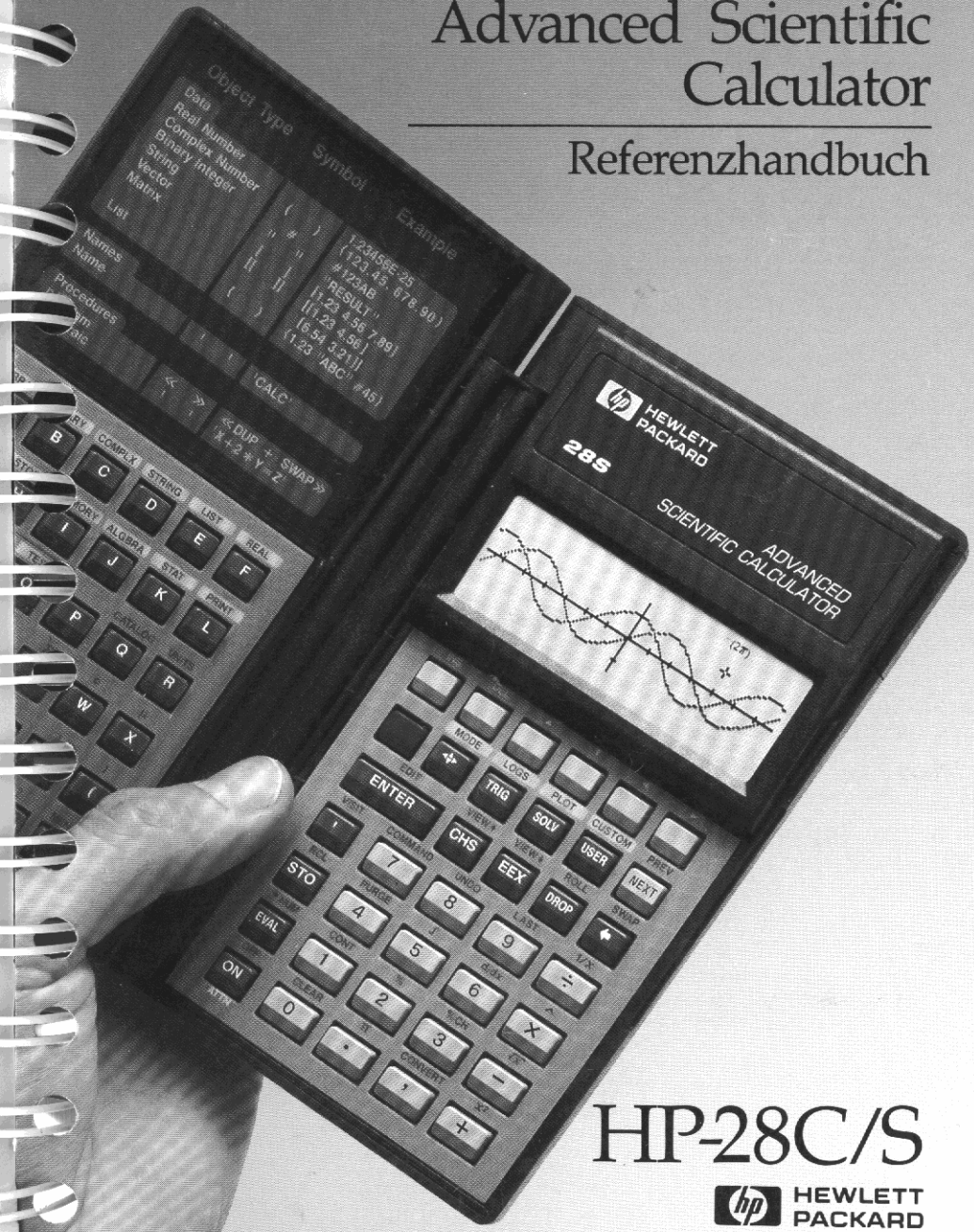
HEWLETT-PACKARD

Advanced Scientific Calculator

Referenzhandbuch



HP-28C/S

HEWLETT PACKARD

HP-28C/S
Advanced Scientific Calculator

Referenzhandbuch

HEWLETT PACKARD

2. Ausgabe Februar 1988
Bestell-Nr. 00028-90025

Hinweis

Änderungen der in dieser Dokumentation enthaltenen Informationen sind vorbehalten.

Hewlett-Packard übernimmt weder ausdrücklich noch stillschweigend irgendwelche Haftung für die in diesem Handbuch dargestellten Programme und Beispiele—weder für deren Funktionsfähigkeit noch deren Eignung für irgendeine spezielle Anwendung. Hewlett-Packard haftet nicht für direkte oder indirekte Schäden im Zusammenhang mit oder als Folge der Lieferung, Benutzung oder Leistung der Programme. (Dies gilt nicht, soweit gesetzlich zwingend gehaftet wird.)

Hewlett-Packard übernimmt keine Verantwortung für den Gebrauch oder die Zuverlässigkeit von HP Software unter Verwendung von Geräten, welche nicht von Hewlett-Packard geliefert wurden.

Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

Druckgeschichte

1. Ausgabe	Oktober 1986	Fertigungsnr. 00028-90026
2. Ausgabe	Februar 1988	Fertigungsnr. 00028-90115

Vorwort

Mit Ihrem HP-28C können Sie auf einfache Weise die komplexesten Probleme lösen, einschließlich der Aufgabenstellungen, die seither mit einem Taschenrechner nicht bearbeitet werden konnten. Der HP-28C kombiniert leistungsstarke numerische Rechenweise mit einer neuen Dimension—*symbolische Rechenweise*. Sie formulieren zuerst ein Problem symbolisch, lassen sich danach die symbolische Lösung anzeigen, aus welcher das globale Verhalten des Problems ersichtlich ist, und erhalten numerische Ergebnisse über die symbolische Lösung.

Der HP-28C bietet Ihnen folgende Leistungsmerkmale:

- Algebraische Manipulation. Sie können in einem Ausdruck Terme erweitern, zusammenfassen oder neuordnen sowie eine Gleichung symbolisch nach einer Variablen auflösen.
- Höhere Mathematik. Sie können Differentiale, bestimmte und unbestimmte Integrale berechnen.
- Numerische Lösungen. Unter Verwendung eines speziellen Lösungsprozesses im HP-28C können Sie eine Gleichung oder einen Ausdruck nach jeder beliebigen Variablen auflösen. Ebenso ist die Lösung eines linearen Gleichungssystems möglich. Durch verschiedene Datentypen lassen sich komplexe Zahlen, Vektoren und Matrizen so einfach wie reelle Zahlen verwenden.
- Abbildungen. Ausdrücke, Gleichungen und statistische Daten lassen sich graphisch darstellen.
- Konvertierung zwischen jeder zulässigen Kombination der eingebauten 120 Einheiten.
- Statistik. Es lassen sich statistische Operationen mit einer oder zwei Variablen sowie Wahrscheinlichkeitsberechnungen durchführen.
- Zahlensysteme. Berechnungen im dualen, oktalen, dezimalen und hexadezimalen Zahlensystem sowie Bitmanipulationen sind möglich.
- Direkte Eingabe für algebraische Ausdrücke und UPN Logik für interaktive Berechnungen.

Das *HP-28S Benutzerhandbuch* macht Sie mit Ihrem Rechner vertraut und führt Sie durch mehrere Anwendungsbeispiele.

Das *HP-28C/S Referenzhandbuch* gibt Ihnen spezifische Informationen über Befehle und die Arbeitsweise des Rechners. Die ersten zwei Kapitel erläutern die Grundlagen sowie die gebräuchlichsten Operationen. Das dritte Kapitel stellt ein Verzeichnis der Menüs dar und beschreibt das Anwendungskonzept und die Befehle der einzelnen Menüs. *Da dieses Handbuch ursprünglich auf den HP-28C ausgerichtet war, sind nicht alle Leistungsmerkmale des HP-28S enthalten. Informationen bezüglich der Anwendung von Verzeichnissen und CUSTOM Menüs finden Sie ausschließlich im Benutzerhandbuch des HP-28S.*

Es wird empfohlen, daß Sie sich zuerst durch die Beispiele im Benutzerhandbuch arbeiten, um mit dem Rechner und seinen Funktionen vertraut zu werden. Wenn Sie näheres über einen bestimmten Befehl wissen möchten, können Sie dazu das Referenzhandbuch zu Hilfe nehmen. Nachdem Sie über den Gebrauch der Befehle Bescheid wissen und ein tieferes Verständnis über die Betriebsweise des Rechners erhalten möchten, können Sie die theoretischen Diskussionen im Referenzhandbuch durchlesen.

Durch die erwähnten Handbücher erfahren Sie, wie Ihr HP-28S zur Lösung mathematischer Aufgabenstellungen eingesetzt werden kann; sie lehren jedoch nicht Mathematik. Es wird angenommen, daß Sie bereits mit den relevanten mathematischen Prinzipien vertraut sind. Um z.B. die Fähigkeiten des HP-28S im Bereich der höheren Mathematik effizient nutzen zu können, sollten Sie die elementaren Gesetze über Differential- und Integralrechnung kennen.

Auf der anderen Seite ist damit nicht gemeint, daß Sie über sämtliche Mathematikgebiete innerhalb des HP-28S Bescheid wissen müssen, um die für Sie interessanten Teile anzuwenden. Wenn Sie z.B. die Statistik-Operationen des HP-28S anwenden möchten, müssen Sie nicht die Integralrechnung verstanden haben.

Inhaltsverzeichnis

Anhänge, Glossar, Index

Verwenden dieses Handbuchs

Dieses Handbuch enthält allgemeine Informationen über die Betriebsweise des HP-28S und spezifische Informationen über die einzelnen Operationen. Sehen Sie sich das Inhaltsverzeichnis an, wenn Sie einen Überblick von diesem Handbuch erhalten möchten. Durch die nachstehende Tabelle können Sie schnell andere Arten von Informationen aufsuchen:

Gesuchte Information über:	Bezugnahme auf:
Eine bestimmte Operation, Funktion oder einen speziellen Befehl.	Das Verzeichnis der Operationen (Seite 381). Alle Operationen, Befehle und Funktionen sind alphabetisch aufgelistet. Jeder Eintrag enthält eine kurze Beschreibung, eine Referenz auf ein Menü oder Thema im Schlüsselwortverzeichnis und eine Seitenangabe. Beziehen Sie sich auf das Menü bzw. das Thema im Verzeichnis, um mehr Hintergrundinformationen zu erhalten; spezifische Informationen finden Sie unter der angegebenen Seitenzahl.
Ein bestimmtes Menü.	Kapitel 3, "Schlüsselwortverzeichnis" (Seite 57). Alle Menüs sind alphabetisch gelistet.
Konzepte und Prinzipien der HP-28S Operationen.	Kapitel 1, "Grundlagen" (Seite 17).
Die Durchführung allgemeiner HP-28S Operationen.	Kapitel 2, "Grundlegende Operationen" (Seite 31).
Die Bedeutung einer angezeigten Meldung.	Anhang A, "Meldungen" (Seite 349).
Die Bedeutung eines unbekannten Begriffs	Glossar (Seite 367).

Aufbau des Handbuchs

Kapitel 1 und 2 enthalten allgemeine Informationen. "Grundlagen" stellt einen Überblick für den erfahrenen Anwender dar und beschreibt die Funktionsweise des HP-28S. Im nächsten Kapitel, "Grundlegende Operationen", wird erläutert, wie ein Objekt in die Befehlszeile eingegeben wird, wie Variablen erzeugt und wie Systemoperationen durchgeführt werden.

Kapitel 3, "Schlüsselwortverzeichnis", stellt den umfangreichsten Teil des Handbuchs dar. Über die einzelnen Menüs gegliedert werden die individuellen Operationen, Funktionen und Befehle detailliert beschrieben. Die Auswirkung jedes Befehls bzw. jeder Funktion ist in einem Stackdiagramm definiert. (Beziehen Sie sich auf den nachfolgenden Abschnitt "Lesen der Stackdiagramme".)

Kapitel 3 enthält außerdem wichtige Themen, die sich nicht direkt auf ein Menü beziehen. Es gibt Beiträge über Arithmetik, höhere Mathematik, CATALOG, Programme und UNITS.

Anhang A, "Meldungen" beschreibt die Status- und Fehlermeldungen, auf die Sie gelegentlich stoßen werden.

Anhang B, "Hinweise für Anwender von HP UPN-Rechnern", und Anhang C, "Hinweise für Anwender von AOS-Rechnern", vergleichen den HP-28S mit anderen Taschenrechnermodellen, mit welchen Sie vielleicht bereits vertraut sind.

Im Glossar sind Begriffe, die in diesem Handbuch verwendet wurden, erklärt.

Das Verzeichnis der Operationen ist eine alphabetische Auflistung aller Operationen, Befehle und Funktionen. Jeder Eintrag enthält eine kurze Beschreibung, eine Referenz auf ein Kapitel oder den Namen eines Menüs, unter welchem Sie weitere Hintergrundinformationen finden können; außerdem ist noch eine Seitenzahl angegeben, wo Sie spezifische Informationen erfahren können.

Lesen der Stackdiagramme

Die Wirkungsweise eines Befehls ist durch die Werte und Reihenfolge seiner Argumente spezifiziert. Ein *Argument* besteht aus einem Objekt, welches von der Stackebene genommen wird, auf die sich der Befehl bezieht. Der Befehl gibt als *Ergebnis* einen Wert in den Stack zurück. (Einige Befehle betreffen Modi, Variable, Flags oder die Anzeige, anstatt Objekte.)

Die Beschreibung jedes Befehls schließt ein *Stackdiagramm* mit ein, das eine tabellarische Auflistung der Argumente und Ergebnisse des Befehls liefert. Ein typisches Stackdiagramm könnte folgender Abbildung gleichen:

BSPL ***Beispiel*** **Funktion**

Ebene 2	Ebene 1		Ebene 1
$Objekt_1$	$Objekt_2$	➧	$Objekt_3$

Dieses Diagramm zeigt:

- Der Name (welcher in der Befehlszeile erscheinen kann) ist "BSPL".
- Die erläuternde Bezeichnung heißt "Beispiel".
- BSPL stellt eine Funktion dar (in algebraischen Ausdrücken erlaubt).
- BSPL benötigt zwei Argumente, $Objekt_1$ und $Objekt_2$, die von den Stackebenen 2 und 1 genommen werden.
- BSPL gibt ein Ergebnis, $Objekt_3$, in Stackebene 1 zurück.

Im Diagramm trennt der Pfeil ➧ die Argumente (auf der linken Seite) von den Ergebnissen (auf der rechten Seite). Es handelt sich um eine abkürzende Notation für "Mit den vorangehenden Argumenten im Stack bewirkt die Ausführung von BSPL das Zurückgeben der folgenden Ergebnisse in den Stack".

Die Argumente und Ergebnisse sind in verschiedener Form in den Diagrammen gelistet und zeigen soviel spezifische Informationen wie möglich an. Spezielle Objekt-Typen sind mit ihren charakteristischen Begrenzungszeichen angegeben. Mit in diese Zeichen eingeschlossene Worte oder Gleichungen bieten zusätzliche Beschreibungen der Objekte. Stackdiagramme verwenden allgemein die nachfolgenden Terme.

In Stackdiagrammen verwendete Terme

Term	Bedeutung
Objekt	Beliebiges Objekt
x oder *y*	Reelle Zahl
hms	Reelle Zahl im Stunden-Minuten-Sekunden-Format
n	Positive ganze reelle Zahl
Flag	Reelle Zahl, Null (falsch) oder ungleich Null (wahr)
z	Reelle oder komplexe Zahl
(*x* , *y*)	Komplexe Zahl in Rechtecksnotation
(*r* , *θ*)	Komplexe Zahl in Polarnotation
# *n*	Binärwert
"*String*"	Zeichenkette
[*Feld*]	Reelle(r) oder komplexe(r) Matrix (Vektor)
[*Vektor*]	Reeller oder komplexer Vektor
[*Matrix*]	Reelle oder komplexe Matrix
[*R-Feld*]	Reelle Matrix oder reeller Vektor
[*K-Feld*]	Komplexe Matrix oder komplexer Vektor
{ *Liste* }	Liste mit Objekten
{ *Index* }	Liste mit einer oder zwei reellen Zahlen, die das Element eines Felds spezifizieren.
{ *Dimension* }	Liste mit einer oder zwei reellen Zahlen, welche die Dimension(en) eines Felds spezifizieren.
' *Name* '	Name oder lokaler Name
« *Programm* »	Programm
' *Symbol* '	Ausdruck, Gleichung oder Name, der als algebraischer Ausdruck behandelt wird.

Das Stackdiagramm eines Befehls kann mehr als nur eine "Argument ➧ Ergebnis" Zeile enthalten, was die vielen möglichen Kombinationen von Argumenten und Ergebnissen reflektiert. Wo es geeignet erscheint, sind die Ergebnisse in einer Form dargestellt, aus welcher die mathematische Kombination der Argumente hervorgeht. So zeigt z.B. das Stackdiagramm für die Operation + die folgenden Einträge (unter anderen):

+ **Addieren** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	z_1+z_2
⊏ *Feld*$_1$ ⊐	⊏ *Feld*$_2$ ⊐	➧	⊏ *Feld*$_1$ + *Feld*$_2$ ⊐
z	' *Symbol* '	➧	' z + ⟨ *Symbol* ⟩ '

Dieses Diagramm zeigt:

- Die Addition zweier reeller oder komplexer Zahlen z_1 und z_2 gibt eine dritte reelle oder komplexe Zahl mit dem Wert z_1+z_2 in den Stack zurück.
- Die Addition zweier Felder ⊏ *Feld*$_1$ ⊐ und ⊏ *Feld*$_2$ ⊐ gibt ein drittes Feld ⊏ *Feld*$_1$ + *Feld*$_2$ ⊐ zurück.
- Die Addition einer reellen oder komplexen Zahl z und einem symbolischen Objekt ' *Symbol* ' gibt ein symbolisches Objekt ' z + ⟨ *Symbol* ⟩ ' in den Stack zurück.

1

Grundlagen

Der HP-28S basiert auf wenigen Grundprinzipien. Diese Prinzipien sind etwas abstrakt, aber ihre Allgemeingültigkeit ist der Schlüssel zu der Leistungsfähigkeit und Flexibilität des Rechners. Es ist kein tiefgreifendes Verständnis des Rechners notwendig, um ihn benutzen zu können; ein geringes Verständnis seiner Prinzipien macht Ihnen jedoch seine volle Leistungsfähigkeit zugänglich.

Falls Sie bisher noch nicht mit dem HP-28S gearbeitet haben, sollten Sie sich zunächst mit dem *HP-28S Benutzerhandbuch* vertraut machen. Dieses Handbuch gibt Ihnen Schritt-für-Schritt Anweisungungen, um typische Aufgaben lösen zu können. Sobald Sie etwas Erfahrung haben, können Sie hierauf zurückgreifen, um die Arbeitsweise des Rechners in einem allgemeineren Zusammenhang zu verstehen.

Dieses Kapitel beginnt mit einer allgemeinen Aussage über die Arbeitsweise des Rechners, mit anschließenden Abschnitten, welche diese allgemeine Aussage näher ausführen. Spätere Abschnitte beschreiben Stack, Modi und Fehler. Informationen über Objekteingabe, Variable und andere grundlegende Themen sind in Kapitel 2, "Grundlegende Operationen", beschrieben. Informationen über individuelle Operationen und Befehle, welche in Menüs zusammengefaßt sind, erscheinen in Kapitel 3, "Schlüsselwortverzeichnis".

Grundprinzipien des Rechners

Die Funktionsweise des Rechners basiert auf der Auswertung der Objekte im Stack. Objekte können Daten, Namen oder Prozeduren sein. Die Auswertung eines Objekts bedeutet die Ausführung der mit dem Objekt verbundenen Aktion. *Daten-Objekte bewirken nichts Besonderes (es sind lediglich Daten), Namen-Objekte beziehen sich auf andere Objekte, und Prozedur-Objekte verarbeiten die Objekte und Befehle nach ihren eigenen Definitionen.*

Ein Vorteil dieses Prinzips ist *Einheitlichkeit*. In Operationen wie Eingeben, Ausgeben, Kopieren, Speichern und Zurückrufen werden alle Objekte gleich behandelt. Durch diese Einheitlichkeit müssen Sie sich weniger Regeln merken.

Ein weiterer Vorteil ist *Flexibilität*. Objekte können in beliebiger Anzahl kombiniert werden, um die Mittel zu erzeugen, die Sie zur Lösung einer bestimmten Aufgabe benötigen. Da Sie frei wählen können, wann—sofern überhaupt— Sie ein symbolisches Objekt auswerten wollen, können Sie ein Problem symbolisch sowie auch numerisch bearbeiten.

Daten-Objekte

Diese Objekte stellen Daten dar, welche wie logische Einheiten behandelt werden: numerische Daten, Strings (Zeichenketten) und Objektlisten.

Daten-Objekte

Typ	Objekt	Bedeutung
Reelle Zahl	Reelle Zahl	Reelle Gleitkommazahl mit reellem Wert
Komplexe Zahl	Komplexe Zahl	Gleitkommazahl mit Real- und Imaginärwert
Binärwert	Binärwert	64-bit umfassende Binärzahl
String	String	Zeichenkette
Reelles Feld	Reeller Vektor	Reeller Vektor mit n Elementen
	Reelle Matrix	Reelle Matrix mit $n \times m$ Elementen
Komplexes Feld	Komplexer Vektor	Komplexer Vektor mit n Elementen
	Komplexe Matrix	Komplexe Matrix mit $n \times m$ Elementen
Liste	Liste	Objektliste

Die Auswertung eines Daten-Objekts bewirkt keine Veränderung. Befindet sich ein Daten-Objekt im Stack und Sie drücken EVAL, so verbleibt das Objekt einfach im Stack. Bitte beachten Sie, daß bei der Auswertung einer Liste die darin enthaltenen Objekte selbst nicht ausgewertet werden.

Namen-Objekte

Diese Objekte ordnen den im Benutzerspeicher gespeicherten Objekten einen Namen zu. Prozeduren können *lokale Namen* erzeugen, welche nach Abschluß der Prozedur automatisch wieder gelöscht werden.

Namen-Objekte

Typ	Objekt	Bedeutung
Name	Name	Bezieht sich auf ein Objekt im Benutzerspeicher
	Lokaler Name	Bezieht sich auf ein Objekt, welches temporär im Benutzerspeicher gespeichert wird.

Variable

Eine Variable ist die Kombination eines willkürlichen Objekts mit einem Namen, welche zusammen abgespeichert werden. Der Name wird zum *Variablennamen;* das andere Objekt ist der *Wert* oder *Inhalt* der Variablen. Sie werden zusammen im *Benutzerspeicher* abgespeichert, welcher vom Stack getrennt ist. Im HP-28S ersetzen Variablen die numerierten Datenregister und Programmspeicher, welche man in den meisten Rechnern vorfindet.

Es gibt zwei Aspekte bei der Auswertung von Variablennamen: Was veranlaßt die Auswertung eines Namens und welcher Art ist das Ergebnis der Auswertung.

Wannn wird ein Name ausgewertet?

- Ein Name in einem USER Menüfeld (Benutzermenü) wird im unmittelbaren Eingabemodus durch Drücken der Menütaste ausgewertet.
- Ein nicht in Anführungszeichen stehender Name in der Befehlszeile wird bei der Auswertung der Befehlszeile ausgewertet.
- Ein nicht in Anführungszeichen stehender Name in einer Prozedur wird bei der Auswertung der Prozedur ausgewertet.
- Ein in einer Variablen enthaltener Name wird bei der Auswertung des Variablennamens ausgewertet.
- Ein Name in Ebene 1 wird bei der Ausführung des EVAL Befehls ausgewertet.

Ergebnisse nach der Auswertung von Namen:

- Die Auswertung eines Variablennamens stellt das gespeicherte Objekt in den Stack und wertet es aus, sofern es ein Name oder ein Programm ist.
- Die Auswertung eines Namens, welcher nicht einer Variablen entspricht, stellt den Namen wieder in den Stack zurück.

Zusätzlich zum soeben Gesagten ist zu beachten:

- Ein in einer Variablen enthaltenes Daten-Objekt läßt sich einfach zurückrufen, indem der Variablenname ausgewertet wird.
- Ein nicht in Anführungszeichen stehender Name, welcher sich auf ein Programm bezieht, wirkt wie ein Befehl zur Auswertung des Programms.
- Falls ein Name sich auf einen anderen Namen bezieht, der sich wiederum auf einen weiteren Namen bezieht, usw., dann bewirkt das Auswerten des ersten Namens die Auswertung aller weiterer Namen.

