28/48 Funktion M2M1,                              *
* 2 zu 1 Multiplexer                                  *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Eingangsbelegungen  : List               *
*  Ebene 1 : a                   : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wert von E(a)       : verschieden        *
*******************************************************

« 1 + GET ->NUM »
```

5.2.5.2 M4M1

Hierbei handelt es sich um einen 4 zu 1 Multiplexer. Aus einer Liste wird der Wert herausgeholt, der mit den als binär aufzufassenden Parametern x_0 und x_1 indiziert wird. Die Reihenfolge der Parameter auf dem Stack entspricht der umgekehrten Reihenfolge der Wertigkeit der Bits.

Stackebene 2:=x_0
Stackebene 1:=x_1

Es wird das Element i aus einer Liste in den Stack kopiert. Wobei

$i = x_1*2+x_0$

M4M1

Ebene 3	Ebenen 2,1		Ebene 1
{Liste}	x_0,x_1	→	Objekt

M4M1 kopiert das mit x_0 und x_1 indizierte Objekt aus der Liste in den Stack.

```
*******************************************************
* HP28/48 Funktion M4M1,                              *
* 4 zu 1 Multiplexer                                  *
*                                                     *
* Eingabe  :                                          *
*  Ebene 3 : Eingangsbelegungen  : List               *
*  Ebene 2 : a                   : Real Number        *
*  Ebene 1 : b                   : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wert von E(2b+a)    : verschieden        *
*******************************************************

« 2 * + 1 + GET ->NUM »
```

5.2.5.3 M8M1

Hierbei handelt es sich um einen 8 zu 1 Multiplexer. Aus einer Liste wird der Wert herausgeholt, der mit den als binär aufzufassenden Parametern x_0 bis x_2 indiziert wird. Die Reihenfolge der Parameter x_0 bis x_2 auf dem Stack entspricht wiederum der umgekehrten Reihenfolge der Wertigkeit der Bits.

Stackebene $3=x_0$
Stackebene $2=x_1$
Stackebene $1=x_2$

Es wird das Element i aus einer Liste in den Stack kopiert. Wobei

$i = x_2*2\text{^}2+x_1*2+x_0$

M8M1

Ebene 4	Ebenen 3-1		Ebene 1
{Liste}	x_0-x_2	→	Objekt

M8M1 kopiert das mit x_0-x_2 indizierte Objekt aus der Liste in den Stack.

```
*******************************************************
* HP28/48 Funktion M8M1,                              *
* 8 zu 1 Multiplexer                                  *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 : Eingangsbelegungen  : List               *
*  Ebene 3 : a                   : Real Number        *
*  Ebene 2 : b                   : Real Number        *
*  Ebene 1 : c                   : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wert von E(4c+2b+a) : verschieden        *
*******************************************************

« 2 * + 2 * + 1 + GET ->NUM »
```

6 Uhrenpaket UHR

Dieses Programmpaket wird zeigen, daß mit dem HP28 Dinge möglich sind, von denen viele noch nicht einmal zu träumen wagten. Es ist nämlich nicht nur möglich, die Zeit quartzgenau zu messen, solange der Rechner eingeschaltet ist, nein, es ist sogar möglich die Messung weiterzuführen, selbst wenn der Rechner ausgeschaltet ist. Das funktioniert so:

Der HP28 besitzt einen internen 64-bit Zähler, der mit 8192 Hz zählt (auch wenn er ausgeschaltet ist). Mit einer von Hewlett Packard nicht dokumentierten Systemfunktion kann man den Zählerstand abfragen. Die Systemfunktion in dem hier vorgestellten Programmpaket läuft auf einem HP28S mit der ROM-Version 2BB. Für alle anderen ROM-Versionen kann die Einsprungadresse eine andere sein (muß aber nicht). Eine falsche Einsprungadresse führt in aller Regel zu einem Rechnerabsturz mit Speicherverlust (Lost Memory).

Beim HP48 wurde das Auslesen des Zählers seitens des Herstellers dokumentiert. Das Auslesen des Zählers erfolgt mit dem Befehl TICKS. Aus Kompatibilitätsgründen wurde beim HP28S der selbe Befehl verwendet. Frequenz und Bitbreite des Zählers sind gleich geblieben. Die entsprechenden HP28 Routinen wie Digitaluhr mit Datumsberechnung sind uneingeschränkt lauffähig. Es macht jedoch wenig Sinn, diese Routinen zu implementieren, da diese in Form einer permanenten Digitaluhr schon eingebaut sind. Für die Features Analoguhr, Stoppuhr, sowie Timer wurde eine entsprechende HP48 Version programmiert. Weiterhin sind beim HP48 zwei zusätzliche Funktionen zur Kalenderwochenberechnung erstellt worden. Das Programmpaket besteht aus mehreren Teilen

- einer Digitaluhr mit Datum,
- einer Analoguhr,
- einer Stoppuhr,
- einem Timer,

sowie aus verschiedenen Service-Routinen.

6.1 UHR (HP28)

Beim Start von UHR zeigt das Display den Wochentag, das Datum und die Zeit an. Zum Verlassen des Programms drückt man eine Taste - nur nicht <ATTN> - und man ist wieder im Editiermodus.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
DAY	\USER\UHR
DATE	\USER\UHR
THMS	\USER\UHR
TICKS	\USER\UHR

Verwendete externe Variablen

Variable	Verzeichnis	Status
ZEIT0	\USER\UHR	gesichert

UHR

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm UHR,                                  *
* schreibt ins LCD Datum, Wochentag und Uhrzeit       *
*******************************************************

« CLLCD STD 64 STWS 0
  DO
    TICKS ZEIT0 - 707788800 /
    DUP2 ÷
    « "    " DAY + "  " + DATE + 1 DISP SWAP »
    IFT
    DROP "       " THMS + 3 DISP
  UNTIL
    KEY
  END
  DROP2 CLMF
»
```

6.2 ANALOG

Die Funktion ANALOG bildet eine Analoguhr nach. Auf Grund der langsamen Verarbeitungsgeschwindigkeit der Taschenrechner mußten Kompromisse geschlossen werden. Ein Zeigeraufbau im Display kam aus diesem Grund nicht in Frage. Ein neues Bild wird daher im Hintergrund aufgebaut. Pro Bildaufbau werden beim HP28S ca. 7 Sekunden und beim HP48 ca. 4 Sekunden benötigt. Da beim Programmstart kein Bild existiert, wird während der Bildaufbauzeit nur die Maske der Analoguhr dargestellt. Die Maske des Programms muß **vorher** einmalig mit dem Programm DEFANALOG erstellt werden. Zum Beenden drückt man eine Taste - nur nicht <ATTN> - und man ist wieder im Editiermodus. Die Reaktionszeit auf den Tastendruck kann im Bereich der Bildaufbauzeit liegen. Das Programm sollte immer ordnungsgemäß beendet werden, da nur dann, weil Systemflags geändert werden, der Ausgangszustand wiederhergestellt wird. Folgende Systemflags werden geändert

HP28 : 37 .. 42, 60
HP48 : -17, -18

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
ZSET	\USER\UHR	X	X
TICKS	\USER\UHR	X	

Verwendete externe Variablen

Variable	Verzeichnis	Status	2	4
ZEIT0	\USER\UHR	gesichert	X	
AMASKE	\USER\UHR	gesichert	X	X

ANALOG

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm ANALOG,                               *
* schreibt die Uhrzeit in analoger Form ins LCD       *
*                                                     *
* Eingabe  :                                          *
*  'AMASKE'  Bildschirmmaske     : String             *
*******************************************************

« AMASKE ->LCD RCLF 64 STWS DEG
  DO
    TICKS ZEIT0 - B->R 29491200 / 12 MOD DUP 30 * DUP
    COS SWAP SIN AMASKE .05 .6 ZSET 4 ROLLD DROP2 FP
    360 * DUP COS SWAP SIN ROT .05 .85 ZSET ->LCD DROP2
  UNTIL
    KEY
  END
  DROP STOF CLMF
»


*******************************************************
* HP48 Programm ANALOG,                               *
* schreibt die Uhrzeit in analoger Form ins LCD       *
*                                                     *
* Eingabe  :                                          *
*  'AMASKE'  Bildschirmmaske     : String             *
*******************************************************

« RCLF DEG AMASKE DUP PICT STO { # 41h # 0h } { # 80h
  # 3Fh } SUB
  -> m
  « { # 0h # 0h } PVIEW
    DO
      PICT DUP { # 7h # 29h } DATE 0 TSTR 1 12 SUB 1
      ->GROB REPL { # 41h # 0h } TIME HMS-> DUP FP 360
      * DUP COS NEG SWAP SIN R->C 32 * m .05 .8 ZSET
      SWAP 12 MOD 30 * DUP COS NEG SWAP SIN R->C 32 *
      SWAP .05 .55 ZSET { # 2Bh # 1Eh } DATE 0 TSTR 5 6
      SUB 1 ->GROB GOR REPL
    UNTIL
      KEY
    END
    DROP STOF
  »
»
```

6.3 Uhr einstellen (HP28)

Wenn man die Zeit einstellen will, muß berücksichtigt werden, daß damit das Datum verstellt wird. Deshalb muß immer zuerst die Zeit und dann das Datum gestellt werden.

6.3.1 TSET (HP28)

Mit der Funktion TSET wird die Uhrzeit eingestellt. Ist es 17:25 und 33 Sekunden, so lautet die Eingabe in Ebene 1 folgendermaßen: 17.2533

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
TICKS	\USER\UHR

Verwendete externe Variablen

Variable	Verzeichnis	Status
ZEIT0	\USER\UHR	gesichert

TSET

Ebene 1	**Ebene 1**
x	→

TSET speichert die Zeit x als Referenzwert in die Variable ZEIT0.

```
*******************************************************
* HP28 Prozedur TSET,                                 *
* stellt Uhrzeit                                      *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Uhrzeit            : Real Number         *
*                                                     *
* Beispiel : 14:30:53 -> 14.3053                      *
*******************************************************

« HMS-> 29491200 * TICKS SWAP - 'ZEIT0' STO »
```

6.3.2 DSET (HP28)

Mit der Funktion DSET wird das Datum eingestellt. Für den 10.02.91 lautet die Eingabe in Ebene 1 folgendermaßen: 10.021991

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
TICKS	\USER\UHR

Verwendete externe Variablen

Variable	Verzeichnis	Status
ZEIT0	\USER\UHR	gesichert

DSET

Ebene 1		Ebene 1
x	→	

DSET verrechnet das Datum x mit der Variablen ZEIT0 und speichert das Ergebnis wieder in ZEIT0 ab.

```
*******************************************************
* HP28 Prozedur DSET,                                 *
* stellt Datum ein, Uhrzeit muß in 'ZEIT0' stehen     *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Datum              : Real Number         *
*                                                     *
* Beispiel : 28.07.1990 -> 28.071990                  *
*******************************************************

« DUP IP SWAP FP 100 * DUP IP SWAP FP 10000 * 1989 -
  DUP 365 * SWAP 1 + 4 / DUP IP SWAP FP NOT 4 PICK 3 <
  AND - + { -1 30 58 89 119 150 180 211 242 272 303 333
  } ROT GET + + R->B TICKS 707788800 OVER ZEIT0 - B->R
  OVER MOD SWAP 4 ROLL * + - 'ZEIT0' STO
»
```

6.4 Stoppuhr

Zur Bedienung der Stoppuhr gibt es nur zwei Funktionen: STA (für Start) und STOP. Wenn man eine Zwischenzeit messen will, drückt man STOP. Der Zähler läuft trotzdem weiter. So können beliebig viele Zwischenzeiten gemessen werden. Um den Zähler zurückzusetzen, gibt man wieder STA ein.

6.4.1 STA

Mit der Funktion STA wird die aktuelle Zeit als Startzeit eingestellt.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
TICKS	\USER\UHR	X	

Verwendete externe Variablen

Variable	Verzeichnis	Status
STOP0	\USER\UHR	gesichert

STA

Ebene 1	**Ebene 1**
→	

STA speichert den aktuellen Timerwert als Referenzwert in die Variable STOP0.

```
**********************************************************
* HP28 Programm STA,                                     *
* startet Stoppuhr                                       *
**********************************************************

« TICKS 534 + 'STOP0' STO »
```

```
*******************************************************
* HP48 Programm STA,                                  *
* startet Stoppuhr                                    *
*******************************************************

« TICKS 300 + 'STOP0' STO »
```

6.4.2 STOP

Mit der Funktion STOP wird die abgelaufene Zeit seit dem STA Befehl angezeigt. Zeiten über 24 Stunden können nicht ausgegeben werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
TICKS	\USER\UHR	X	

Verwendete externe Variablen

Variable	Verzeichnis	Status
STOP0	\USER\UHR	gesichert

STOP

Ebene 1		Ebene 1
	→	"String"

STOP zeigt in "String" die abgelaufene Uhrzeit im Format "Stunden:Minuten:Sekunden.Sekunden/100" an.

```
*******************************************************
* HP28/48 Funktion STOP,                              *
* liest Stoppuhr aus                                  *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : abgelaufende Zeit   : String             *
*******************************************************

« TICKS STOP0 - B->R 29491200 / 24 MOD ->HMS DUP IP
  ->STR 58 CHR ROT FP 6 FIX ->STR STD DUP2 3 4 SUB +
  ROT + OVER 5 6 SUB + "." + SWAP 7 8 SUB + +
»
```

6.5 Timer

Der Timer ist ein einfacher Rückwärtszähler, der die verbleibende Zeit bis 0:00 anzeigt. Er kann allerdings maximal 24 Stunden zählen.

6.5.1 CSET

Mit der Funktion CSET legt man den Timerstartwert fest. Will man beispielsweise 2 Stunden und eine Sekunde zählen, so lautet das Eingabeformat : 2.0001

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
TICKS	\USER\UHR	X	

Verwendete externe Variablen

Variable	Verzeichnis	Status
COUNT0	\USER\UHR	gesichert

CSET

Ebene 1	Ebene 1
x	→

CSET speichert den Timerwert x als Referenzwert in die Variable COUNT0.

```
*******************************************************
* HP28/48 Prozedur CSET,                              *
* setzt Countdownzähler                               *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Zeit                : Real Number        *
*                                                     *
* Beispiel : 14:30:53 -> 14.3053                      *
*******************************************************

« HMS-> 29491200 * TICKS + 'COUNT0' STO »
```

6.5.2 CTR

Mit der Funktion CTR kann man den Timerwert auslesen. Die Rückgabe erfolgt in einem String.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
TICKS	\USER\UHR	X	

Verwendete externe Variablen

Variable	Verzeichnis	Status
COUNT0	\USER\UHR	gesichert

CTR

Ebene 1		**Ebene 1**
	→	"String"

CTR zeigt in "String" die noch verbleibende Zeit im Format "Stunden:Minuten:Sekunden" an.

```
*******************************************************
* HP28 Funktion CTR,                                  *
* liest Countdownzähler aus                           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : verbleibende Zeit   : String             *
*******************************************************

« COUNT0 TICKS - B->R 29491200 / 24 MOD ->HMS DUP IP
  ->STR 58 CHR ROT FP 8 FIX RND STD ->STR DUP 2 3 SUB 3
  PICK ROT 4 5 SUB + + + +
»
```

```
*******************************************************
* HP48 Funktion CTR,                                  *
* liest Countdownzähler aus                           *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : verbleibende Zeit   : String             *
*******************************************************

« COUNTO TICKS - B->R 29491200 / 24 MOD ->HMS DUP IP
  ->STR ":" ROT FP 8 RND ->STR DUP 2 3 SUB 3 PICK ROT 4
  5 SUB + + + +
»
```

6.6 Kalenderwoche (HP48)

Die folgenden Funktionen dienen der Umwandlung einer Kalenderwoche in das entsprechende Datum und umgekehrt. Sie wurden auf dem HP28 nicht implementiert, da spezielle Funktionen der Datumsarithmetik des HP48 verwendet werden.

6.6.1 D→KW (HP48)

Die Funktion D→KW bestimmt zum aktuellen Datum die Kalenderwoche. Durch einfache Modifikation der Funktion (weglassen des DATE-Befehls) kann die Kalenderwoche für ein beliebiges Datum ermittelt werden. Das Datumsformat muß bei der modifizierten Funktion nach HP48-Datumskonvention eingegeben werden.

Das Programm gibt 0 zurück, falls ein Datum eingegeben wird, das vor dem ersten Werktag im Jahr liegt.

D→KW

Ebene 1		Ebene 1
	→	n

Bestimme zum aktuellen Datum die Kalenderwoche n.

```
*****************************************************
* HP48 Funktion D->KW,                              *
* bestimmt die aktuelle Kalenderwoche               *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Kalenderwoche       : Real Number      *
*****************************************************

« DATE DUP 100 * FP 101 + 100 / "MOTUWETHFRSASU" OVER 0
  TSTR 1 2 SUB POS 1 - 2 / DUP 3 > 7 * - SWAP ROT DDAYS
  + 7 / FLOOR 1 +
»
```

6.6.2 KW→D (HP48)

Die Funktion KW→D ermittelt zur Kalenderwoche das entsprechende Datum des ersten Wochentags (Montag). Als Eingabe kann entweder nur die Woche, oder die Woche und das Jahr, durch Punkt getrennt, eingegeben werden.

Beispiel: KW28 im Jahre 1985

28.1985 → "08.07.85"

KW→D

Ebene 1		Ebene 1
x	→	"String"

Bestimme zur Kalenderwoche x das Datum "String" des Wochenanfangs.

```
*******************************************************
* HP48 Funktion KW->D,                                *
* wandelt die eingegebene Kalenderwoche in das Datum  *
* des Wochenanfangs um                                *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : Kalenderwoche       : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Datum               : String             *
*                                                     *
* Beispiel Eingabe : 12     -> KW 12 aktuelles Jahr   *
*                    12.94  -> KW 12 1994             *
*******************************************************

« 1 -
  IF DUP FP NOT THEN
    DATE 100 *
  ELSE
    DUP SWAP IP SWAP
  END
  FP 101 + 100 / "MOTUWETHFRSASU" OVER 0 TSTR 1 2 SUB
  POS 1 - 2 / DUP 3 > 7 * - NEG ROT 7 * + DUP 0 ≥ *
  DATE+ 0 TSTR 5 12 SUB
»
```

6.7 Serviceroutinen

Die folgenden Funktionen sind Unterprogramme zu den oben genannten Funktionen. Sie können aber für eigene Anwendungen verwendet werden.

6.7.1 THMS (HP28)

Die Funktion THMS gibt als Rückgabewert die aktuelle Uhrzeit in Ebene 1 als String zurück.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
TICKS	\USER\UHR

Verwendete externe Variablen

Variable	Verzeichnis	Status
ZEIT0	\USER\UHR	gesichert

THMS

Ebene 1		**Ebene 1**
	→	"String"

THMS zeigt in "String" die Uhrzeit im Format "Stunden:Minuten:Sekunden" an.

```
*******************************************************
* HP28 Funktion THMS,                                 *
* liefert aktuelle Uhrzeit                            *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Uhrzeit            : String              *
*******************************************************

« TICKS ZEIT0 - B->R 29491200 / 24 MOD ->HMS DUP IP
  ->STR 58 CHR ROT FP 6 FIX ->STR STD DUP2 3 4 SUB +
  ROT + SWAP 5 6 SUB + +
»
```

6.7.2 THM (HP28)

Die Funktion THM gibt als Rückgabewert die aktuelle Uhrzeit in Ebene 1 als String zurück.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
THMS	\USER\UHR

THM

Ebene 1		Ebene 1
	→	"String"

THM zeigt in "String" die Uhrzeit im Format "Stunden:Minuten" an.

```
*******************************************************
* HP28 Funktion THM,                                  *
* liefert aktuelle Uhrzeit ohne Sekunden              *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Uhrzeit            : String              *
*******************************************************

« THMS 1 OVER SIZE 3 - SUB »
```

6.7.3 DATE (HP28)

Die Funktion DATE gibt als Rückgabewert das aktuelle Datum in Ebene 1 als String zurück. Achtung: Aufgrund eines schnelleren Algorithmus wird das Datum nur bis zum 28.02.2100 korrekt ausgegeben!

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
TICKS	\USER\UHR

Verwendete externe Variablen

Variable	Verzeichnis	Status
ZEIT0	\USER\UHR	gesichert

DATE

Ebene 1		Ebene 1
	→	"String"

DATE zeigt in "String" das Datum im Format "Tag.Monat.Jahr" an.

```
*******************************************************
* HP28 Funktion DATE,                                 *
* liest Datum aus                                     *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Datum              : String              *
*******************************************************

« TICKS ZEIT0 - 707788800 / B->R 1461 DUP2 / IP 4 * ROT
  ROT MOD DUP 365 / IP DUP 4 SAME - ROT OVER + 1989 +
  ROT ROT 365 * - OVER 4 MOD NOT 31 28 ROT + { 31 30 31
  30 31 31 30 31 30 31 } + + 1
  WHILE
    DUP2 GET 4 ROLL DUP2 ≤
  REPEAT
    SWAP - ROT ROT 1 +
  END
  ROT 100 / + 1 + 2 FIX ->STR STD "." + 4 ROLLD DROP2
  ->STR +
»
```

6.7.4 DAY (HP28)

Die Funktion DAY gibt als Rückgabewert den aktuellen Wochentag in Ebene 1 als String zurück.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
TICKS	\USER\UHR

Verwendete externe Variablen

Variable	Verzeichnis	Status
ZEIT0	\USER\UHR	gesichert

DAY

Ebene 1		**Ebene 1**
	→	"String"

DAY zeigt in "String" den Wochentag.

```
*******************************************************
* HP28 Funktion DAY,                                  *
* liest Wochentag aus                                 *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Wochentag          : String              *
*******************************************************

« { "Sun" "Mon" "Tue" "Wed" "Thu" "Fri" "Sat" } TICKS
  ZEIT0 - 707788800 / B->R 7 MOD 1 + GET
»
```

6.7.5 TICKS (HP28)

Die Funktion TICKS gibt als Rückgabewert den internen Timerwert des Rechners zurück. Diese Funktion gilt nur für den HP28S ROM-Version 2BB. Beim HP48 ist dies eine eingebaute Funktion!

TICKS

Ebene 1		Ebene 1
	→	# n

```
*******************************************************
* HP28 Funktion TICKS,                                *
* liefert aktuellen Timerwert                         *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Timerwert          : Binary Integer      *
*                                                     *
* ACHTUNG : nicht dokumentierter Systemaufruf, gilt   *
*           nur für HP28S Rom-Version 2BB             *
*******************************************************

« # 11CAh SYSEVAL »
```

6.7.6 ZSET

Die Funktion ZSET ist ein spezielles Unterprogramm der Funktion ANALOG zum Zeichnen der Zeiger. Deshalb werden die Übergabeparameter nicht dokumentiert. Als Hilfe sei hier auf den Kopf des Listings von ZSET verwiesen.

ZSET

Ebene 1..		Ebene 1..
Werte	→	Werte

```
*******************************************************
* HP28 Funktion ZSET,                                 *
* zeichnet Zeiger in Bildschirmstring                 *
*                                                     *
* Eingabe  :                                          *
*  Ebene 5 : y-Position          : Real Number        *
*  Ebene 4 : x-Position          : Real Number        *
*  Ebene 3 : Bildschirmstring    : String             *
*  Ebene 2 : Zeigerfaktor Anfang : Real Number        *
*  Ebene 1 : Zeigerfaktor Ende   : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 3 : y-Position          : Real Number        *
*  Ebene 2 : x-Position          : Real Number        *
*  Ebene 1 : Bildschirmstring    : String             *
*                                                     *
* ZSET ist ein Unterprogramm zu ANALOG                *
*******************************************************

« FOR i
    OVER i * 15 5 PICK i * 15.5 * .5 - IP - SWAP 6 + 17
    * .5 + IP OVER 8 / IP 137 * + 2 ROT 8 MOD ^ CHR
    OVER 4 PICK SWAP 1 + DUP SUB OR ROT ROT DUP2 1 SWAP
    SUB 4 ROLLD 2 + OVER SIZE SUB + +
  .06 STEP
»
```

```
*****************************************************
* HP48 Funktion ZSET,                               *
* zeichnet Zeiger in Bildschirmstring               *
*                                                   *
* Eingabe  :                                        *
*  Ebene 4 : y/x-Position        : Complex Number   *
*  Ebene 3 : Grafikobjekt        : GROB             *
*  Ebene 2 : Zeigerfaktor Anfang : Real Number      *
*  Ebene 1 : Zeigerfaktor Ende   : Real Number      *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Grafikobjekt        : GROB             *
*                                                   *
* ZSET ist ein Unterprogramm zu ANALOG              *
*****************************************************

« FOR i
    OVER C->R i * 32 + R->B SWAP i * 32 + R->B 2 ->LIST
    GROB 1 1 10 REPL
  .03125 STEP
  SWAP DROP
»
```

6.7.7 DEFANALOG

Die Funktion DEFANALOG definiert die LCD-Maske für die Analoguhr. Die Maske wird in der Variablen 'AMASKE' abgespeichert. Es kann auch eine eigene Maske erstellt werden. Hierbei ist jedoch zu beachten, daß die Zeiger immer an dieselbe Position geschrieben werden und die Maske den Namen 'AMASKE' besitzen muß.

Verwendete externe Variablen

Variable	Verzeichnis	Status
AMASKE	\USER\UHR	gesichert

DEFANALOG

Ebene 1	**Ebene 1**
	→

```
*******************************************************
* HP28 Funktion DEFANALOG,                            *
* definiert Bildschirmmaske für Analoguhr             *
*                                                     *
* Ausgabe  :                                          *
*  'AMASKE'  Bildschirmmaske     : String             *
*******************************************************

« RCLF CLLCD " Hewlett" 1 DISP " Packard" 2 DISP
  " Timer" 4 DISP (-6,-1.0009) PMIN (2,1) PMAX DEG -.06
  .066
  FOR i
    i 0 R->C PIXEL 0 i R->C PIXEL
  .063 STEP
  0 180 DUP2
  FOR i
    i COS i SIN R->C DUP NEG PIXEL PIXEL
  3 STEP
  FOR i
    i COS i SIN R->C .8 .88
    FOR j
      DUP j * DUP NEG PIXEL PIXEL
    .04 STEP
    DROP
  30 STEP
  STOF 'PPAR' PURGE LCD-> 0 CHR 1 9
  START
    DUP +
  NEXT
  1 367 SUB 56 CHR 68 CHR + 120 CHR + 0 CHR + 120 CHR +
  64 CHR + 120 CHR + 0 CHR + 104 CHR DUP + + 112 CHR +
  0 CHR + 112 CHR + 8 CHR DUP + + 0 CHR + 104 CHR + 88
  CHR + 72 CHR + OVER + + 1 548 SUB OR 'AMASKE' STO
  CLMF
»
```

```
*******************************************************
* HP48 Funktion DEFANALOG,                            *
* definiert Bildschirmmaske für Analoguhr             *
*                                                     *
* Ausgabe  :                                          *
*  'AMASKE'  Bildschirmmaske     : String             *
*******************************************************

« RCLF (-3.05,-1) PMIN (1.05,1) PMAX ERASE DEG (0,0)
  .04 0 360 ARC (0,0) .96 0 360 ARC 0 150
  FOR i
    i COS i SIN R->C DUP .7 * SWAP .88 * DUP2 LINE NEG
    SWAP NEG LINE
  30 STEP
  STOF 'PPAR' PURGE PICT RCL { # 57h # 29h } "Quarz" 1
  ->GROB REPL { # 8h # 8h } "Hewlett" 3 ->GROB REPL {
  # 8h # 12h } "Packard" 3 ->GROB REPL 'AMASKE' STO
»
```

7 Datei

DATEI ist ein Programmpaket zur Verwaltung von Daten. Der Aufbau des Datensatzes ist hierbei beliebig, kann also frei definiert werden, ohne daß das Programm geändert werden muß.

7.1 HP28

Sämtliche Daten stehen in Listen, die alle zu einer übergeordneten Liste mit dem Namen VER zusammengefaßt sind. Die Liste enthält als erstes Element die Beschreibung der Datei, also eine Liste mit den Namen der einzelnen Positionen und dem Programm zum Aufbau des Displays. Die weiteren Elemente sind dann die Listen mit den eigentlichen Daten.

Die Funktionen des Paketes DATEI umfassen das Suchen nach einem bestimmten Kriterium und das Auflisten sämtlicher Daten. Man kann auch einzelne Elemente löschen. Die Bedienung des Programms ist am einfachsten an einem Beispiel zu erklären:

Es soll eine Telefondatei verwaltet werden. Die Daten, die gespeichert werden, sind: Name, Straße, Wohnort, Geburtstag und Telefon. Die Anordnung auf dem LCD soll so aussehen:

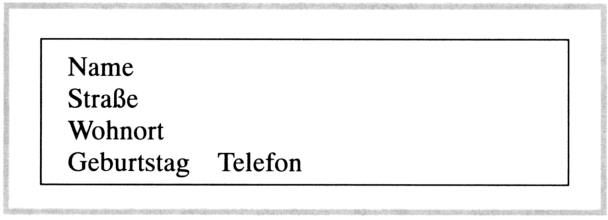


Abb. 7-1
Ausgabemaske (HP28)

7.1.1 Konfiguration

Vor dem ersten Aufruf muß eine Datei erst einmal initialisiert werden. Man erzeugt eine Liste, die ihrerseits eine Liste mit den Namen der Parameter sowie das Ausgabeprogramm enthält.

```
*******************************************************
* HP28: Leere Liste                                   *
*******************************************************

{
  « " " SWAP + + 4 DISP 3 DISP 2 DISP 1 DISP »
}
```

Das Hauptprogramm legt die einzelnen Datenelemente ("Strings") für den Aufbau des LCDs so auf dem Stack ab, daß der letzte Eintrag der Liste nach Ebene 1 kommt. Das Ausgabeprogramm, welches hinter den Aufbaudaten liegt, muß diese vom Stack entfernen und dabei die gewünschte Datenmaske im LCD erstellen. Die Liste VER wird für dieses Beispiel wie folgt eingegeben:

```
{
  { "Name" "Strasse" "Wohnort" "Geburtstag" "Telefon"
  « " " SWAP + + 4 DISP 3 DISP 2 DISP 1 DISP »
  }
}
```

7.1.2 SUCH

Will man etwas in der Datei suchen, ruft man SUCH auf und wird nach dem Suchkriterium gefragt.

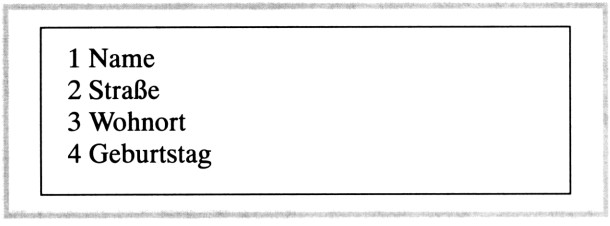


Abb. 7-2
Suchkriterien (HP28)

Man kann immer nur nach maximal vier Kriterien suchen, die ersten vier der Definitionsliste. Nach Eingabe einer der aufgeführten Zahlen wird man aufgefordert, den Suchbegriff einzugeben. Der Suchbegriff kann auch nur ein Teil des zu suchenden Wortes sein. Gibt man eine Zahl größer als vier ein, wird alles ausgegeben.

Wenn das Objekt gefunden wurde, wird das LCD entsprechend des Algorithmus im Definitionsteil der Datei aufgebaut. Durch Betätigung der Taste <D> wird der Datensatz gelöscht. Die Taste <E> beendet das Programm.

SUCH erzeugt eine ausführbare Datei mit dem Namen ALGO. Hierbei handelt es sich um das Ausgabeprogramm, welches beim Verlassen des Programmes SUCH wieder gelöscht wird.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM
AKTION	\USER\DATEN

Verwendete externe Variablen

Variable	Verzeichnis	Status
VER	\USER\DATEN	gesichert
ALGO	\USER\DATEN	gelöscht

SUCH

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm SUCH,                                 *
* durchsucht Datensatz nach Kriterien                 *
*******************************************************

« VER 1 GET DUP DUP SIZE GET 'ALGO' STO 1 OVER SIZE 1 -
  DUP 4 > 4 ROT
  IFTE
  FOR i
    i ->STR " " + OVER i GET + i DISP
  NEXT
  DO
  UNTIL
    KEY
  END
  STR-> SWAP OVER 5 <
```

```
  « CLLCD OVER GET "÷" + 1 PROGRAM SGET DATEN »
  IFT
  -> x such
  « VER 2 OVER SIZE
    FOR i
      DUP i GET x 4 > 1
      « DUP x GET such POS 1
        « DROP 0 »
        IFTE
      »
      IFTE
      « LIST-> DROP ALGO i AKTION »
      IFT
    NEXT
  »
  DROP CLMF 'ALGO' PURGE
»
```

7.1.3 NEU

Um Daten einzugeben, startet man NEU. Die Funktion fragt nacheinander jedes Datenelement ab, bis alle Daten eines Datensatzes eingegeben sind. Um weitere Daten einzugeben, muß NEU ein zweites Mal gestartet werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
VER	\USER\DATEN	gesichert

NEU

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm NEU,                                  *
* Nimmt neues Element in die Liste auf                *
*******************************************************

« CLLCD { } VER 1 GET 1 OVER SIZE 1 - PROGRAM
  FOR i
    DUP i GET "╪" + i SGET ROT SWAP + SWAP
  NEXT
  DATEN DROP 1 ->LIST VER SWAP + 'VER' STO CLMF
»
```

7.1.4 AKTION

AKTION ist ein Unterprogramm von SUCH und startet verschiedene Routinen, die die aktuelle Seite bearbeiten (z.B. löschen). Im Stapel wird der Index dieser Seite übergeben.

Verwendete externe Variablen

Variable	Verzeichnis	Status
VER	\USER\DATEN	gesichert

AKTION

Ebene 1	Ebene 1
x →	

```
*******************************************************
* HP28 Programm AKTION,                               *
* Bearbeitet gerade gelistete Seite                   *
*******************************************************

« -> i
  « DO
    UNTIL
      KEY
    END
    IF DUP "D" SAME THEN
      OVER DUP 1 i 1 - SUB SWAP i 1 + OVER SIZE SUB +
      'VER' STO DROP
    ELSE
```

```
      IF "E" SAME THEN
        DROP ABORT
      END
    END
  »
»
```

7.2 HP48

Für den HP48 wurde eine neue Datenbank entworfen. Diese ist wesentlich komfortabler als die des HP28.

Wo beim HP28 nur eine Datenbank möglich ist, sind beim HP48 mehrere möglich. Die Datenbank wird nun grafisch definiert.

In den Beispielen soll von einer Telefondatei ausgegangen werden. Die Einträge hierzu sind Nachname, Vorname, Straße, Ort und Telefonnummer.

7.2.1 SUCH

Mit SUCH können Sie in einer Datenbank Datensätze suchen, editieren oder löschen.

Als erstes werden Sie nach dem Kriterium gefragt, nach dem Sie suchen wollen. Durch Betätigung der Taste <Cursor ab> können Sie weitere Kriterien auswählen. In der obersten Zeile wird die aktuelle Datenbank angezeigt.

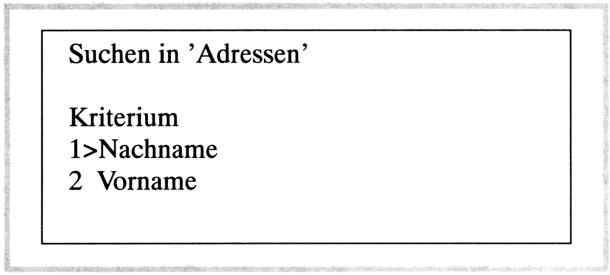


Abb. 7-3
Suchkriterien (HP48)

Nach Auswahl und Quittierung mit der Taste <ENTER> wird in der untersten Zeile nach dem Suchbegriff gefragt. Das Programm sucht auch nach Teilbegriffen, falls Sie

z.B. nur noch den Anfangsbuchstaben des Suchkriteriums wissen. Durch Eingabe eines Leer-Strings (nur die <ENTER>-Taste betätigen), können alle Datensätze nacheinander angezeigt werden.

Existiert der Begriff, wird eine Maske aufgebaut, die den Datensatz darstellt. Nun können unterschiedliche Aktionen gestartet werden.

Betätigen Sie die Taste <+>, so wird der nächste Datensatz gesucht, der das Suchkriterium erfüllt. Es wird vorwärts geblättert. Ist das Ende der Datenbank erreicht, wird wieder von vorne angefangen.

Mit der Taste <-> kann man rückwärts blättern. Am Anfang angelangt, geht die Suche am Ende der Datenbank weiter.

Mit <ENTER> wird der nächste Datensatz gesucht, und bei Erreichen des Datenbankendes das Programm verlassen.

Durch Eingabe von <E> wird SUCH sofort beendet.

Wenn Sie den gerade gezeigten Datensatz löschen wollen, betätigen Sie die Taste <DEL>. Aber Vorsicht, der Datensatz wird ohne Rückfrage gelöscht!

Ein Datensatz kann mit der <+/-> Taste editiert werden. Folgendes Menü erscheint auf der Anzeige.

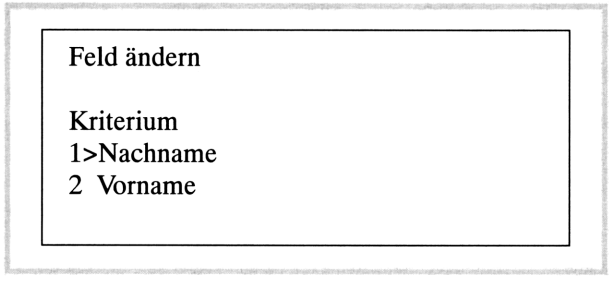


Abb. 7-4
Kriterium wählen, das editiert werden soll (HP48)

Nach Auswahl des Kriteriums wird der zu ändernde Eintrag dargestellt. Sie wollen z.B. 'Nachname' ändern.

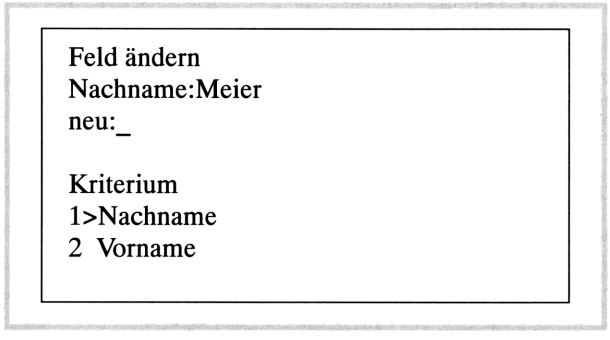


Abb. 7-5
Kriterium editieren (HP48)

Hinter 'neu:' wird der neue Nachname eingegeben. Anschließend gelangen Sie wieder in die Ausgabemaske.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM
ILIST	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ADB	\USER\DATEN	gesichert
Datenbankname	\USER\DATEN	gesichert

SUCH

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP48 Programm SUCH,                                 *
* durchsucht Datenbank nach Kriterien                 *
*******************************************************

« CLLCD "'" ADB + "'" + "Suchen in " OVER + 1 DISP
  "Kriterium" 3 DISP OBJ-> DUP RCL SIZE OVER 1 GET DUP
  SIZE
  -> nl lal il lil
  « { } 1 lil
    FOR i
      il i GET 2 GET +
    NEXT
    DUP PROGRAM 4 2 ILIST "Suchbegriff:" 7 SGET DATEN
    -> kl kr su
    « (2,0)
      WHILE
        DUP RE lal ≤ OVER RE 1 > AND
      REPEAT
        IF nl OVER RE GET DUP kr GET su POS su SIZE NOT
        OR THEN
          ERASE { # 0h # 0h } PVIEW 1 lil
          FOR j
            PICT il j GET LIST-> DROP ":" + 4 PICK j
            GET + 1 ->GROB REPL
          NEXT
          DO
            .05 WAIT
```

```
    UNTIL
      KEY
    END
    CASE
      DUP 52 SAME THEN
        DROP TEXT CLLCD "Feld ändern" 1 DISP
        "Kriterium" 5 DISP PROGRAM kl DUP 6 2
        ILIST SWAP OVER GET ":" + 3 DUPN DROP GET
        + 2 DISP "neu:" 3 SGET DATEN PUT nl 3
        PICK RE ROT PUT C->R DROP 1 - 0 R->C
      END
      SWAP DROP DUP 15 SAME THEN
        DROP2 (0,0)
      END
      DUP 95 SAME THEN
        DROP C->R DROP 1 R->C
      END
      DUP 85 SAME THEN
        DROP C->R DROP -1 R->C
      END
      DUP 51 SAME THEN
        DROP C->R DROP 0 R->C
      END
      54 SAME THEN
        'lal' 1 STO- RE nl RCL DUP 1 4 PICK 1 -
        SUB SWAP ROT 1 + OVER SIZE SUB + nl STO
        (1,0)
      END
      C->R DROP 1 - 0 R->C
    END
  ELSE
    DROP
  END
  IF DUP IM DUP THEN
    + DUP RE
    CASE
      DUP 2 < THEN
        - lal +
      END
      DUP lal > THEN
        - 2 +
      END
      DROP
    END
  ELSE
    DROP 1 +
  END
END
```

```
    »
  »
  DROP TEXT
»
```