Hinweis

Definieren Sie keine Variable, welche ihren eigenen Namen beinhaltet, ein Fall, welcher durch Ausführen von `'X' 'X' STO` oder `'X+Y' 'X' STO` eintritt. Die Auswertung einer solchen Variablen bewirkt eine Endlos-Schleife. Um eine Endlos-Schleife abzubrechen, müssen Sie einen Systemstopp durchführen (ON ▲, wie unter "Grundlegende Operationen" beschrieben), wodurch gleichzeitig auch der Stack gelöscht wird.

Definieren Sie ebenso keine Variablen, welche eine kreisförmige Beziehung zueinander haben. Die Auswertung dieser Variablen würde ebenfalls zu einer Endlos-Schleife führen.

Lokale Variable

Lokale Variable werden nur innerhalb der Programmstruktur, welche die Variable erzeugt, angewandt. So verwenden z.B. benutzerdefinierte Funktionen und FOR...NEXT Programmstrukturen lokale Variable. Namen, welche lokale Variable identifizieren, werden als *lokale Namen* bezeichnet und sind unter "Programme" beschrieben. Beim Auswerten eines lokalen Namens wird einfach der Inhalt der lokalen Variablen in den Stack übertragen.

Formale Variable

In symbolischen Berechnungen können Namen-Objekte als Variable verwendet werden (im mathematischen Sinne), bevor den Variablen ein Wert zugeordnet wird. Wenn Sie ein symbolisches Ergebnis—z.B. die Ableitung eines Ausdrucks—anstreben, kann es vorkommen, daß Sie auf die Zuweisung eines numerischen Werts ganz verzichten.

Namen, welche als mathematische Variablen benutzt werden, aber nicht mit gespeicherten Objekten in Verbindung stehen, werden als formale Variable bezeichnet. Dieser Ausdruck wird nur benutzt, wenn diese Unterscheidung von Bedeutung ist. Nach der Auswertung einer formalen Variablen verbleibt ihr Name im Stack.

Prozedur-Objekte

Diese Objekte enthalten *Prozeduren*—Objektfolgen und Befehle, welche gleichzeitig mit der Auswertung der Prozedur-Objekte verarbeitet werden. Ein *Programm-Objekt* kann jede beliebige Folge von Objekten und Befehlen enthalten, auch solche, welche einen Einfluß auf den Stack, den Benutzerspeicher oder die Rechner-Modi haben. Ein *algebraisches Objekt* enthält eine begrenzte Anzahl an Objekttypen und Befehlen, und seine Syntax hat Ähnlichkeit mit mathematischen Ausdrücken und Gleichungen.

Prozedur-Objekte

Typ	Objekt	Bedeutung
Programm	Programm	Enthält jede beliebige Objektfolge.
Algebraisch	Ausdruck	Enthält einen mathematischen Ausdruck.
	Gleichung	Enthält eine mathematische Gleichung, welche zwei Ausdrücke in Beziehung setzt.

Programme

Ein Programm ist im Grunde genommen die Objektform einer Befehlszeile. Die in der Befehlszeile eingegebenen Objekte und Befehle bilden eine Prozedur. Durch Einschluß einer Prozedur in Programm-Begrenzungszeichen wird angedeutet, daß die Prozedur wie ein Objekt behandelt werden soll, welches später ausgewertet wird.

Wann wird ein Programm ausgewertet?

- Ein Programm in Ebene 1 kann durch Ausführen des EVAL Befehls ausgewertet werden.
- Ein Programm, welches in einer Variablen gespeichert ist, wird bei der Auswertung der Variablen ausgewertet.
- Befehle wie DRAW, ∫ und ROOT werten ein Programm wiederholt aus.
- Ein Programm, welches der Prozedurteil einer lokalen Variablen Struktur ist, wird bei der Auswertung der Struktur ausgewertet.

Ergebnisse nach der Auswertung von Namen:

- Die Auswertung eines Programms stellt jedes Objekt in den Stack und, im Falle eines Befehls oder eines nicht in Anführungszeichen stehenden Namens, wertet es aus.

Zusätzlich zum soeben Gesagten ist zu beachten:

- Angenommen, ein Programm enthält einen nicht in Anführungszeichen stehenden Namen, welcher sich auf ein weiteres Programm bezieht, und dieses Programm enthält wiederum einen nicht in Anführungszeichen stehenden Namen, der sich auf ein weiteres Programm bezieht, usw., so werden durch Auswertung des ersten Programms alle weiteren Programme ausgewertet. (Das zuletzt erwähnte ist das zuerst vollständig ausgewertete.)
- Durch Ausführen eines Befehls, der ein Programm zuordnet, kann es passieren, daß die Befehle in diesem Programm die ursprünglichen Argumente, welche in LAST gespeichert sind, überschreiben.

Ausdrücke

Ein Ausdruck ist eine Prozedur, welche einen mathematischen Ausdruck darstellt. Die Eingabe und Anzeige des Ausdrucks erfolgt analog der Syntax herkömmlicher mathematischer Formen.

Wann wird ein Ausdruck ausgewertet?

- Ein Ausdruck kann durch Ausführen eines EVAL Befehls ausgewertet werden. (Die Auswertung von Ausdrücken ist der häufigste Gebrauch von EVAL.)

- Befehle wie DRAW, ∫, ROOT, TAYLR und QUAD werten einen ihnen als Argument zugeordneten Ausdruck wiederholt aus.
- Ein Ausdruck, welcher eine durch den Benutzer definierte Funktion bestimmt, wird bei der Auswertung der Funktion ausgewertet.

Ergebnisse nach der Auswertung von Namen:

- Die Auswertung eines Ausdrucks stellt jedes Objekt in den Stack und wertet es aus. Diese Objekte werden in UPN-Reihenfolge (die Folge des äquivalenten Programms) ausgewertet und nicht wie sie im Ausdruck erscheinen.

Zusätzlich zum soeben Gesagten ist zu beachten:

- Obwohl die Auswertung eines Namens, welcher sich auf ein Programm bezieht, das Programm auswertet, wird bei der Auswertung eines Namens, welcher sich auf einen Ausdruck bezieht, der Ausdruck in den Stack übernommen.
- Falls ein Name in einem Ausdruck sich auf einen zweiten Ausdruck bezieht, wertet der erste Ausdruck *nicht* auch den zweiten Ausdruck aus. Stattdessen wird der zweite Ausdruck jedesmal eingesetzt, wenn dessen Name im ersten Ausdruck erscheint.

Gleichungen

Gleichungen sind Ausdrücke, welche durch ein Gleichheitszeichen "=" verbunden sind. Die Auswertung einer Gleichung führt zu einer neuen Gleichung. Der neue Ausdruck auf der linken Seite ist das Ergebnis der Auswertung der ursprünglichen linken Seite. Der neue Ausdruck auf der rechten Seite ist das Ergebnis der Auswertung der ursprünglichen rechten Seite.

Befehle

Befehle sind interne Prozeduren, welche in Programme mit einbezogen werden können. Ein Befehlsname, wie er in einer Befehlszeile erscheint (zum Beispiel: DROP oder SIN), kann als der nicht in Anführungszeichen stehende Name einer gespeicherten Prozedur angesehen werden. Dies ist mit den Namen und Inhalten Ihrer eigenen Variablen vergleichbar. In der Praxis existiert kein Unterschied zwischen einem nicht in Anführungszeichen stehenden Namen und einer internen Prozedur.

Eingebaute Prozeduren werden entsprechend ihres Gebrauchs klassifiziert:

- Eine *Operation* ist jede interne Prozedur des Rechners wie ENTER, CATALOG oder TRACE.
- Ein *Befehl* ist eine programmierbare Operation wie z.B. SWAP oder STO.
- Eine *Funktion* ist ein in Algebra zulässiger Befehl wie IP oder MIN.
- Eine *analytische Funktion* ist eine Funktion, für welche der Rechner eine Ableitung oder Umkehrung zur Verfügung stellt, wie SIN oder +.

Eingebaute Prozeduren werden normalerweise durch ihre wichtigste Fähigkeit charakterisiert. Zum Beispiel stellt SWAP einen Befehl und eine Operation dar, während IP Funktion, Befehl und Operation darstellt. SWAP wird jedoch als Befehl und IP als Funktion charakterisiert.

Der "Stack"

Der Stack ist eine Folge von numerierten *Ebenen*, von welchen jede ein Objekt enthält. Objekte werden auf Ebene 1 eingegeben, wodurch bereits im Stack befindliche Objekte nach oben verschoben werden. Objekte werden in Ebene 1 vom Stack genommen, wodurch im Stack verbleibende Objekte nach unten verschoben werden. Alle Objekte im Stack werden identisch behandelt—einfach als Objekte.

Der Rechner stellt Befehle zur Verfügung, um Objekte im Stack zu kopieren, zu löschen oder neu zu ordnen. Viele dieser Befehle sind über das Tastenfeld aufrufbar ([DROP], ■[SWAP], ■[ROLL] und ■[CLEAR]); andere befinden sich im Stack-Menü.

Die meisten Befehle nehmen Eingabe-Objekte (*Argumente* genannt) aus dem Stack und geben Ausgabe-Objekte (*Ergebnisse* genannt) wieder in den Stack zurück. Argumente müssen im Stack vorhanden sein bevor der Befehl ausgeführt wird. Der Befehl entfernt seine Argumente und tauscht sie gegen Ergebnisse aus. Zum Beispiel nimmt die Funktion SIN einen Wert (eine reelle oder komplexe Zahl oder einen algebraischen Ausdruck) aus Ebene 1, berechnet seinen Sinus und gibt das Ergebnis wieder in Ebene 1 zurück. Die Funktion + nimmt zwei Werte aus dem Stack und gibt deren Summe zurück.

Diese Art der Logik, bei welcher der Befehl auf die Argumente folgt, wird *postfix logic* oder *UPN* für *Umgekehrte Polnische Notation* genannt, nach dem polnischen Mathematiker Jan Lukasiewicz (1878-1956).

(Bitte beachten Sie, daß diese UPN Befehle Operationen beinhalten, die in manchen UPN Rechnern eine vorangestellte Syntax benutzen. Zum Beispiel muß für das FIX Anzeigeformat mit zwei Nachkomma-Stellen auf dem HP-28S die Tastenfolge `2 FIX` gewählt werden. Auf ähnliche Weise muß, um die Zahl 12 unter einer Variablen mit dem Namen FIRST abzuspeichern, die Tastenfolge `12 'FIRST' STO` gewählt werden.)

Modi

Durch die Wahl des Modus können bei vielen Operationen die Ergebnisse gesteuert werden. Zum Beispiel können Sie steuern, ob trigonometrische Funktionen Zahlen als Grad oder Bogenmaß interpretieren, indem Sie für den Winkelmodus Grad oder Bogenmaß spezifizieren.

Die folgende Tabelle zeigt die verwendeten Modi, gruppiert nach verwandten Themen. Die Voreinstellung für jeden Modus, welche nach jedem Memory Reset durchgeführt wird, ist durch ein Sternchen gekennzeichnet. Die meisten Modi sind durch einen Flag, einen Indikator oder ein Menüfeld gekennzeichnet. Der dritte Fall tritt ein, wenn der Modus durch Menütasten gewählt wird: Das Menüfeld für die gerade vorgenommene Wahl erscheint in dunklen Zeichen, nicht in inverser Darstellung, die für Menüfelder typisch ist. Das Drücken einer Menütaste, welches eine Veränderung des Modus bewirkt, läßt gleichzeitig das Menüfeld in inverser Zeichendarstellung erscheinen.

HP-28C/S Modi

Modus	Auswahl	Indikator
Winkel	Grad*/Bogenmaß	MODE-Menü Kennzeichen, Flag 60 gelöscht*/gesetzt, (2π) Indikator
Akustiksignal	Aktiviert*/desaktiviert	Flag 51 gelöscht*/gesetzt
Hauptwert	Aus*/ein	Flag 34 gelöscht*/gesetzt

* Voreinstellung

HP-28C/S Modi (*Fortsetzung*)

Modus	Auswahl	Indikator
Allgemeine Eingabe und Anzeige		
Eingabemodus	Unmittelbarer*/ algebraischer/Alpha	Cursor, α Indikator
Schreibweise	Groß*/klein	Keiner
Anzeige in Ebene 1	Mehrfachzeile*/ kompakt	MODE-Menü, Flag 45, gesetzt*/gelöscht
Eingabe und Anzeige von reellen Zahlen		
Dezimalzeichen	Punkt*/Komma	MODE-Menü, Flag 48, gelöscht*/gesetzt
Format	Standard*/fest/wissensch./technisch	MODE-Menü, Flags 49–50
Anzahl Dezimalstellen	0* bis 11	Flags 53–56
Binärwert-Eingabe und -Anzeige		
Binärwert-Basis	Dezimal*/Hexadezimal/ Oktal/Binär	BINARY-Menü, Flags 43–44,
Binärwert-Wortlänge	1 bis 64*	Flags 37–42
Rücksicherung		
COMMAND	Aktiviert*/desaktiviert	MODE-Menü
UNDO	Aktiviert*/desaktiviert	MODE-Menü
LAST	Aktiviert*/desaktiviert	MODE-Menü, Flag 31, gesetzt*/gelöscht
Auswertung		
Auswertung von symbolischen Konstanten	Symbolisch*/numerisch	Flag 35 gesetzt*/gelöscht
Auswertung von Funktionen	Symbolisch*/numerisch	Flag 36 gesetzt*/gelöscht
Drucker		
Protokoll	Desaktiviert*/aktiviert	PRINT-Menü, Flag 32, gelöscht*/gesetzt
Druckkopf rechts	Aktiviert*/desaktiviert	Flag 33 gelöscht*/gesetzt
Schnelldruck	Desaktiviert*/aktiviert	Flag 52 gelöscht*/gesetzt

* Voreinstellung

Indikatoren

Indikatoren am oberen Ende der Anzeige weisen auf den Winkelmodus, den Eingabemodus und andere Statusinformationen hin.

Indikatoren

Indikator	Bedeutung
O	Eine Programmausführung wurde ausgesetzt.
⬏	Die Umschalttaste wurde gedrückt.
α	Der Alpha-Eingabemodus wurde aktiviert.
((•))	Der Rechner ist am Arbeiten—das heißt, es ist keine Eingabe über das Tastenfeld möglich.
▭	Schwache Batterie.
(2π)	Als Winkelmodus wurde Bogenmaß gewählt.
🖷	Der Rechner sendet Daten an den Drucker.

Flags

Ein *Flag* steht für eine logische Einheit, die den Wert *richtig* oder *falsch* annehmen kann. Flags treten als *numerische* Flags oder als *Benutzerflags* auf.

Numerische Flags. Im Stack stellt eine reelle Zahl ungleich Null *wahr* und die Zahl 0 *falsch* dar. Numerische Flags werden zusammen mit Programm-Verzweigungsstrukturen, wie IF...THEN...ELSE, und mit logischen Tests, wie XOR, benutzt.

Benutzerflags. Ein getrennter Teil des Speichers im Rechner enthält Benutzerflags, von welchen jeder in zwei möglichen Formen auftritt: *gesetzt* (wahr) oder *gelöscht* (falsch). Der Wert *wahr* wird im Flag durch Setzen des Flags und der Wert *falsch* durch Löschen des Flags gespeichert. Der Wert im Flag kann getestet werden. Dabei wird die entsprechende numerische Zahl, 0 (falsch) oder 1 (wahr), in den Stack zurückgegeben.

Es gibt 64 Benutzerflags mit den Nummern 1 bis 64. Die Flags 1 bis 30 dienen dem allgemeinen Gebrauch. Die Flags 31—64 haben die nachstehenden Spezialbedeutungen—durch Setzen oder Löschen der Flags werden die mit ihnen verbundenen Modi verändert.

Reservierte Benutzerflags

Nummer	Bedeutung	Vorgabe
31	LAST aktivieren	Gesetzt
32	Protokoll	Gelöscht
33	Druckkopf rechts	Gelöscht
34	Hauptwert	Gelöscht
35	Symbolische Auswertung von Konstanten	Gesetzt
36	Symbolische Auswertung von Funktionen	Gesetzt
37-42	Binärwert-Wortlänge	Gesetzt
43-44	Binärwert-Basis	Gelöscht
45	Anzeige in Ebene 1	Gesetzt
46	Reserviert	Gelöscht
47	Reserviert	Gelöscht
48	Dezimalzeichen	Gelöscht
49-50	Format einer reellen Zahl	Gelöscht
51	Akustiksignal	Gelöscht
52	Schnelldruck	Gelöscht
53-56	Anzahl Dezimalstellen	Gelöscht
57	`Underflow` Maßnahme	Gelöscht
58	`Overflow` Maßnahme	Gelöscht
59	`Infinite Result` Maßnahme	Gesetzt
60	Winkel	Gelöscht
61	`Underflow-` Ausnahme	Gelöscht
62	`Underflow+` Ausnahme	Gelöscht
63	`Overflow` Ausnahme	Gelöscht
64	`Infinite Result` Ausnahme	Gelöscht

Fehler und Ausnahmen

Bei Auftreten eines Fehlers gibt der Rechner ein Akustiksignal und eine Fehlermeldung in der obersten Zeile der Anzeige aus. Fehlermeldungen werden in Anhang A, "Meldungen" beschrieben.

Tritt der Fehler während der Ausführung eines Befehls, welcher Argumente vom Stack nimmt, auf, so werden die Argumente bei aktiviertem LAST in den Stack zurückgespeichert. Die Argumente gehen verloren, wenn LAST desaktiviert ist.

Fehler in der Befehlszeile

Ein Fehler kann während ENTER, d.h. während der Rechner den Text in der Befehlszeile verarbeitet, auftreten. Ist das der Fall, gibt der Rechner ein Akustiksignal aus und zeigt `Syntax Error` an; er gibt die Befehlszeile wieder aus und versucht das Problem aufzuzeigen. Wurde der Fehler durch eine unzulässige Syntax verursacht, erscheint der unkorrekte Text in inverser Darstellung, gefolgt vom Cursor. Wenn eine unvollständige Eingabe die Fehlerursache war, dann steht der Cursor am Ende der Zeile.

Fehler in Programmen

Tritt ein Fehler in einem Programm auf, dann wird das Programm an dieser Stelle abgebrochen. Wurde die Auswertung des Programms durch ein anderes Programm begonnen, wird auch dieses Programm abgebrochen.

Mathematische Ausnahmen

Gewisse Fehler, welche während gewöhnlichen Rechnungen mit reellen Zahlen auftreten können, werden als *mathematische Ausnahmen* klassifiziert. Eine Ausnahme kann wie ein gewöhnlicher Fehler die Unterbrechung einer Berechnung bewirken, oder sie kann ein Standard (Vorgabe)-Ergebnis liefern, wodurch die Fortsetzung der Berechnung möglich ist. Die Auswirkungen der Ausnahmen können durch Setzen oder Löschen der Flags 57, 58 oder 59 gewählt werden. Die folgende Liste beschreibt die mathematischen Ausnahmen und die entsprechenden Flags.

Mathematische Ausnahmen

Ausnahme	Bedeutung
Infinite Result (unendliches Ergebnis)	Diese Ausnahme tritt auf, wenn das Ergebnis einer Berechnung 'unendlich' wäre, wie z.B. LN(0), TAN(90°) und Division durch Null. Falls Flag 59 *gesetzt ist, werden Infinite Result Ausnahmen als Fehler behandelt. Falls Flag 59 gelöscht ist, gibt Infinite Result das Standardergebnis ±MAXR zurück und setzt Flag 64, den Infinite Result Indikator.
Overflow (Überlauf)	Diese Ausnahme tritt auf, wenn das Ergebnis einer Berechnung ein absoluter Wert wäre, der größer ist, als die größte im Rechner zulässige Zahl MAXR. Z.B. 9E499 + 9E499, EXP(5000), FACT(2000). Falls Flag 58 gesetzt ist, werden Overflow Ausnahmen als Fehler behandelt. Falls Flag 58 * gelöscht ist, gibt Overflow das Standardergebnis ±MAXR zurück und setzt Flag 63, den Overflow Indikator.
Underflow (Unterlauf)	Diese Ausnahme tritt auf, wenn das Ergebnis der Berechnung ein absolutes Ergebnis wäre, dessen Wert kleiner ist als die kleinste im Rechner zulässige Zahl MINR. Z.B. 1E-499/2 und EXP(-5000). Falls Flag 57 gesetzt ist, wirkt eine Underflow Ausnahme wie ein Fehler. Je nach Vorzeichen des tatsächlichen Ergebnisses gibt sie die Fehlermeldung Negative Underflow oder Positive Underflow aus. Falls der Flag 57 * gelöscht ist, gibt Underflow das Standardergebnis 0 aus und setzt je nach Vorzeichen des tatsächlichen Ergebnisses Flag 62, den Underflow+ Indikator, oder Flag 61, den Underflow- Indikator.

* Voreinstellung

2

Grundlegende Operationen

Dieses Kapitel beschreibt die Eingabe von Objekten in die Befehlszeile und das Erzeugen, Aufrufen und Löschen von Variablen. Weiterhin wird das Zurückrufen vorhergehender Befehlszeilen, Stack-Einträge und Argumente, sowie der Umgang mit zu wenig freiem Speicherplatz erläutert. Eine Beschreibung der Ausführung von Systemoperationen schließt das Kapitel ab.

Eingabe von Objekten

Sobald Sie eine Taste drücken, um mit der Eingabe von Objekten zu beginnen, wird das zugehörige Zeichen in die *Befehlszeile* übernommen. Die Befehlszeile kann eine beliebige Anzahl von Objekten enthalten, die in Textform dargestellt sind. Sie erscheint am unteren Rand der Anzeige (über den Menüfeldern, falls diese vorhanden sind). Die Befehlszeile erscheint auch, wenn Sie ■ EDIT oder ■ VISIT drücken, um den Inhalt eines schon existierenden Objekts anzusehen oder abzuändern.

Der Inhalt der Befehlszeile wird verarbeitet, sobald Sie ENTER (oder irgendeine Befehls- oder Funktionstaste, die automatisch ENTER ausführt) drücken. Er wird als Programm ausgewertet, und die Befehlszeile verschwindet aus der Anzeige.

Sie können eine beliebige Anzahl von Zeichen in die Befehlszeile eingeben. Außerdem können Sie die Zeile in mehrere Reihen aufteilen indem Sie ■ NEWLINE drücken. Dies fügt ein "Newline"-Zeichen (Zeilenvorschub) an der momentanen Cursor-Position in die Befehlszeile ein. "Newline"-Zeichen dienen zur Trennung von Objekten, werden jedoch bei der Auswertung der Befehlszeile ignoriert.

Wenn Sie mehr als 23 Zeichen in die Befehlszeile eingeben, werden sie über den linken Anzeigerand hinaus verschoben. Eine Ellipse (...) erscheint in der linken Anzeigeposition und deutet auf die Existenz nicht dargestellter Zeichen hin. Sobald Sie versuchen, den Cursor über den linken Anzeigerand hinaus zu bewegen, kommen die "versteckten" Zeichen wieder zum Vorschein, während Zeichen am rechten Anzeigerand verschwinden. In diesem Fall erscheint die Ellipse am rechten Anzeigerand. Enthält die Befehlszeile mehrere Textreihen, so werden alle zusammen nach links oder rechts verschoben.

Eingabe von Zahlen

Die Eingabe von reellen Zahlen erfolgt mit Hilfe der Zifferntasten, [CHS] und [EEX]. Zifferntasten fügen jeweils eine einzelne Ziffer in der Befehlszeile ein.

Vorzeichenwechsel ([CHS]): Das Drücken der Taste [CHS] (*change sign*) ändert das Vorzeichen einer Zahl in der Befehlszeile. (Wenn keine Befehlszeile vorhanden ist, führt das Drücken von [CHS] den Befehl NEG aus, der das Objekt in Ebene 1 negiert. NEG wird in "Arithmetik" beschrieben.)

"Zahl" bedeutet entweder Mantisse oder Exponent einer Zahl—die Position des Cursors bestimmt, was geändert wird. Wenn kein Vorzeichen vorhanden ist, wird ein Minuszeichen (−) vor der Zahl eingefügt. Ist ein positives (+) oder negatives Vorzeichen vorhanden, so wird es invertiert.

Wenn Sie eine negative Zahl als erstes Objekt einer Befehlszeile eingeben wollen, so müssen Sie zuerst wenigstens eine Ziffer eingeben, um die Befehlszeile zu erzeugen. Danach können Sie [CHS] drücken. In allen anderen Fällen können Sie [CHS] vor, während oder nach Eingabe einer Zahl drücken.