7.2.2 NEU

Um Daten einzugeben, startet man NEU. Es wird nach jedem einzelnen Kriterium gefragt, bis alle Daten eines Datensatzes eingegeben sind. Zum Schluß werden die eingegebenen Daten so dargestellt, wie sie nacher auf der Anzeige erscheinen. Um weitere Daten einzugeben, muß NEU ein zweites Mal gestartet werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ADB	\USER\DATEN	gesichert
Datenbankname	\USER\DATEN	gesichert

NEU

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP48 Programm NEU,                                  *
* nimmt neues Element in die aktuelle Datenbank auf   *
*******************************************************

« ERASE CLLCD "'" ADB + OBJ-> { } OVER 1 GET 1 OVER
  SIZE PROGRAM
  FOR i
    PICT OVER i GET LIST-> DROP ":" + DUP i SGET 6 ROLL
    OVER + 6 ROLLD + 1 ->GROB REPL
  NEXT
  DROP DATEN 1 ->LIST STO+ { # 0h # 0h } PVIEW 7 FREEZE
»
```

7.2.3 AKTU

AKTU erzeugt eine Auswahl aller gespeicherten Datenbanken. Sie wählen hier die Datenbank aus, die bearbeitet werden soll.

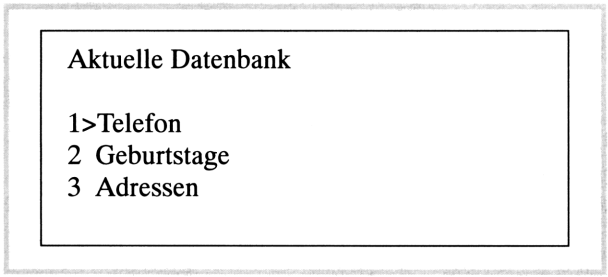


Abb. 7-6
Datenbank wählen (HP48)

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
ILIST	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ADB	\USER\DATEN	gesichert
DBLISTE	\USER\DATEN	gesichert

AKTU

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP48 Programm AKTU,                                 *
* setzt die aktuelle Datenbank                        *
*******************************************************

« CLLCD "Aktuelle Datenbank" 1 DISP DBLISTE DUP PROGRAM
  3 3 ILIST DATEN GET 'ADB' STO
»
```

7.2.4 DSORT

Das Programm DSORT sortiert die aktuelle Datenbank alphabetisch in aufsteigender Reihenfolge nach dem Quick-Sort-Verfahren. Das Sortierkriterium kann über die Pfeiltasten eingestellt werden.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
ILIST	\USER\PROGRAM
UDS	\USER\DATEN

Verwendete externe Variablen

Variable	Verzeichnis	Status
ADB	\USER\DATEN	gesichert
Datenbankname	\USER\DATEN	gesichert

DSORT

Ebene 1	**Ebene 1**
→	

Die in der Variablen 'ADB' gewählte Datenbank wird mit dem interaktiv eingegebenen Sortierkriterium alphabetisch sortiert.

```
**********************************************************
* HP48 Programm DSORT,                                   *
* sortiert aktuelle Datenbank nach einem Kriterium       *
**********************************************************

« CLLCD "'" ADB + "'" + "Sortiere " OVER + 1 DISP STR->
  DUP RCL "Sortierkriterium" 3 DISP { } 1 3 PICK 1 GET
  SIZE
  FOR i
    OVER 1 GET i GET 2 GET +
  NEXT
  PROGRAM 4 2 ILIST DATEN "Sortierung läuft..." 7 DISP
  SWAP 2 OVER SIZE
  IF DUP2 < THEN
    UDS
  ELSE
    DROP2
  END
  ROT STO DROP
»
```

7.2.5 UDS

Die Funktion UDS ist ein Unterprogramm zu DSORT. Hier wurde der Quick-Sort-Algorithmus rekursiv implementiert. Das Programm entspricht im wesentlichen der Funktion QSORT. Es wurde lediglich auf den vorhandenen Datensatz mit dem Sortierkriterium angepaßt.

UDS

Ebene 4	Ebene 3	Ebene 2	Ebene 1		Ebene 2	Ebene 1
z	{Liste$_1$}	x	y	→	z	{Liste$_2$}

Das Datenfeld in {Liste$_1$} wird nach dem Sortierkriterium z vom Datensatz x bis zum Datensatz y in alphabetischer Reihenfolge sortiert und in {Liste$_2$} abgelegt.

```
*******************************************************
* HP48 Funktion UDS,                                  *
* sortiert einen Datensatz nach einem Sortierkrite-   *
* rium mit dem Quick-Sort-Verfahren vom Element       *
* Startwert bis zum Element Endwert                   *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 : Sortierkriterium    : Real Number        *
*  Ebene 3 : Datensatz           : List               *
*  Ebene 2 : Startwert           : Real Number        *
*  Ebene 1 : Endwert             : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 2 : Sortierkriterium    : Real Number        *
*  Ebene 1 : Datensatz           : List               *
*******************************************************

« DUP2
  -> l r
  « 3 DUPN + 2 / IP GET 5 PICK GET
    -> p
    « DO
        ROT SWAP
        WHILE
          DUP2 GET 5 PICK GET p >
        REPEAT
          1 -
        END
        SWAP ROT
        WHILE
```

```
        DUP2 GET 5 PICK GET p <
      REPEAT
        1 +
      END
      ROT
      IF DUP2 ≤ THEN
        3 DUPN 3 PICK ROT GET PUT ROT 4 ROLL 4 PICK
        GET PUTI ROT 1 -
      END
    UNTIL
      DUP2 >
    END
  »
  -> i j
  « l j <
    « l j UDS »
    IFT
    i r <
    « i r UDS »
    IFT
  »
»
»
```

7.2.6 DEF

Zuerst einmal muß eine Datenbank definiert werden. Man benötigt dazu die Anzahl der Felder und die Position eines Eintrages auf dem Display.

Nach Aufruf von DEF wird nach dem Namen der neu anzulegenden Datenbak gefragt.

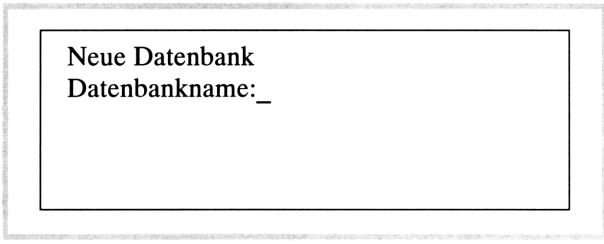


Abb. 7-7
Datenbank definieren (HP48)

Danach erscheint das GRAPH-Display. Bewegen Sie nun den Cursor mit den Cursortasten an die Stelle, wohin Sie einen Eintrag haben wollen und quittieren Sie mit <ENTER>. Die Position des Cursors markiert die linke, obere Ecke des Eintragstextes.

Drücken Sie danach <ATTN>, und Sie gelangen in das folgende Bild.

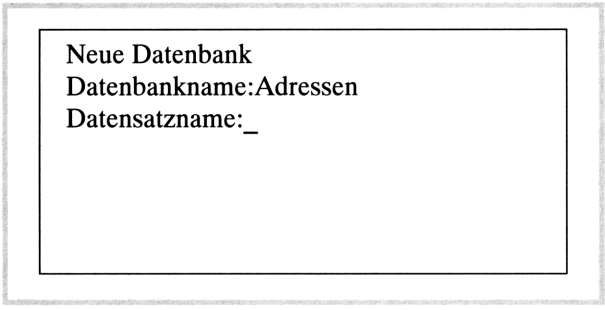


Abb. 7-8
Datensätze anlegen (HP48)

Geben Sie hier die Bezeichnung des Eintrages ein (z.B. 'Name'). Im Folgenden werden Sie dann gefragt, ob Sie das Programm beenden möchten, oder ob Sie einen weiteren Eintrag definieren wollen.

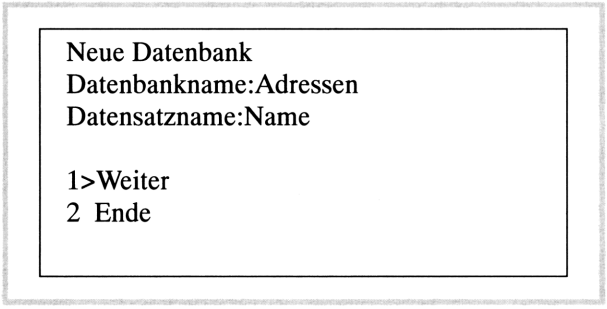


Abb. 7-9
Datensätze anlegen (HP48)

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM
ILIST	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
DBLISTE	\USER\DATEN	gesichert
Datenbankname	\USER\DATEN	gesichert

DEF

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP48 Programm DEF,                                  *
* definiert eine neue Datenbank                       *
*                                                     *
* Achtung: LF in String wurde im Listing durch ein    *
* • Zeichen ersetzt!                                  *
*******************************************************

« VARS CLLCD "Neue Datenbank" 1 DISP 'DBLISTE' DUP RCL
  PROGRAM
  WHILE
    "Datenbankname:" 2 SGET DUP2 POS
  REPEAT
    DROP
  END
  DATEN SWAP OVER + ROT STO ERASE { }
  DO
    DO
      GRAPH
    UNTIL
      DUP TYPE 1 SAME
    END
    WHILE
      OVER TYPE 5 ≠
    REPEAT
      SWAP DROP
    END
    C->PX TEXT PROGRAM "Datensatzname:" 3 SGET DUP2 ":"
    + PICT ROT ROT 1 ->GROB GOR 2 ->LIST 1 ->LIST + {
    "Weiter" "Ende" } 5 2 ILIST "•" 5 DISP DATEN
  UNTIL
    1 -
  END
  { # 0h # 0h } PVIEW 7 FREEZE 1 ->LIST "'" ROT + OBJ->
  STO ORDER
»
```

7.2.7 Konfiguration

Vor dem ersten Aufruf muß das System initialisiert werden. Speichern Sie unter dem Variablennamen DBLISTE eine leere Liste ab. In diese Liste trägt DEF eine zu definierende Datenbank ein.

```
*******************************************************
* HP48 Liste DBLISTE,                                 *
* verfügbare Datenbanken des Datenbankpakets          *
*******************************************************

{ "TELP" }


*******************************************************
* HP48 Liste TELP,                                    *
* Datenbankmaske für eine Telefondatei                *
*******************************************************

{
  {
    { { # 2h # 2h } "Nachname" }
    { { # 2h # Bh } "Vorname" }
    { { # 2h # 14h } "Strasse" }
    { { # 2h # 1Dh } "PLZ." }
    { { # 2Ch # 1Dh } "Ort" }
    { { # 2h # 26h } "Tel." }
    { { # 2h # 2Fh } "Geburtstag" }
  }
}
```

8 GELD

8.1 Tabellenkalkulation

Das Programm TAB ist eine Tabellenkalkulation mit einem festgelegten, einfachen Rahmen. Es handelt sich demnach nicht um eine frei programmierbare Tabellenkalkulation, sondern um ein Programm, daß die Einnahmen den Ausgaben eines Jahres gegenüberstellt. Die Tabelle besteht aus 12 Spalten für die Monate und aus 5 Zeilen für die Ausgaben pro Monat (also 5 Wochen pro Monat) und eine Zeile für die Einnahmen. Die Tabelle sähe auf einer Tabellenkalkulation etwa so aus:

	A	B	C	D	...	N
1						
2			Januar	Februar	...	Dezember
3						
4	Woche 1		526.44	59.25		0.00
5	Woche 2		781.73	285.00		0.00
6	Woche 3		200.00	799.88		0.00
7	Woche 4		815.15	143.05		0.00
8	Woche 5		110.29	0.00		0.00
9						
10	Ausgaben		ΣC4:C8	ΣD4:D8		ΣN4:N8
11						
12	Einnahmen		4229.32	4015.00		0.00
13						
14	Differenz		C12-C10	D12-D10		N12-N10
15						
16	Gesamt:		C14	ΣC14:D14		ΣC14:N14

Abb. 8-1 *Ausschnitt aus einer Tabellenkalkulation*

8.1.1 TAB

Als erstes wird der Monat eingegeben, den man bearbeiten will (1..12).

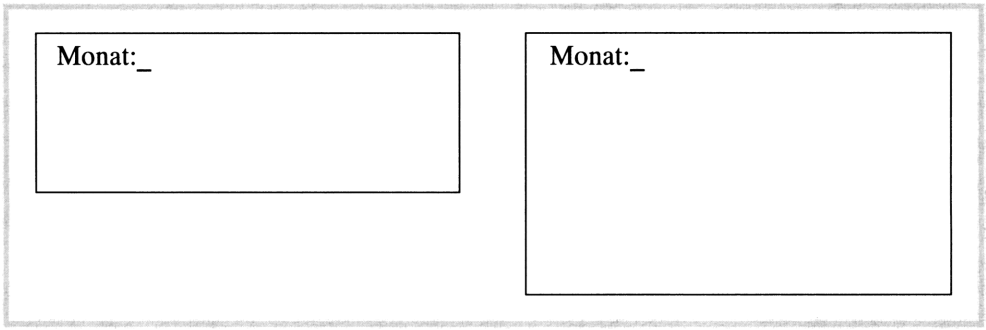


Abb. 8-2 *links HP28S, rechts HP48*

Als Beispiel wird hier der Monat Januar verwendet. Anschließend wird man aufgefordert, die Woche einzugeben.

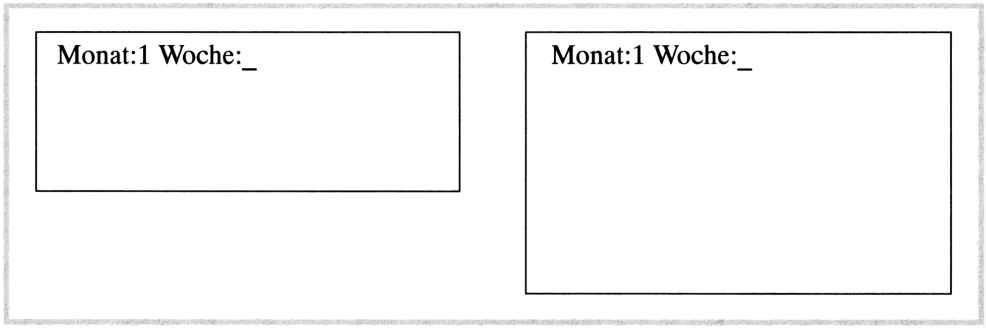


Abb. 8-3 *Eingabe der Woche*

Wir nehmen die vierte Woche. Dann sieht das LCD laut obiger Tabelle so aus:

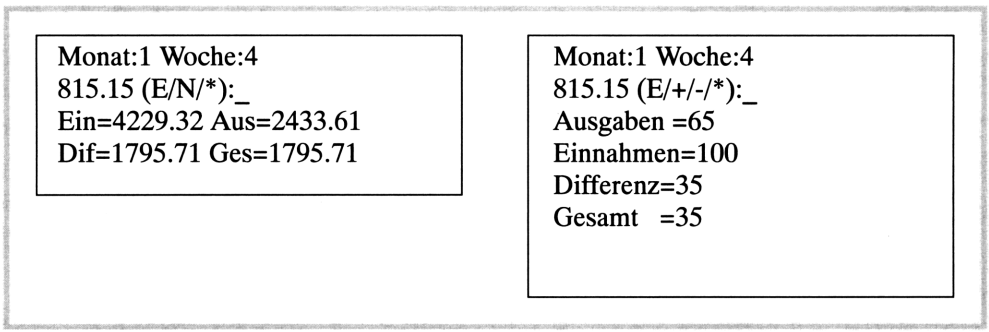


Abb. 8-4 *Gesamt-Anzeige*

Bedienung HP28

In Zeile 2 steht als erstes die Summe der Ausgaben in der ausgewählten Woche. Darunter stehen hinter 'Ein=' die Einnahmen und hinter 'Aus=' die bisherigen Gesamtausgaben des Monats. 'Dif=' in Zeile 4 steht vor der Differenz zwischen Einnahmen und Gesamtausgaben. Hinter 'Ges=' stehen alle Differenzen zusammenaddiert bis zum angezeigten Monat. In der zweiten Zeile nach '(E/N/):' befindet sich der Eingabecursor.*

Drückt man nur <ENTER>, blättert man zur nächsten Woche weiter (Nach Woche fünf kommt wieder Woche eins desselben Monats).

Gibt man aber eine Zahl ein, dann wird diese als Summe der Ausgaben der aktuellen Woche interpretiert und entsprechend eingetragen. Das Programm berechnet die neue Tabelle und blättert auch hier zur nächsten Woche weiter.

Die Eingabe 'E' beendet das Programm.

Mit 'N' kann eine neue Woche ausgewählt werden.

Nach Eingabe von '' wird der Betrag der Einnahmen und anschließend die Eingabe der darzustellenden Woche erwartet.*

Bedienung HP48

In Zeile 2 steht die Summe der Ausgaben der gewünschten Woche, gefolgt von Kürzeln der zugelassenen Kommandos. In Zeile 3 stehen die Gesamtausgaben des Monats gefolgt von den Gesamteinnahmen in Zeile 4. Zeile 5 zeigt die Differenz zwischen Einnahmen und Gesamtausgaben an. Hinter 'Gesamt' steht die Differenz aller Einnahmen und Ausgaben des laufenden Jahres, bis zum angezeigten Monat. In der zweiten Zeile nach '(E/+/-/):' befindet sich der Eingabecursor.*

Durch <ENTER>, wird zur nächsten Woche geblättert, bis die letzte Woche erreicht ist. Das Programm wird dann beendet.

Gibt man '+' ein, wird endlos durchgeblättert (Nach Woche fünf kommt wieder Woche eins des selben Monats). Mit Eingabe von '-' passiert das Gleiche rückwärts.

Durch Eingabe einer Zahl wird diese als Summe der Ausgaben der aktuellen Woche interpretiert und entsprechend eingetragen. Das Programm berechnet die neue Tabelle und blättert auch hier zur nächsten Woche weiter.

Ein 'E' beendet das Programm.

Nach Eingabe von '*' wird der Betrag der Einnahmen und anschließend die Eingabe der darzustellenden Woche erwartet.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
AKTUAL	\GELD\CALC
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\GELD\CALC	gesichert
AUS	\GELD\CALC	gesichert
EIN	\GELD\CALC	gesichert
GES	\GELD\CALC	gesichert

TAB

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm TAB,                                  *
* Tabellenkalkulation                                 *
*******************************************************

« CLEAR 2 CF CLLCD AKTUAL USER PROGRAM
  DO
    "Monat#" 1 SGET STR->
  UNTIL
    DUP 0 > OVER 13 < AND
  END
  -> m
  « DO
      USER PROGRAM
      DO
        "Monat" 58 CHR + m ->STR + " Woche" + 58 CHR +
        IF DEPTH 2 SAME THEN
          OVER ->STR + 1 DISP
```

```
      ELSE
        1 SGET STR->
      END
    UNTIL
      DUP DUP 0 > SWAP 6 < AND
    END
    HOME GELD CALC DUP ΣDAT SWAP m 2 ->LIST GET AUS m
    1 ->LIST GET EIN m 1 ->LIST GET DUP2 SWAP - GES m
    1 ->LIST GET SWAP "Dif=" SWAP ->STR + SWAP
    "  Ges=" SWAP ->STR + + 4 DISP "Ein=" SWAP ->STR
    + SWAP "  Aus=" SWAP ->STR + + 3 DISP ->STR
    " DM (E/N/*)∓" + USER PROGRAM 2 SGET HOME GELD
    CALC
    IF DUP "N" SAME THEN
      DROP2
    ELSE
      IF DUP "E" SAME THEN
        DROP2 2 SF
      ELSE
        IF DUP "*" SAME THEN
          "Ein∓" 2 USER PROGRAM SGET HOME GELD CALC
          STR->
          IF DUP 0 ≥ THEN
            m 1 ->LIST SWAP 'EIN' ROT ROT PUT
          ELSE
            DROP
          END
          AKTUAL DROP2
        ELSE
          IF DUP "" ≠ THEN
            STR-> OVER m 2 ->LIST SWAP 'ΣDAT' 3 ROLLD
            PUT AKTUAL
          ELSE
            DROP
          END
          1 +
          IF DUP 5 > THEN
            DROP 1
          END
        END
      END
    END
  UNTIL
    2 FS?C
  END
»
CLMF
»
```

```
*******************************************************
* HP48 Programm TAB,                                  *
* Tabellenkalkulation                                 *
*******************************************************

« RCLF 2 SF CLLCD AKTUAL USER PROGRAM
  DO
    "Monat≠" 1 SGET STR->
  UNTIL
    DUP 0 > OVER 13 < AND
  END
  -> m
  « 1
    DO
      USER PROGRAM
      DO
        "Monat:" m + " Woche:" +
        IF 2 FS?C THEN
          1 SGET STR-> SWAP DROP
        ELSE
          OVER + 1 DISP
        END
      UNTIL
        DUP DUP 0 > SWAP 6 < AND
      END
      HOME GELD CALC "Differenz=" EIN m GET
      "Einnahmen=" OVER + 4 DISP AUS m GET "Ausgaben ="
      OVER + 3 DISP - + 5 DISP "Gesamt    =" GES m GET +
      6 DISP ∑DAT OVER m 2 ->LIST GET " DM (E/+/-/*)≠"
      + USER PROGRAM 2 SGET HOME GELD CALC
      CASE
        DUP "+" SAME THEN
          DROP 1 +
          IF DUP 5 > THEN
            DROP 1
          END
        END
        DUP "-" SAME THEN
          DROP 1 -
          IF DUP 1 < THEN
            DROP 5
          END
        END
        DUP "E" SAME THEN
          DROP2 6
        END
        DUP "" SAME THEN
          DROP 1 +
```

```
        END
        DUP "*" SAME THEN
          'EIN' "Ein≠" 2 USER PROGRAM SGET HOME GELD
          CALC STR->
          IF DUP 0 ≥ THEN
            m SWAP PUT
          ELSE
            DROP2
          END
          DROP AKTUAL 2 SF
        END
        STR-> OVER m 2 ->LIST SWAP '∑DAT' 3 ROLLD PUT
        AKTUAL 1 +
        IF DUP 5 > THEN
          DROP 1
        END
      END
    UNTIL
      DUP 5 >
    END
    DROP
  »
  STOF
»
```

8.1.2 AKTUAL

AKTUAL ist ein Unterprogramm von TAB. Es aktualisiert die gesamte Tabelle, d.h. es berechnet die Spaltensumme und Gesamtdifferenz zwischen Einnahmen und Ausgaben neu.

Verwendete externe Variablen

Variable	Verzeichnis	Status
AUS	\GELD\CALC	gesichert
GES	\GELD\CALC	gesichert
SUM	\GELD\CALC	gelöscht

AKTUAL

Ebene 1	Ebene 1
→	

```
*****************************************************
* HP28/48 Programm AKTUAL,                          *
* Berechnet die Tabelle neu                         *
*****************************************************

« TOT DUP 'AUS' STO EIN SWAP - 'SUM' STO 0 1 12
  FOR n
    'SUM' n 1 ->LIST GET + DUP 'GES' n 1 ->LIST ROT PUT
  NEXT
  DROP 'SUM' PURGE
»
```

8.1.3 NEW

Vor der ersten Verwendung des Programmes TAB muß NEW aufgerufen werden. Dieses Programm legt alle nötigen Variablen an.

Verwendete externe Variablen

Variable	Verzeichnis	Status
ΣDAT	\GELD\CALC	gesichert
AUS	\GELD\CALC	gesichert
EIN	\GELD\CALC	gesichert
GES	\GELD\CALC	gesichert

NEW

Ebene 1	**Ebene 1**
→	

NEW Definiert Variablen für das Programm TAB.

```
*****************************************************
* HP28/48 Programm NEW,                             *
* legt Tabelle neu an                               *
*****************************************************

« VARS 12 IDN 0 * DUP {5 12} RDM 'ΣDAT' STO {12} RDM
  DUP 'AUS' STO DUP 'EIN' STO 'GES' STO ORDER
»
```

8.2 Kontostand-Anzeige

Dieses Programmpaket dient der Verwaltung eines Girokontos. Damit ist man jederzeit in der Lage, den aktuellen Kontostand zu ermitteln.

8.2.1 STAND

Die Funktion STAND gibt den aktuellen Kontostand in den Stack zurück.

Verwendete externe Variablen

Variable	Verzeichnis	Status
EIN	\GELD\KONTO	gesichert
AUS	\GELD\KONTO	gesichert
KTO	\GELD\KONTO	gesichert

STAND

Ebene 1		Ebene 1
	→	x

```
*******************************************************
* HP28/48 Funktion STAND,                             *
* gibt Kontostand in den Stack                        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Kontostand       : Real Number           *
*******************************************************

« EIN AUS - KTO + »
```

8.2.2 KONT

Nach Aufruf des Programms muß angegeben werden, ob auf das Konto Einzahlungen erfolgen sollen oder etwas davon abgehoben wird.

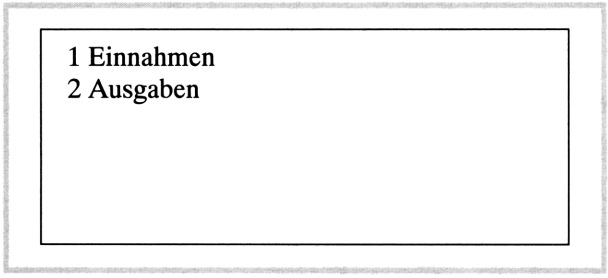


Abb. 8-5

Als nächstes gibt man den entsprechenden Betrag ein.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
EIN	\GELD\KONTO	gesichert
AUS	\GELD\KONTO	gesichert

KONT

Ebene 1	**Ebene 1**
→	

```
*****************************************************
***************** Kontoführungs-Paket *****************
*****************************************************
* HP28/48 Programm KONT,                            *
* Kontoführungsprogramm, registriert Einnahmen      *
* und Ausgaben                                      *
*****************************************************

« CLLCD "1 Einnahmen" 1 DISP "2 Ausgaben" 2 DISP
  WHILE
    DO
    UNTIL
      KEY
    END
    DUP DUP "1" < SWAP "2" > OR
  REPEAT
    DROP
  END
  USER PROGRAM "DM " 3 SGET STR-> GELD KONTO SWAP
  IF "2" SAME THEN
    'AUS' STO+
  ELSE
    'EIN' STO+
  END
»
```

8.2.3 BILANZ

Das Programm Bilanz dient der Gegenüberstellung der Einnahmen und Ausgaben am Ende eines Jahres. Gleichzeitig wird KTO (Kontostand zu Jahresbeginn) aktualisiert und EIN sowie AUS auf Null gesetzt.

Verwendete externe Variablen

Variable	Verzeichnis	Status
EIN	\GELD\KONTO	gesichert
AUS	\GELD\KONTO	gesichert
KTO	\GELD\KONTO	gesichert

BILANZ

Ebene 1	**Ebene 1**
→	

```
*****************************************************
* HP28/48 Programm BILANZ,                          *
* macht Jahresbilanz und speichert aktuellen        *
* Kontostand in KTO ab                              *
*****************************************************

« CLLCD STAND 'KTO' STO
  "Einnahmen" 58 CHR + EIN ->STR + 1 DISP 0 'EIN' STO
  "Ausgaben" 58 CHR + AUS ->STR + 2 DISP 0 'AUS' STO
»
```

8.2.4 Initialisierung

Vor dem ersten Aufruf müssen alle globalen Variablen initialisiert werden.

KTO: Kontostand am Anfang des Jahres
AUS: Gesamtausgaben
EIN: Gesamteinnahmen

Zu definierende Variablen

Variable	Verzeichnis	Inhalt
EIN	\GELD\KONTO	0
AUS	\GELD\KONTO	0
KTO	\GELD\KONTO	Kontostand

8.3 Passwortschutz

Um Daten in einem Computer zumindest im Ansatz vor neugierigen Blicken zu schützen, versieht man sie üblicherweise mit einem Passwort. Nur eine begrenzte Anzahl von Usern ist dann in der Lage, diese Daten zu nutzen. Leider gibt es aber auch Personen, die unbefugt in Datenbestände eindringen und großen Schaden anrichten können. Es wird auch nie ein System geben, das wirklich sicher ist.

Um einen einfachen Passwortschutz geht es auch in diesem Programm für den HP. Dieser Schutz kann jedoch von jedem versierten HP Benutzer umgangen werden. Gegen versehentliches Bedienen ist er aber allemal gut genug.

8.3.1 Passwortprogramm

Das Passwort-Programm xxxx sollte den gleichen Namen haben, wie das zu schützende Menü. xxxx ist also durch die gewünschte Bezeichnung zu ersetzen. Nach dem Aufruf wird, wenn der Passwortschutz gesetzt ist, nach dem Passwort gefragt, das in der Variablen PAS im Menü xxxx steht.

Wenn in der Variablen PAS2 der leere String "" steht, ist der Schutz aktiv. Wenn aber der leere String "" in PAS steht, dann ist der Schutz inaktiv (das Passwort steht hier in PAS2).

In unserem Beispiel hat das zu schützende Menü den Namen GELD. Also wird im User-Menü das Passwortprogramm GELD definiert. GELD bezieht sich dann auf das gleichnamige Menue im HOME-Verzeichnis.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
PAS	GELD	gesichert
PAS2	GELD	gesichert

GELD

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm Geld,                                 *
* schützt ein Menue mit einem Passwort                *
*                                                     *
* xxxx ist ein Menuename, der unter HOME zu erreichen *
* ist                                                 *
*******************************************************

« HOME GELD
  IF PAS "" + THEN
    CLLCD USER PROGRAM "Passwort+" 1 SGET HOME GELD
    IF PAS + THEN
      USER
    END
```

```
  END
  CLMF
»

*******************************************************
* HP48 Programm Geld,                                 *
* schützt ein Menue mit einem Passwort                *
*                                                     *
* xxxx ist ein Menuename, der unter HOME zu erreichen *
* ist                                                 *
*******************************************************

« HOME GELD
  IF PAS "" ≠ THEN
    CLLCD USER PROGRAM "Passwort+" 1 SGET HOME GELD
    IF PAS ≠ THEN
      USER
    END
  END
»
```

8.3.2 EXIT

Durch Aufruf von EXIT in xxxx wird der Passwortschutz nicht gesetzt, und man gelangt wieder ins USER Menü. Startet man jetzt erneut xxxx, so verhält sich das Programm wie ein normales Menü.

Verwendete externe Variablen

Variable	Verzeichnis	Status
PAS	GELD	gesichert
PAS2	GELD	gesichert

EXIT

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28/48 Programm EXIT,                              *
* kehrt aus dem geschützten Menue zurück              *
* der Passwortschutz bleibt aufgehoben                *
*******************************************************

« IF PAS "" ≠ THEN
    PAS 'PAS2' STO
  END
  "" 'PAS' STO USER
»
```

8.3.3 SAVE

Nach Aufruf von SAVE wird dagegen der Passwortschutz gesetzt, und man gelangt ebenfalls ins USER Menü.

Verwendete externe Variablen

Variable	Verzeichnis	Status
PAS	GELD	gesichert
PAS2	GELD	gesichert

SAVE

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28/48 Programm SAVE,                              *
* kehrt aus dem geschützten Menue zurück              *
* der Passwortschutz  wird gesetzt                    *
*******************************************************

« IF PAS "" SAME THEN
    PAS2 'PAS' STO
  END
  "" 'PAS2' STO USER
»
```

8.3.4 Initialisierung

Bevor man das Programm das erste Mal benutzt, müssen zwei Variablen im zu schützenden Menue deklariert werden: PAS und PAS2. Die Variable PAS enthält den leeren String ”“ und PAS2 das Passwort. Das Menü ist jetzt nicht geschützt und kann mit den Befehlen EXIT und SAVE verlassen werden.

9 SPIELE

Ein Rechner, insbesondere ein HP-Taschenrechner, kann nicht nur Berechnungen ausführen, er läßt sich auch für Spiele nutzen. Die folgenden Spiele sind hervorragend geeignet für Vorlesungen, Seminare, Besprechungen, Qualitätszirkel..., da sie keinen Lärm verursachen.

9.1 FIRE2

FIRE2 ist ein Geschicklichkeitsspiel, bei dem es auf Reaktionsvermögen und Geschwindigkeit ankommt. Ich könnte Ihnen nun erzählen, daß Sie Kapitän eines Raumkreuzers seien, und die Aufgabe haben ...

Es ist aber viel banaler, Sie sitzen also gelangweilt vor ihrem HP (und würden viel lieber einen Kaffee trinken) und starten FIRE2. Sie haben die Aufgabe, Zahlen aus einer nach links wandernden Zahlenreihe zu entfernen, ehe die Zahlenreihe zu groß wird.

Mit der <SPACE> Taste (HP28), oder der <0> Taste (HP48) können Sie die Zahl ganz links vor dem Doppelpunkt durchtasten. Stimmt die Zahl mit einer in der Zeichenkette überein, quittieren Sie dies mit der <+> Taste, die Zahl verschwindet aus der Zahlenreihe. Sollte die Zahl mehrfach vorkommen, so drücken sie die Taste so oft, wie die Zahl vorhanden ist. Sie haben wie üblich drei Versuche, ein Durchbrechen der Zahlenreihe zu verhindern, andernfalls wäre das Spiel beendet. Nach der Meldung 'THE END' wird das Programm mit einer beliebigen Taste (außer <ATTN>) korrekt verlassen. Sollten Sie jedoch versuchen, Zahlen zu entfernen, die nicht in der Zeichenkette vorhanden sind, so „belohnt" Sie das Spiel mit einer höheren Geschwindigkeit. Aber trösten Sie sich, es wird auch ohne „Belohnungen" schneller.

Sollte Ihnen die Spielidee bekannt vorkommen, stimmt, es ist eine weitere Adaption eines Taschenrechnerspiels.

Damit das Spiel nicht so langweilig wird, sorgen die HPs noch für ein paar Überraschungen. Wie Sie sicher schon bei Anwendungen bemerkt haben, legt der HP dann und wann mal eine kleine Pause ein, so auch hier. Denken Sie bitte daran: Der HP übernimmt trotzdem die Zeichen von der Tastatur und legt sie im Tastaturbuffer ab. Er bearbeitet den Buffer dann sofort nach Wiederbeginn. Es mag am Anfang leicht störend wirken, aber man gewöhnt sich daran.

Das Programm besteht aus zwei Dateien. Das Hauptprogramm FIRE2 beinhaltet den eigentlichen Programmcode, die Datei SCORE Programmdaten. Die Datei SCORE muß vor dem Start des Programms schon vorhanden sein. Die Liste SCORE beinhaltet Variablen mit folgender Bedeutung:

Element 1: Name der Person mit der Highscore
Element 2: Highscore
Element 3: Der Punktestand des letzten Spielers

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

Verwendete externe Variablen

Variable	Verzeichnis	Status
SCORE	\USER\SPIELE	gesichert

FIRE2

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm FIRE2,                                *
* Spiel                                               *
*******************************************************

« 'SCORE' 3 0 PUT 3 1
  FOR n
    CLLCD "===" 1 n SUB "  ship(s)" + 4 DISP 140 14
    "        " "0"
    WHILE
      IF KEY THEN
        IF DUP " " SAME THEN
          DROP NUM 1 + DUP 57 > 10 * - CHR DUP
```

```
      END
      IF "+" SAME THEN
        DUP2 POS
        IF DUP THEN
          'SCORE' 3 DUP2 GET 4 PICK + PUT " " 4 ROLL
          + DUP2 1 ROT SUB SWAP ROT 2 + 9 SUB + SWAP
        ELSE
          DROP ROT DROP ROT DUP 10 MOD - ROT ROT 1
          ROT ROT
        END
      END
    END
    DUP2 58 CHR + SWAP + 2 DISP
    IF ROT 1 - DUP 4 ROLLD DUP NOT THEN
      DROP ROT DROP ROT 2 - DUP 4 ROLLD 10 / IP
      DUP 4 ≥ SWAP 4
      IFTE
      SWAP ROT RAND 10 * IP ->STR + DUP 2 9 SUB ROT
      ROT NUM 32 SAME
    END
  REPEAT
  END
  KEY CLEAR
-1 STEP
SCORE LIST-> DROP ROT DROP CLLCD "Score " 58 CHR +
" " + OVER ->STR + 2 DISP
IF < THEN
  "* * New high score * *" 1 DISP 'SCORE' 3 GETI ROT
  ROT PROGRAM "Enter name ∓ " 4 SGET SPIELE PUTI ROT
  PUT
ELSE
  "  * * Your score * *" 1 DISP 4 WAIT
END
CLLCD "    * * THE END * *" 2 DISP
DO
  DO
  UNTIL
    KEY
  END
UNTIL
  "ENTER" SAME
END
CLMF
»
```

```
*******************************************************
* HP48 Programm FIRE2,                                 *
* Spiel                                                *
*******************************************************

« CLLCD "    * * Fire2 * *" 1 DISP 'SCORE' 3 0 PUT 3 1
  FOR n
    "===" 1 n SUB "  ship(s)" + 5 DISP 140 14
    "        " "0"
    WHILE
      IF KEY THEN
        IF DUP 92 SAME THEN
          DROP NUM 1 + DUP 57 > 10 * - CHR DUP
        END
        IF 95 SAME THEN
          DUP2 POS
          IF DUP THEN
            'SCORE' 3 DUP2 GET 4 PICK + PUT " " 4 ROLL
            + DUP2 1 ROT SUB SWAP ROT 2 + 9 SUB + SWAP
          ELSE
            DROP ROT DROP ROT DUP 10 MOD - ROT ROT 1
            ROT ROT
          END
        END
      END
      DUP2 ":" + SWAP + 3 DISP
      IF ROT 1 - DUP 4 ROLLD DUP NOT THEN
        DROP ROT DROP ROT 2 - DUP 4 ROLLD 10 / IP
        DUP 4 ≥
        SWAP
        4
        IFTE
        SWAP ROT RAND 10 * IP ->STR + DUP 2 9 SUB ROT
        ROT NUM 32 SAME
      END
    REPEAT
    END
    KEY CLEAR
  -1 STEP
  SCORE LIST-> DROP ROT DROP CLLCD "Score : " OVER
  ->STR + 3 DISP
  IF < THEN
    "* * New high score * *" 1 DISP 'SCORE' 3 GETI ROT
    ROT PROGRAM "Enter name : " 5 SGET SPIELE PUTI ROT
    PUT
  ELSE
    "  * * Your score * *" 1 DISP
  END
  "   * * THE END * *" 7 DISP 3 FREEZE
»
```