Eingabe des Exponenten ([EEX]): Sehr große oder sehr kleine Zahlen können unter Verwendung der wissenschaftlichen Darstellungsweise eingegeben werden. Eine Zahl kann durch *Mantisse* und *Exponent* dargestellt werden, wobei sich der Zahlenwert als Produkt von Mantisse und 10, potenziert mit dem Exponenten, ergibt.

Um eine Zahl in wissenschaftlicher Darstellung einzugeben, tippen Sie die Mantisse ein, drücken [EEX] (*enter exponent bzw. Eingabe Exponent*) und tippen dann den Exponenten ein. Das Drücken von [EEX] fügt das Zeichen E in die Befehlszeile ein und trennt damit Mantisse und Exponent.

Wenn der Cursor nicht auf einer zulässigen Zahl steht (oder keine Befehlszeile vorhanden ist), erzeugt das Drücken von [EEX] die Zeichenfolge 1E in der Befehlszeile. Ist der Cursor auf einer Zahl positioniert, die schon einen Exponenten besitzt, so wird durch Drücken von [EEX] der Cursor zur ersten Ziffer des Exponenten bewegt.

Rückschrittaste: [←]

Durch Drücken von [←] wird das Zeichen links vom Cursor gelöscht und der Cursor (sowie alle Zeichen rechts davon) um eine Stelle nach links bewegt. Das Niederhalten von [←] wiederholt diesen Vorgang. Das Drücken von [←] hat keine Auswirkung, wenn der Cursor am linken Zeilenanfang steht.

Kleinbuchstaben: [LC]

Wenn Sie die Taste [LC] (*lower case*) drücken, erzeugen die Buchstabentasten [A] bis [Z] die Zeichen a bis z in der Befehlszeile. Die Eingabe von Kleinbuchstaben wird fortgesetzt, bis [LC] erneut gedrückt, ENTER ausgeführt oder [ON] gedrückt wird, um die Befehlszeile zu löschen.

Begrenzungs- und Trennzeichen für Objekte

Die verschiedenen Objekt-Typen werden in derselben Form eingegeben, in der sie angezeigt werden. Aufeinanderfolgende Objekte oder Befehle, welche in der Befehlszeile eingegeben werden, müssen voneinander getrennt werden durch:

- Ein Objekt-Begrenzungszeichen ‹, ›, [,], {, }, #, ", ', «, ».
- Ein Leerzeichen oder eine neue Zeile ("Newline").
- Ein Punkt oder ein Komma, je nachdem, welches Zeichen gerade *nicht* als Dezimalzeichen verwendet wird. (Wenn Flag 48 gelöscht ist, sind Punkte Dezimalzeichen und Kommas Trennzeichen; ist er gesetzt, so sind Kommas Dezimalzeichen und Punkte Trennzeichen.)

In algebraischen Objekten werden Leerzeichen ignoriert (außer bei den Operatoren AND, OR, XOR und NOT); Argumente innerhalb von Klammern (wie zum Beispiel MOD(A,B)) müssen durch das momentane Trennzeichen, entweder Punkt oder Komma, getrennt werden.

Wenn Sie ein Objekt eingeben, müssen Sie der korrekten Syntax für dieses Objekt folgen. Die meisten Objekttypen beginnen und enden mit *Begrenzungszeichen*—speziellen Interpunktionszeichen, die das Objekt identifizieren. So sind zum Beispiel Strings mit doppelten Anführungszeichen begrenzt, wie in `"Guten Tag"` oder `"m^2"`, und Vektoren sind durch eckige Klammern begegrenzt, wie in `[ 1 2 3 ]` oder `[ -5 6 7 -10.2 ]`. Der Rechner folgt denselben Formatregeln, wenn er die Objekte anzeigt.

Die folgende Tabelle ist eine erweiterte Version der Tabelle, die oberhalb des linken Tastenfeldes auf dem Rechner wiedergegeben ist. Beide zeigen die entsprechenden Begrenzungszeichen für verschiedene Objekte. (Beachten Sie, daß in den Beispielen als Dezimalzeichen ein Punkt verwendet wurde.)

Objekt-Formate

Objekt	Format	Beispiel
Reelle Zahl	*reell*	`-1.234E24`
Komplexe Zahl	⟨*reell*, *reell*⟩	`(1.23,4.56)`
Binärwert	# *Zeichen*	`# 123AF`
String	"*Text*"	`"HALLO THOMAS"`
Reeller Vektor	[*reell reell ...*]	`[1 2 3 4]`

Reelle Matrix	[[*reell reell ...*]
	[*reell reell ...*]
	⋮
	[*reell reell ...*]]
		`[[1 2 3 4]`
		`[5 6 7 8]`
		`[9 0 1 2]`
		`[3 4 5 6]]`

Komplexer Vektor	[⟨*reell*, *reell*⟩ ...]	`[(1,2) (3,4)]`
Komplexe Matrix	[[⟨*reell*, *reell*⟩ ...] [⟨*reell*, *reell*⟩ ...] ⋮ [⟨*reell*, *reell*⟩ ...]]	`[[(1,2) (3,4)]` `[(5,6) (7,8)]]`
Liste	⟨*Objekt Objekt ...* ⟩	`{1 "HP" (1,2)}`

Objekt-Formate (*Fortsetzung*)

Objekt	Format	Beispiel
Name	'*Name*'	'WOLFGANG'
Lokaler Name	'*Name*'	'GEORG'
Programm	«*Objekt Objekt ...* »	«DUP 4 ROLL»
Ausdruck	'*Ausdruck*'	'A+B'
Gleichung	'*Ausdruck=Ausdruck*'	'A+B=SIN(X)'

Fehlende Begrenzungszeichen am Ende der Befehlszeile werden automatisch hinzugefügt, sobald Sie [ENTER] drücken.

Wie der Cursor den jeweiligen Modus anzeigt

Die Form des Cursors zeigt den gegenwärtigen Eingabemodus und die Wahl für Einfügungs- oder Ersetzungsmodus. (Eingabemodi werden als nächstes beschrieben, gefolgt vom Cursor-Menü, das Einfügungs- und Ersetzungsmodus mit einschließt.) Die folgende Tabelle zeigt die sechs möglichen Kombinationen von Eingabemodus und Einfügungs- oder Ersetzungsmodus.

Eingabemodus	Einfügungsmodus	Ersetzungsmodus
Unmittelbar	▷	□
Algebraisch	▷	▤
Alpha	▷	▦

Eingabemodi

Es gibt drei Modi, um verschiedene Objekt-Typen einzugeben. Im allgemeinen wird der *unmittelbare Eingabemodus* für Daten-Objekte verwendet, der *algebraische Eingabemodus* für Namen-Objekte und algebraische Objekte und der *Alpha-Eingabemodus* für Programme und Zeichenketten. Sie können den Alpha-Eingabemodus jederzeit durch Drücken von [α] ein- oder ausschalten; in einigen Fällen wird er automatisch umgeschaltet, wenn Sie anfangen, ein neues Objekt einzugeben.

Der Eingabemodus bestimmt in erster Linie die Funktion der Tasten und der damit verbundenen Befehle—ob das Drücken einer Taste die Ausführung des Befehls oder das Hinzufügen des Befehlsnamens in die Befehlszeile bewirkt. In dieser Beschreibung schließt das Wort "Taste" auch umgeschaltete Tasten wie ■ π und zugeordnete Menütasten wie SIN mit ein.

Die folgenden Tasten werden nicht vom gegenwärtigen Eingabemodus beeinflußt:

- Tasten mit nicht programmierbaren Operationen wie ENTER, ■ CATALOG oder +ML. Das Drücken einer Operationstaste führt stets die Operation aus.
- Alle Tasten auf dem linken Tastenfeld, welche ein *einzelnes* Zeichen in der Befehlszeile wiedergeben (also auch SPACE, "Newline" und Tasten wie ■ >, die dem Namen der Funktion entsprechen, aber als Zeichentasten fungieren).
- Die Zeichentasten 0 bis 9, ., , und ■ π auf dem rechten Tastenfeld. Das Drücken der Taste fügt das Zeichen der Befehlszeile hinzu.
- CLUSR im USER-Menü, HALT im Programmsteuerungs-Menü (PROGRAM ConTRoL) und jede beliebige Taste im Programmverzweigungs-Menü (PROGRAM BRANCH). Das Drücken einer dieser Tasten fügt den Befehlsnamen der Befehlszeile hinzu.

Zusammen mit den Befehlstasten werden die Menütasten, die Variablennamen im USER-Menü zugeordnet sind, vom gegenwärtigen Eingabemodus beeinflußt. Der folgende Teil beschreibt die einzelnen Eingabemodi und wie diese die einzelnen Tastenarten beeinflussen. Die betroffenen Tasten befinden sich alle auf dem rechten Tastenfeld und sind hauptsächlich Menütasten.

Unmittelbarer Eingabemodus. Dies ist der vorgegebene Eingabemodus—eine neue Befehlszeile beginnt normalerweise in diesem Modus. Der Cursor erscheint als □ oder ◇. Im unmittelbaren Eingabemodus bewirkt das Drücken einer entsprechenden Taste:

- Eine Befehlstaste (wie STO) führt den Befehl aus.
- Eine Funktionstaste (wie +) führt die Funktion aus.
- Eine Variablentaste im USER-Menü wertet die jeweilige Variable aus.

Um die Tastenfolge zu reduzieren, bewirken die meisten Befehlstasten ENTER vor der eigentlichen Befehlsausführung. Ausnahmen von dieser Regel sind STD, DEG und RAD im MODE-Menü und DEC, HEX, OCT und BIN im BINARY-Menü; diese Tasten führen den Befehl aus, ohne die Befehlszeile zu beeinflussen.

Algebraischer Eingabemodus. Falls Sie sich im unmittelbaren Eingabemodus befinden, schalten Sie auf algebraischen Eingabemodus um, indem Sie die Taste ['] drücken. Der Cursor erscheint als ⊟ oder ÷. Wenn Sie ['] ein zweites Mal drücken, um das Objekt zu beenden, so sind Sie wieder im unmittelbaren Eingabemodus. Im algebraischen Eingabemodus gilt:

- Das Drücken einer Befehlstaste führt den Befehl aus, genau wie im unmittelbaren Eingabemodus.
- Das Drücken einer Funktionstaste fügt den Funktionsnamen zur Befehlszeile hinzu. Falls die Funktion die Eingabe des Arguments in Klammern erfordert, wie bei SIN(X), wird die Klammer geöffnet.
- Das Drücken einer Variablentaste im USER-Menü fügt den Variablennamen—ohne Anführungszeichen—zur Befehlszeile hinzu.

Alpha-Eingabemodus. Um zur Eingabe eines Programms oder Strings in den Alpha-Eingabemodus zu gelangen, drücken Sie [«] oder ["] (es erscheint der α Indikator). Der Cursor wird als ■ oder ✦ dargestellt. Wenn Sie sich nicht im Alpha-Eingabemodus befinden, können Sie durch Drücken von [α] zwischen momentanem Modus und Alpha-Modus umschalten. Das Drücken von [α LOCK] bewirkt eine permanente Einstellung des Alpha-Eingabemodus, wobei mit [α] diese Einstellung wieder aufgehoben werden kann.

Im Alpha-Eingabemodus gilt:

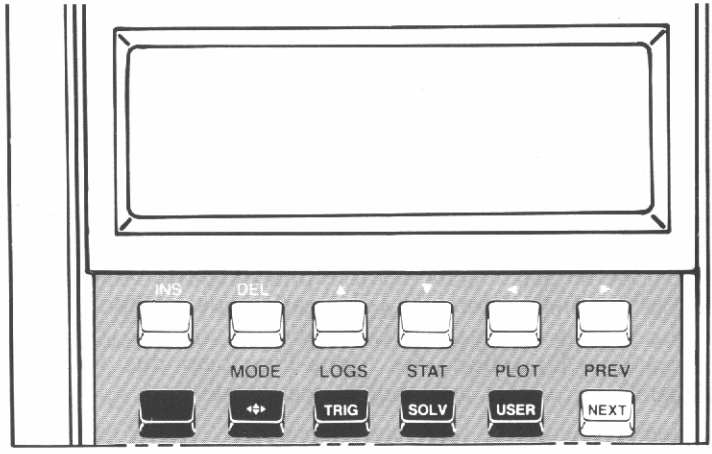
- Das Drücken einer Befehlstaste fügt den Befehlsnamen zur Befehlszeile hinzu.
- Das Drücken einer Funktionstaste fügt den Funktionsnamen zur Befehlszeile hinzu.
- Das Drücken einer Variablentaste im USER-Menü fügt den Variablennamen—ohne Anführungszeichen—zur Befehlszeile hinzu.

Befindet sich der Cursor im Einfügungsmodus oder am Ende der Befehlszeile, wenn Sie irgendeine der beschriebenen Tasten drücken, so wird zur Trennung der Befehle ein Leerzeichen vor und nach dem angehängten Text eingefügt.

Das Cursor-Menü: [◂▸]

Durch Drücken von [◂▸] wird den Menütasten das Cursor-Menü zugeteilt. In diesem Fall erscheinen keine Menüfelder in der Anzeige; stattdessen zeigt die weiße Beschriftung oberhalb der Tasten ihre Bedeutung an. Das Cursor-Menü stellt Editier-Operationen zur Verfügung, die über die Rückschrittfunktion ([◀]) hinausgehen. Wenn Sie [◂▸] erneut drücken, sind Sie wieder im vorhergehenden Menü, dessen Menüfelder nun in der Anzeige erscheinen.

Das Cursor-Menü beinhaltet sowohl die Funktion der einfachen als auch der umgeschalteten Tasten. Die weiße Beschriftung oberhalb der Tasten bezieht sich auf ihre einfache Funktion.



Die folgende Tabelle beschreibt die Bedeutung der einfachen Cursor-Menütasten. Wenn Sie irgendeine dieser Tasten (außer [INS]) drücken und niederhalten, wird die Operation solange wiederholt, bis Sie die Taste wieder freigeben.

Einfache Cursor-Menütasten

Taste	Bedeutung
INS	Schaltet zwischen Ersetzungs- und Einfügungsmodus (insert mode) um. Im Ersetzungsmodus treten neue Zeichen an Stelle der ehemaligen; der Cursor erscheint als □, ▯ oder ■. Im Einfügungsmodus werden neue Zeichen zwischen die bereits bestehenden eingefügt; der Cursor erscheint als ◁, ◆ oder ◆.
DEL	Löscht (delete) das Zeichen an der Cursor-Position.
▲	Bewegt den Cursor eine Zeile nach oben.
▼	Bewegt den Cursor eine Zeile nach unten.
◀	Bewegt den Cursor eine Position nach links.
▶	Bewegt den Cursor eine Position nach rechts.

Die nun folgende Tabelle beschreibt die Bedeutung der umgeschalteten Cursor-Menütasten. Mit Ausnahme von ■ INS entsprechen sie dem mehrfachen Drücken der einfachen Cursor-Menütasten.

Umgeschaltete Cursor-Menütasten

Taste	Bedeutung
■ INS	Löscht alle Zeichen links des Cursors.
■ DEL	Löscht das Zeichen an der gegenwärtigen Cursorposition und alle Zeichen rechts davon.
■ ▲	Bewegt den Cursor zur obersten Reihe der Befehlszeile.
■ ▼	Bewegt den Cursor zur untersten Reihe der Befehlszeile.
■ ◀	Bewegt den Cursor an das linke Ende der Befehlszeile.
■ ▶	Bewegt den Cursor an das rechte Ende der Befehlszeile.

Beenden einer Befehlszeile: ENTER

Das Drücken von ENTER wertet die Befehlszeile aus. (Ist keine Befehlszeile vorhanden, so wird der Befehl DUP ausgeführt, der den Inhalt der Ebene 1 dupliziert. DUP wird in "STACK" beschrieben.)

Um die Befehlszeile auszuwerten, muß ENTER den Text in der Befehlszeile *analysieren*. Dazu werden die Objekte erkannt und zu einem Programm zusammengefaßt, und schließlich wird dieses Programm ausgewertet. Im einzelnen laufen folgende Prozesse ab, sobald Sie ENTER drücken:

1. Der "busy"-Indikator ((•)) wird eingeschaltet.
2. Falls UNDO aktiv ist, wird der momentane Stackinhalt abgespeichert.
3. Die Zeichenkette in der Befehlszeile wird nach Begrenzungs- und Trennzeichen durchsucht und in entsprechende Teile zerlegt.
4. Jeder so entstandene Textteil wird mit Hilfe von Syntaxregeln getestet, um den Typ des Objekts zu bestimmen. Danach wird das entsprechende Objekt im Stack gespeichert.
5. Wenn COMMAND aktiv ist, wird eine Kopie der Befehlszeile im Befehls-Stack gespeichert.
6. Die Objekte im Stack werden zu einem einzigen Programm-Objekt zusammengefaßt, das dann ausgewertet wird.
7. Der "busy"-Indikator ((•)) wird ausgeschaltet.

Wenn ein Textteil die Syntaxtests von Punkt 4 nicht erfüllt, wird `Syntax Error` angezeigt. Die Objekte, die ENTER im Stack gespeichert hat, werden gelöscht und die Befehlszeile wird in die Anzeige zurückgerufen. Der fehlerhafte Text wird durch inverse Darstellung hervorgehoben, und der Cursor befindet sich unmittelbar dahinter. Wurde der Fehler durch unvollständige Syntax hervorgerufen, so erscheint der Cursor am Ende der Zeile.

Ansehen von Objekten: ■ VIEW↑, ■ VIEW↓

Mit ■ VIEW↑ und ■ VIEW↓ können Sie die nicht sichtbaren Zeilen eines Objekts ansehen, das mehr als die verfügbaren Anzeigezeilen in Anspruch nimmt. ■ VIEW↑ bewegt das Anzeigefenster um eine Zeile nach oben und ■ VIEW↓ um eine Zeile nach unten. Auf dieselbe Art können Sie auch die nicht sichtbaren Stackebenen ansehen. ■ VIEW↑ und ■ VIEW↓ sind Tasten mit automatischer Wiederholfunktion und können auch während der Objekt-Eingabe oder dem Editieren von Text verwendet werden.

Ändern vorhandener Objekte

Ein existierendes Objekt kann in die Befehlszeile zurückgerufen werden. Hier können Sie seine gesamte Definition betrachten oder abändern, indem Sie die normalen EDIT-Funktionen in der Befehlszeile anwenden. EDIT bringt ein Objekt aus der Ebene 1 in die Befehlszeile. VISIT holt ein Objekt aus höheren Stackebenen oder aus dem Benutzerspeicher in die Befehlszeile.

Editieren in Ebene 1: ■ EDIT

EDIT wirkt als Umkehrfunktion von ENTER. Er bringt ein Objekt aus Ebene 1 zurück in die Befehlszeile. Die Auswirkungen von EDIT im einzelnen:

1. Eine Kopie des Objekts in Ebene 1 wird in Textform umgewandelt und in die Befehlszeile gebracht.
2. Der Alpha-Eingabemodus wird aktiviert.
3. Das Cursor-Menü wird für das Editieren aktiviert.

Das ursprüngliche Objekt in Ebene 1 wird hervorgehoben, um Sie daran zu erinnern, daß Sie dieses Objekt editieren und es in ursprünglicher Form noch vorhanden ist.

Solange das Objekt sich in Textform in der Befehlszeile befindet, können Sie es beliebig ändern. Wenn Sie den Änderungsvorgang beendet haben:

- Drücken Sie ON, um die EDIT-Funktion abzubrechen, die Befehlszeile zu löschen und das ursprüngliche Objekt in Ebene 1 zu erhalten.
- Drücken Sie ENTER (oder eine Taste, die ENTER ausführt), um das ursprüngliche Objekt in Ebene 1 zu ersetzen. (Genauer gesagt wird das ursprüngliche Objekt in Ebene 1 gelöscht und die Befehlszeile ausgewertet.)

Wenn das Cursor-Menü nach Beenden des Editierens immer noch aktiv ist, so wird das vorangehende Menü wieder aufgerufen.

Ändern einer Variablen oder höheren Stackebene:

■ VISIT

VISIT ist eine erweiterte Version von EDIT. Es ermöglicht Ihnen, ein als Variable oder in einer höheren Stackebene (> 1) gespeichertes Objekt anzusehen oder zu editieren, ohne das Objekt vorher in Ebene 1 zu bringen.

Ändern einer Variabeln. Um ein Objekt zu ändern, das als Variable gespeichert ist, geben Sie den Variablennamen in Ebene 1 ein und drücken ■ VISIT. Das gespeicherte Objekt wird in die Befehlszeile kopiert und der Alpha-Eingabemodus sowie das Cursor-Menü werden aktiviert.

Ändern in einer Stack-Ebene. Um ein Objekt in der Stackebene *n* zu ändern, geben Sie *n* in Ebene 1 ein und drücken ■ VISIT. Das Objekt wird in die Befehlszeile kopiert und der Alpha-Eingabemodus sowie das Cursor-Menü werden aktiviert. Das ursprüngliche Objekt wird hervorgehoben, um Sie daran zu erinnern, daß Sie beim Editieren sind und die ursprüngliche Kopie noch vorhanden ist.

Sie beenden VISIT auf dieselbe Art wie EDIT:

- Drücken Sie ON, um die Editierfunktion abzubrechen, die Befehlszeile zu löschen und das ursprüngliche Objekt in Ebene 1 zu erhalten.
- Drücken Sie ENTER (oder eine Taste, die ENTER ausführt), um das ursprüngliche Objekt in Ebene 1 zu ersetzen. (Genauer gesagt wird das ursprüngliche Objekt in Ebene 1 gelöscht und die Befehlszeile ausgewertet.)

Wenn das Cursor-Menü nach Beenden des Editierens immer noch aktiv ist, so wurde das vorangehende Menü wieder aufgerufen.

Auswerten von Objekten

EVAL — ***Objekt auswerten*** — **Befehl**

Ebene 1	
Objekt ➡	

EVAL (*EVALuate*) wertet das Objekt in Ebene 1 aus. Das Resultat dieses Prozesses, einschließlich der Resultate im Stack, hängt vom ausgewerteten Objekt ab. Die Auswertung von Objekten wird detailliert in "Grundlagen" erklärt. Die Auswertung von Funktionen wird von Flag 36 beeinflußt, das den symbolischen oder den numerischen Auswertungsmodus aktiviert. Eine Beschreibung dazu finden Sie unter "ALGEBRA".

→NUM — *Als Zahl auswerten* — Befehl

Ebene 1	Ebene 1
Objekt ➧	*z*

→NUM ist bis auf den Punkt mit EVAL identisch, daß er zeitweise den numerischen Auswertungsmodus aktiviert (beschrieben in "ALGEBRA"), um sicherzustellen, daß Funktionen numerische Werte liefern. Der vorherige Auswertungsmodus wird wieder aktiviert, sobald →NUM beendet ist.

SYSEVAL — *System-Objekt auswerten* — Befehl

Ebene 1	
# n ➧	

SYSEVAL ist ausschließlich für die Verwendung in Anwendungsprogrammen durch Hewlett-Packard bestimmt. Die allgemeine Anwendung von SYSEVAL kann den Speicherbereich durcheinander bringen oder den Verlust des Speicherinhalts bedeuten. *Verwenden Sie SYSEVAL ausschließlich so, wie es in Anwendungen von Hewlett-Packard spezifiziert ist.*

SYSEVAL wertet das System-Objekt an der absoluten Adresse *# n* aus. Sie können die Version Ihres Rechners 28S feststellen, indem Sie `# 10 SYSEVAL` ausführen (dezimale Basis vorausgesetzt, die auch Voreinstellung ist).

Namen

Namen können bis zu 127 Zeichen enthalten, obwohl praktische Gesichtspunkte Namen mit maximal fünf oder sechs Zeichen nahelegen. Das erste Zeichen muß ein Buchstabe sein. Kleinbuchstaben werden intern von Großbuchstaben unterschieden, erscheinen jedoch immer als Großbuchstaben im USER-Menü. Sie können die Namen von HP-28S Befehlen nicht als Variablennamen verwenden.

Die auf dem Tastenfeld verfügbaren zulässigen Zeichen sind Buchstaben, Ziffern und die Zeichen ?, Σ, π, →, µ und °. Die folgenden Zeichen dürfen nicht in Variablennamen enthalten sein:

- Objekt-Begrenzungszeichen (#, ⌊, ⌋, ", ', ⟨, ⟩, (,), «, »).
- Algebraische Operatoren (+, −, ×, /, ^, √, =, <, >, ≤, ≥, ≠, ∂, ∫).
- Das momentane Trennzeichen (. oder ,).