9.2 MASTERMINDS

Dies ist eine Adaption eines ähnlich lautenden Spiels. Vielen dürfte es bekannt sein. Trotzdem wird hier eine kleine Bedienungsanleitung geliefert und auf Besonderheiten des Programms aufmerksam gemacht.

MASTERMINDS ist ein Logik-Spiel, bei dem es im wesentlichen auf Kombinationsvermögen und nicht, wie bei FIRE2, auf Geschwindigkeit ankommt.

Ein Gegenspieler (in diesem Fall der Taschenrechner) generiert eine Zufallszahlenreihe, die Ihnen nicht bekannt ist (Das Original verwendet statt Zahlen verschiedene Farben, die aus verständlichen Gründen nicht dargestellt werden können). Ihre Aufgabe besteht nun darin, diese Zahlenreihe herauszufinden. Dazu unterstützt Sie der Rechner mit einer Aussage über die Anzahl der Ziffern, die an der richtigen Stelle in der Zahlenreihe stehen, und der Anzahl der Ziffern, die in der Zahlenreihe vorhanden sind, aber an der falschen Stelle stehen. Durch verschiedene Eingaben und deren Verifizierung läßt sich so durch Kombination der Ergebnisse mit zunehmender Anzahl der Versuche eine Aussage über die richtige Zeichenkette vornehmen. Auf Grund der Größe des Displays wird immer nur die letzte Kombination mit der Antwort auf dem LCD dargestellt. Daher ist es unabdingbar, sich die vorangegangenen Versuche auf einem Zettel zu notieren. Wahlweise bietet sich hierzu die Protokollierung auf einem Drucker an. Im Gegensatz zum Original wird jede Ziffer in der Zeichenkette nur einmal verwendet!

Programmbedienung

Zuerst fragt Sie der HP, ob die Daten auf dem Drucker protokolliert werden sollen (jede andere als die <J> Taste wird als „Nein“ interpretiert). Sämtliche Abfragen müssen nun mit der <ENTER> Taste quittiert werden. Mit der Abfrage der ”Stellen:“ wird nach der Länge der zu generierenden Zeichenkette gefragt. Die Länge ist auf neun Ziffern begrenzt. Eine Fehleingabe wird nicht abgefangen. Mit der Abfrage ”Farben:“ wird nach der Anzahl der verschiedenen Symbole gefragt. Die Anzahl der Farben ist ebenfalls auf neun Ziffern begrenzt. Eine Fehleingabe wird nicht abgefangen. Ist die Anzahl der Stellen gleich der Anzahl der Farben, tritt jede Ziffer genau einmal auf. Ist die Anzahl der Farben größer als die Anzahl der Stellen, werden einige Ziffern aus dem Zeichenvorrat nicht verwendet. Sollte die Anzahl der Ziffern kleiner als die Anzahl der Stellen sein, wird die Eingabe der Stellen und Farben wiederholt. Als Zeichenvorrat werden die Ziffern ’1’ bis zur gewählten Anzahl der Farben verwendet.

Der HP fordert Sie nun zur Eingabe einer Zeichenkette auf. Die Abkürzung ’Kom.’ steht für Kombination. Geben Sie nun eine Zeichenkette ein. Sollte die Zeichenkette länger oder kürzer als die geforderte Länge sein, muß die Eingabe wiederholt werden. Der HP wertet nun die Zeichenkette aus und schreibt das Ergebnis seiner Auswertung in die Zeile eins des Displays (und optional auf den Drucker). Der Ausdruck der Auswertung hat folgendes Format:

Zuerst erfolgt eine Wiederholung der letzen Eingabe, der zwei Leerzeichen folgen. Die nachfolgende zweistellige Zahl gibt Aufschluß über die Anzahl der Farben an der richtigen und der Anzahl der Farben an der falschen Stelle. Die erste Ziffer gibt die Anzahl der Farben an der richtigen Stelle, die zweite Ziffer die Anzahl der Farben an der falschen Stelle an. Sollte die Kombination korrekt sein, wird nur die korrekte Kombination ausgegeben und das Programm beendet.

Als Eingabeparameter erweisen sich vier Stellen bei sechs Farben als sinnvoll. Die Anzahl der Versuche bis zur Lösung liegt dann in der Regel zwischen vier und sechs.

Beispiel:

Anzahl der Stellen: 2
Anzahl der Farben: 4

Vom Rechner ermittelte unbekannte Kombination: 43

Eingabe1: 12 Resultat: 12 00
Eingabe2: 34 Resultat: 34 02
Eingabe3: 43 Resultat: 43 Ende

Das Resultat der Eingabe1 bedeutet, daß die Ziffern 1 und 2 in der Zeichenkette nicht vorhanden sind. Da aber der Zeichenvorrat nur aus den Ziffern 1, 2, 3 und 4 besteht, können in der Zeichenkette nur die Ziffer 3 und 4 vorkommen, die Reihenfolge ist aber noch unbekannt. Ein Versuch bei der Eingabe2 zeigt, daß die Ziffern in der Zeichenkette enthalten sind (was zu erwarten war), aber an der falschen Position stehen. Da nur eine Umkehrung der Ziffern möglich ist, muß bei der Eingabe3 zwangsläufig das richtige Ergebnis herauskommen.

Verwendete selbstdefinierte Unterprogramme:

Name	Verzeichnis
SGET	\USER\PROGRAM

MASTERMINDS

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP28 Programm MASTERMINDS,                          *
* Spiel                                               *
*******************************************************

« PROGRAM CLLCD "   ***Masterminds***" 1 DISP
  "-------------" 2 DISP
  "Protokoll Drucker (J/N)" 3 DISP
  DO
  UNTIL
    KEY
  END
  WHILE
    "Stellen ≠ " 3 SGET STR-> "Farben  ≠ " 4 SGET STR->
    DUP2 >
  REPEAT
    DROP2
  END
  { } DUP 1 4 ROLL
  FOR c
    c +
  NEXT
  1 4 PICK
  START
    DUP SIZE RAND * IP 1 + DUP2 GET 4 ROLL + ROT ROT
    DUP2 1 - 1 SWAP SUB ROT ROT 1 + OVER SIZE SUB +
  NEXT
  DROP CLLCD
  -> p a k
  « DO
      a ->STR " Stellen eingeben " + 58 CHR + 2 DISP
      WHILE
        "Kom.= " 3 SGET DUP SIZE a ≠
      REPEAT
        DROP
      END
      0 1 a
      FOR c
        1 a
        FOR b
          OVER b DUP SUB STR-> k c GET SAME
          « 1 + b c ≠ .9 * - »
          IFT
        NEXT
      NEXT
    UNTIL
      IF DUP a SAME THEN
        DROP CLLCD 1
```

```
      ELSE
        "  " OVER IP ->STR + SWAP FP 10 * ->STR + + 0
      END
      SWAP
      p "J" SAME
      « PR1 »
      IFT
      1 DISP
    END
  »
  SPIELE
»

*******************************************************
* HP48 Programm MASTERMINDS,                          *
* Spiel                                               *
*******************************************************

« PROGRAM CLLCD "  ***Masterminds***
  ————————————

  Druckerprotokoll (J/N)" 1 DISP SKEY
  WHILE
    "Stellen : " 4 SGET STR-> "Farben  : " 5 SGET STR->
    DUP2 >
  REPEAT
    DROP2
  END
  { } DUP 1 4 ROLL
  FOR c
    c +
  NEXT
  1 4 PICK
  START
    DUP SIZE RAND * IP 1 + DUP2 GET 4 ROLL + ROT ROT
    DUP2 1 - 1 SWAP SUB ROT ROT 1 + OVER SIZE SUB +
  NEXT
  DROP "" 1 DISP
  -> p a k
  « DO
      a " Stellen eingeben :" + 4 DISP
      WHILE
        "Kom.= " 5 SGET DUP SIZE a ≠
      REPEAT
        DROP
      END
      0 1 a
      FOR c
```

```
        1 a
        FOR b
          OVER b DUP SUB STR-> k c GET SAME
          « 1 + b c + .9 * - »
          IFT
        NEXT
      NEXT
    UNTIL
      IF DUP a SAME THEN
        DROP CLLCD 1
      ELSE
        "  " OVER IP ->STR + SWAP FP 10 * ->STR + + 0
      END
      SWAP p "J" SAME
      « PR1 »
      IFT
      1 DISP
    END
  »
  SPIELE 1 FREEZE
»
```

9.3 LSORT

LSORT ist ein weiteres Spiel, bei dem es im wesentlichen auf das logische Denkvermögen ankommt. Bei diesem Spiel müssen Sie eine Zeichenkette alphabetisch sortieren. Dazu gibt es ein spezielles Sortierverfahren.

Programmbeschreibung

Zuerst fordert Sie das Programm zur Eingabe der Buchstabenanzahl in der Zeichenkette auf. Gerade Zahlen sind nicht erlaubt, um eine lösbare Zeichenkette gewährleisten zu können. Die Zeichenkette muß aus mindestens 3 Zeichen und darf maximal aus 21 Zeichen bestehen. Falsche Eingaben werden zurückgewiesen. Unter dem Programmtitel wird die unsortierte Zeichenkette dargestellt, die es zu sortieren gilt. Mit der Eingabe "Stelle :" können Sie die Zeichenkette modifizieren.

Gehen wir von der Annahme aus, daß die fünfstellige Zeichenkette "BECDA" modifiziert werden soll. Geben Sie auf die Anfrage "Stelle :" die Zahl 3 ein, bedeutet dies, daß die Stelle, auf die die Zahl zeigt (in diesem Fall auf das "C"), an die erste Stelle der Zeichenkette verschoben wird. Das Zeichen, das hinter der Stelle steht (in diesem Fall

das „D"), wird an das Ende der Zeichenkette verschoben. Daraus ergibt sich nach der Modifikation die Zeichenkette „CBEAD". Um die Sache zu erschweren, darf als Stelle weder Null noch die letzte Stelle eingegeben werden.

Durch die Eingabe der richtigen Stellen kann die Zeichenkette sortiert werden. Es gibt immer eine Lösung. Sollten Sie den Lösungsweg gefunden haben, läßt sich auch eine Zeichenkette mit 21 Buchstaben problemlos sortieren.

Dabei sollte angemerkt werden, daß das Spiel seinen Reiz verliert, wenn ein Lösungsweg gefunden wurde. Aus diesem Grund wird auch kein Beispiel angegeben, da dieses einen Hinweis auf den Lösungsweg geben könnte.

Haben Sie die Zeichenkette erfolgreich sortiert, wird die Anzahl der Versuche angegeben. Das Programm läßt sich durch Drücken der <J> Taste wiederholen, eine andere Taste bricht das Programm ab.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

LSORT

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm LSORT,                                *
* Spiel                                               *
*******************************************************

« DO
    CLLCD "   ** Letter-Sort **" 1 DISP
    WHILE
      "Stellen ≠ " 3 PROGRAM SGET SPIELE STR-> DUP 22 >
      OVER 2 ≤ OR OVER 2 MOD NOT OR
    REPEAT
      DROP
    END
    "" 3 DISP
    -> n
    « 0 "" 65 n 64 +
      FOR i
        i CHR +
      NEXT
      "" OVER 1 n
      START
```

```
        DUP SIZE RAND * 1 + IP DUP2 DUP SUB 4 ROLL +
        ROT ROT DUP2 1 - 1 SWAP SUB ROT ROT 1 + OVER
        SIZE SUB +
      NEXT
      DROP
      DO
        ROT 1 + ROT ROT DUP 2 DISP
        WHILE
          "Stelle ≠ " 4 PROGRAM SGET SPIELE STR-> DUP n
          1 - > OVER 0 ≤ OR
        REPEAT
          DROP
        END
        DUP2 1 SWAP SUB DUP SIZE DUP2 DUP SUB ROT ROT 1
        - 1 SWAP SUB + ROT ROT 1 + n SUB DUP 2 OVER
        SIZE SUB SWAP 1 DUP SUB + +
      UNTIL
        DUP2 SAME
      END
      DROP 2 DISP ->STR "Versuche " 58 CHR + " " + SWAP
      + 3 DISP
    »
  UNTIL
    "Weiter (J) ?" 4 DISP
    DO
    UNTIL
      KEY
    END
    "J" ≠
  END
  CLMF
»


*******************************************************
* HP48 Programm LSORT,                                *
* Spiel                                               *
*******************************************************

« PROGRAM
  DO
    CLLCD "  ** Letter-Sort **" 1 DISP
    WHILE
      "Stellen : " 3 SGET STR-> DUP 22 > OVER 2 ≤ OR
      OVER 2 MOD NOT OR
    REPEAT
      DROP
    END
```

```
    -> n
    « 0 "" 65 n 64 +
      FOR i
        i CHR +
      NEXT
      "" OVER 1 n
      START
        DUP SIZE RAND * 1 + IP DUP2 DUP SUB 4 ROLL +
        ROT ROT DUP2 1 - 1 SWAP SUB ROT ROT 1 + OVER
        SIZE SUB +
      NEXT
      DROP
      DO
        ROT 1 + ROT ROT DUP 3 DISP
        WHILE
          "Stelle : " 5 SGET STR-> DUP n 1 - > OVER 0 ≤
          OR
        REPEAT
          DROP
        END
        DUP2 1 SWAP SUB DUP SIZE DUP2 DUP SUB ROT ROT 1
        - 1 SWAP SUB + ROT ROT 1 + n SUB DUP 2 OVER
        SIZE SUB SWAP 1 DUP SUB + +
      UNTIL
        DUP2 SAME
      END
      DROP 3 DISP "Versuche : " SWAP + 5 DISP
    »
  UNTIL
    "Weiter (J) ?" 7 DISP SKEY "J" ≠
  END
  SPIELE
»
```

10 PROGRAM

10.1 SGET

Die Funktion SGET ist ein universelles Programm zur Eingabe während des Programmablaufs. So können Strings, aber auch Zahlen zur Zeit der Programmausführung angefordert werden. Das Programm wird für die Eingabe nicht verlassen. Es entspricht im wesentlichen dem INPUT-Befehl unter BASIC. Die Funktion erfordert zwei Eingabeparameter. In Ebene 2 muß ein String mit einer Eingabeaufforderung (z.B. "Zahl ?") stehen. Da bei Eingabeaufforderungen häufig ein Doppelpunkt verwendet wird, und dieser beim HP28 nicht über die Tastatur erreichbar ist (nur über das Konstrukt 58 CHR), kann in der Eingabeaufforderung an Stelle des Doppelpunktes auch ein "+"-Zeichen stehen. Die Funktion wandelt das erste "+"-Zeichen im String in einen Doppelpunkt um. In der Ebene 1 muß die Nummer der Zeile stehen, in der die Eingabe erfolgen soll. Wird eine Zahl größer als vier (HP28), bzw. sieben (HP48) eingegeben, so wird der Displayinhalt um eine Zeile nach oben geschoben, und die unterste Displayzeile verwendet. Als Rückgabewert wird die eingegebene Zeichenkette als String in Ebene 1 zurückgegeben. Zur Weiterverarbeitung als Zahl, natürlich nur bei einer Zahleneingabe, kann der String mit dem Befehl STR→ in eine Zahl umgewandelt werden.

Bedeutung der Tasten während der Eingabe

HP28

Während der Eingabe über die Tastatur kann nur die Hauptbedeutung der Taste verwendet werden, die "^"-Taste (<SHIFT> <>) kann zum Beispiel nicht direkt aufgerufen werden. Weiter sind einige Tasten anders belegt.*

Besondere Funktionstasten

– *Taste <ENTER> :*
  Beendet die Eingabe.

- *Taste ← :*
  Löscht das letzte eingegebene Zeichen.
- *Taste <DROP> :*
  Löscht alle eingegebenen Zeichen.
- *Taste <EEX> :*
  Gibt ein E für die Eingabe von Zahlen in Exponentialdarstellung aus.
- *Taste <LC> :*
  Schaltet zwischen Groß- und Kleinschrift und umgekehrt um. Bei Kleinschrift ändert sich der Cursor in ein ". "-Zeichen.
- *Taste <SHIFT> (rote Taste) :*
  Ermöglicht die Eingabe eines ASCII-Zeichens (vgl. HP28C/S - Referenzhandbuch; Seite 318, 319).
- *Tastenfolge <SHIFT> und dreistellige Dezimalzahl.*
  Beispiel: SHIFT 146 ist das "«"-Zeichen (vgl. ALT 174 bei PC's).
  So kann zum Beispiel das "^"-Zeichen für Funktionen erzeugt werden.