Reservierte Namen

Die folgenden Namen sind für spezielle Anwendungen reserviert:

- EQ bezieht sich auf die momentan vom Gleichungslöser (SOLVE) und von PLOT Befehlen verwendetete Gleichung.
- ΣPAR bezieht sich auf eine Parameterliste für Statistikbefehle.
- PPAR bezieht sich auf eine Parameterliste für PLOT Befehle.
- ΣDAT bezieht sich auf das momentane Statistikfeld.
- s1, s2, ... werden von ISOL und QUAD erzeugt, um beliebige Vorzeichen bei symbolischen Lösungen darzustellen.
- n1, n2, ... werden von ISOL und QUAD erzeugt, um beliebige ganze Zahlen bei symbolischen Lösungen darzustellen.

Sie können jeden dieser Namen für Ihre eigenen Zwecke verwenden. Denken Sie aber daran, daß bestimmte Befehle diese Namen als implizite Argumente verwenden.

Namen mit und ohne Anführungszeichen

Sie können einen Namen mit oder ohne Anführungszeichen in die Befehlszeile eingeben, abhängig davon, ob Sie den Namen auswerten möchten.

Mit Anführungszeichen: Die Eingabe eines Namens in einfachen Anführungszeichen bedeutet "Speichere diesen Namen im Stack". Damit wird der Name im Stack gespeichert, bei der Ausführung der Befehlszeile erfolgt jedoch keine Auswertung des Namens.

Ohne Anführungszeichen: Die Eingabe eines Namens ohne Anführungszeichen bedeutet "Werte das Objekt aus, das durch diesen Namen gekennzeichnet ist". Das heißt, der Name wird in den Stack gebracht und ausgewertet, sobald die Befehlszeile ausgeführt wird.

Identische Namen

Im allgemeinen können Sie keine zwei Benutzervariablen definieren, die denselben Namen haben. Bestimmte Befehle (DRAW, ∫, QUAD, TAYLR) erzeugen jedoch eine temporäre Benutzervariable, deren Name das Namen-Argument dupliziert, das Sie für den Befehl spezifiziert haben. Nach der Ausführung des Befehls wird die temporäre Variable wieder gelöscht. Wird die Ausführung jedoch durch einen Systemstopp ([ON] [▲]) oder durch einen `Out of Memory` Fehler (kein freier Speicherplatz) abgebrochen, so bleibt die temporäre Variable im Benutzerspeicher.

Wenn dies der Fall ist und Sie zuvor eine Variable mit demselben Namen definiert haben, gibt es zwei Variablen mit gleichen Namen im USER-Menü. Verwenden Sie PURGE, um den doppelten Namen aus dem USER-Menü zu entfernen. (PURGE löscht die Variable, die zuletzt erzeugt wurde, was hier die temporäre Variable ist.)

Erzeugen, Aufrufen und Löschen von Variablen

Eine Variable ist die Kombination eines Namen-Objekts und eines beliebigen anderen Objektes. Diese werden zusammen im Benutzerspeicher abgelegt. Das Namen-Objekt stellt den Namen der Variablen dar; das andere Objekt entspricht dem Wert oder dem Inhalt der Variablen.

Dieser Abschnitt zeigt Ihnen, wie man ein Objekt erzeugt, wie man den Inhalt einer Variablen in den Stack ruft, ohne ihn auszuwerten, und wie man eine Variable löscht.

STO — *Speichern* — Befehl

Ebene 2	Ebene 1	
Objekt	*'Name'*	➡

Dieser Befehl definiert eine Variable, deren Name *Name* und deren Wert *Objekt* ist. Eine darauffolgende Auswertung von *Name* bringt *Objekt* in den Stack und wertet *Objekt* aus, falls *Objekt* ein Name oder ein Programm ist.

RCL — *Zurückrufen* — Befehl

Ebene 1		Ebene 1
'Name'	➡	*Objekt*

Dieser Befehl durchsucht den Benutzerspeicher nach der Variablen *Name* und kehrt mit ihren Inhalt *Objekt* zurück. Das zurückgebrachte *Objekt* wird nicht ausgewertet.

Es gibt einen wichtigen Unterschied zwischen RCL (*recall*) und EVAL (*evaluate*). RCL erfordert einen Variablennamen als Argument und kehrt mit dem Inhalt zurück. EVAL akzeptiert ein beliebiges *Objekt* als Argument und wertet es gemäß den entsprechenden Regeln aus. RCL und EVAL haben nur dann denselben Effekt, wenn das Argument ein Name ist, der sich auf ein Daten-Objekt oder eine algebraische oder lokale Variable bezieht. In diesen Fällen bringen beide das gespeicherte Objekt in den Stack zurück.

Die folgende Tabelle faßt die Resultate der Ausführung von EVAL und RCL (mit dem Namen `'ABC'` in Ebene 1) für verschiedene Werte der damit verbundenen Variablen ABC zusammen.

Vergleich zwischen EVAL und RCL

Inhalt von 'ABC':	'ABC' EVAL:	'ABC' RCL:
(nicht definiert)	Bringt 'ABC' zurück	Verursacht Undefined Name Fehlermeldung
Der Name 'DEF'	Wertet 'DEF' aus	Bringt 'DEF' zurück
Ein Programm	Führt Programm aus	Bringt das Programm zurück
Irgendein Objekt	Bringt das Objekt zurück	Bringt das Objekt zurück

PURGE ***Löschen*** **Befehl**

Ebene 1	
'Name'	➡
{ $Name_1$ $Name_2$ ... }	➡

PURGE löscht eine oder mehrere Variablen aus dem Benutzerspeicher. Wenn das Argument ein Name ist, löscht PURGE die entsprechende Variable. Ist das Argument eine Liste von Namen, so löscht PURGE alle aufgeführten Variablen.

Rücksicherungen

Der HP-28S speichert automatisch Kopien von Befehlszeile, Stack und Argumenten. Diese Kopien ermöglichen es Ihnen, nach dem Auftreten eines Fehlers zum vorherigen Zustand zurückzukehren. Sie können dann eine Berechnung wiederholen und fehlerfrei ausführen, ohne noch einmal von vorne anfangen zu müssen. Die Kopien von Befehlszeile und Argumenten eignen sich auch bestens zur Wiederholung von Berechnungen.

Diese Kopien können einen beträchtlichen Teil des Speichers belegen. Jede einzelne dieser Rücksicherungsmöglichkeiten—Befehlszeile, Stack, Argumente—können Sie aktivieren oder ausschalten. (Die dazu notwendigen Operationen finden Sie im MODE-Menü.) Dieser Abschnitt setzt voraus, daß alle Möglichkeiten aktiviert sind, wie es nach einem Zurücksetzen des Speichers der Fall ist.

Die Rücksicherungs-Operationen sind ■ COMMAND, was Kopien der Befehlszeile zurückruft, ■ UNDO, was eine Kopie des Stacks zurückruft, und ■ LAST, was die zuletzt verwendeten Argumente zurückruft.

Rücksicherung der Befehlszeile: ■ COMMAND

Jede Ausführung von ENTER speichert eine Kopie der Befehlszeile. Bis zu vier Befehlszeilen werden gleichzeitig im *Befehls-Stack* gespeichert. Einmaliges Drücken von ■ COMMAND bringt die erste (zuletzt gespeicherte) Befehlszeile zurück und ersetzt den momentanen Inhalt der Befehlszeile. Ein zweites Drücken von ■ COMMAND bringt die zweite Befehlszeile zurück, usw. Wenn Sie ■ COMMAND mehr als viermal betätigen, beginnt die Sequenz wieder mit der ersten Befehlszeile.

Rücksicherung des Stacks: ■ UNDO

Jede Ausführung von ENTER speichert eine Kopie des Stacks, bevor die Befehlszeile ausgewertet wird. Drücken Sie ■ UNDO, so wird der gegenwärtige Stack gelöscht und durch den gespeicherten ersetzt. UNDO bringt den Stack in denselben Zustand wie vor dem Drücken von ENTER (oder einer Taste, die ein ENTER ausführte), wobei mögliche Änderungen in Benutzerflags oder im Benutzerspeicher nicht beeinflußt werden.

Während die Ausführung eines Programms ausgesetzt ist, hängt die UNDO-Möglichkeit (einschließlich +UND, -UND und UNDO selbst) von der Umgebung des unterbrochenen Programms ab. Das bedeutet, UNDO wird den Stack zurückbringen, der vor dem letzten ENTER, aber nach der Aussetzung des (zuletzt ausgeführten) Programms vorhanden war. Nachdem ein Programm fortgesetzt und vollständig ausgeführt wurde, bezieht sich UNDO auf den Stack, der vor der Programmausführung gespeichert worden war.

Rücksicherung der letzten Argumente

LAST ***Letzte Argumente*** **Befehl**

	Ebene 3	Ebene 2	Ebene 1
•			$Objekt_1$
•		$Objekt_1$	$Objekt_2$
•	$Objekt_1$	$Objekt_2$	$Objekt_3$

Befehle, die Argumente aus dem Stack nehmen, speichern Kopien dieser Objekte. Die Ausführung von ■ LAST bringt die Objekte zurück, die zuletzt durch einen Befehl gespeichert wurden. Die Objekte gelangen in dieselben Stackebenen, in denen sie ursprünglich waren. Befehle, die keine Argumente erfordern, verändern die gespeicherte Argumentenliste nicht.

Wenn LAST einem Befehl folgt, der eine Prozedur (wie $\int$, ∂, ISOL, EVAL, ROOT, usw.) auswertet, dann stammen die zuletzt gespeicherten Argumente von der Prozedur und nicht vom ursprünglichen Befehl.

Zu kleiner Speicherbereich

Der HP-28S besitzt 32 KBytes als Benutzerspeicher, wobei etwa 400 Bytes für Systemzwecke reserviert sind, so daß etwa 31.6 KBytes für den allgemeinen Gebrauch zur Verfügung stehen. Praktisch benötigt jede HP-28S Operation Speicherplatz—sogar die Analyse der Befehlszeile. Einige algebraischen Befehle (COLCT, EXPAN, TAYLR) haben einen beträchtlichen Speicherbedarf, vor allem, wenn ihre Argumente komplizierter werden.

Wenn Sie Ihren Rechner wirkungsvoll einsetzen wollen, sollten Sie daran denken, daß Ihr HP-28S ein *Rechner zum interaktiven Lösen von Problemen ist*. Seine Leistungsfähigkeit liegt in den eingebauten Operationen—nicht in der Fähigkeit, große Datenmengen oder Programmbibliotheken zu speichern. Versuchen Sie, wenigstens einige hundert Bytes des Speichers für die dynamische Systembelegung freizuhalten.

Da sich das Betriebssystem des HP-28S Teile des Speichers mit den Benutzer-Objekten teilt, kann der Speicher so voller Benutzer-Objekte sein, daß normale Operationen des Rechners schwierig oder undurchführbar werden. Der HP-28S besitzt eine Reihe von Warnungen für knapp werdenden freien Speicherplatz. In der Reihenfolge zunehmender Dringlichkeit—das heißt abnehmenden freien Speicherplatzes, sind dies:

1. `Insufficient Memory` — Unzureichender Speicherplatz
2. `No Room for UNDO` — Kein Platz für UNDO
3. `No Room to ENTER` — Kein Platz für ENTER
4. `Low Memory!` — Zu kleiner Speicherbereich
5. `No Room to Show Stack` — Kein Platz für Stack-Anzeige
6. `Out of Memory` — Kein freier Speicherplatz

Unzureichender Speicherplatz

Wenn nicht genügend Speicherplatz zur Ausführung eines Befehls vorhanden ist, wird die Ausführung gestoppt und es wird `Insufficient Memory` angezeigt. Ist LAST aktiv, so werden die ursprünglichen Argumente in den Stack zurückgespeichert, andernfalls gehen sie verloren.

Kein Platz für UNDO

Angenommen UNDO ist aktiv, und Sie haben eine 11 × 11 Matrix im Stack. In diesem Fall können Sie nicht einmal eine einfache Operation wie NEG ausführen, da nicht genügend Speicherplatz für die Ausgangs- und die Ergebnismatrix vorhanden ist. Es erscheint eine `No Room for UNDO` Meldung, die automatisch UNDO ausschaltet. Sie können nun die gescheiterte Operation noch einmal ausführen und anschließend UNDO reaktivieren.

Kein Platz für ENTER

Wenn nicht genügend Speicherplatz vorhanden ist, um die Befehlszeile zu verarbeiten, löscht der Rechner die Befehlszeile und zeigt `No Room to ENTER` an. Eine Kopie der unzulässigen Befehlszeile ist im Befehls-Stack gespeichert, falls er aktiviert ist.

Wenn Sie ein existierendes Objekt mit EDIT oder VISIT editieren wollen und eine Kopie der unzulässigen Befehlszeile im Befehls-Stack gespeichert ist, so löschen Sie zunächst die ursprüngliche Form des Objekts. Drücken Sie dann ■ [COMMAND], um die Befehlszeile zurückzurufen, die das zu editierende Objekt enthält; das Drücken von [ENTER] bewirkt die Eingabe der modifizierten Version.

Zu kleiner Speicherbereich

Sind weniger als 128 Bytes an freiem Speicherplatz vorhanden, so blinkt die Meldung `Low Memory!` einmal in der obersten Zeile der Anzeige. Sie blinkt von nun an nach jeder Tasteneingabe, bis zusätzlicher Speicherplatz verfügbar ist. Löschen Sie unnötige Objekte aus dem Speicher, bevor Sie Ihre Berechnungen fortsetzen.

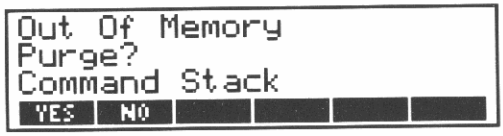
Kein Platz für Stack-Anzeige

Es kann vorkommen, daß der HP-28S alle anstehenden Operationen abschließt und dann nicht genügend Speicherplatz für die normale Stackanzeige übrig hat. In diesen Fällen meldet der Rechner `No Room to Show Stack` in der obersten Anzeigezeile. Die Zeilen der Anzeige, die normalerweise die Stack-Objekte wiedergeben, zeigen nun nur noch die Objekttypen, zum Beispiel `Real Number`, `Algebraic`, usw.

Der zur Anzeige eines Stack-Objekts notwendige Speicherplatz hängt vom Objekttyp ab—algebraische Typen haben im allgemeinen den geringsten Speicherbedarf. Löschen Sie ein oder mehrere Objekte aus dem Speicher oder speichern Sie ein Stack-Objekt als Variable, so daß es nicht angezeigt werden muß.

Kein freier Speicherplatz

Der Extremfall von geringem Speicherplatz liegt dann vor, wenn nicht mehr genügend Speicher vorhanden ist, um irgendeine Funktion auszuführen—Anzeigen des Stacks oder der Menüfelder, Aufbauen der Befehlszeile oder ähnliches. In dieser Situation *müssen* Sie Teile des Speichers löschen, bevor Sie fortfahren können. Eine spezielle `Out of Memory` Prozedur wird aktiviert, welche die folgende Anzeige hervorruft:



Der Rechner wird Sie der Reihe nach fragen, ob nachfolgendes gelöscht werden soll:

1. Der COMMAND-Stack (wenn aktiviert).
2. Der UNDO-Stack (wenn aktiviert).
3. Die LAST Argumente (wenn aktiviert).
4. Der Stack.
5. Jede einzelne Benutzervariable mit Namen.

Für Elemente, die Sie löschen wollen, drücken Sie die Menütaste YES; für solche, die Sie beibehalten wollen, drücken Sie NO. Nachdem Sie YES mindestens einmal gedrückt haben, können Sie versuchen, die `Out of Memory` Prozedur durch Drücken von ATTN abzubrechen. Wenn genügend Speicherplatz verfügbar ist, kehrt der Rechner zur normalen Anzeige zurück; andernfalls ertönt ein Akustiksignal und die Löschsequenz wird fortgesetzt. Nach einem Durchlauf aller fünf Optionen versucht die `Out of Memory` Prozedur zur normalen Operationsweise zurückzukehren. Wenn immer noch nicht genügend Speicherplatz frei ist, beginnt die Prozedur noch einmal von vorne.

Systemoperationen

Es gibt spezielle Tastenkombinationen, die gewöhnliche Operationen unterbrechen, um Systemoperationen auszuführen. Diese Systemoperationen beinhalten das Anpassen des Anzeigekontrasts, Stoppen von Endlos-Schleifen, die nicht auf die [ON] Taste reagieren, oder Ausführen eines elektronischen Systemtests.

Grundstellung: [ON]

Das Drücken von [ON] löscht die Befehlszeile und zeigt den Stack an. Wenn gerade eine Prozedur ausgeführt wird, stoppt [ON] die Prozedur. Das Resultat ist ähnlich wie bei einer Ausführung von ABORT (in "PROGRAM CONTROL" beschrieben).

Das Betätigen von ■[OFF] schaltet den HP-28S aus. Beim nächsten Drücken von [ON] nimmt der HP-28S sein Tätigkeit im selben Zustand wieder auf, in dem er sich vor dem Ausschalten befunden hat. Wenn 10 Minuten lang keine Eingabe erfolgt, schaltet sich der HP-28S automatisch ab.

Kontrastregelung: [ON][+], [ON][−]

Sie können den Anzeigekontrast des HP-28S folgenderweise ändern:

1. Betätigen Sie die [ON] Taste und halten Sie sie gedrückt.
2. Drücken Sie [+] für stärkeren und [−] für schwächeren Kontrast. Solange Sie [ON] gedrückt halten, können Sie mehrmals oder dauernd [+] oder [−] drücken, bis Sie die beste Kontrasteinstellung gefunden haben.
3. Geben Sie die [ON] Taste frei.

Systemstopp: ON ▲

Um einen *Systemstopp* auszuführen, drücken Sie gleichzeitig die ON und die ▲ Taste. Ein Systemstopp:

- Stoppt die Ausführung aller Befehle und Prozeduren.
- Löscht alle lokalen Variablen.
- Löscht den Stack.
- Aktiviert das Cursor-Menü.
- Nimmt die normale Operationsweise des Tastenfelds wieder auf.

Der häufigste Gebrauch eines Systemstopps besteht darin, eine "Endlos-Schleife" zur Auswertung eines Namens anzuhalten. Denken Sie daran, daß die Auswertung eines Namens, der sich auf einen zweiten Namen bezieht, die Auswertung des zweiten Namens zur Folge hat. Wenn sich der zweite Name nun wiederum auf den ersten bezieht, so ruft der Versuch, einen der beiden Namen auszuwerten, eine Endlos-Schleife hervor. Zum Beispiel resultiert

```
'Y' 'X' STO  'X' 'Y' STO X
```

in einer Endlos-Schleife. Da die Auswertung von Namen für symbolische Algebra kritisch ist, ist sie auf Geschwindigkeit getrimmt und kann nicht mit der ON Taste angehalten werden. Sie müssen einen Systemstopp verwenden, um die Schleife zu unterbrechen.

Zurücksetzen des Speichers: ON INS ▶

Um den gesamten Speicherbereich des HP-28S zu löschen und zurückzusetzen:

1. Halten Sie ON gedrückt.
2. Halten Sie INS und ▶ gedrückt.
3. Geben Sie INS und ▶ frei.
4. Geben Sie ON frei.

Ein Zurücksetzen des Speichers bewirkt:

- Stoppen der Ausführung aller Befehle und Prozeduren.
- Löschen aller lokalen Variablen.
- Löschen aller Benutzervariablen.
- Löschen des Stacks.
- Setzen aller Benutzerflags auf ihre Voreinstellung.
- Aktivieren des Cursor-Menüs.
- Ausgeben eines Akustiksignals und Anzeigen von `Memory Lost.`
- Wiederaufnehmen der normalen Operationsweise des Tastenfelds.

Aufheben des Zurücksetzens: [ON] [DEL]

Wenn Sie einen Systemstopp ([ON] [▲]) oder ein Zurücksetzen des Speichers ([ON] [INS] [▶]) einleiten, wird dies erst nach Loslassen der [ON] Taste ausgeführt. Bevor Sie die [ON] Taste freigeben, können Sie den auszuführenden Befehl zu jeder Zeit aufheben, indem Sie:

1. Alle Tasten außer [ON] freigeben.
2. Die [DEL] Taste drücken und wieder freigeben.
3. Die [ON] Taste freigeben.

Testprogramme: [ON] [▼], [ON] [◀]

Der HP-28S besitzt Testmöglichkeiten seiner Systemelektronik für Hersteller- und Serviceoperationen. Sie können die Tests starten, indem Sie [ON] und [▼] gleichzeitig drücken und sie dann wieder freigeben. Nach jedem Abschluß eines Tests, was durch ein neues Anzeigemuster angedeutet wird, können Sie durch Drücken einer beliebigen Taste Ihre Berechnungen fortsetzen.

Wenn KEYBOARD TEST angezeigt wird, drücken Sie [A] bis [F], dann [G] bis [L], dann [M] bis [R], und so weiter. Sind Sie mit dem linken Tastenfeld fertig, so testen Sie das rechte Tastenfeld, beginnend mit [INS].

Wenn der Rechner alle Tests erfolgreich beendet, zeigt er OK-28S an.

Sie können die Tests wiederholt ausführen, indem Sie gleichzeitig [ON] und [◀] drücken. Der Rechner wiederholt die Testserie, wobei er allerdings den Tastenfeld-Test ausläßt, bis Sie eine Taste drücken. Er zeigt dann FAIL an, was durch die Testunterbrechung verursacht wurde.

Die Systemtests führen auch einen Systemstopp ([ON][▲]) durch.

Schlüsselwortverzeichnis

Alle Rechneroperationen (außer dem besonderen Befehl SYSEVAL) können durch eine Taste aufgerufen werden. Häufig verwendete Operationen wie [ENTER], [+] und ■[√] sind stets über eine Taste mit dem zugehörigen Namen zugreifbar. Alle anderen Operationen sind über die *Menütasten*—die sechs Tasten am unteren Rand der Anzeige—erreichbar. Die von Ihnen erzeugten Variablen sind auch über die Menütasten erreichbar, was in "SOLVE" und "USER" beschrieben wird.)

Menüs

Die momentane Belegung jeder Menütaste erscheint in ihrem *Menüfeld* direkt oberhalb der Taste. Ihr Name wird in inverser Darstellung gezeigt. Wenn keine Menüfelder sichtbar sind, ist das Cursor-Menü aktiv. Die Cursor-Operationen werden zum Editieren verwendet und sind im Kapitel "Grundlegende Operationen" beschrieben.

Operationen sind nach gemeinsamen Anwendungen (z.B. Trigonometrie) oder Argument-Typen (z.B. Felder) in Menüs zusammengefaßt. Sie wählen ein Menü durch Drücken einer *Menüwahl-Taste*, wie [TRIG] oder ■[ARRAY]. Jedes Menü besteht aus *Menüzeilen*—ein Satz mit sechs Befehlen, die zur selben Zeit den Menütasten zugeordnet sind. Wenn Sie eine Menüwahl-Taste drücken, wird die erste Menüzeile den Menütasten zugeteilt. Drücken Sie [NEXT], wird die nächste Menüzeile zugeordnet, bis Sie schließlich zur ersten zurückkehren. Drücken Sie ■[PREV], so werden die Menüzeilen in der umgekehrten Reihenfolge durchlaufen.

Das Drücken einer Menütaste führt die zugeordnete Operation aus oder fügt ihren Namen in die Befehlszeile ein. Eine vollständige Beschreibung finden Sie unter "Objekt-Eingabe" im Kapitel "Grundlegende Operationen."

In der folgenden Tabelle sind die Menüwahl-Tasten und eine Kurzbeschreibung jedes HP-28C/S Menüs aufgeführt.

Menüwahl-Taste	Beschreibung
◀▶	Cursorbewegung, Editieroperationen.
■ ALGEBRA	Algebra-Befehle.
■ ARRAY	Vektor- und Matrizen-Befehle.
■ BINARY	Arithmetik ganzer Zahlen, Basisumwandlungen, Bitmanipulationen.
■ BRANCH	Programmverzweigungs-Strukturen.
■ CMPLX	Befehle für komplexe Zahlen.
■ CTRL	Programmsteuerung, Stopp und Einzelschritt-Operationen
■ LIST	List-Befehle.
■ LOGS	Logarithmische und exponentielle Funktionen, hyperbolische Funktionen.
■ MODE	Wahl des Anzeige-, Rücksicherungs- und Winkelmodus sowie des Dezimalzeichens.
■ PLOT	Zeichenbefehle (PLOT).
■ PRINT	Druckbefehle.
■ REAL	Befehle für reelle Zahlen.
SOLV	Befehle für numerische und symbolische Lösungen, der Gleichungslöser (Solver).
■ STACK	Stackmanipulations-Befehle.
■ STAT	Befehle zur Statistik und Wahrscheinlichkeitsrechnung.
■ STORE	Arithmetische Speicherungsbefehle, ersetzende Matrizen-Befehle.
■ STRING	Befehle für Zeichenketten.
■ TEST	Flag-Befehle und logische Testfunktionen.
TRIG	Trigonometrische Funktionen, Umwandlungen zwischen kartesischen und Polarkoordinaten, zwischen Grad und Bogenmaß; Befehle für Stunden/Minuten/Sekunden Arithmetik.
USER	Variable, Benutzerspeicher-Befehle.

ALGEBRA

COLCT	EXPAN	SIZE	FORM	OBSUB	EXSUB
TAYLR	ISOL	QUAD	SHOW	OBGET	EXGET

Algebraische Objekte

Ein algebraisches Objekt ist eine Prozedur, die in mathematischer Form eingegeben und angezeigt wird. Es kann Zahlen, Variablennamen, Funktionen und Operatoren beinhalten, die wie folgt definiert sind:

Zahl: Eine reelle oder komplexe Zahl.

Variablenname: Ein beliebiger Name, wobei momentan diesem Namen eine Variable zugeordnet sein kann oder nicht. Mit dem Ausdruck *formale Variable* wird ein Name bezeichnet, der momentan nicht einer Benutzervariablen zugeordnet ist. Bei der Auswertung eines solchen Namens wird der Name selbst zurückgegeben.

Funktion: Ein Befehl des HP-28S, der in einer algebraischen Prozedur zulässig ist. Funktionen müssen genau ein Ergebnis zurückgeben. Wenn ein oder mehrere Argumente der Funktion algebraische Objekte sind, so ist das Ergebnis algebraisch. Die meisten Funktionen erscheinen als Funktionsname, gefolgt von einem oder mehreren Argumenten in Klammern; zum Beispiel `'SIN(X)'`.

Operator: Eine Funktion, die im allgemeinen keine Klammern für ihre Argumente benötigt. Die Operatoren NOT, $\sqrt{}$ und NEG (was in algebraischen Ausdrücken als "$-$" Zeichen erscheint) sind *vorangestellte* Operatoren: ihr Name erscheint vor ihren Argumenten. Die Operatoren $+$, $-$, $*$, $/$, $\hat{}$, $=$, $==$, $\neq$, $<$, $>$, $\leq$, $\geq$, AND, OR und XOR sind *zwischengestellte* Operatoren: ihr Name erscheint zwischen ihren zwei Argumenten.

...ALGEBRA

Rangordnung

Die *Rangordnung* von Operatoren bestimmt die Reihenfolge der Bearbeitung, falls Ausdrücke ohne Klammern eingegeben werden. Die Operationen höheren Ranges werden zuerst ausgeführt. Für Operatoren gleichen Ranges werden Ausdrücke von links nach rechts ausgewertet. Die folgende Liste zeigt die algebraischen Funktionen des HP-28S nach Rang geordnet, vom höchsten zum niedersten:

1. Ausdrücke innerhalb von Klammern. Ausdrücke innerhalb verschachtelter Klammern werden von innen nach außen ausgewertet.
2. Funktionen wie SIN, LOG und FACT, die Argumente in Klammern erfordern.
3. Potenzen (^) und Wurzeln ($\sqrt{}$).
4. Negation (−), Multiplikation (*) und Division (/).
5. Addition (+) und Subtraktion (−).
6. Relations-Operatoren (==, ≠, <, >, ≤, ≥).
7. AND und NOT.
8. OR und XOR.
9. =

Algebraische Objekte und Programme besitzen eine identische interne Struktur. Beide Prozedur-Typen sind Reihen von Objekten, die bei der Prozedurausführung sequentiell abgearbeitet werden. Der algebraische Ausdruck `'X+Y'` und das Programm `« X Y + »` werden als dieselbe Sequenz (in UPN-Form) gespeichert. Algebraische Ausdrücke werden als solche *markiert*, damit sie als mathematische Ausdrücke angezeigt werden, und um anzudeuten, daß sie die algebraische Syntax erfüllen.

Algebraische Syntax und Teilausdrücke

Eine Prozedur hält sich an die algebraische Syntax, wenn sie bei ihrer Auswertung keine Argumente vom Stack nimmt und genau ein Argument in den Stack zurückbringt und wenn sie vollständig in eine Hierarchie von *Teilausdrücken* unterteilt werden kann. Ein Teilausdruck kann eine Zahl, ein Name oder eine Funktion mit ihren Argumenten sein. *Hierarchie* bedeutet, daß jeder Teilausdruck selbst ein Argument einer Funktion sein kann. Betrachten Sie zum Beispiel den Ausdruck:

```
'1-SIN(X+Y)'
```

Der Ausdruck besteht aus einer Zahl, `1`, und zwei Namen, `X` und `Y`, die beide als einfache Ausdrücke betrachtet werden können. Der Ausdruck beinhaltet außerdem drei Funktionen, `+`, `-` und `SIN`, von denen jede zusammen mit ihren Argumenten einen Teilausdruck bildet. Die Argumente von `+` sind `X` und `Y`; `X+Y` ist das Argument von `SIN` und `1` sowie `SIN(X+Y)` sind die Argumente von `-`. Die Hierarchie wird deutlicher, wenn der Ausdruck mit seinen Operatoren noch einmal mit Hilfe von gewöhnlichen Funktionen niedergeschrieben wird (*UPN*):

$$-(1,\ \mathrm{SIN}\ (+(X,\ Y)))$$

Ein Objekt oder Teilausdruck innerhalb eines Ausdrucks wird durch *Position* und *Ebene* charakterisiert.

Die *Position* eines Objekts wird bestimmt, indem man innerhalb des Ausdrucks von links nach rechts zählt. Zum Beispiel hat in dem Ausdruck `'1-SIN(X+Y)'`, `1` die Position 1, `-` hat Position 2, `SIN` hat Position 3, und so weiter.

Die Position eines Teilausdrucks ist die Position des Objekts, das diesen Teilausdruck definiert. Im selben Beispiel hat `'SIN(X+Y)'` die Position 3, da es durch `SIN` in Position 3 definiert wird.

...ALGEBRA

Die *Ebene* eines Objekts innerhalb eines algebraischen Ausdrucks ist die Anzahl von Klammerpaaren, die das Objekt umgeben, wenn der Ausdruck in Funktionsform dargestellt ist. Zum Beispiel hat in dem Ausdruck `'1-SIN(X+Y)'` "-" die Ebene 0, 1 und `SIN` haben Ebene 1, "+" hat Ebene 2 und `X` und `Y` haben Ebene 3. Jeder algebraische Ausdruck hat genau ein Objekt in Ebene 0.

(Benutzerdefinierte Funktionen sind eine offensichtliche Ausnahme von der Regel zur Bestimmung der Ebene eines Teilausdrucks. Zum Beispiel haben in dem Ausdruck `'F(A,B)'`, wo `F` eine vom Benutzer definierte Funktion ist, `F`, `A` und `B` alle die Ebene 1; es gibt keine explizite Funktion der Ebene 0. Dies liegt daran, daß `F` und seine Argumente `A` und `B` Argumente für eine spezielle "unsichtbare" Funktion sind, die Anzeige und Auswertelogik für benutzerdefinierte Funktionen zur Verfügung stellt.)

Wenn Sie den obigen Ausdruck noch einmal ohne Klammern und mit den Funktionen hinter die dazugehörigen Argumente schreiben, erhalten Sie die UPN-Form dieses Ausdrucks:

```
1 X Y + SIN -
```

Dies definiert ein *Programm*, das algebraische Syntax besitzt und praktisch äquivalent zum entsprechenden algebraischen Objekt ist. Programme sind jedoch flexibler als algebraische Objekte; zum Beispiel könnten Sie ein DUP an jeder beliebigen Stelle dieses Progamms einfügen und immer noch ein zulässiges Programm haben, aber es würde sich dann nicht mehr an die algebraische Syntax halten. Da DUP ein Argument nimmt und zwei zurückgibt, kann es nicht einen algebraischen Ausdruck definieren oder Teil davon sein.

Gleichungen

Eine algebraische *Gleichung* ist ein algebraisches Objekt, das zwei Ausdrücke enthält, die durch ein Gleichheitszeichen (=) verbunden sind. Mathematisch gesehen impliziert das Gleichheitszeichen die Gleichheit der beiden Teilausdrücke. Im HP-28S ist "=" eine Funktion von zwei Argumenten. Es wird als zwischengestellter Operator angezeigt, der die zwei Teilausdrücke, seine Argumente, trennt. Intern ist eine Gleichung ein Ausdruck mit = als Objekt der Ebene 0.

Wenn eine Gleichung numerisch ausgewertet wird, ist = äquivalent zu -. Diese Einrichtung gestattet, daß Ausdrücke und Gleichungen als Argumente für symbolische und numerische Nullstellenberechnungen austauschbar sind. Eine Gleichung ist äquivalent zu einem Ausdruck, in welchem = durch - ersetzt ist, und ein Ausdruck ist äquivalent zur linken Seite einer Gleichung, in der die rechte Seite Null ist.

Wenn eine Gleichung das Argument einer Funktion ist, so ist das Resultat wieder eine Gleichung, bei der die Funktion auf beide Seiten angewendet wurde. Daher resultiert

```
'X=Y' SIN      in      'SIN(X)=SIN(Y)'.
```

Der übliche mathematische Gebrauch des Gleichheitszeichens ist zweideutig. Das Gleichheitszeichen wird verwendet, um zwei Ausdrücke wie in $x + \sin y = 2z + t$ gleichzusetzen. Diese Art von Gleichung kann gelöst werden, das heißt eine oder mehrere Varablen werden angepaßt, um die Gleichheit der beiden Seiten herzustellen.

Das Gleichheitszeichen wird auch verwendet, um einer Varablen einen Wert zuzuweisen, wie z.B. in $x = 2y + z$. Diese Gleichung bedeutet, daß das Symbol x den längeren Ausdruck $2y + z$ ersetzt; es macht keinen Sinn, diese Gleichung zu "lösen".

Die Zweideutigkeit des Gleichheitszeichens wird durch gewisse Programmiersprachen wie z.B. BASIC verstärkt, in denen "=" die Bedeutung von "ersetze durch" einnimmt, wie z.B. in X = Y + Z. Eine solche Notation hat überhaupt keine Verbindung zu einer mathematischen Gleichung.

Im HP-28S bedeutet ein Gleichheitszeichen immer das Gleichsetzen zweier Ausdrücke, so daß das Lösen der Gleichung äquivalent ist zu dem Vorgang, der die Differenz zwischen den beiden Ausdrücken zu Null macht. (Die Zuordnung wird mit dem STO Befehl durchgeführt, der genau genommen ein nachgestellter Befehl ist, welcher zwei Argumente erfordert.)

...ALGEBRA

= **Gleichsetzen** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	'z_1=z_2'
z	'*Symbol*'	➧	'*z=Symbol*'
'*Symbol*'	z	➧	'*Symbol=z*'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$=$Symbol_2$'

Diese Funktion verbindet zwei Argumente, die Namen, Ausdrücke, reelle Zahlen oder komplexe Zahlen sein müssen.

Wenn sich der Rechner im symbolischen Auswertungsmodus befindet (Flag 36 gesetzt), ist das Ergebnis eine algebraische Gleichung, wobei das Argument aus Ebene 2 sich auf der linken Seite der Gleichung und das Argument aus Ebene 1 sich auf der rechten Seite befindet.

Ist der Rechner im numerischen Auswertungsmodus (Flag 36 gelöscht), so ist das Ergebnis die numerische Differenz der beiden Argumente. Von der Wirkungsweise her gesehen verhält sich = dann wie der − Operator im numerischen Auswertungsmodus.

Funktionen symbolischer Argumente

Auswertungsmodi für Funktionen

Symbolischer Auswertungsmodus (Flag 36 gesetzt). Im symbolischen Auswertungsmodus resultieren Funktionen in symbolischen Ergebnissen, wenn ihre Argumente in symbolischer Form vorliegen. Dies ist der voreingestellte Auswertungsmodus. Als Beispiel:

```
    'X' SIN ergibt 'SIN(X)'
'X^2+5' LN ergibt 'LN(X^2+5)'.
   3 'X' + ergibt '3+X'
2 'X' + SIN ergibt 'SIN(2+X)'.
'X' 1 2 IFTE ergibt 'IFTE(X,1,2)'.
```

Numerischer Auswerte-Modus (Flag 36 gelöscht). Im numerischen Auswertungsmodus versucht jede Funktion symbolische Objekte in Daten-Objekte umzuwandeln. Sind die Argumente in Zahlen umgewandelt, so wird die Funktion auf sie angewendet und das Ergebnis ist numerisch. Die Argumente werden wiederholt ausgewertet, bis sie schließlich zu Daten-Objekten oder formalen Variablen werden. Wenn die endgültigen Argumente formale Variablen sind, erscheint eine `Undefined Name` Fehlermeldung.

Automatische Vereinfachung

Bestimmte Funktionen ersetzen bei ihrer Auswertung bestimmte Argumente oder Kombinationen von Argumenten durch einfachere Formen. Im Beispiel `'1*X'` stellt die `*` Funktion fest, daß eines der Argumente eine `1` ist, daher wird der Ausdruck durch `'X'` ersetzt. Automatische Vereinfachung findet in den folgenden Fällen statt:

Ursprünglicher Ausdruck	Vereinfachter Ausdruck
Negation, Inversion, Quadrieren	
`-(-X)`	`X`
`INV(INV(X))`	`X`
`SQ(√X)`	`X`
`SQ(X^Y)`	`X^(Y*2)`
`SQ(i)`	`-1`
Addition und Subtraktion	
`0+X` oder `X+0`	`X`
`X-0`	`X`
`0-X`	`-X`
`X-X`	`0`
Multiplikation	
`X*0` oder `0*X`	`0`
`X*1` oder `1*X`	`X`
`X*(-1)` oder `-1*X`	`-X`
`-X*(-1)` oder `-1*(-X)`	`X`
`i*i`	`-1`
`-X*INV(Y)`	`-(X/Y)`
`-X*Y`	`-(X*Y)`
`X*INV(Y)`	`X/Y`

...ALGEBRA

Ursprünglicher Ausdruck	Vereinfachter Ausdruck
Division	
X/1	X
0/X	0
1/INV(X)	X
1/X	INV(X)
Potenz	
1^X	1
X^0	1
X^1	X
(√X)^2	X
INV(X)^(-1)	X
X^(-1)	INV(X)
i^2	-1 oder (-1,0)*
i^(2,0)	(-1,0)
SIN, COS, TAN	
SIN(ASIN(X))	X
SIN(-X)	-SIN(X)
SIN(π)	0†
SIN(π/2)	1†
COS(ACOS(X))	X
COS(-X)	COS(X)
COS(π)	-1†
COS(π/2)	0†
TAN(ATAN(X))	X
TAN(-X)	-TAN(X)
TAN(π)	0†
ABS, MAX, MIN, MOD, SIGN	
ABS(ABS(X))	ABS(X)
ABS(-X)	ABS(X)
MAX(X,X)	X
MIN(X,X)	X
MOD(X,0)	X
MOD(0,X)	0
MOD(X,X)	0
MOD(MOD(X,Y),Y)	MOD(X,Y)
SIGN(SIGN(X))	SIGN(X)

* Hängt vom symbolischen (Flag 36 gesetzt) oder numerischen (Flag 36 gelöscht) Auswertungsmodus ab.

† Gilt nur, wenn als Winkelmodus Bogenmaß spezifiziert wurde.

Ursprünglicher Ausdruck	Vereinfachter Ausdruck
ALOG, EXP, EXPM, SINH, COSH, TANH	
ALOG(LOG(X))	X
EXP(LN(X))	X
EXPM(LNP1(X))	X
SINH(ASINH(X))	X
COSH(ACOSH(X))	X
TANH(ATANH(X))	
IM, RE, CONJ	
IM(IM(X))	0
IM(RE(X))	0
IM(CONJ(X))	-IM(X)
IM(i)	1
RE(RE(X))	RE(X)
RE(IM(X))	IM(X)
RE(CONJ(X))	RE(X)
RE(i)	0
CONJ(CONJ(X))	X
CONJ(RE(X))	RE(X)
CONJ(IM(X))	IM(X)
CONJ(i)	-i

Funktionen von Gleichungen

Funktionen, die im symbolischen Auswertungsmodus auf Gleichungen angewendet werden, geben Gleichungen als Ergebnis zurück.

Wenn die Funktion eines einzigen Arguments auf eine Gleichung angewendet wird, ist das Ergebnis eine Gleichung, die durch getrennte Anwendung der Funktion auf die linke und rechte Seite der Gleichung erhalten wird. Zum Beispiel:

'X+2=Y' SIN ergibt 'SIN(X+2)=SIN(Y)'.

...ALGEBRA

Wenn beide Argumente einer Funktion, welche 2 Argumente erfordert, Gleichungen sind, so ist das Ergebnis wiederum eine Gleichung. Sie wurde durch Gleichsetzen derjenigen Ausdrücke erhalten, die durch getrennte Anwendung der Funktion auf die beiden linken Seiten als Argumente und auf die beiden rechten Seiten entstand. Zum Beispiel:

```
'X+Y=Z+T' 'SIN(Q)=5' + ergibt 'X+Y+SIN(Q)=Z+T+5'.
```

Ist ein Argument einer Funktion, welche 2 Argumente erfordert, ein numerisches Objekt oder ein algebraischer Ausdruck und ist das andere eine Gleichung, dann wird das Original zu einer Gleichung mit dem urprünglichen Objekt auf beiden Seiten. Danach wird die Funktion wie im Falle zweier Gleichungen als Argumente ausgeführt. Zum Beispiel:

```
'X=Y' 3 - ergibt 'X-3=Y-3'.
```

Diese Eigenschaften definieren das Verhalten algebraischer Objekte bei ihrer Auswertung (siehe nächster Abschnitt) und erlauben Ihnen algebraische Berechnungen in einem interaktiven UPN-Stil auszuführen.

Auswertung algebraischer Objekte

Die *Auswertung* algebraischer Objekte ist eine leistungsfähige Eigenschaft des HP-28S, die Ihnen ermöglicht, Ausdrücke durch Ausführung expliziter numerischer Berechnungen zusammenzufassen und Zahlen oder Ausdrücke durch Variable zu ersetzen. Um zu verstehen, was Sie bei der Anwendung eines algebraischen Objekts erwarten können, ist zu beachten, daß ein algebraisches Objekt äquivalent zu einem Programm ist. Die Ausführung eines Programms erfolgt, indem jedes Objekt des Programms in den Stack gebracht und ausgewertet wird, falls es ein Befehl oder ein Name ist.

Um zu demonstrieren, was dies bedeutet, soll angenommen werden, daß die Variable X den Wert 3 hat (das heißt `3 'X' STO`), Y den Wert 4 und Z den Wert `'X+T'`. Außerdem sei der symbolische Auswertungsmodus aktiviert (Flag 36 gesetzt), so daß Funktionen symbolische Argumente akzeptieren.

Betrachten Sie zunächst den Ausdruck '%+Y'. Wenn Sie diesen Ausdruck auswerten ('%+Y' EVAL), erhalten Sie 7 als Ergebnis. Hier der Grund: Intern wird '%+Y' als X Y + dargestellt und in dieser Reihenfolge ausgewertet:

1. Da es sich bei % um einen Namen handelt, ist seine Auswertung gleichwertig mit der Auswertung des in der Variablen % gespeicherten Objekts, der Zahl 3. Die Auswertung von X bringt 3 in Ebene 1.
2. Analog dazu bringt die Auswertung von Y 4 in Ebene 1 und schiebt 3 in Ebene 2.
3. Nun wird + mit den numerischen Argumenten 3 und 4 ausgeführt. Dies löscht die 3 und die 4 und ergibt das numerische Resultat 7.

Nun zur Auswertung von '%+T':

1. Die Auswertung von % bringt 3 in Ebene 1.
2. T ist ein Name, der nicht mit einer Variablen verbunden ist, so daß T selbst in Ebene 1 kommt und 3 in Ebene 2 schiebt.
3. Dieses Mal hat + 3 und T als Argumente; da T symbolisch ist, gibt + ein algebraisches Ergebnis, '3+T', zurück.

Zum Schluß zu '%+Y+Z'. Dieser Ausdruck wird intern als X Y + Z + dargestellt. Derselben Logik wie in den obigen Beispielen folgend ergibt die Auswertung das Ergebnis '7+(%+T)'. Dieses Ergebnis kann noch einmal ausgewertet werden, was das neue Ergebnis '7+(3+T)' ergibt. Weitere Auswertungen ergeben keine weiteren Veränderungen, da T kein Wert zugeordnet ist

Die erhaltenen Werte 7 und 3 sind nicht Argumente desselben + Operators und werden daher nicht zusammengefaßt. Wenn Sie die 7 und die 3 zusammenfassen wollen, können Sie entweder den COLCT Befehl zum automatischen Zusammenfassen von Termen oder den FORM Befehl zur allgemeineren Neuordnung des Ausdrucks verwenden.

...ALGEBRA

Symbolische Konstanten: e, π, i, MAXR, und MINR

Es gibt fünf interne algebraische Objekte, die in einer numerischen Darstellung bestimmter Konstanten resultieren. Diese Objekte besitzen die spezielle Eigenschaft, daß ihre Auswertung vom Modus für symbolische Konstanten (Flag 35) und für Funktionen (Flag 36) kontrolliert wird. Wenn Flag 36 gelöscht ist (numerischer Modus), werden diese Objekte zu einem numerischen Wert ausgewertet (ohne Rücksichtnahme auf den Auswertungsmodus für Konstanten (Flag 35). Falls Flag 36 gesetzt ist (symbolischer Auswertungsmodus):

- Wenn Flag 35 gelöscht ist, werden diese Objekte als numerische Werte ausgewertet. Als Beispiel:

  `'2*i' EVAL` ergibt `(0,2)`.

- Wenn Flag 35 gesetzt ist, behalten diese Objekte ihre symbolische Form bei. Als Beispiel:

  `'2*i' EVAL` ergibt `'2*i'`.

Die folgende Tabelle zeigt die fünf Objekte und ihre numerischen Werte.

HP-28S symbolische Konstanten

Objektname	Numerischer Wert
e	2.71828182846
π	3.14159265359
i	(0.00000000000,1.00000000000)
MAXR	9.99999999999E499
MINR	1.00000000000E-499

Die numerischen Werte für e und π sind die besten Näherungen für die Konstanten ϵ und π, die mit 12-stelliger Genauigkeit ausgedrückt werden können. Der numerische Wert für i ist die exakte Darstellung der Konstanten i. MAXR und MINR sind die größten und kleinsten Werte ungleich Null, die im Rechner dargestellt werden können.

Für größere numerische Genauigkeit sollten Sie den Ausdruck 'EXP(X)' an Stelle von 'e^X' verwenden. Die Funktion EXP wendet einen speziellen Algorithmus an, um die Exponentialfunktion mit größerer Genauigkeit zu berechnen.

Wenn als Winkelmodus das Bogenmaß spezifiziert wurde und die Flags 35 und 36 gesetzt sind, werden trigonometrische Funktionen von π und π/2 automatisch vereinfacht. Zum Beispiel ergibt die Auswertung von 'SIN(π)' das Resultat 0.

COLCT EXPAN SIZE FORM OBSUB EXSUB

Diese Befehle ändern die Form algebraischer Ausdrücke, ähnlich wie Sie diese Ausdrücke "auf Papier" behandeln würden. COLCT, EXPAN und FORM sind Identitäts-Operationen, das heißt sie ändern die Form eines Ausdrucks, ohne seinen Wert zu verändern. OBSUB und EXSUB gestatten Ihnen, den Wert eines Ausdrucks zu verändern, indem Sie neue Objekte oder Teilausdrücke in den Ausdruck einbauen.