HP48

Beim HP48 gilt bei den Tasten mit zusätzlicher alphanumerischer Belegung der alphanumerische Aufdruck, während Zahlen und Rechenzeichen ihre Hauptfunktion behalten.

Sondertasten

- *Taste <ENTER> :*
  Beendet die Eingabe.
- *Taste <- >*
  Löscht das letzte eingegebene Zeichen.
- *Taste <DEL> :*
  Löscht alle eingegebenen Zeichen.
- *Taste <α> :*
  Schaltet zwischen Groß- und Kleinschrift und umgekehrt um. Bei Kleinschrift ändert sich der Cursor in ein ". "-Zeichen.
- *rote Taste :*
  Ermöglicht die Eingabe eines ASCII-Zeichens
  Tastenfolge rote Taste und dreistellige Dezimalzahl.
  Beispiel: rote Taste 171 ist das "«"-Zeichen (vgl. ALT 174 bei PC's).
- *blaue Taste :*
  Gibt ein "^"-Zeichen aus.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis	2	4
SKEY	\USER\PROGRAM		X

SGET

Ebene 2	Ebene 1		Ebene 1
"String$_1$"	n	→	"String$_2$"

"String$_1$" wird in Zeile n des Displays geschrieben und zur Texteingabe aufgefordert. Die eingegebene Zeichenkette steht in "String$_2$".

```
*******************************************************
* HP28 Funktion SGET,                                 *
* liest String von Tastatur ein                       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Ausgabestring       : String             *
*  Ebene 1 : Displayzeile        : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Eingegebener String : String             *
*                                                     *
* Spezielle Funktionstasten :                         *
*   DROP             : Clear String                   *
*   SHIFT + 3 Zahlen : Druckt ASCII-Code der 3 Zahlen *
*                                                     *
* Dekodierung des Ausgabestring :                     *
*   Das erste "+" Zeichen wird durch ein ":" Zeichen  *
*   ersetzt.                                          *
*                                                     *
* Dekodierung der Displayzeile :                      *
*   Bei Displayzeilen > 4 wird der LCD-Inhalt eine    *
*   Zeile nach oben gescrollt und Zeile 4 verwendet.  *
*******************************************************

« SWAP DUP "∓" POS DUP2 1 - 1 SWAP SUB
  OVER
  « 58 CHR + »
  IFT
  ROT ROT 1 + OVER SIZE SUB +
  OVER 4.5 ≥
  « LCD-> 138 OVER SIZE SUB ->LCD »
  IFT
  -> p s
  « 0 ""
    WHILE
      s OVER + 3 PICK NOT 95 * CHR +
      DUP SIZE 23 >
      « DUP SIZE DUP 22 - SWAP SUB »
      IFT
```

```
      p DISP
      DO
      UNTIL
        KEY
      END
      DUP "ENTER" ≠
    REPEAT
      DUP "BACK" SAME
      « DROP 1 OVER SIZE 1 - SUB "" »
      IFT
      DUP "DROP" SAME
      « DROP2 "" DUP »
      IFT
      DUP "EEX" SAME
      « NUM CHR »
      IFT
      DUP "1" SAME
      « DROP SWAP NOT SWAP "" »
      IFT
      IF DUP SIZE 1 SAME THEN
        DUP "A" ≥ OVER "Z" ≤ AND 4 PICK AND
        « " " OR »
        IFT
      ELSE
        DUP "SHIFT" SAME
        « TYPE 4
          START
            WHILE
              DO
              UNTIL
                KEY
              END
              DUP "0" < OVER "9" > OR
            REPEAT
              DROP
            END
          NEXT
          + + STR-> CHR
        »
        IFT
      END
      +
    END
    ROT DROP2 s OVER + p DISP
  »
»
```

```
*******************************************************
* HP48 Funktion SGET,                                 *
* liest String von Tastatur ein                       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Ausgabestring       : String             *
*  Ebene 1 : Displayzeile        : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Eingegebener String : String             *
*                                                     *
* Spezielle Funktionstasten :                         *
*   DEL                  : Clear String               *
*   rote Taste + 3 Zahlen : ASCII-Code der 3 Zahlen   *
*   blaue Taste           : '^' Zeichen               *
*   'a' Taste             : Umschalttaste Groß/Klein  *
*                                                     *
* Dekodierung des Ausgabestring :                     *
*   Das erste "≠" Zeichen wird durch ein ":" Zeichen  *
*   ersetzt.                                          *
*                                                     *
* Dekodierung der Displayzeile :                      *
*   Bei Displayzeilen > 7 wird der LCD-Inhalt eine    *
*   Zeile nach oben gescrollt und Zeile 7 verwendet.  *
*******************************************************

« SWAP DUP "+" POS DUP
  « ":" REPL 0 »
  IFT
  DROP OVER 7.5 ≥
  « LCD-> { # 0h # 8h } { # 82h # 37h } SUB ->LCD »
  IFT
  -> p s
  « 0 ""
    WHILE
      s OVER + 3 PICK NOT 95 * CHR + DUP SIZE 22 >
      « DUP SIZE DUP 21 - SWAP SUB »
      IFT
      p DISP SKEY DUP "e" ≠
    REPEAT
      CASE
        DUP "b" SAME THEN
          DROP 1 OVER SIZE 1 - SUB
        END
        DUP "d" SAME THEN
          DROP2 ""
        END
        DUP "l" SAME THEN
```

```
        DROP SWAP NOT SWAP
      END
      DUP "A" ≥ OVER "Z" ≤ AND 4 PICK AND THEN
        " " OR +
      END
      DUP "s" SAME THEN
        TYPE 4
        START
          WHILE
            SKEY DUP "0" < OVER "9" > OR
          REPEAT
            DROP
          END
        NEXT
        + + STR-> CHR +
      END
      +
    END
  END
  ROT DROP2 s OVER + p DISP
»
»
```

10.2 SKEY (HP48)

Die Funktion SKEY wartet auf eine Taste und wandelt den empfangenen Matrixcode in einen Buchstaben um. Durch die internen Wartezeiten wird ein Systemfehler, welcher Tasten ignoriert, in den ROM-Versionen HPHP48-D und ...-E weitestgehend umgangen.

Die Tastenbezeichnung ' ' steht für ein Leerzeichen. Die <ON> Taste liefert keinen Code zurück, da sie die Bedeutung <ATTN> hat.

SKEY

Ebene 1		Ebene 1
	→	"String"

SKEY wartet auf eine Taste und gibt den Zeichencode in "String" zurück.

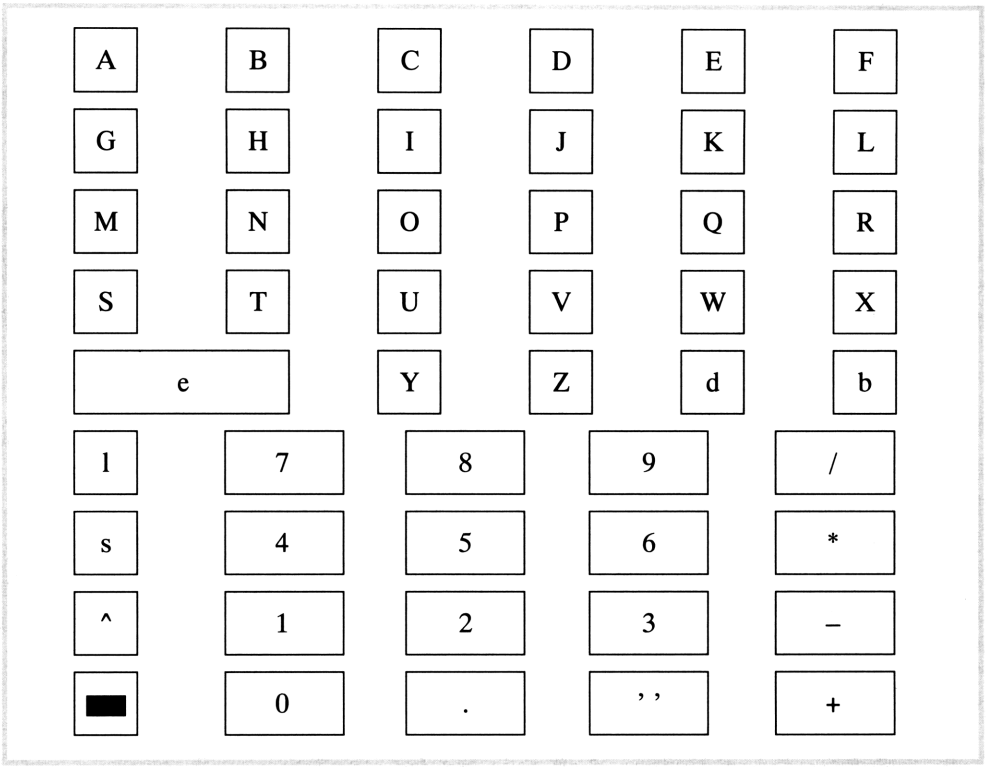


Abb. 10-1 *Returncodes der Tasten bei SKEY*

```
*********************************************************
* HP48 Funktion SKEY,                                   *
* wartet auf eine Taste und konvertiert sie             *
*                                                       *
* Ausgabe  :                                            *
*  Ebene 1 : Zeichen             : String               *
*********************************************************

« ""
  DO
    IF KEY THEN
      "ABCDEFGHIJKLMNOPQRSTUVWXeYZdb 1789/ s456* ^123-  0. +"
      OVER 10 / IP 1 - 6 * ROT 10 MOD + DUP SUB +
    ELSE
      .05 WAIT
    END
  UNTIL
    DUP SIZE
  END
»
```

10.3 ILIST

Die Funktion ILIST ist ein Menügenerator zur Auswahl von Schlagwörtern. Die Schlagwörter, die im Menü dargestellt werden sollen, werden als Strings in einer Liste zusammengefaßt. Die Auflistung der Menüpunkte erfolgt linksbündig und zeilenweise, wobei der erste Listeneintrag oben steht. Position und die darzustellende Anzahl von Einträgen sind variabel. Sollten mehr Listeneinträge definiert worden sein, als aktuell angezeigt werden dürfen, können sie durch Scrollen der Menüs angewählt werden. Die Auswahl der Einträge erfolgt durch Betätigung der <HOCH> beziehungsweise der <RUNTER> Taste und wird mit der <ENTER> Taste abgeschlossen. Die Nummer des gewählten Menüpunktes wird dann zurückgegeben.

Mit dem Benutzerflag 1 kann beim HP48 die Darstellungsweise des Menüs geändert werden. Bei gelöschtem Bit wird vor den Menüpunkten die laufende Nummer des Eintrags und die aktuelle Position durch einen Einfachpfeil angezeigt. Dies ist die Darstellungsart des HP28. Die Einträge können **nicht** durch Drücken der vorangestellten Nummer angewählt werden. Bei gesetztem Bit 1 wird die Eintragsnummer weggelassen und die aktuelle Position durch einen Doppelpfeil links und rechts des Eintrags gekennzeichnet.

Beispielprogramm

« { "Eintrag1" "Eintrag2" "Eintrag3" "Eintrag4" } 1 3 ILIST »

Darstellung des Beispiels bei gelöschtem Bit 1

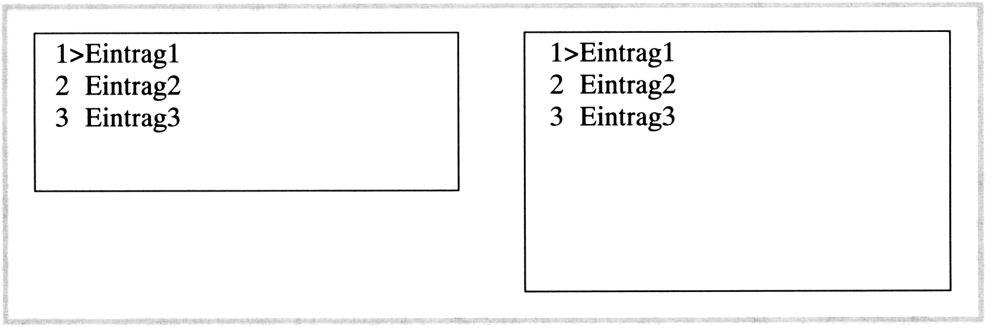


Abb. 10-2 *links HP28, rechts HP48*

Darstellung des Beispiels bei gesetztem Bit 1

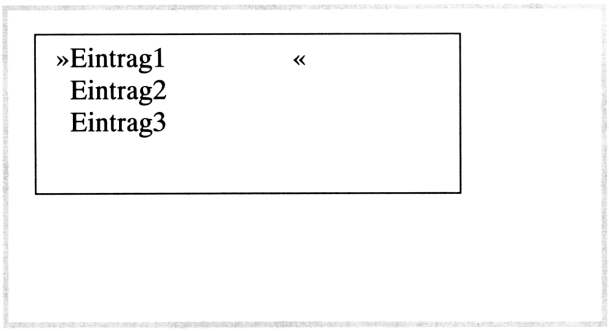


Abb. 10-3
nur HP48

ILIST

Ebene 3	Ebene 2	Ebene 1		Ebene 1
{Liste}	x	y	→	z

ILIST generiert ein Menü mit y Menüpunkten aus {Liste} ab der Zeile x auf dem Display. Nach der interaktiven Auswahl wird die Eintragsnummer z zurückgegeben.

```
*********************************************************
* HP28 Funktion ILIST,                                  *
* erstellt ein interaktives Eingabemenü aus einer       *
* Liste von Strings                                     *
*                                                       *
* Eingabe  :                                            *
*  Ebene 3 : Menüeinträge        : List                 *
*  Ebene 2 : Anfangszeile im LCD : Real number          *
*  Ebene 1 : Länge des Menüs     : Real number          *
*                                                       *
* Ausgabe  :                                            *
*  Ebene 1 : Eintragsnummer      : Real number          *
*                                                       *
* Achtung: LF in String wurde im Listing durch ein      *
* • Zeichen ersetzt!                                    *
*********************************************************

« 3 PICK SIZE
  -> L p h l
  « 1 'a' STO 1 'c' STO 1 'd' STO 0
    DO
      a 'd' STO
      WHILE
        d a h 1 - + ≤ d l ≤ AND
```

```
      REPEAT
        d ->STR
        IF c d SAME THEN
          ">"
        ELSE
          " "
        END
        + L d GET ->STR + d a - p + DISP 1 'd' STO+
      END
      DO
      UNTIL
        KEY
      END
      IF DUP "DOWN" SAME THEN
        IF c l < THEN
          IF a h 1 - + c SAME THEN
            'a' 1 STO+
          END
          'c' 1 STO+
        END
        DROP
      ELSE
        IF DUP "UP" SAME THEN
          IF c 1 ≠ THEN
            IF c a SAME THEN
              'a' -1 STO+
            END
            'c' -1 STO+
          END
          DROP
        ELSE
          IF "ENTER" SAME THEN
            DROP c
          END
        END
      END
    UNTIL
      DUP
    END
  »
  { d c a } PURGE
»
```

```
*******************************************************
* HP48 Funktion ILIST,                                *
* erstellt ein interaktives Eingabemenü aus einer     *
* Liste von Strings                                   *
*                                                     *
* Eingabe  :                                          *
*  Ebene 3 : Menüeinträge        : List               *
*  Ebene 2 : Anfangszeile im LCD : Real Number        *
*  Ebene 1 : Länge des Menüs     : Real Number        *
*  Flag 1  : 0 = Anzeige Einfachpfeil mit Nummer      *
*            1 = Anzeige Doppelpfeil ohne Nummer      *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Eintragsnummer      : Real number        *
*                                                     *
* Achtung: LF in String wurde im Listing durch ein    *
* • Zeichen ersetzt!                                  *
*******************************************************

« ROT DUP SIZE
  -> t m
  « DUP m ≤ SWAP m
    IFTE
    "" 1 m
    FOR i
      1 FC? i ""
      IFTE
      " " + t i GET + "                    " + 1 22
      SUB "
      " + +
    NEXT
    0 DUP
    -> s e g p l
    « DO
        g p 23 * 1 + DUP e 23 * + 2 - SUB l 23 * 1 +
        IF 1 FC? THEN
          1 + p l + 1 + XPON + 134 CHR
        ELSE
          SWAP OVER "»" REPL SWAP 21 + "«"
        END
        REPL s DISP
        DO
          .05 WAIT
        UNTIL
          KEY
        END
        IF DUP 25 SAME THEN
          IF l THEN
```

```
            'l' DECR DROP
          ELSE
            IF p THEN
              'p' DECR DROP
            END
          END
        END
        IF DUP 35 SAME THEN
          IF l 1 + e < THEN
            'l' INCR DROP
          ELSE
            IF p e + m < THEN
              'p' INCR DROP
            END
          END
        END
      UNTIL
        51 SAME
      END
      p l + 1 +
    »
  »
»
```

10.4 INPUT (HP28)

Die Funktion INPUT simuliert den INPUT-Befehl des HP48 in seiner einfachsten Form. Sie liest einen String von der Tastatur ein. Das Ende der Eingabe wird mit der <ENTER> Taste quittiert. Der Befehl sollte immer dann verwendet werden, wenn Portabilität zum HP48 gewährleistet werden soll. Für reine HP28 Anwendungen ist die Funktion SGET flexibler anwendbar. Verzeichniswechsel oder Sondertastenaufrufe sind bei der Simulation nicht möglich. Es gelten die gleichen Einschränkungen, sowie die Besonderheiten der Funktion SGET. Die Funktion INPUT löscht zu Beginn das Display und erwartet zwei Eingabeparameter. In Ebene 2 muß ein String stehen, der in Zeile eins der Anzeige geschrieben wird. In Ebene 1 muß ebenfalls ein String stehen. Dieser dient als Eingabeaufforderung in Zeile vier der Anzeige. Es dürfen auch leere Strings übergeben werden. Als Ergebnis wird die eingegebene Zeichenkette in Ebene 1 zurückgegeben.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM

INPUT

Ebene 2	Ebene 1		Ebene 1
"String$_1$"	"String$_2$"	→	"String$_3$"

"String$_1$" wird in Zeile eins, "String$_2$" in Zeile vier des Displays geschrieben und danach in Zeile vier zur Texteingabe aufgefordert. Die eingegebene Zeichenkette steht in "String$_3$".

```
*******************************************************
* HP28 Funktion INPUT,                                *
* simuliert den INPUT-Befehl des HP48SX der einen     *
* String von Tastatur einliest                        *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Displaystring       : String             *
*  Ebene 1 : Ausgabestring       : String             *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Eingegebener String : String             *
*******************************************************

« CLLCD SWAP 1 DISP 4 SGET CLMF »
```

10.5 C→D (HP28)

Die Funktion C→D wandelt die Grafik-Koordinaten in komplexer Form (x,y) in die Koordinaten eines durch LCD→ erzeugten Grafik-Strings um. Die Eingangsvariable in Ebene 1 beschreibt die Position des Pixels innerhalb des durch 'PPAR' festgelegten Bereiches. Zur Festlegung der Randpunkte des LCDs können die Befehle PMIN, PMAX im Menü PLOT verwendet werden. Die komplexe Ausgangsvariable beinhaltet im Realteil die Adresse im Grafik-String. Der Imaginärteil beinhaltet den ASCII-Code der

Bitmaske. Die Kombination aus Adresse und Bitmaske legt das Pixel fest (vgl. 1-2, Der Grafik-String). Mit Hilfe dieser Information kann nun ein Grafik-String modifiziert werden.

Diese Funktion ist in Bezug auf die Koordinaten mit dem PIXEL-Befehl nicht vollständig kompatibel, da es durch eine unterschiedliche interne Berechnung zu Positionsdifferenzen von einem Pixel kommen kann.

1. Setzen eines Pixels im Grafik-String:
   Der Inhalt der errechneten Adresse im Grafik-String muß mit der Bitmaske geodert (OR) werden.
2. Löschen eines Pixels im Grafik-String:
   Der Inhalt der errechneten Adresse im Grafik-String muß mit dem Einer-Komplement der Bitmaske geundet (AND) werden.

Formeln zur Berechnung

$$\text{xpixel} = \text{INT}\left(136\,\frac{x - x_{min}}{x_{max} - x_{min}} + \frac{1}{2}\right)$$

$$\text{ypixel} = 31 - \text{INT}\left(31\,\frac{y - y_{min}}{y_{max} - y_{min}} + \frac{1}{2}\right)$$

$$\text{adresse} = \text{xpixel} + 137 \cdot \text{INT}\left(136\,\frac{\text{ypixel}}{8}\right) + 1$$

$$\text{char} = 2^{\text{ypixel MOD 8}}$$

C→D

Ebene 1		Ebene 1
(x,y)	→	(adr,mask)

Die Koordinaten (x,y) im durch 'PPAR' festgelegten Bereich werden in die Adresse und die Bitmaske eines Grafik-Strings umgewandelt.

```
*******************************************************
* HP28 Funktion C->D,                                 *
* Umrechnung von x/y-Koordinaten in Bildschirmadresse *
* und Charakterbyte                                   *
*                                                     *
* Eingabe  :                                          *
*  Ebene 1 : x/y-Koordinate      : Complex Number     *
*  'PPAR'    Abbildungsparameter : List               *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Adresse / Charakter : Complex Number     *
*                                                     *
* Als Bereichsgrenzen pmin und pmax werden die Werte  *
* aus 'PPAR' verwendet.                               *
*******************************************************

« PPAR LIST-> 4 DROPN OVER - ROT ROT - C->R ROT C->R
  ROT SWAP / 31 * .5 + IP 31 SWAP - ROT ROT / 136 * .5
  + IP OVER 8 / IP 137 * + 1 + 2 ROT 8 MOD ^ R->C
»
```

10.6 SPIX (HP28)

Die Funktion SPIX setzt das Pixel im Grafik-String, das von der komplexen Ausgangsvariable der Funktion C→D bestimmt wurde. Der Realteil der Variable beinhaltet die Adresse im Grafik-String. Der Imaginärteil der Variable bestimmt das zu setzende Bit.

SPIX

Ebene 2	Ebene 1		Ebene 1
„String$_1$“	(adr,mask)	→	„String$_2$“

SPIX setzt im String$_1$ das durch (adr,mask) spezifizierte Bit und speichert das Ergebnis in String$_2$.

```
*******************************************************
* HP28 Funktion SPIX,                                 *
* setzt das, in der Funktion C->D, durch              *
* Adresse / Charakter bestimmte Pixel im Grafikstring *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Grafikstring        : String             *
*  Ebene 1 : Adresse / Charakter : Complex Number     *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Grafikstring        : String             *
*******************************************************

« C->R CHR ROT ROT DUP2 1 - 1 SWAP SUB ROT ROT DUP2 DUP
  SUB 5 ROLL OR ROT ROT 1 + OVER SIZE SUB + +
»
```

10.7 RPIX (HP28)

Die Funktion RPIX löscht das Pixel im Grafik-String, das von der komplexen Ausgangsvariable der Funktion C→D bestimmt wurde. Der Realteil der Variable beinhaltet die Adresse im Grafik-String. Der Imaginärteil der Variable bestimmt das zu löschende Bit.

RPIX

Ebene 2	Ebene 1		Ebene 1
"String$_1$"	(adr,mask)	→	"String$_2$"

RPIX löscht im String$_1$ das durch (adr,mask) spezifizierte Bit und speichert das Ergebnis in String$_2$.

```
*****************************************************
* HP28 Funktion RPIX,                               *
* löscht das, in der Funktion C->D, durch           *
* Adresse / Charakter bestimmte Pixel im Grafikstring *
*                                                   *
* Eingabe  :                                        *
*  Ebene 2 : Grafikstring       : String            *
*  Ebene 1 : Adresse / Charakter : Complex Number   *
*                                                   *
* Ausgabe  :                                        *
*  Ebene 1 : Grafikstring       : String            *
*****************************************************

« C->R CHR NOT ROT ROT DUP2 1 - 1 SWAP SUB ROT ROT DUP2
  DUP SUB 5 ROLL AND ROT ROT 1 + OVER SIZE SUB + +
»
```

10.8 LROT

Dieses Programm rotiert eine Zeile um eine Anzahl Pixel nach links oder rechts.

Die Richtung hängt von Flag 1 ab: 1 rotiert nach links
0 rotiert nach rechts

Beim HP48 wird ein Zeichen im Grafik-Modus rotiert. Die Zeilennummerierung geht dabei von einer Zeichenhöhe von acht Pixel aus. Dies entspricht der Darstellung des Text-Modus (sieben Zeilen + Statuszeile).

LROT

Ebene 2	Ebene 1		Ebene 1
x	y	→	

LROT rotiert die Zeile x um y Pixel in die von Flag 1 abhängige Richtung.

```
*******************************************************
* HP28 Prozedur LROT,                                 *
* rotiert pixelweise eine Zeile                       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Zeilennummer        : Real Number        *
*  Ebene 1 : Rotationszahl       : Real Number        *
*  Flag 1  : 0 = Rotation nach rechts                 *
*            1 = Rotation nach links                  *
*                                                     *
* Ausgabe  :                                          *
*            Rotiert Zeile im LCD                     *
*******************************************************

« LCD-> ROT 1 - 137 * DUP2 1 SWAP SUB ROT ROT 1 + DUP
  136 + SUB
  -> a m
  « 1 SWAP
    FOR x
      a m
      1 FS?
      x
      137 OVER -
      IFTE
      DUP2 1 + OVER SIZE SUB ROT ROT 1 SWAP SUB + +
      ->LCD
    NEXT
  »
»
```

```
*******************************************************
* HP48 Prozedur LROT,                                 *
* rotiert pixelweise eine Zeile                       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Zeilennummer        : Real Number        *
*  Ebene 1 : Rotationszahl       : Real Number        *
*  Flag 1  : 0 = Rotation nach rechts                 *
*            1 = Rotation nach links                  *
*                                                     *
* Ausgabe  :                                          *
*            Rotiert Zeile im LCD                     *
*******************************************************

« { # 0h # 0h } PVIEW { # 0h } ROT 1 - 8 * R->B { # 82h
  } OVER 7 + + ROT ROT + PICT OVER 4 ROLL SUB
  -> p g
  « 1 SWAP
    IF DUP2 ≤ THEN
      START
        g
        IF 1 FC? THEN
          { # 1h # 0h } OVER { # 0h # 0h } { # 81h # 7h
          } SUB REPL { # 0h # 0h } g { # 82h # 0h } {
          # 82h # 7h }
        ELSE
          { # 0h # 0h } OVER { # 1h # 0h } { # 82h # 7h
          } SUB REPL { # 82h # 0h } g { # 0h # 0h } {
          # 0h # 7h }
        END
        SUB REPL DUP 'g' STO PICT p ROT REPL
      NEXT
    ELSE
      DROP2
    END
  »
»
```

10.9 ROTD

Mit ROTD wird das gesamte Display um eine Zeile nach unten rotiert. Die unten hinausgeschobene Zeile taucht oben wieder auf.

ROTD

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm ROTD,                                 *
* rotiert LCD um eine Zeile nach unten                *
*******************************************************

« LCD-> DUP 412 548 SUB SWAP 1 411 SUB + ->LCD »

*******************************************************
* HP48 Programm ROTD,                                 *
* rotiert LCD um eine Zeile nach unten                *
*******************************************************

« LCD-> DUP { # 0d # 0d } { # 130d # 47d } SUB OVER {
  # 0d # 48d } { # 130d # 55d } SUB ROT ROT { # 0d # 8d
  } SWAP REPL SWAP { # 0d # 0d } SWAP REPL ->LCD
»
```

10.10 ROTU

Mit ROTU wird das gesamte Display um eine Zeile nach oben rotiert. Die oben hinausgeschobene Zeile taucht unten wieder auf.

ROTU

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm ROTU,                                 *
* rotiert LCD um eine Zeile nach oben                 *
*******************************************************

« LCD-> DUP 138 548 SUB SWAP 1 137 SUB + ->LCD »

*******************************************************
* HP28 Programm ROTU,                                 *
* rotiert LCD um eine Zeile nach oben                 *
*******************************************************

« LCD-> DUP { # 0d # 8d } { # 130d # 55d } SUB OVER {
  # 0d # 0d } { # 130d # 7d } SUB ROT ROT { # 0d # 0d }
  SWAP REPL SWAP { # 0d # 48d } SWAP REPL ->LCD
»
```

10.11 SCRD

Mit SCRD wird das gesamte Display um eine Zeile nach unten geschoben. Die unten hinausgeschobene Zeile ist verloren. In der obersten Zeile wird stattdessen eine Leerzeile eingefügt.

SCRD

Ebene 1	Ebene 1
→	

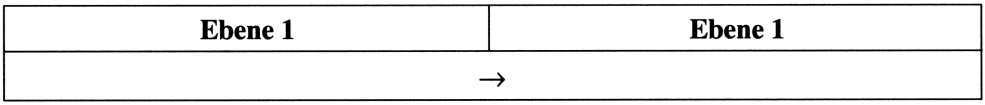
```
*******************************************************
* HP28 Programm SCRD,                                 *
* schiebt LCD um eine Zeile nach unten                *
*******************************************************

« LCD-> DUP 1 137 SUB SWAP 1 412 SUB + ->LCD "" 1 DISP
»
```

```
*******************************************************
* HP48 Programm SCRD,                                 *
* schiebt LCD um eine Zeile nach unten                *
*******************************************************

« LCD-> { # 0d # 8d } OVER { # 0d # 0d } { # 130d # 47d
  } SUB CLLCD REPL ->LCD "" 1 DISP
»
```

10.12 SCRU

Mit SCRU wird das gesamte Display um eine Zeile nach oben geschoben. Die oben hinausgeschobene Zeile ist verloren. In der untersten Zeile wird stattdessen eine Leerzeile eingefügt.

SCRU




```
*******************************************************
* HP28 Programm SCRU,                                 *
* schiebt LCD um eine Zeile nach oben                 *
*******************************************************

« LCD-> 138 548 SUB ->LCD "" 4 DISP »

*******************************************************
* HP48 Programm SCRU                                  *
* schiebt LCD um eine Zeile nach oben                 *
*******************************************************

« LCD-> { # 0h # 8h } { # 82h # 37h } SUB ->LCD »
```

10.13 CIRCLE (HP28)

CIRCLE zeichnet ein Kreissegment in das LCD. Die Parameter im Stack haben folgende Bedeutung:

Ebene 4 :	Anfangswinkel:	Real Number
Ebene 3 :	Endwinkel:	Real Number
Ebene 2 :	Mittelpunkt:	Complex Number
Ebene 1 :	Radius:	Real Number

Die Winkel sind alle in Grad anzugeben.

CIRCLE

Ebenen 1..4	Ebene 1
s.o.	→

```
*******************************************************
* HP28 Funktion CIRCLE,                               *
* Zeichnet einen Kreis auf dem Display                *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 : Anfangswinkel    : Real Number           *
*  Ebene 3 : Endwinkel        : Real Number           *
*  Ebene 2 : Mittelpunkt      : Complex Number        *
*  Ebene 1 : Radius           : Real Number           *
*                                                     *
* Ausgabe                                             *
*******************************************************

« 60 FS?C
  -> a e p r f
  « a e
    FOR t
      t COS r * p RE + t SIN r * p IM + R->C PIXEL r
      INV 5
    * STEP
    f
  »
  IF THEN
    60 SF
  END
»
```

10.14 LINE (HP28)

LINE zeichnet eine Linie auf dem Display. Die Parameter im Stack haben folgende Bedeutung:

Ebene 2:	Startpunkt:	Complex Number
Ebene 1:	Endpunkt:	Complex Number

LINE

Ebene 2	Ebene 1	Ebene 1
z_1	z_2	→

LINE zieht eine Linie von Startpunkt z_1 zum Endpunkt z_2 im Display.

```
*******************************************************
* HP28 Funktion LINE,                                 *
* Zeichnet einen Linie auf dem Display                *
*                                                     *
* Eingabe  :                                          *
*  Ebene 2 : Startpunkt       : Complex Number        *
*  Ebene 1 : Endpunkt         : Complex Number        *
*******************************************************

« C->R ROT C->R
  IF OVER 5 PICK > THEN
    4 ROLL 4 ROLL
  END
  -> x2 y2 x1 y1
  « y2 y1 - x2 x1 - /
    -> f
    « IF f .5 ≤ THEN
        x1 x2
        FOR x
          x f * y1 + x1 f * - x SWAP R->C PIXEL
        .1 STEP
      ELSE
        y1 y2
        FOR y
          y f / x1 + y1 f / - y R->C PIXEL
        .1 STEP
      END
    »
  »
  'PPAR' PURGE
»
```

10.15 QSORT

Die Funktion QSORT sortiert eine Liste in aufsteigender Reihenfolge nach dem Quicksort-Verfahren. Die Funktion arbeitet rekursiv, deshalb sollte der Programmname beibehalten werden, andernfalls ist eine Anpassung des Quelltextes notwendig, da sich bei einem rekursiven Algorithmus die Funktion bis zu einer Abbruchbedingung immer wieder selbst aufuft.

Als zu sortierendes Feld kann entweder eine Liste oder ein Vektor übergeben werden. Innerhalb von Listen können Real Number, Binary Integer oder Strings sortiert werden. Der Funktion muß der zu sortierende Bereich in Ebene 2 und 1 übergeben werden. Dabei muß $n_1 < n_2$ sein.

Beispiel: Liste vollständig sortieren

« 1 OVER SIZE QSORT »

QSORT

Ebene 3	Ebene 2	Ebene 1		Ebene 1
$\{Liste_1\}$	n_1	n_2	→	$\{Liste_2\}$
$[R\text{-}Feld_1]$	n_1	n_2	→	$[R\text{-}Feld_2]$

QSORT sortiert $\{Liste_1\}$ oder $[R\text{-}Feld_1]$ im Bereich zwischen n_1 und n_2. Das Ergebnis steht in $\{Liste_2\}$ oder $[R\text{-}Feld_2]$.

```
*******************************************************
* HP28/48 Funktion QSORT,                             *
* sortiert eine Liste nach dem Quick-Sort-Verfahren   *
* von Element Startwert bis zum Element Endwert       *
*                                                     *
* Eingabe  :                                          *
*  Ebene 3 : Datenfeld          : List                *
*  Ebene 2 : Startwert          : Real number         *
*  Ebene 1 : Endwert            : Real number         *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Datenfeld          : List                *
*******************************************************

« DUP2
  -> l r
  « 3 DUPN + 2 / IP GET
    -> p
    « DO
```

```
        ROT SWAP
        WHILE
          DUP2 GET p >
        REPEAT
          1 -
        END
        SWAP ROT
        WHILE
          DUP2 GET p <
        REPEAT
          1 +
        END
        ROT
        IF DUP2 ≤ THEN
          3 DUPN 3 PICK ROT GET PUT ROT 4 ROLL 4 PICK
          GET PUTI ROT 1 -
        END
      UNTIL
        DUP2 >
      END
    »
    -> i j
    « l j <
      « l j QSORT »
      IFT
      i r <
      « i r QSORT »
      IFT
    »
  »
»
```

11 Demoprogramme

11.1 Lisaujou-Figuren

Wer behauptet denn, die HP-Rechner seien langsam und schnelle Grafik unmöglich? Mit dem Programmpaket LIS wird man eines Besseren belehrt. Dieses Paket demonstriert eine interessante Grafik-Animation, die man recht gern als „Leistungsbeweis" für Computer gebraucht. Der Nachteil des Ganzen: Man belegt damit etwa 12k freien Speicher.

11.1.1 GO

GO startet die Animation.

Verwendete externe Variablen

Variable	Verzeichnis	Status
Sx	\USER\DEMO\LIS	gesichert

'x' ist eine beliebige Zahl

GO

Ebene 1	**Ebene 1**
→	

```
*****************************************************
* HP28 Programm GO,                                 *
* spielt Folge von Bildern ab                       *
*****************************************************

« 1 10
  START
    0 17
    FOR f
      "S" f ->STR + STR-> ->LCD
    NEXT
  NEXT
  CLMF
»


*****************************************************
* HP48 Programm GO,                                 *
* spielt Folge von Bildern ab                       *
*****************************************************

« ERASE { # 0h # 0h } PVIEW 1 10
  START
    0 17
    FOR f
      PICT (-1,1) "S" f ->STR + STR-> REPL
    NEXT
  NEXT
»
```

11.1.2 LIS

Mit LIS wird interaktiv eine Lisaujou-Figur definiert. Als erstes wird man nach dem Verhältnis der Frequenzen fx/fy gefragt. Für eine '1' bekommt man einen Kreis. Dann ist die Drehung des Kreises / der Ellipse dran. Als nächstes wird Schrittweite und dann die gesamte Anzahl der Schritte abgefragt.