COLCT *Terme zusammenfassen* Befehl

Ebene 1	Ebene 1
'*Symbol*$_1$' ➧ '*Symbol*$_2$'	

...ALGEBRA

COLCT (*COLleCT terms*) stellt ein algebraisches Objekt neu und vereinfacht dar, indem ähnliche Terme "zusammengefaßt" werden. COLCT:

- Wertet numerische Teilausdrücke aus. Als Beispiel: `'1+2+LOG(10)'` wird ersetzt durch 4.
- Faßt numerische Terme zusammen. Als Beispiel: `'1+X+2'` wird ersetzt durch `'3+X'`.
- Ordnet Faktoren (Argumente von *) und kombiniert ähnliche Faktoren. Zum Beispiel: `'X^Z*Y*X^T*Y'` wird ersetzt durch `'X^(T+Z)*Y^2'`.
- Ordnet Summanden (Argumente von +) und kombiniert ähnliche Terme, die sich nur in einem numerischen Koeffizienten unterscheiden. Zum Beispiel: `'X+X+Y+3*X'` wird ersetzt durch `'5*X+Y'`.

COLCT arbeitet getrennt für beide Seiten einer Gleichung, so daß ähnliche Terme auf verschiedenen Seiten der Gleichung nicht kombiniert werden.

Der Algorithmus zur Ermittlung der Reihenfolge (d.h., ob das X dem Y vorausgeht), der von COLCT verwendet wird, richtet sich nach Geschwindigkeit und nicht nach irgendwelchen offensichtlichen oder standardisierten Formen. Wenn die Anordnung von Termen in einem resultierenden Ausdruck nicht Ihren Wünschen entspricht, können Sie FORM verwenden, um eine neue Reihenfolge herzustellen.

EXPAN *Produkte erweitern* Befehl

Ebene 1	Ebene 1
'*Symbol*$_1$' •	'*Symbol*$_2$'

EXPAN (*EXPANd*) schreibt ein algebraisches Objekt um, indem Produkte und Potenzen erweitert werden. EXPAN:

- Erweitert Multiplikation und Division von Summen. Als Beispiel: `'A*(B+C)'` wird zu `'A*B+A*C'`; `'(B+C)/A'` wird zu `'B/A+C/A'`.
- Erweitert Terme, deren Exponenten aus einer Summe besteht. Als Beispiel: `'A^(B+C)'` wird zu `'A^B*A^C'`.
- Erweitert Potenzen positiver ganzer Zahlen. Als Beispiel: `'X^5'` wird zu `'X*X^4'`. Das Quadrat einer Summe `'(X+Y)^2'` oder `'SQ(X+Y)'` wird zu `'X^2+2*X*Y+Y^2'`.

EXPAN versucht nicht, alle möglichen Erweiterungen eines Ausdrucks in einem Durchlauf auszuführen. Stattdessen arbeitet sich EXPAN durch die Hierarchie der Teilausdrücke und stoppt in jedem Zweig einer Hierarchieebene, wenn es einen Teilausdruck findet, der erweitert werden kann. Zuerst werden die Teilausdrücke der Ebene 0 untersucht; wenn sich dort geeignete Ausdrücke finden, werden sie erweitert und EXPAN stoppt. Wenn nicht, untersucht EXPAN jeden Teilausdruck der Ebene 1. Alle passenden Teilausdrücke werden erweitert; sind keine vorhanden, wird zur Ebene 2 übergegangen. Dieser Prozeß setzt sich durch die gesamte Hierarchie fort, bis eine Erweiterung ein weiteres Durchsuchen beendet. Zum Beispiel:

Erweitern Sie den Ausdruck `'A^(B*(C^2+D))'`.

1. Der Operator der Ebene 0 ist das linke ^. Da es nicht erweitert werden kann, wird der Operator * der Ebene 1 untersucht. Eines seiner Argumente ist eine Summe, und daher wird das Produkt aufgespalten:

   ```
   'A^(B*C^2+B*D)'
   ```

2. Der Operator der Ebene 0 ist immer noch das linke ^, aber nun ist sein Exponent eine Summe. Daher wird die Potenz erweitert, wenn EXPAN noch einmal ausgeführt wird:

   ```
   'A^(B*C^2)*A^(B*D)'
   ```

3. Noch eine Erweiterung ist möglich. Der Operator der Ebene 0 ist nun das mittlere *. Da es nicht erweitert werden kann, werden die Operatoren der Ebene 1 (die außerhalb der Klammer stehenden ^) untersucht. Da auch sie nicht erweitert werden können, werden die Operatoren der Ebene 2 (die außen stehenden *) untersucht. Auch hier ist keine Erweiterung möglich und es wird zur Ebene 3 (dem mittleren ^) übergegangen. Sein Exponent ist eine positive ganze Zahl, so daß eine Erweiterung durchgeführt wird:

```
'A^(B*(C*C))*A^(B*D)'
```

SIZE — *Größe anzeigen* — Befehl

Ebene 1		Ebene 1
"*String*"	➧	*n*
⟨ *Liste* ⟩	➧	*n*
⟦ *Feld* ⟧	➧	⟨ *Liste* ⟩
'*Symbol*'	➧	*n*

SIZE ergibt die Anzahl von Objekten, aus denen ein algebraisches Objekt zusammengesetzt ist.

Schlagen Sie unter "ARRAY," "LIST" und "STRING" über den Gebrauch von SIZE mit anderen Objekt-Typen nach.

FORM — *Algebraischen Ausdruck formen* — Befehl

Ebene 1		Ebene 3	Ebene 2	Ebene 1
'$Symbol_1$'	➧			'$Symbol_2$'
'$Symbol_1$'	➧	'$Symbol_2$'	*n*	'$Symbol_3$'

FORM ist ein interaktiver Editor für Ausdrücke, der es Ihnen erlaubt, einen algebraischen Ausdruck oder eine Gleichung nach festgesetzten mathematischen Regeln umzustellen. Seine Arbeitsweise wird im nächsten Abschnitt, "ALGEBRA (FORM)," beschrieben.

OBSUB — *Objekt substituieren* — Befehl

Ebene 3	Ebene 2	Ebene 1	Ebene 1
'*Symbol*$_1$'	*n*	⟨ *Objekt* ⟩	➧ '*Symbol*$_2$'

OBSUB (*OBject SUBstitute*) ersetzt eine Zahl, einen Namen oder eine Funktion in der bezeichneten Position eines algebraischen Objekts. Das Objekt ist der Inhalt einer Liste in Ebene 1, die Position *n* befindet sich in Ebene 2 und das algebraische Objekt in Ebene 3. Als Beispiel:

```
'A*B' 3 ⟨ C ⟩ OBSUB ergibt 'A*C'.
```

Sie können sowohl Funktionen als auch Benutzervariable ersetzen. Als Beispiel:

```
'A*B' 2 ⟨ + ⟩ OBSUB ergibt 'A+B'.
```

EXSUB — *Ausdruck substituieren* — Befehl

Ebene 3	Ebene 2	Ebene 1	Ebene 1
'*Symbol*$_1$'	*n*	'*Symbol*$_2$'	➧ '*Symbol*$_3$'

EXSUB (*EXpression SUBstitute*) ersetzt den Teilausdruck in der *n*-ten Position des algebraischen Ausdrucks '*Symbol*$_1$' durch den algebraischen Ausdruck (oder Namen) '*Symbol*$_2$' und bringt als Ergebnis den Ausdruck '*Symbol*$_3$' zurück. Der *n*-te Teilausdruck besteht aus dem *n*-ten Objekt in der Definition eines algebraischen Objekts und den dazugehörigen Argumenten, falls vorhanden. Als Beispiel:

```
'(A+B)*C' 2 'E^F' EXSUB ergibt 'E^F*C'.
```

...ALGEBRA

TAYLR ISOL QUAD SHOW OBGET EXGET

TAYLR wird in "Höhere Mathematik" zusammen mit ∂ und $\int$ beschrieben. ISOL, QUAD und SHOW werden in "SOLVE" erklärt.

OBGET *Objekt holen* Befehl

Ebene 2	Ebene 1		Ebene 1
'Symbol'	*n*	➡	*{ Objekt }*

OBGET (*OBject GET*) holt das Objekt in der n-ten Position des algebraischen Objekts *Symbol* aus Ebene 2. Das Objekt wird als einziges Objekt einer Liste zurückgebracht. Als Beispiel:

'(A+B)*C' 2 OBGET ergibt { + }.

Wenn n gößer als die Anzahl der Objekte ist, holt OBGET das Objekt der Ebene 0 zurück.

EXGET *Ausdruck holen* Befehl

Ebene 2	Ebene 1		Ebene 1
'Symbol$_1$'	*n*	➡	*'Symbol$_2$'*

EXGET (*EXpression GET*) holt den Teilausdruck in der n-ten Position des algebraischen Ausdrucks *Symbol*$_1$ aus Ebene 2. Der n-te Teilausdruck besteht aus dem n-ten Objekt in der Definition eines algebraischen Objekts und seinen Argumenten, falls solche vorhanden sind. Als Beispiel:

'(A+B)*C' 2 EXGET ergibt 'A+B'.

Wenn n größer ist als die Anzahl der Objekte, holt EXGET den Teilausdruck der Ebene 0 zurück.

ALGEBRA (FORM)

FORM ***Algebraischen Ausdruck formen*** **Befehl**

Ebene 1		Ebene 3	Ebene 2	Ebene 1
'$Symbol_1$'	➧			'$Symbol_2$'
'$Symbol_1$'	➧	'$Symbol_2$'	n	'$Symbol_3$'

FORM ist ein interaktiver Editor für Ausdrücke. Er erlaubt Ihnen, algebraische Ausdrücke und Gleichungen nach festgesetzten mathematischen Regeln umzustellen. Alle mathematischen Operationen von FORM sind Identitäten; das heißt, daß der resultierende Ausdruck $Symbol_2$ denselben Wert wie der ursprüngliche Ausdruck $Symbol_1$ besitzt, obwohl beide verschiedene Form haben können. Zum Beispiel können Sie mit FORM `'A+B'` neu ordnen zu `'B+A'`, was die Form aber nicht den Wert des Ausdrucks ändert.

Eine Variation des Befehls EXGET ist verfügbar, solange FORM aktiv ist. Sie erlaubt Ihnen, einen Teilausdruck $Symbol_3$, der in $Symbol_1$ enthalten ist, und seine Position in den Stack zu kopieren.

Bei der Ausführung von FORM wird die normale Stack-Anzeige durch eine spezielle Anzeige des algebraischen Objekts ersetzt, und ein Menü von FORM-Operationen erscheint am unteren Anzeigerand. Die spezielle Anzeige beginnt in Zeile 2 (zweite von oben) und wird in Zeile 3 fortgesetzt, wenn das Objekt zu lang ist, um in einer Zeile dargestellt werden zu können. Benötigt das Objekt mehr als zwei Anzeigezeilen, müssen Sie den FORM-Cursor über das Objekt bewegen, um den Rest zu betrachten.

...ALGEBRA (FORM)

Um FORM abzubrechen und andere Rechenoperationen fortzusetzen, drücken Sie [ON]. Als Alternative können Sie die EXGET Menütaste drücken, die außerdem den ausgewählten Teilausdruck $Symbol_3$ und seine Position n in den Stack kopiert.

Der FORM-Cursor hebt ein einzelnes Objekt in der Anzeige des Ausdrucks hervor. (Es ist kein Zeichen-Cursor wie in der Befehlszeile.) Das hervorgehobene Objekt erscheint als helles Zeichen auf dunklem Hintergrund. Der Cursor identifiziert sowohl das *gewählte Objekt*, das hervorgehoben wird, als auch den *gewählten Teilausdruck*, der aus dem gewählten Objekt und seinen Argumenten besteht (falls solche vorhanden sind).

Sie können den Cursor im Ausdruck nach links oder nach rechts bewegen, indem Sie die Menütasten [←] oder [→] drücken; der Cursor bewegt sich direkt von einem zum nächsten Objekt und überspringt dazwischenliegende Klammern. Er befindet sich stets in der zweiten Zeile der Anzeige. Wenn Sie versuchen, den Cursor über das Ende der Zeile 2 hinaus zu bewegen, schiebt sich der Ausdruck in der Anzeige um eine Zeile nach oben, und der Cursor kehrt zum Anfang der zweiten Zeile zurück. Analog dazu wird der Ausdruck um eine Zeile nach unten bewegt, und der Cursor erscheint am Ende von Zeile 2, wenn Sie versuchen, ihn über den Zeilenanfang hinaus zu bewegen.

Die Anzeige des Ausdrucks unterscheidet sich von der normalen Stack-Anzeige für algebraische Objekte dadurch, daß zusätzliche Klammern eingefügt werden, um die Rangordnung der Operatoren deutlich zu machen. Diese Eigenschaft hilft Ihnen, den gewählten Teilausdruck auszumachen, der zu dem vom Cursor hervorgehobenen Objekt gehört. Dies ist wichtig, da sich alle Operationen des FORM-Menüs auf den gewählten Teilausdruck beziehen.

Während FORM aktiv ist, sind spezielle Operationen über die Menütasten verfügbar. Das anfängliche Menü besteht aus sechs Operationen, die auf alle Teilausdrücke anwendbar sind. Weitere Menüs für zusätzliche Operationen sind über die [NEXT] und [PREV] Tasten verfügbar; der Inhalt dieser Menüs hängt vom gewählten Objekt ab. Es werden nur die Operationen angezeigt, die auf das gewählte Objekt anwendber sind.

Sie können zu jeder Zeit zu den ersten sechs Menütasten zurückkehren, indem Sie [ENTER] drücken.

FORM-Operationen

In den folgenden Unterabschnitten werden alle Operationen beschrieben, die im FORM-Menü erscheinen können. Die Beschreibungen bestehen hauptsächlich aus Beispielen der Strukturen "vor" und "nach" der Anwendung jeder einzelnen Operation auf die ausgewählten Teilausdrücke. Jede mögliche Funktion wird durch ein Beispiel wie das folgende repräsentiert:

←D Distribution links

Vorher	Nachher
((A+B)*C)	((A*C)+(B*C))

Um eine einfache Darstellung zu ermöglichen, werden Variablen wie A, B und C verwendet, aber jede Variable kann ein allgemeines Objekt oder einen Teilausdruck repräsentieren. Das Beispiel zeigt, daß die Anwendung von ←D (nach links distribuieren) auf '(A+B)*C' in 'A*C+B*C' resultiert.

Einzelne FORM-Operationen erscheinen im FORM-Menü, wenn sie für das gewählte Objekt relevant sind. Zum Beispiel erscheint ←D im Menü, wenn + das gewählte Objekt ist, nicht aber wenn SIN ausgewählt wird. Außerdem können Sie die Operation nur dann verwenden, wenn sie sich auf den Teilausdruck anwenden läßt. Zum Beispiel erscheint D→, wenn * das gewählte Objekt ist, da die Distribution eine Eigenschaft der Multiplikation ist. Die Menütaste ist jedoch nicht aktiv (sie erzeugt nur einen Warnton), wenn der Teilausdruck nicht die Form '(A+B)*C' oder '(A-B)*C' besitzt, auf die eine Distribution angewendet werden kann.

...ALGEBRA (FORM)

Das anfängliche Menü besteht aus den folgenden Operationen:

Operationen, die auf alle Teilausdrücke anwendbar sind

Operation	Bedeutung
COLCT	Faßt ähnliche Terme in dem gewählten Teilausdruck zusammen (*collect terms*). Diese Operation wirkt wie der Befehl COLCT, außer daß sie sich nur auf den gewählten Teilausdruck auswirkt. Der FORM-Cursor wird an den Anfang der Anzeige des Ausdrucks zurückbewegt.
EXPAN	Erweitert (*expand*) Produkte und Potenzen in dem gewählten Teilausdruck. Diese Operation wirkt wie der Befehl EXPAN, außer daß sie sich nur auf den gewählten Teilausdruck bezieht. Der FORM-Cursor wird an den Anfang der Anzeige des Ausdrucks zurückbewegt.
LEVEL	Zeigt die Ebene (*level*) des gewählten Objekts oder der mit ihm verbundenen Teilausdrücke an. Die Ebene wird angezeigt, solange Sie die LEVEL Taste gedrückt halten.
EXGET	Bricht FORM ab und hinterläßt die gegenwärtige Form des editierten Ausdrucks in Ebene 3, eine Kopie des gewählten Teilausdrucks in Ebene 1 und seine Position in Ebene 2.
[←]	Bewegt den FORM-Cursor zum vorangehenden Objekt (nach links) im selben Ausdruck.
[→]	Bewegt den FORM-Cursor zum nachfolgenden Objekt (nach rechts) im selben Ausdruck.

Kommutation, Assoziation und Distribution

←→ Argumente eines Operators vertauschen

Vorher	Nachher
(A+B)	(B+A)
(-(A)+B)	(B-A)
(A-B)	(-(B)+A)
(A*B)	(B*A)
(INV(A)*B)	(B/A)
(A/B)	(INV(B)*A)

←A Nach links assoziieren. Der Pfeil zeigt die Richtung an, in die sich die Klammern "bewegen" werden.

Vorher	Nachher
(A+(B+C))	((A+B)+C)
(A+(B-C))	((A+B)-C)
(A-(B+C))	((A-B)-C)
(A-(B-C))	((A-B)+C)
(A*(B*C))	((A*B)*C)
(A*(B/C))	((A*B)/C)
(A/(B*C))	((A/B)/C)
(A/(B/C))	((A/B)*C)
(A^(B*C))	((A^B)^C)

...ALGEBRA (FORM)

A→ **Nach rechts assoziieren.** Der Pfeil zeigt die Richtung an, in die sich die Klammern "bewegen" werden.

Vorher	Nachher
((A+B)+C)	(A+(B+C))
((A-B)+C)	(A-(B-C))
((A+B)-C)	(A+(B-C))
((A-B)-C)	(A-(B+C))
((A*B)*C)	(A*(B*C))
((A/B)*C)	(A/(B/C))
((A*B)/C)	(A*(B/C))
((A/B)/C)	(A/(B*C))
((A^B)^C)	(A^(B*C))

→() **Vorangestellten Operator distribuieren**

Vorher	Nachher
-(A+B)	(-(A)-B)
-(A-B)	(-(A)+B)
-(A*B)	(-(A)*B)
-(A/B)	(-(A)/B)
-(LOG(A))	LOG(INV(A))
-(LN(A))	LN(INV(A))
INV(A*B)	(INV(A)/B)
INV(A/B)	(INV(A)*B)
INV(A^B)	(A^-(B))
INV(ALOG(A))	ALOG(-(A))
INV(EXP(A))	EXP(-(A))

...ALGEBRA (FORM)

Beachten Sie, daß nach jeder neuen Darstellung eines Ausdrucks die Sequenz * INV durch / und die Sequenz + − durch − ersetzt wird.

←D Nach links distribuieren. Der Pfeil zeigt auf den Teilausdruck, der distribuiert wird.

Vorher	Nachher
((A+B)*C)	((A*C)+(B*C))
((A-B)*C)	((A*C)-(B*C))
((A+B)/C)	((A/C)+(B/C))
((A-B)/C)	((A/C)-(B/C))
((A*B)^C)	((A^C)*(B^C))
((A/B)^C)	((A^C)/(B^C))

D→ Nach rechts distribuieren. Der Pfeil zeigt auf den Teilausdruck, der distribuiert wird.

Vorher	Nachher
(A*(B+C))	((A*B)+(A*C))
(A*(B-C))	((A*B)-(A*C))
(A/(B+C))	INV((INV(A)*B)+(INV(A)*C))
(A/(B-C))	INV((INV(A)*B)-(INV(A)*C))
(A^(B+C))	((A^B)*(A^C))
(A^(B-C))	((A^B)/(A^C))
LOG(A*B)	(LOG(A)+LOG(B))
LOG(A/B)	(LOG(A)-LOG(B))
ALOG(A+B)	(ALOG(A)*ALOG(B))
ALOG(A-B)	(ALOG(A)/ALOG(B))
LN(A*B)	(LN(A)+LN(B))

...ALGEBRA (FORM)

(Fortsetzung)

Vorher	Nachher
LN(A/B)	(LN(A)-LN(B))
EXP(A+B)	(EXP(A)*EXP(B))
EXP(A-B)	(EXP(A)/EXP(B))

←M Linke Faktoren zusammenführen. Diese Operation führt Argumente von +, −, * und / immer dort zusammen (*merge*), wo die Argumente einen gemeinsamen Faktor oder eine gemeinsame einwertige Funktion EXP, ALOG, LN oder LOG besitzen. Im Falle gemeinsamer Faktoren zeigt der Pfeil an, daß die Faktoren der linken Seite gleich sind.

Vorher	Nachher
((A*B)+(A*C))	(A*(B+C))
((A*B)-(A*C))	(A*(B-C))
((A^B)*(A^C))	(A^(B+C))
((A^B)/(A^C))	(A^(B-C))
(LN(A)+LN(B))	LN(A*B)
(LN(A)-LN(B))	LN(A/B)
(LOG(A)+LOG(B))	LOG(A*B)
(LOG(A)-LOG(B))	LOG(A/B)
(EXP(A)*EXP(B))	EXP(A+B)
(EXP(A)/EXP(B))	EXP(A-B)
(ALOG(A)*ALOG(B))	ALOG(A+B)
(ALOG(A)/ALOG(B))	ALOG(A-B)

M→ Rechte Faktoren zusammenführen. Diese Operation führt Argumente von +, −, * und / dort zusammen (*merge*), wo sie einen gemeinsamen Faktor besitzen. Der Pfeil zeigt an, daß die Faktoren der rechten Seite gleich sind.

Vorher	Nachher
((A*C)+(B*C))	((A+B)*C)
((A/C)+(B/C))	((A+B)/C)
((A*C)-(B*C))	((A-B)*C)
((A/C)-(B/C))	((A-B)/C)
((A^C)*(B^C))	((A*B)^C)
((A^C)/(B^C))	((A/B)^C)

Doppelte Negation und doppelte Inversion

DNEG Doppelt negieren. Negiert einen Teilausdruck zweimal.

Vorher	Nachher
A	-(-(A))

...ALGEBRA (FORM)

-() **Doppelt negieren und distribuieren.** Diese Operation ist äquivalent zu einer doppelten Negation **DNEG**, gefolgt von einer Distribution **→()** der resultierenden inneren Negation.

Vorher	Nachher
(A+B)	-(-(A)-B)
(A-B)	-(-(A)+B)
(-(A)-B)	-(A+B)
(A*B)	-(-(A)*B)
(-(A)*B)	-(A*B)
(-(A)/B)	-(A/B)
(A/B)	-(-(A)/B)
LOG(A)	-(LOG(INV(A)))
LOG(INV(A))	-(LOG(A))
LN(A)	-(LN(INV(A)))
LN(INV(A))	-(LN(A))

DINV **Doppelt invertieren.** Invertiere einen Teilausdruck zweimal.

Vorher	Nachher
A	INV(INV(A))

1/( ) Doppelt invertieren und distribuieren. Diese Operation ist äquivalent zu einer doppelten Inversion **DINV**, gefolgt von einer Distribution **→( )** der resultierenden inneren INV:

Vorher	Nachher
(A*B)	INV(INV(A)/B)
(A/B)	INV(INV(A)*B)
(A^B)	INV(A^-(B))
(A^-(B))	INV(A^B)
ALOG(A)	INV(ALOG(-(A)))
ALOG(-(A))	INV(ALOG(A))
EXP(A)	INV(EXP(-(A)))
EXP(-(A))	INV(EXP(A))

Identitäten

*1 Mit 1 multiplizieren

Vorher	Nachher
A	A*1

/1 Durch 1 dividieren

Vorher	Nachher
A	A / 1

...ALGEBRA (FORM)

^1 **Mit 1 potenzieren**

Vorher	Nachher
A	A ^ 1

+1-1 **1 addieren und 1 subtrahieren**

Vorher	Nachher
A	(A + 1) - 1

Neuordnung von exponentiellen Funktionen

L* **Logarithmus einer Potenz durch Produkt eines Logarithmus ersetzen**

Vorher	Nachher
LOG(A^B)	(LOG(A)*B)
LN(A^B)	(LN(A)*B)

L() **Produkt eines Logarithmus durch Logarithmus einer Potenz ersetzen**

Vorher	Nachher
(LOG(A)*B)	LOG(A^B)
(LN(A)*B)	LN(A^B)

E^ Produktform in Potenzausdruck durch Potenz einer Potenz ersetzen

Vorher	Nachher
ALOG(A*B)	(ALOG(A)^B)
ALOG(A/B)	(ALOG(A)^INV(B))
EXP(A*B)	(EXP(A)^B)
EXP(A/B)	(EXP(A)^INV(B))

E() Potenz einer Potenz durch Produktform ersetzen

Vorher	Nachher
(ALOG(A)^B)	ALOG(A*B)
(ALOG(A)^INV(B))	ALOG(A/B)
(EXP(A)^B)	EXP(A*B)
(EXP(A)^INV(B))	EXP(A/B)

Brüche addieren

AF Brüche durch Erzeugen gemeinsamer Nenner addieren

Vorher	Nachher
(A+(B/C))	(((A*C)+B)/C)
((A/B)+C)	((A+(B*C))/B)
((A/B)+(C/D))	(((A*D)+(B*C))/(B*D))
(A-(B/C))	(((A*C)-B)/C)
((A/B)-C)	((A-(B*C))/B)
((A/B)-(C/D))	(((A*D)-(B*C))/(B*D))

Beachten Sie: Wenn zwei Brüche bereits einen gemeinsamen Nenner besitzen, ist M→ oder ←M zu verwenden.