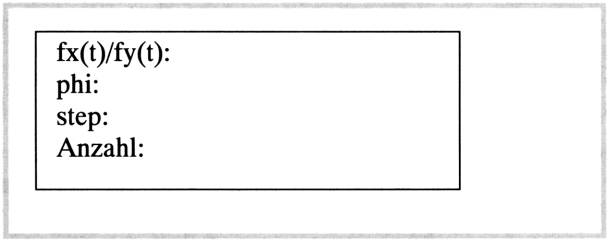


Abb. 11-1
Darstellung der interaktiven Eingabe

Alle Winkelangaben erfolgen in Grad

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
SGET	\USER\PROGRAM
LISGO	\USER\DEMO\LIS

LIS

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm LIS,                                  *
* Zeichnet Lisaujou-Figur. Die Abfrage ist Interaktiv.*
*******************************************************

« RCLF RAD CLLCD PROGRAM "x(t)/y(t)+" 1 SGET STR->
  "phi÷" 2 SGET STR-> "step÷" 3 SGET STR-> "Anzahl÷" 4
  SGET STR-> DEMO LIS LISGO STOF
»


*******************************************************
* HP48 Programm LIS,                                  *
* Zeichnet Lisaujou-Figur. Die Abfrage ist Interaktiv.*
*******************************************************

« RCLF GRAD CLLCD PROGRAM "x(t)/y(t)+" 1 SGET STR->
  "phi÷" 2 SGET STR-> "step÷" 3 SGET STR-> "Anzahl÷" 4
  SGET STR-> DEMO LIS { # 0h # 0h } PVIEW LISGO STOF
»
```

11.1.3 LISDEF

Mit LISDEF wird eine komplette Folge von Bildern für die Animation definiert. Das gelistete LISDEF ist ein Beispiel für einen scheinbar rotierenden Kreis. Die Ausführungszeit des Programms ist sehr hoch!

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
LISGO	\USER\DEMO\LIS

Verwendete externe Variablen

Variable	Verzeichnis	Status
Sx	\USER\DEMO\LIS	gesichert

'x' ist eine beliebige Zahl

LISDEF

Ebene 1	Ebene 1
→	

```
*******************************************************
* HP28 Programm LISDEF,                               *
* Definiert Folge von Lisaujou-Figuren                *
*******************************************************

« VARS 0 17
  FOR i
    1 i 20 * 6 400 LISGO LCD-> "S" i ->STR + STR-> STO
  NEXT
  ORDER CLMF
»


*******************************************************
* HP48 Programm LISDEF,                               *
* Definiert Folge von Lisaujou-Figuren                *
*******************************************************

« VARS { # 0h # 0h } PVIEW 0 17
  FOR i
    1 i 20 * 6 400 LISGO PICT RCL (-1,-1) (1,1) SUB
    "'S" i ->STR + "'" + STR-> STO
  NEXT
  ORDER
»
```

11.1.4 LISGO

Das ist das eigentliche Zeichenprogramm. Die Übergabeparameter stehen im Listing.

LISGO

Ebene 1..4	Ebene 1
x →	

```
*******************************************************
* HP28 Programm LISGO,                                *
* Zeichnet Lisaujou-Figur                             *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 :      Verhältnis        : Real Number     *
*  Ebene 3 :      Phase             : Real Number     *
*  Ebene 2 :      Schrittweite      : Real Number     *
*  Ebene 1 :      Anzahl            : Real Number     *
*******************************************************

« -> v p s n
  « CLLCD 0 n
    FOR t
      t SIN t v / p + SIN R->C PIXEL
    s STEP
  »
»


*******************************************************
* HP48 Programm LISGO,                                *
* Zeichnet Lisaujou-Figur                             *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 :      Verhältnis        : Real Number     *
*  Ebene 3 :      Phase             : Real Number     *
*  Ebene 2 :      Schrittweite      : Real Number     *
*  Ebene 1 :      Anzahl            : Real Number     *
*******************************************************

« -> v p s n
  « ERASE 0 n
    FOR t
      t SIN t v / p + SIN R->C PIXON
    s STEP
  »
»
```

11.2 Türme von Hanoi

11.2.1 HANOI (HP48)

Das Programm HANOI simuliert grafisch eine Lösung für das Logikproblem „Türme von Hanoi". Bei den Türmen von Hanoi muß ein Stapel von runden Steinen mit unterschiedlichem Durchmesser auf einen anderen Stapel umsortiert werden. Die Steine sind dem Durchmesser entsprechend sortiert, wobei der Stein mit dem geringsten Durchmesser oben liegt. Die Steine müssen einzeln bewegt werden. Es darf nie ein größerer Stein auf einem kleineren zu liegen kommen. Ein weiterer Hilfsstapel darf verwendet werden.

Das Programm fragt nach dem Start nach der Nummer des Start- und des Zielstapels. Es sind Zahlen für die Stapel von eins bis drei erlaubt. Weiterhin wird nach der Anzahl von übereinander liegenden Steinen gefragt. Die Software verarbeitet maximal acht Steine. Die Begrenzung liegt jedoch nicht im Lösungsalgorithmus, sondern in der Grafik begründet. Die Wartezeit bestimmt den Zeitraum zwischen zwei Umsortierungen. Wenn Sie nur die <ENTER> Taste betätigen, arbeitet das Programm mit den Defaultwerten.

Verwendete selbstdefinierte Unterprogramme

Name	Verzeichnis
UPH	\USER\DEMO

HANOI

Ebene 1	**Ebene 1**
→	

```
*******************************************************
* HP48 Programm HANOI,                                 *
* grafische Darstellung der Lösung "Türme von Hanoi"  *
*******************************************************

« "** Türme von Hanoi **" { ":Start:1     :Ziel:2
  :Anzahl der Steine:7
  :Wartezeit:0.1" { -1 8 } V } INPUT OBJ-> { 0 0 0 }
  SWAP + 4 PICK 3 PICK PUT
  -> x y a s
  « ERASE PICT RCL { # 3h # 0h }
```

```
    "** Türme von Hanoi **" 2 ->GROB REPL { # 16h # 39h
    } "1" 1 ->GROB REPL { # 40h # 39h } "2" 1 ->GROB
    REPL { # 6Ah # 39h } "3" 1 ->GROB REPL PICT STO {
    # 0h # 8h } { # 82h # 8h } LINE { # 0h # 37h } {
    # 82h # 3Fh } BOX { # 0h # 0h } PVIEW 9 a - 8
    FOR i
      8 i - DUP + x 1 - 42 * + 7 + R->B i 1 - 5 * 15 +
      R->B DUP2 2 ->LIST ROT i 4 * + ROT 4 + 2 ->LIST
      BOX
    NEXT
    s a x y UPH DROP
  »
  PICT { # 12h # 39h } "ENDE, WEITER MIT TASTE !" 1
  ->GROB REPL
  DO
    .05 WAIT
  UNTIL
    KEY
  END
  DROP
»
```

11.2.2 UPH (HP48)

Das Unterprogramm UPH beinhaltet den Algorithmus des Programms HANOI und „bewegt" die Steine anhand der Lösung von einem Stapel zum nächsten. Die Problemlösung erfolgt rekursiv, deshalb sollte der Programmname nicht verändert werden.

UPH

Ebene 4	Ebene 3	Ebene 2	Ebene 1		Ebene 1
{Liste$_1$}	n	x	y	→	{Liste$_2$}

Bewegt n Steine vom Stapel x zum Stapel y. Dabei wird der Inhalt der {Liste$_1$} mit den aktuellen Positionsdaten zur {Liste$_2$} aktualisiert.

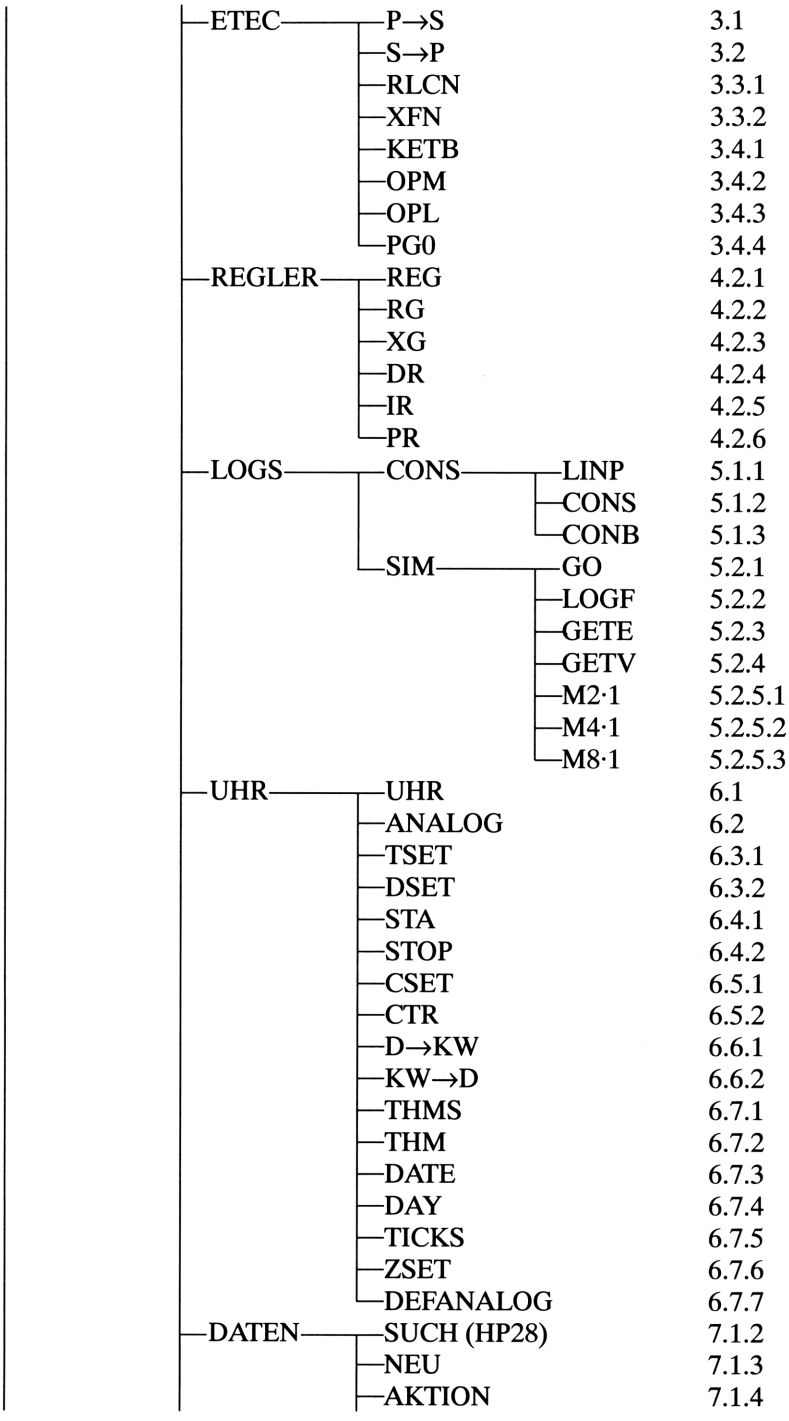
```
*******************************************************
* HP48 Funktion UPH,                                  *
* bewegt die Steine grafisch von der Start- zu der    *
* Zielposition                                        *
*                                                     *
* Eingabe  :                                          *
*  Ebene 4 : Positionsliste      : List               *
*  Ebene 3 : Anzahl der Steine   : Real Number        *
*  Ebene 2 : Startpunkt          : Real Number        *
*  Ebene 1 : Endpunkt            : Real Number        *
*                                                     *
* Ausgabe  :                                          *
*  Ebene 1 : Positionsliste      : List               *
*                                                     *
* UPH ist ein Unterprogramm zu HANOI                  *
*******************************************************

« -> x y
  « 1 - x R->B y R->B XOR B->R
    -> n z
    « IF n THEN
        n x z UPH
      END
      x DUP2 GET PICT DUP 3 PICK NEG 8 + 5 * 15 + R->B
      5 PICK 1 - 42 * 7 + R->B SWAP DUP2 2 ->LIST DUP 5
      ROLLD ROT 32 + ROT 4 + 2 ->LIST SUB DUP 7 ROLLD
      GXOR 1 - PUT y DUP2 GET 1 + OVER 1 - 42 * 7 +
      R->B 8 3 PICK - 5 * 15 + R->B 2 ->LIST PICT SWAP
      6 ROLL REPL PUT DUP 4 GET WAIT
      IF n THEN
        n z y UPH
      END
    »
  »
»
```

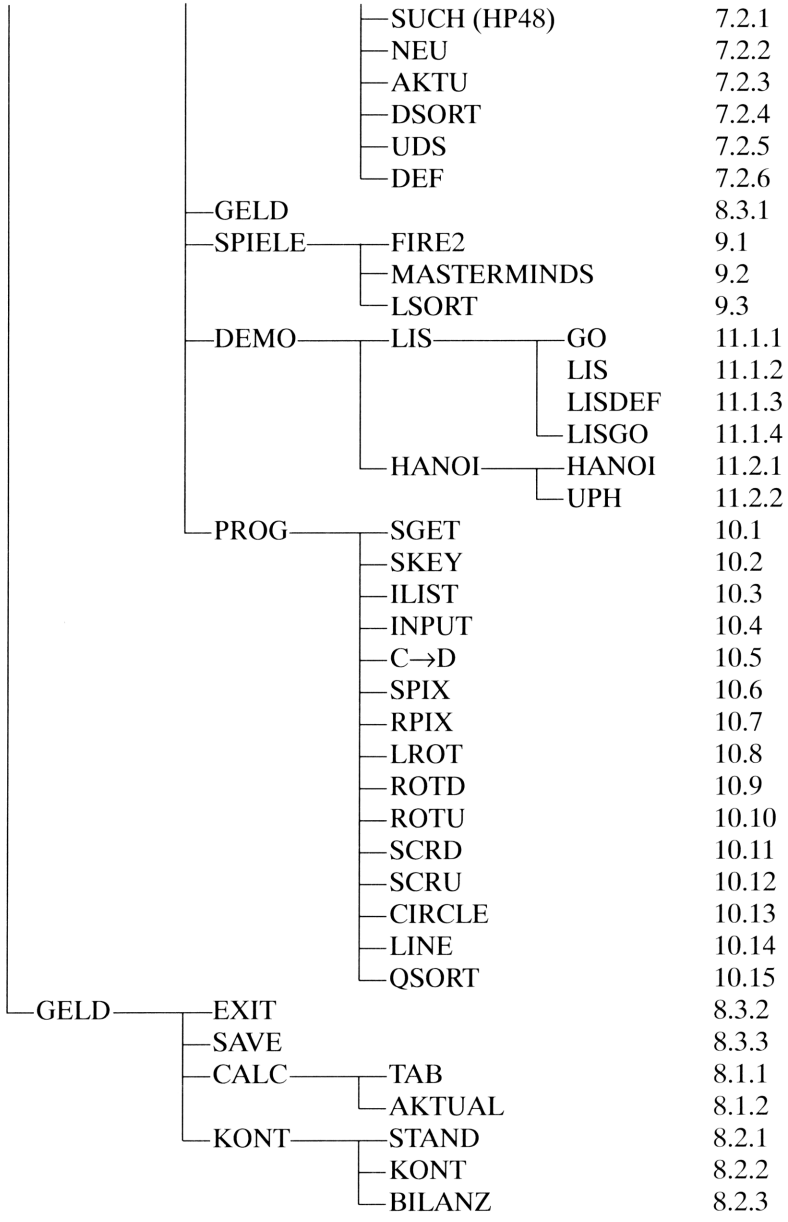
A Anhang

A1 Verzeichnisstruktur

Die folgende Liste zeigt die hierarchische Anordnung der Programme im HP28 und HP48.

					Kapitel
HOME	USER	MATH	BRUCH		2.1
			SQUAD		2.2
			F→L		2.3
			L→F		2.4
			PDIV		2.5
			NEWTON		2.6
			INTER		2.7
			SPLINE	SPLINE	2.8.1
				ASPLINE	2.8.2
				NSPLINE	2.8.3
				PSPLINE	2.8.4
				PARA	2.8.5
				HORNER	2.8.6
				FUNC	2.8.7
				SDRAW	2.8.8
			APPROXIMATION		2.9
			TPOL		2.10
			PASCAL		2.11
			PRIM		2.12
			FAK		2.13.1
			FAKU		2.13.2
			GGT		2.14
			B→D		2.15
			C→PT		2.16


ETEC
P→S 3.1
S→P 3.2
RLCN 3.3.1
XFN 3.3.2
KETB 3.4.1
OPM 3.4.2
OPL 3.4.3
PG0 3.4.4
REGLER
REG 4.2.1
RG 4.2.2
XG 4.2.3
DR 4.2.4
IR 4.2.5
PR 4.2.6
LOGS
CONS
LINP 5.1.1
CONS 5.1.2
CONB 5.1.3
SIM
GO 5.2.1
LOGF 5.2.2
GETE 5.2.3
GETV 5.2.4
M2·1 5.2.5.1
M4·1 5.2.5.2
M8·1 5.2.5.3
UHR
UHR 6.1
ANALOG 6.2
TSET 6.3.1
DSET 6.3.2
STA 6.4.1
STOP 6.4.2
CSET 6.5.1
CTR 6.5.2
D→KW 6.6.1
KW→D 6.6.2
THMS 6.7.1
THM 6.7.2
DATE 6.7.3
DAY 6.7.4
TICKS 6.7.5
ZSET 6.7.6
DEFANALOG 6.7.7
DATEN
SUCH (HP28) 7.1.2
NEU 7.1.3
AKTION 7.1.4


SUCH (HP48) 7.2.1
NEU 7.2.2
AKTU 7.2.3
DSORT 7.2.4
UDS 7.2.5
DEF 7.2.6
GELD 8.3.1
SPIELE
FIRE2 9.1
MASTERMINDS 9.2
LSORT 9.3
DEMO
LIS
GO 11.1.1
LIS 11.1.2
LISDEF 11.1.3
LISGO 11.1.4
HANOI
HANOI 11.2.1
UPH 11.2.2
PROG
SGET 10.1
SKEY 10.2
ILIST 10.3
INPUT 10.4
C→D 10.5
SPIX 10.6
RPIX 10.7
LROT 10.8
ROTD 10.9
ROTU 10.10
SCRD 10.11
SCRU 10.12
CIRCLE 10.13
LINE 10.14
QSORT 10.15
GELD
EXIT 8.3.2
SAVE 8.3.3
CALC
TAB 8.1.1
AKTUAL 8.1.2
KONT
STAND 8.2.1
KONT 8.2.2
BILANZ 8.2.3

A.2 Literaturverzeichnis

Kettenbruchverfahren Kapitel 2
Tobias Haist, *c't* 2-89 S. 104, Heise-Verlag

RLCN Kapitel 3
D. Pilz, Skriptum zu den Vorlesungen „Grundgebiete der Elektrotechnik A und D"

Kettenbruchverfahren und RLCN Kapitel 3
H. Wolf, *Lineare Systeme und Netzwerke*, Springer-Verlag (1989)

Reglersimulation Kapitel 4
O. Föllinger, *Regelungstechnik*, Hüthig-Verlag (1989)

Logik-Optimierung Kapitel 5
H. M. Lipp, Skriptum zur Vorlesung „Entwurf digitaler Schaltungen 1-A"

Logik-Simulation Kapitel 5
H. J. Wunderlich, *Hochintegrierte Schaltungen: Prüfgerechter Entwurf und Test*, Springer-Verlag (1990)

Sortierverfahren,
c't-Kartei, *c't* 3/92 S.263, Heise-Verlag

Quickersort
Uwe Thiemann, **c't** 4/92 S.264, Heise-Verlag

Grundwerte der Meßwiderstände für Widerstandsthermometer
DIN 43760

HP48 Programmer's Reference Manual
Hewlett Packard

HP 48 I/O TECHNICAL INTERFACING GUIDE,
Hewlett Packard, Ausgabe 14 Juni 1990

Die ideale Ergänzung zu den Taschenrechnern HP 28S und HP 48. Christoph Gießelink und Stefan Rau stellen konkrete Problemlösungen aus der Praxis für die Bereiche Elektrotechnik und Mathematik vor.

Im einzelnen werden folgende Schwerpunkte gesetzt:

- Spline-Interpolation;
- Polynomdivision;
- grafische Darstellung eines Frequenzganges;
- Optimierung logischer Ausdrücke;
- Datenbank.

Mit diesen Tools sind Ihr HP 28S und HP 48 noch mächtiger! Die Programme für den HP 48 liegen auf der beigefügten Diskette vor.

ISBN 3-88229-031-5