...ALGEBRA (FORM)

FORM-Operationen von Funktionen

Die folgende Tabelle zeigt, welche Operationen im FORM-Menü erscheinen, wenn eine gegebene Funktion das gewählte Objekt ist. Die Form des ursprünglichen Teilausdrucks und des Ergebnisses wird für jede Operation gesondert aufgeführt.

Die Operationen COLCT, EXPAN, LEVEL, DNEG, DINV, *1, /1 und +1-1 stehen für alle Funktionen und Variablen zur Verfügung und sind deshalb in den nachfolgenden Tabellen nicht gelistet. Wenn nur die gemeinsamen Operationen für eine Funktion verfügbar sind, wurde auf die Darstellung einer Tabelle verzichtet. (Es handelt sich um die Funktionen √ und SQ; Sie können diese jedoch durch ^.5 und ^2 ersetzen, wenn Sie andere Operationen verwenden möchten.)

Addition (+)

Operation	Vorher	Nachher
←→	(A+B) (-(A)+B)	(B+A) (B-A)
←A	(A+(B+C)) (A+(B-C))	((A+B)+C) ((A+B)-C)
A→	((A+B)+C) ((A-B)+C)	(A+(B+C)) (A-(B-C))
←M	((A*B)+(A*C)) (LN(A)+LN(B)) (LOG(A)+LOG(B))	(A*(B+C)) LN(A*B) LOG(A*B)
M→	((A*C)+(B*C)) ((A/C)+(B/C))	((A+B)*C) ((A+B)/C)
-()	(A+B) -(A)+B	-(-(A)-B) -(A-B)
AF	(A+(B/C)) ((A/B)+(C/D)) ((A/B)+C)	(((A*C)+B)/C) (((A*D)+(B*C))/(B*D)) ((A+(B*C))/B)

Subtraktion (−)

Operation	Vorher	Nachher
←→	(A−B)	(−(B)+A)
←A	(A−(B+C)) (A−(B−C))	((A−B)−C) ((A−B)+C)
A→	((A+B)−C) ((A−B)−C)	(A+(B−C)) (A−(B+C))
←M	((A∗B)−(A∗C)) (LN(A)−LN(B)) (LOG(A)−LOG(B))	(A∗(B−C)) LN(A/B) LOG(A/B)
M→	((A∗C)−(B∗C)) ((A/C)−(B/C))	((A−B)∗C) ((A−B)/C)
−()	(A−B) (−(A)−B)	−(−(A)+B) −(A+B)
AF	(A−(B/C)) ((A/B)−C) ((A/B)−(C/D))	(((A∗C)−B)/C) ((A−(B∗C))/B) (((A∗D)−(B∗C))/(B∗D))

Multiplikation (∗)

Operation	Vorher	Nachher
←→	(A∗B) (INV(A)∗B)	(B∗A) (B/A)
←A	(A∗(B∗C)) (A∗(B/C))	((A∗B)∗C) ((A∗B)/C)
A→	((A∗B)∗C) ((A/B)∗C)	(A∗(B∗C)) (A/(B/C))
←D	((A+B)∗C) ((A−B)∗C)	((A∗C)+(B∗C)) ((A∗C)−(B∗C))
D→	(A∗(B+C)) (A∗(B−C))	((A∗B)+(A∗C)) ((A∗B)−(A∗C))

...ALGEBRA (FORM)

(*Fortsetzung*)

Operation	Vorher	Nachher
←M	((A^B)*(A^C)) (ALOG(A)*ALOG(B)) (EXP(A)*EXP(B))	(A^(B+C)) ALOG(A+B) EXP(A+B)
M→	((A^C)*(B^C))	((A*B)^C)
-()	(A*B) (-(A)*B)	-(-(A)*B) -(A*B)
1/()	(A*B) (INV(A)*B)	INV(INV(A)/B) INV(A/B)
L()	(LOG(A)*B) (LN(A)*B)	LOG(A^B) LN(A^B)

Division (/)

Operation	Vorher	Nachher
←→	(A/B)	(INV(B)*A)
←A	(A/(B*C)) (A/(B/C))	((A/B)/C) ((A/B)*C)
A→	((A*B)/C) ((A/B)/C)	(A*(B/C)) (A/(B*C))
←D	((A+B)/C) ((A-B)/C)	((A/C)+(B/C)) ((A/C)-(B/C))
D→	(A/(B+C)) (A/(B-C))	INV((INV(A)*B) +(INV(A)*C)) INV((INV(A)*B) -(INV(A)*C))
←M	((A^B)/(A^C)) (ALOG(A)/ALOG(B)) (EXP(A)/EXP(B))	(A^(B-C)) ALOG(A-B) EXP(A-B)

(Fortsetzung)

Operation	Vorher	Nachher
M→	((A^C)/(B^C))	((A/B)^C)
-()	(A/B) (-(A)/B)	-(-(A)/B) -(A/B)
L()	(LN(A)/B) (LOG(A)/B)	LN(A^INV(B)) LOG(A^INV(B))
1/()	(A/B)	INV(INV(A)*B)

Potenz (^)

Operation	Vorher	Nachher
←A	(A^(B*C))	((A^B)^C)
A→	((A^B)^C)	(A^(B*C))
←D	((A*B)^C) ((A/B)^C)	((A^C)*(B^C)) ((A^C)/(B^C))
D→	(A^(B+C)) (A^(B-C))	((A^B)*(A^C)) ((A^B)/(A^C))
1/()	(A^B) (A^-(B))	INV(A^-(B)) INV(A^B)
E()	(ALOG(A)^B) (ALOG(A)^INV(B)) (EXP(A)^B) (EXP(A)^INV(B))	ALOG(A*B) ALOG(A/B) EXP(A*B) EXP(A/B)

...ALGEBRA (FORM)

Negation (−)

Operation	Vorher	Nachher
→()	-(A+B)	(-(A)-B)
	-(A-B)	(-(A)+B)
	-(A*B)	(-(A)*B)
	-(A/B)	(-(A)/B)
	-(LOG(A))	LOG(INV(A))
	-(LN(A))	LN(INV(A))

Inversion (INV)

Operation	Vorher	Nachher
→()	INV(A*B)	(INV(A)/B)
	INV(A/B)	(INV(A)*B)
	INV(A^B)	(A^-(B))
	INV(ALOG(A))	ALOG(-(A))
	INV(EXP(A))	EXP(-(A))

Logarithmus (LOG)

Operation	Vorher	Nachher
D→	LOG(A*B)	(LOG(A)+LOG(B))
	LOG(A/B)	(LOG(A)-LOG(B))
-()	LOG(A)	-(LOG(INV(A)))
	LOG(INV(A))	-(LOG(A))
L*	LOG(A^B)	(LOG(A)*B)
	LOG(A^INV(B))	(LOG(A)/B)

...ALGEBRA (FORM)

Antilogarithmus (ALOG)

Operation	Vorher	Nachher
D→	ALOG(A+B) ALOG(A-B)	(ALOG(A)*ALOG(B)) (ALOG(A)/ALOG(B))
1/()	ALOG(A) ALOG(-(A))	INV(ALOG(-(A))) INV(ALOG(A))
E^	ALOG(A*B) ALOG(A/B)	(ALOG(A)^B) (ALOG(A)^INV(B))

Natürlicher Logarithmus (LN)

Operation	Vorher	Nachher
D→	LN(A*B) LN(A/B)	(LN(A)+LN(B)) (LN(A)-LN(B))
-()	LN(A) LN(INV(A))	-(LN(INV(A))) -(LN(A))
L*	LN(A^INV(B))	(LN(A)*B)

Exponentialfunktion (EXP)

Operation	Vorher	Nachher
D→	EXP(A+B) EXP(A-B)	(EXP(A)*EXP(B)) (EXP(A)/EXP(B))
1/()	EXP(A) EXP(-(A))	INV(EXP(-(A))) INV(EXP(A))
E^	EXP(A*B) EXP(A/B)	(EXP(A)^B) (EXP(A)^INV(B))

Arithmetik

Dieser Abschnitt beschreibt die arithmetischen Funktionen +, −, ∗, /, ^, INV, √, SQ und NEG. Diese Funktionen können auf verschiedene Objekttypen angewendet werden. Sie werden hier für alle geeigneten Objekttypen behandelt; in anderen Abschnitten wie "ARRAY" und "COMPLEX" sind sie nur dann beschrieben, wenn sie auf diesen speziellen Objekttyp angewendet werden können.

+ ***Addieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	z_1+z_2
[$Feld_1$]	[$Feld_2$]	➧	[$Feld_1+Feld_2$]
z	' *Symbol* '	➧	' $z+$ (*Symbol*) '
' *Symbol* '	z	➧	' *Symbol*$+z$ '
' $Symbol_1$ '	' $Symbol_2$ '	➧	' $Symbol_1+$ ($Symbol_2$) '
{ $Liste_1$ }	{ $Liste_2$ }	➧	{ $Liste_1 Liste_2$ }
" $String_1$ "	" $String_2$ "	➧	" $String_1 String_2$ "
# n_1	n_2	➧	# n_1+n_2
n_1	# n_2	➧	# n_1+n_2
# n_1	# n_2	➧	# n_1+n_2

+ ergibt die Summe seiner Argumente, wobei die Art der Summe vom Typ ihrer Argumente abhängt. Sind die Argumente:

Zwei reelle Zahlen: Das Ergebnis ist die gewöhnliche reelle Summe der Argumente.

Eine reelle Zahl *u* und eine komplexe Zahl (*x*, *y*): Das Ergebnis ist die komplexe Zahl $(x + u, y)$, die durch Behandeln der reellen Zahl als komplexe Zahl mit dem Imaginärteil Null entsteht.

Zwei komplexe Zahlen (x_1, y_1) und (x_2, y_2): Das Ergebnis ist die komplexe Summe $(x_1 + x_2, y_1 + y_2)$.

Eine Zahl und ein algebraischer Ausdruck: Das Ergebnis ist ein algebraischer Ausdruck, der die symbolische Summe darstellt.

Zwei algebraische Ausdrücke: Das Ergebnis ist ein algebraischer Ausdruck, der die symbolische Summe darstellt.

Zwei Listen: Das Ergebnis ist eine Liste, die durch Anhängen von Objekten der Liste in Ebene 1 an das Ende der Liste von Objekten in Ebene 2 entsteht.

Zwei Strings: Das Ergebnis ist ein String, der durch Anhängen von Zeichen des Strings in Ebene 1 an das Ende des Strings in Ebene 2 entsteht.

Zwei Felder: Das Ergebnis ist die Summe der Felder, wobei jedes Element die reelle oder komplexe Summe der sich entsprechenden Feldelemente ist. Die zwei Felder müssen dieselbe Dimension besitzen.

Ein Binärwert und eine reelle Zahl: Das Ergebnis ist ein Binärwert, der sich als Summe der beiden Argumente, gekürzt auf die momentane Wortlänge, ergibt. Die reelle Zahl wird vor der Addition in einen Binärwert umgewandelt.

Zwei Binärwerte: Das Ergebnis ist ein Binärwert, der sich als Summe der beiden Argumente, gekürzt auf die momentane Wortlänge, ergibt.

...Arithmetik

– **Subtrahieren** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	$z_1 - z_2$
$[\,Feld_1\,]$	$[\,Feld_2\,]$	➧	$[\,Feld_1 - Feld_2\,]$
z	$'\,Symbol\,'$	➧	$'\,z - Symbol\,'$
$'\,Symbol\,'$	z	➧	$'\,Symbol - z\,'$
$'\,Symbol_1\,'$	$'\,Symbol_2\,'$	➧	$'\,Symbol_1 - Symbol_2\,'$
$\#\ n_1$	n_2	➧	$\#\ n_1 - n_2$
n_1	$\#\ n_2$	➧	$\#\ n_1 - n_2$
$\#\ n_1$	$\#\ n_2$	➧	$\#\ n_1 - n_2$

– ergibt die Differenz seiner Argumente, wobei die Art der Differenz vom Typ ihrer Argumente abhängt. Das Objekt in Ebene 1 wird vom Objekt in Ebene 2 subtrahiert. Sind die Argumente:

Zwei reelle Zahlen: Das Ergebnis ist die gewöhnliche reelle Differenz der Argumente.

Eine reelle Zahl *u* und eine komplexe Zahl (x, y): Das Ergebnis ist die komplexe Zahl $(x - u, y)$ oder $(u - x, -y)$, die durch Behandeln der reellen Zahl als komplexe Zahl mit dem Imaginärteil Null entsteht.

Zwei komplexe Zahlen (x_1, y_1) und (x_2, y_2): Das Ergebnis ist die komplexe Differenz $(x_1 - x_2, y_1 - y_2)$.

Eine Zahl und ein algebraischer Ausdruck: Das Ergebnis ist ein algebraischer Ausdruck, der die symbolische Differenz darstellt.

Zwei algebraische Ausdrücke: Das Ergebnis ist ein algebraischer Ausdruck, der die symbolische Differenz darstellt.

Zwei Felder: Das Ergebnis ist die Differenz der Felder, wobei jedes Element die reelle oder komplexe Differenz der sich entsprechenden Feldelemente ist. Die zwei Felder müssen dieselbe Dimension besitzen.

Ein Binärwert und eine reelle Zahl: Das Ergebnis ist ein Binärwert, der sich als Summe der Zahl in Ebene 2 und des Zweierkomplements der Zahl in Ebene 1 ergibt. Die reelle Zahl wird vor der Subtraktion in einen Binärwert umgewandelt.

Zwei Binärwerte: Das Ergebnis ist ein Binärwert, der sich als Summe der Zahl in Ebene 2 und des Zweierkomplements der Zahl in Ebene 1 ergibt.

* ***Multiplizieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	$z_1 z_2$
[*Matrix*]	[*Feld*]	➧	[*Matrix* × *Feld*]
z	[*Feld*]	➧	[z × *Feld*]
[*Feld*]	z	➧	[*Feld* × z]
z	' *Symbol* '	➧	' z ✱ (*Symbol*) '
' *Symbol* '	z	➧	' (*Symbol*) ✱ z '
' $Symbol_1$ '	' $Symbol_2$ '	➧	' $Symbol_1$ ✱ $Symbol_2$ '
# n_1	n_2	➧	# $n_1 n_2$
n_1	# n_2	➧	# $n_1 n_2$
# n_1	# n_2	➧	# $n_1 n_2$

...Arithmetik

* ergibt das Produkt seiner Argumente, wobei die Art des Produkts vom Typ der Argumente bestimmt wird. Sind die Argumente:

Zwei reelle Zahlen: Das Ergebnis ist das gewöhnliche reelle Produkt der Argumente.

Eine reelle Zahl *u* und eine komplexe Zahl (*x*, *y*): Das Ergebnis ist die komplexe Zahl (xu, yu), die durch Behandeln der reellen Zahl als komplexe Zahl mit dem Imaginärteil Null erhalten wird.

Zwei komplexe Zahlen (x_1, y_1) und (x_2, y_2): Das Ergebnis ist das komplexe Produkt $(x_1x_2 - y_1y_2, x_1y_2 + x_2y_1)$.

Eine Zahl und ein algebraischer Ausdruck: Das Ergebnis ist ein algebraischer Ausdruck, der das symbolische Produkt darstellt.

Zwei algebraische Ausdrücke: Das Ergebnis ist ein algebraischer Ausdruck, der das symbolische Produkt darstellt.

Eine Zahl und ein Feld: Das Ergebnis ist das Produkt, das durch Multiplikation jedes Feldelements mit der Zahl erhalten wird.

Eine Matrix und ein Feld: Das Ergebnis ist das Matrizenprodukt der Argumente. Die Anzahl von Zeilen (Elementen, falls es sich um einen Vektor handelt) des Felds in Ebene 1 muß der Anzahl von Spalten der Matrix in Ebene 2 entsprechen.

Ein Binärwert und eine reelle Zahl: Das Ergebnis ist ein Binärwert, der sich als Produkt der zwei Argumente, gekürzt auf die momentane binäre Wortlänge, ergibt. Die reelle Zahl wird vor der Multiplikation in einen Binärwert umgewandelt.

Zwei Binärwerte: Das Ergebnis ist ein Binärwert, der sich als Produkt der zwei Argumente, gekürzt auf die momentane binäre Wortlänge, ergibt.

/ ***Dividieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	z_1/z_2
⊏ *Feld* ⊐	⊏ *Matrix* ⊐	➧	⊏ *Feld* × *Matrix*$^{-1}$ ⊐
z	' *Symbol* '	➧	' z ⁄ (*Symbol*) '
' *Symbol* '	z	➧	' (*Symbol*) ⁄ z '
' $Symbol_1$ '	' $Symbol_2$ '	➧	' $Symbol_1$ ⁄ $Symbol_2$ '
# n_1	n_2	➧	# n_1/n_2
n_1	# n_2	➧	# n_1/n_2
# n_1	# n_2	➧	# n_1/n_2

/ ([÷]) ergibt den Quotienten (das Objekt in Ebene 2 dividiert durch das Objekt in Ebene 1) der zwei Argumente, wobei die Art des Quotienten vom Typ seiner Argumente bestimmt wird. Sind die Argumente:

Zwei reelle Zahlen: Das Ergebnis ist der gewöhnliche reelle Quotient der Argumente.

Eine reelle Zahl *u* in Ebene 2 und eine komplexe Zahl (x, y) in Ebene 1: Das Ergebnis ist die komplexe Zahl

$$(ux/(x^2 + y^2),\ -uy/(x^2 + y^2))$$

die durch Behandeln der reellen Zahl als komplexe Zahl mit dem Imaginärteil Null erhalten wird.

Eine komplexe Zahl (x, y) in Ebene 2 und eine reelle Zahl *u* in Ebene 1: Das Ergebnis ist die komplexe Zahl $(x/u,\ y/u)$, die durch Behandeln der reellen Zahl als komplexe Zahl mit dem Imaginärteil Null erhalten wird.

...Arithmetik

Eine komplexe Zahl (x_1, y_1) in Ebene 2 und eine komplexe Zahl (x_2, y_2) in Ebene 1: Das Ergebnis ist der komplexe Quotient

$$((x_1x_2 + y_1y_2)/(x_2^2 + y_2^2), (y_1x_2 - x_1y_2)/(x_2^2 + y_2^2))$$

Eine Zahl und ein algebraischer Ausdruck: Das Ergebnis ist ein algebraischer Ausdruck, der den symbolischen Quotienten darstellt.

Zwei algebraische Ausdrücke: Das Ergebnis ist ein algebraischer Ausdruck, der den symbolischen Quotienten darstellt.

Ein Feld und eine Matrix: Das Ergebnis ist das Matrizenprodukt des Feldes in Ebene 2 und der invertierten Matrix in Ebene 1. Die Anzahl von Zeilen (Elementen, falls es sich um einen Vektor handelt) des Felds in Ebene 2 muß der Anzahl von Spalten der Matrix in Ebene 1 entsprechen.

Ein Binärwert und eine reelle Zahl: Das Ergebnis ist ein Binärwert, der sich als ganzzahliger Teil des Quotienten der beiden Argumente, gekürzt auf die momentane Wortlänge, ergibt. Die reelle Zahl wird vor der Division in einen Binärwert umgewandelt. Null als Divisor ergibt # 0.

Zwei Binärwerte: Das Ergebnis ist ein Binärwert, der sich als ganzzahliger Teil des Quotienten der beiden Argumente, gekürzt auf die momentane Wortlänge, ergibt. Null als Divisor ergibt # 0.

^ ***Potenzieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
z_1	z_2	➧	$z_1^{z_2}$
z	'*Symbol*'	➧	'z^(*Symbol*)'
'*Symbol*'	z	➧	'(*Symbol*)^z'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$^($Symbol_2$)'

^ ergibt den Wert des Objekts in Ebene 2 potenziert mit dem Wert des Objekts in Ebene 1. Alle möglichen Kombinationen von reellen Zahlen, komplexen Zahlen und algebraischen Argumenten können verwendet werden. Wenn eines der zwei Argumente komplex ist, ergibt ^ ein komplexes Resultat.

INV ***Invertieren*** **Analyt. Fkt.**

Ebene 1		Ebene 1
z	➧	*1/z*
⟦ *Matrix* ⟧	➧	⟦ *Matrix*$^{-1}$ ⟧
' *Symbol* '	➧	' INV (*Symbol*) '

INV (■ 1/x) ergibt den invertierten (reziproken) Wert seines Arguments.

Für ein komplexes Argument (x, y) ist der invertierte Wert die komplexe Zahl

$$(x/(x^2 + y^2),\ -y/(x^2 + y^2)).$$

Zur Inversion verwendete Felder müssen aus quadratischen Matrizen bestehen.

√ ***Quadratwurzel ziehen*** **Analyt. Fkt.**

Ebene 1		Ebene 1
z	➧	$\sqrt{z}$
' *Symbol* '	➧	' √ (*Symbol*) '

$\sqrt{}$ ergibt die (positive) Quadratwurzel seines Arguments. Für eine komplexe Zahl (x_1, y_1), ist die Quadratwurzel die komplexe Zahl

$$(x_2, y_2) = (\sqrt{r} \cos \theta/2, \sqrt{r} \sin \theta/2)$$

wobei

$$r = \text{abs } (x_1, y_1)$$

$$\theta = \arg (x_1, y_1).$$

Wenn $(x_1, y_1) = (0, 0)$, dann ist die Quadratwurzel $(0, 0)$.

Sehen Sie auch unter "Hauptverzweigungen und allgemeine Lösungen" in "COMPLEX" nach.

SQ — *Quadrieren* — Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	z^2
⟦ *Matrix* ⟧	➧	⟦ *Matrix* × *Matrix* ⟧
' *Symbol* '	➧	' SQ (*Symbol*) '

SQ (■ x^2) (*SQuare*) ergibt das Quadrat seines Arguments.

Für ein komplexes Argument (x,y) ist das Quadrat die komplexe Zahl

$$(x^2 - y^2, 2xy).$$

Die verwendeten Felder müssen aus quadratischen Matrizen bestehen.

NEG — *Negieren* — Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	$-z$
⟦ *Feld* ⟧	➧	⟦ −*Feld* ⟧
' *Symbol* '	➧	' − (*Symbol*) '

NEG ergibt das Negative seines Arguments.

Für ein Feld ist das Negative ein Feld, das aus den negierten Elementen des ursprünglichen Felds zusammengesetzt ist. Wenn keine Befehlszeile vorhanden ist, kann die [CHS] Taste zur Ausführung von NEG verwendet werden. Ist eine Befehlszeile vorhanden, so wirkt [CHS] auf die Befehlszeile, entsprechend der Beschreibung in "Grundlegende Operationen".

Menütasten für NEG kommen im REAL- und im ARRAY-Menü vor.

ARRAY

→ARRY	ARRY→	PUT	GET	PUTI	GETI
SIZE	RDM	TRN	CON	IDN	RSD
CROSS	DOT	DET	ABS	RNRM	CNRM
R→C	C→R	RE	IM	CONJ	NEG

Felder (Arrays) sind geordnete Sammlungen von reellen oder komplexen Zahlen unter Beachtung verschiedener mathematischer Regeln. Im HP-28S werden eindimensionale Felder als *Vektoren*, zweidimensionale Felder als *Matrizen* bezeichnet. Der Begriff *Feld* wird dann verwendet, wenn auf beide Arten Bezug genommen werden soll.

Obwohl Vektoren zeilenweise eingegeben und angezeigt werden, behandelt der HP-28S Vektoren aus Gründen der Matrizenrechnung als $n \times 1$ Matrix.

Ein Feld kann reelle oder komplexe Zahlen enthalten. Die Begriffe *reelles Feld* (*reeller Vektor* oder *reelle Matrix*) und *komplexes Feld* werden dann verwendet, wenn Eigenschaften von Feldern beschrieben werden, welche spezifisch für reelle und komplexe Zahlen sind.

Felder werden entsprechend dem folgenden Format eingegeben und angezeigt:

Vektor	[*Zahl Zahl ...*]
Matrix	[[*Zahl Zahl ...*] [*Zahl Zahl ...*] ⋮ [*Zahl Zahl ...*]]

wobei *Zahl* für eine reelle oder komplexe Zahl steht.

Bei der Eingabe eines Feldes können Sie zwischen reellen und komplexen Zahlen beliebig abwechseln. Ist nur ein Element eines Feldes komplex, so wird das resultierende Feld ebenfalls komplex sein.

Sie können eine beliebige Anzahl neuer Zeilen an einer beliebigen Stelle bei der Eingabe vornehmen—oder Sie geben das ganze Feld in einer einzigen Befehlszeile ein.

Bei der Eingabe von Matrizen kann das Begrenzungszeichen] am Ende jeder Zeile weggelassen werden. Lediglich das am Anfang jeder Zeile stehende [ist erforderlich. Wenn zusätzliche Objekte dem Feld in der Befehlszeile folgen, dann muß das Feld mit]] abgeschlossen werden, bevor Sie die neue Objekteingabe beginnen.

Der Begriff *Reihenfolge* bezieht sich auf die sequentielle Anordnung der Feldelemente, beginnend mit dem ersten Element (erste Zeile, erste Spalte): von links nach rechts innerhalb der Zeile; von der obersten Zeile zur untersten Zeile (bei Matrizen).

Das STORE-Menü enthält Befehle, mit welchen Sie Feld-Operationen unter Verwendung eines Variablennamens (unter dem ein Feld gespeichert ist) durchführen können; hierbei muß das Feld selbst nicht im Stack gespeichert sein. In diesen Fällen wird das Ergebnis einer Operation wieder in der Variable gespeichert, wobei der alte Inhalt überschrieben wird. Dieses Verfahren nimmt, im Gegensatz zu Operationen im Stack, weniger Speicherplatz in Anspruch und erlaubt Ihnen daher auch den Umgang mit größeren Feldern.

Lang andauernde Operationen auf größere Felder können durch Drücken von ON unterbrochen werden. Wenn Sie während einer solchen Operation ON drücken, beendet der HP-28S die Ausführung des Befehls und löscht die Feld-Argumente im Stack. Sie können die ursprünglichen Argumente durch Verwenden von UNDO oder LAST wieder erhalten.

Zusätzlich zu den im ARRAY- und STACK-Menü enthaltenen Funktionen können die nachfolgend beschriebenen Tastenfeld-Funktionen auf Felder angewendet werden.

Tastenfeld-Funktionen

Vollständige Stackdiagramme dieser Funktionen erscheinen unter "Arithmetik".

+ ***Addieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
[$Feld_1$]	[$Feld_2$]	➡	[$Feld_1 + Feld_2$]

+ gibt die Summe von zwei Feldern zurück. Die zwei Felder müssen die gleiche Dimension besitzen. Die Summe eines reellen Feldes und eines komplexen Feldes ist ein komplexes Feld, wobei jedes Element x des reellen Feldes als komplexes Element $(x, 0)$ behandelt wird.

– ***Subtrahieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
[$Feld_1$]	[$Feld_2$]	➡	[$Feld_1 - Feld_2$]

– gibt die Differenz von zwei Feldern zurück. Die zwei Felder müssen die gleiche Dimension besitzen. Die Differenz eines reellen Feldes und eines komplexen Feldes ist ein komplexes Feld, wobei jedes Element x des reellen Feldes als komplexes Element $(x, 0)$ behandelt wird.

* *Multiplizieren* Analyt. Fkt.

Ebene 2	Ebene 1		Ebene 1
z	[*Feld*]	➧	[*z* × *Feld*]
[*Feld*]	*z*	➧	[*z* × *Feld*]
[*Matrix*]	[*Feld*]	➧	[*Matrix* × *Feld*]

* gibt das Produkt seiner Argumente zurück, wobei die Art des Produkts durch den Typ der Argumente festgelegt ist. Sind die Argumente:

Ein Feld und eine Zahl: Die Ergebnismatrix ergibt sich aus der Multiplikation der reellen oder komplexen Zahl (Skalar) mit den einzelnen Feldelementen.

Zwei Felder: Das Produkt ist das Matrizenprodukt der beiden Felder. Das Feld in der Ebene 2 muß eine Matrix sein. Die Ebene 1 kann eine Matrix oder einen Vektor enthalten. Die Anzahl der Zeilen des Felds in Ebene 1 muß gleich der Anzahl von Matrixspalten in Ebene 2 sein.

Das Produkt eines reellen Feldes und eines komplexen Feldes ist ein komplexes Feld. Jedes Element x des reellen Feldes wird als komplexes Element $(x,0)$ behandelt.

/ *Dividieren* Analyt. Fkt.

Ebene 2	Ebene 1		Ebene 1
[*Matrix* **B**]	[*Matrix* **A**]	➧	[*Matrix* **X**]
[*Vektor* **B**]	[*Matrix* **A**]	➧	[*Vektor* **X**]

...ARRAY

Die Anwendung von "/" ([÷]) auf zwei Argumente, welche Felder darstellen, löst das Gleichungssystem **AX** = **B**. Das bedeutet, "/" berechnet $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$. "/" verwendet eine 16-stellige interne Genauigkeit, um ein genaueres Ergebnis zu erhalten, als bei der Anwendung von INV auf **A** und anschließender Multiplikation des Ergebnisses mit **B** erreicht wird.

A muß eine quadratische Matix sein, wobei **B** entweder eine Matrix oder ein Vektor sein kann. Wenn **B** eine Matrix darstellt, so muß sie dieselbe Anzahl an Zeilen wie **A** haben. Wenn **B** ein Vektor ist, dann muß seine Anzahl an Elementen der Anzahl der Spalten von **A** entsprechen.

In vielen Fällen ermittelt Ihr Rechner 28S eine korrekte Lösung, auch wenn das Koeffizientenfeld singulär ist (**A** kann nicht invertiert werden). Diese Fähigkeit des Rechners ermöglicht Ihnen das Lösen von Gleichungssystemen, in welchen die Anzahl der Unbekannten von der Anzahl der Gleichungen abweicht.

In einem System, wo mehr Variable als Gleichungen vorliegen, enthält das Koeffizientenfeld weniger Zeilen als Spalten. Um in diesem Fall eine Lösung zu finden:

1. Hängen Sie genügend Null-Zeilen am unteren Ende des Koeffizientenfelds an, um ein quadratisches Feld zu erhalten.
2. Fügen Sie die entsprechende Anzahl Nullen dem Spalten-bzw. Konstantenfeld hinzu.

Danach können Sie auf diese Felder "/" anwenden, um eine Lösung für das ursprüngliche System zu finden.

In einem System, wo mehr Gleichungen als Variable vorliegen, enthält das Koeffizientenfeld mehr Zeilen als Spalten. Um in diesem Fall eine Lösung zu finden:

1. Hängen Sie genügend Null-Spalten an der rechten Seite des Koeffizientenfelds an, um ein quadratisches Feld zu erhalten.
2. Fügen Sie die entsprechende Anzahl Nullen dem Spalten-bzw. Konstantenfeld hinzu.

Danach kann auf diese Felder "/" angewendet werden, um eine Lösung für das ursprüngliche Gleichungssystem zu finden. Dabei sind allerdings nur die Elemente des Ergebnisfelds von Bedeutung, welche den ursprünglichen Variablen entsprechen.

In beiden Fällen (Anzahl der Unbekannten ungleich der Anzahl von Gleichungen) ist das Koeffizientenfeld singulär und Sie sollten das durch "/" erhaltene Ergebnis im Hinblick auf das ursprüngliche Gleichungssystem überprüfen.

Erhöhen der Genauigkeit beim Lösen linearer Gleichungssysteme

Aufgrund von Rundungsfehlern während einer Berechnung ist im allgemeinen die ermittelte Lösung **Z** nicht generell die Lösung des ursprünglichen Systems $\mathbf{AX} = \mathbf{B}$, sondern eher die Lösung des gestörten Systems $(\mathbf{A} + \Delta\mathbf{A})\ \mathbf{Z} = \mathbf{B} + \Delta\mathbf{B}$. Die Störungen $\Delta\mathbf{A}$ und $\Delta\mathbf{B}$ erfüllen $\|\Delta\mathbf{A}\| \leqslant \epsilon\|\mathbf{A}\|$ und $\|\Delta\mathbf{B}\| \leqslant \epsilon\|\mathbf{B}\|$, wobei ϵ eine kleine Zahl und $\|\mathbf{A}\|$ die *Norm* von **A** ist. In vielen Fällen wird $\Delta\mathbf{A}$ und $\Delta\mathbf{B}$ einen Betrag annehmen, welcher sich erst ab der 12. Stelle von jedem Element in **A** und **B** unterscheidet.

Für eine numerische Lösung **Z** ergibt sich das *Residuum* als $\mathbf{R} = \mathbf{B} - \mathbf{AZ}$. Dabei gilt $\|\mathbf{R}\| \leqslant \epsilon\|\mathbf{A}\|\ \|\mathbf{Z}\|$; das erwartete Residuum einer numerischen Lösung ist somit klein. Trotzallem kann der *Fehler* $\mathbf{Z} - \mathbf{X}$ groß ausfallen, wenn **A** schlechtkonditioniert ist, d.h. wenn $\|\mathbf{Z} - \mathbf{X}\| \leqslant \epsilon\|\mathbf{A}\|\ \|\mathbf{A}^{-1}\|\ \|\mathbf{Z}\|$ gilt.

Als Daumenregel für die Genauigkeit einer numerischen Lösung gilt:

(Anzahl korrekter Stellen)
$\geqslant$ (Anzahl der mitgeführten Stellen) $-$ log $(\|\mathbf{A}\|\ \|\mathbf{A}^{-1}\|) - \log 10n$

wobei n die Dimension von **A** ist. Da der Rechner 12 signifikante Stellen hat, gilt

$$\text{(Anzahl korrekter Stellen)} \geqslant 11 - \log(\|\mathbf{A}\|\ \|\mathbf{A}^{-1}\|) - \log n.$$

...ARRAY

Diese Genauigkeit mag für viele Applikationen ausreichend sein. Wenn jedoch eine höhere Genauigkeit erforderlich ist, kann die berechnete Lösung **Z** gewöhnlich durch eine *Nachiteration* (auch *Residuum-Korrektur* genannt) verbessert werden. Die Nachiteration beinhaltet die Lösung eines Gleichungssystems und anschließender Verbesserung der Genauigkeit, indem das mit der Lösung verbundene Residuum zur Modifikation der Lösung verwendet wird.

Um die Nachiteration anzuwenden, berechnen Sie zuerst die Lösung **Z** des Ausgangssystems **AX** = **B**. Danach wird **Z** als Näherung an **X** betrachtet, mit dem Fehler **E** = **X** − **Z**. Damit erfüllt **E** das lineare Gleichungssystem

$$\mathbf{AE} = \mathbf{AX} - \mathbf{AZ} = \mathbf{R},$$

wobei **R** das Residuum für **Z** darstellt. Im nächsten Schritt ist das Residuum zu berechnen, wonach die Lösung für **E** in **AE** = **R** gefunden werden kann. Die berechnete Lösung—durch **F** gekennzeichnet—wird als Näherung zu **E** betrachtet und wird zu **Z** addiert, um eine neue Näherung für **X** zu erhalten.

Um mit **F** + **Z** eine bessere Näherung an **X** als mit **Z** zu erhalten, muß das Residuum **R** = **B** − **AZ** berechnet werden. Dies erfolgt über die Funktion RSD (detaillierte Hinweise zur Anwendung finden Sie in nachstehender Beschreibung von RSD).

Der Nachiterationsprozeß kann wiederholt werden, die größte Verbesserung wird allerdings bereits nach dem ersten Schritt erreicht. Die Funktion "/" unternimmt keine Residuum-Korrektur, da zuviel Speicherplatz zur Erhaltung mehrerer Kopien der Ausgangsfelder benötigt würde. Hier nun ein Beispiel für ein Benutzerprogramm, welches ein Gleichungssystem, einschließlich einem Iterationsschritt unter Verwendung von RSD, löst:

```
« → B A « B A / B A 3 PICK RSD A / + » »
```

Das Programm nimmt zwei Felder, **B** und **A**, vom Stack—ebenso wie "/"—und gibt das Ergebnisfeld **Z** zurück. **Z** stellt eine verbesserte Näherung der Lösung **X** dar.

INV — *Invertieren* — Analyt. Fkt.

Ebene 1	Ebene 1
[*Matrix*] ➡	[*Matrix*$^{-1}$]

INV (■ 1/x) gibt die invertierte Matrix des Arguments zurück. Als Argument muß eine quadratische Matrix, entweder reell oder komplex, spezifiziert sein.

SQ — *Quadrieren* — Analyt. Fkt.

Ebene 1	Ebene 1
[*Matrix*$_1$] ➡	[*Matrix*$_2$]

SQ (*SQuare*, ■ x²) gibt das Produkt einer mit sich selbst multiplizierten quadratischen Matrix zurück.

NEG — *Negieren* — Analyt. Fkt.

Ebene 1	Ebene 1
[*Feld*] ➡	[−*Feld*]

Ist keine Befehlszeile vorhanden, dann bewirkt das Drücken von CHS die Ausführung der Funktion NEG. Bei einem Feld ist jedes Element des Ergebnisses die Negation des zugehörigen Felds im Argument.

Um die Funktion NEG in die Befehlszeile einzugeben, ist die Menütaste NEG (in der vierten Menüzeile des des ARRAY-Menüs) zu verwenden.

...ARRAY

→ARRY ARRY→ PUT GET PUTI GETI

Diese Befehlsgruppe erlaubt Ihnen das Zurückrufen oder das Ändern individueller Feldelemente.

→ARRY *Stack in Feld* Befehl

Ebene *nm*+1 ... Ebene 2	Ebene 1		Ebene 1
$x_1 \ldots x_n$	$\{ n \}$	➧	⊏ *Vektor* ⊐
$x_{11} \ldots x_{nm}$	$\{ n\ m \}$	➧	⊏ *Matrix* ⊐

→ARRY ergibt ein Feld, welches aus reellen oder komplexen Elementen besteht; die Elemente werden nacheinander aus der Ebene $\geqslant 2$ des Stacks genommen. →ARRY entnimmt die Größe des Ergebnisfelds aus einer Liste in Ebene 1:

Vektoren: Wenn die Liste *eine* ganze Zahl n enthält, werden n Werte vom Stack genommen und in einem Vektor mit n Elementen angegeben.

Matrizen: Wenn die Liste *zwei* ganze Zahlen n und m enthält, werden nm Werte vom Stack genommen und in einer $n \times m$ Matrix angegeben.

Die Elemente des Ergebnisfelds sollten in Zeilenreihenfolge in den Stack eingegeben werden, mit x_{11} (oder x_1) in Ebene $nm + 1$ (oder $n + 1$), und x_{nm} (oder x_n) in Ebene 2. Wenn eines oder mehrere der Elemente eine komplexe Zahl ist, dann ist das Ergebnisfeld ebenfalls komplex.

ARRY→ *Feld in Stack* Befehl

Ebene 1		Ebene n+1 ... Ebene 2	Ebene 1
⊏ *Vektor* ⊐	➧	$x_1 \ldots x_n$	$\{ n \}$
⊏ *Matrix* ⊐	➧	$x_{11} \ldots x_{nm}$	$\{ n\ m \}$

ARRY→ nimmt ein Feld vom Stack und speichert seine Elemente als einzelne reelle oder komplexe Zahlen in den Stack zurück. ARRY→ gibt außerdem eine Liste, welche die Dimension des Felds enthält, in die Ebene 1 zurück. Die Feldelemente werden zeilenweise in den Stack geschrieben.

Vektoren: Wenn das Argument ein Vektor mit n Elementen ist, dann wird das erste Element in die Ebene $n + 1$ und das letzte in Ebene 2 zurückgegeben. Ebene 1 enthält dann die Liste $\{ n \}$.

Matrizen: Wenn das Argument eine $n \times m$ Matrix ist, dann wird das Element x_{nm} in Ebene 2 und das Element x_{11} in Ebene $(nm + 1)$ zurückgegeben.

PUT *Element übernehmen* Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
⌈ *Feld*$_1$ ⌋	{ *Index* }	*x*	➧	⌈ *Feld*$_2$ ⌋
' *Name* '	{ *Index* }	*x*	➧	
⌈ *K-Feld*$_1$ ⌋	{ *Index* }	*z*	➧	⌈ *K-Feld*$_2$ ⌋
' *Name* '	{ *Index* }	*z*	➧	
{ *Liste*$_1$ }	*n*	*Objekt*	➧	{ *Liste*$_2$ }
' *Name* '	*n*	*Objekt*	➧	

PUT schreibt ein Element in ein Feld oder eine Liste. Dieser Abschnitt beschreibt die Verwendung mit einem Feld; der Gebrauch mit einer Liste wird unter "LIST" erläutert.

PUT nimmt 3 Argumente vom Stack. Ebene 1 muß die Zahl enthalten, welche Sie in das Feld übernehmen möchten. Wenn es sich um eine komplexe Zahl handelt, muß das Feld auch komplex sein.

Ebene 2 enthält den Index des Feldelements, welches ersetzt werden soll. Handelt es sich bei dem Feld um einen Vektor, so enthält die Liste eine ganze Zahl, welche den Index des Vektorelements darstellt. Liegt als Feld eine Matrix vor, so hat die Liste das Format { *Zeile Spalte* }.

...ARRAY

Ebene 3 kann entweder ein Feld oder einen Namen enthalten. Wenn es sich um ein Feld handelt, gibt PUT dieses Feld in den Stack zurück, wobei ein Element durch den Wert in Ebene 1 ersetzt wurde.

Enthält Ebene 3 einen Namen, so schreibt PUT den Wert von Ebene 1 als ein Element in das Feld, welches unter der Variable *Name* gespeichert ist. (*Name* kann kein lokaler Name sein.)

GET ist die umgekehrte Operation zu PUT.

GET ***Element holen*** **Befehl**

Ebene 2	Ebene 1		Ebene 1
[*Feld*]	{ *Index* }	➧	*z*
' *Name* '	{ *Index* }	➧	*z*
{ *Liste* }	*n*	➧	*Objekt*
' *Name* '	*n*	➧	*Objekt*

GET ist ein allgemeines Verfahren, um ein Element aus einem Feld oder einer Liste zu holen. In diesem Abschnitt wird die Verwendung mit einem Feld beschrieben. Der Gebrauch im Zusammenhang mit einer Liste ist unter "LIST" erläutert.

GET nimmt zwei Argumente vom Stack. Ebene 1 sollte eine Index-Liste enthalten, welche das abzurufende Element spezifiziert. Handelt es sich bei dem Feld um einen Vektor, muß die Liste eine ganze Zahl zur Spezifikation des gewünschten Vektorelements enthalten. Liegt bei dem Feld eine Matrix vor, so muß die Index-Liste das Format { *Zeile Spalte* } besitzen.

Ebene 2 kann entweder ein Feld oder einen Namen enthalten. Bei einem Feld holt GET das indizierte Element dieses Felds in den Stack. Enthält Ebene 2 einen Namen, so holt GET das indizierte Element des unter *Name* gespeicherten Felds in den Stack zurück.

PUT ist die umgekehrte Operation zu GET.

PUTI — *Übernehmen und Index erhöhen* — Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 2	Ebene 1
[$Feld_1$]	{ $Index_1$ }	x	➧	[$Feld_2$]	{ $Index_2$ }
' *Name* '	{ $Index_1$ }	x	➧	' *Name* '	{ $Index_2$ }
[$K\text{-}Feld_1$]	{ $Index_1$ }	z	➧	[$K\text{-}Feld_2$]	{ $Index_2$ }
' *Name* '	{ $Index_1$ }	z	➧	' *Name* '	{ $Index_2$ }
{ $Liste_1$ }	n_1	*Objekt*	➧	{ $Liste_2$ }	n_2
' *Name* '	n_1	*Objekt*	➧	' *Name* '	n_2

PUTI (*PUT and Increment*) ist ein allgemeines Verfahren, um ein Element in ein Feld oder ein Objekt in eine Liste zu schreiben. Der Gebrauch in Zusammenhang mit einer Liste ist unter "LIST" beschrieben.

PUTI übernimmt einen Wert in ein Feld in der gleichen Art und Weise wie PUT, aber gleichzeitig wird der Index der Feldelemente für das nächste Element erhöht. Damit bleibt der Stack immer für die nächste Eingabe eines neuen Elements bereit; führen Sie nach dem Eintippen PUTI erneut aus, um den Wert als nächstes Element in das Feld zu übernehmen. Wenn der Index das Maximum für das vorliegende Feld erreicht hat, gibt PUTI den Index {1} (für Vektoren) oder {1 1} (für Matrizen) zurück und startet den Eingabeprozeß erneut vom Anfang des Feldes.

PUTI nimmt drei Argumente vom Stack. Ebene 1 muß die Zahl enthalten, welche Sie in das Feld schreiben möchten. Wenn es sich um eine komplexe Zahl handelt, muß das Feld ebenfalls komplex sein.

Ebene 2 enthält die Index-Liste, welche das zu ersetzende Feldelement spezifiziert. Wenn das Feld ein Vektor ist, sollte die Liste einen Index enthalten; der das entsprechende Element spezifiziert. Liegt als Feld eine Matrix vor, so muß die Index-Liste das Format { *Zeile Spalte* } besitzen.

Ebene 3 kann entweder ein Feld oder einen Namen enthalten. Wenn ein Feld vorliegt, gibt PUTI dieses Feld in Ebene 2 zurück, wobei das indizierte Element durch den Wert aus Ebene 1 ersetzt wurde; in Ebene 1 wird der nächste Index geschrieben.

Enthält Ebene 3 einen Namen, so schreibt PUTI die Zahl von Ebene 1 als *n*-tes Element in das unter *Name* gespeicherte Feld. Name wird dabei in Ebene 2 zurückgegeben und der nächste Index wird in Ebene 1 gestellt.

GETI ist die umgekehrte Operation zu PUTI.

GETI ***Holen und Index erhöhen*** **Befehl**

Ebene 2	Ebene 1		Ebene 3	Ebene 2	Ebene 1
⊏ *Feld* ⊐	⟨ $Index_1$ ⟩	➧	⊏ *Feld* ⊐	⟨ $Index_2$ ⟩	z
' *Name* '	⟨ $Index_1$ ⟩	➧	' *Name* '	⟨ $Index_2$ ⟩	z
⟨ *Liste* ⟩	n_1	➧	⟨ *Liste* ⟩	n_2	*Objekt*
' *Name* '	n_1	➧	' *Name* '	n_2	*Objekt*

GETI (*GET and Increment*) ist ein allgemeines Verfahren, um ein Element aus einer Liste oder einem Feld zu holen. Der Gebrauch von GETI im Zusammenhang mit Listen wird unter "LIST" beschrieben.

GETI ruft ein Element aus einem Feld in der gleichen Art und Weise zurück, wie es durch GET erfolgt. Als zusätzliche Fähigkeit jedoch läßt GETI das Feld im Stack und erhöht den Index für die Feldelemente auf das nächste Element. Damit wird das fortlaufende Abrufen der Elemente aus demselben Feld ermöglicht. Wenn der Index das Maximum für das vorliegende Feld erreicht hat, gibt GETI den Index {1} (für Vektoren) und {1 1} (für Matrizen) zurück und beginnt erneut mit dem Abrufen des ersten Elements.

GETI nimmt zwei Argumente vom Stack. Ebene 1 sollte dabei einen Index (eine oder zwei ganze Zahlen) enthalten, welcher das zu holende Feldelement spezifiziert. Wenn als Feld ein Vektor vorliegt, sollte die Index-Liste einen Wert enthalten, welcher das jeweilige Element spezifiziert. Handelt es sich bei dem Feld um eine Matrix, so muß die Index-Liste das Format { *Zeile Spalte* } besitzen.

Ebene 2 kann ein Feld oder einen Namen enthalten. Für ein Feld holt GETI das indizierte Element dieses Felds in Ebene 1. GETI holt außerdem das Feld in Ebene 3 und den Index des nächsten Elements in Ebene 2.

Wenn Ebene 2 einen Namen enthält, holt GETI das indizierte Element des unter der Variablen *Name* gespeicherten Felds in Ebene 1. Weiterhin wird *Name* in Ebene 3 und der Index des nächsten Elements in Ebene 2 geholt.

PUTI ist die umgekehrte Operation zu GETI.

SIZE **RDM** **TRN** **CON** **IDN** **RSD**

SIZE ***Größe*** **Befehl**

Ebene 1		**Ebene 1**
"String"	➧	*n*
{ Liste }	➧	*n*
[Feld]	➧	*{ Liste }*
' Symbol '	➧	*n*

...ARRAY

SIZE gibt ein Objekt zurück, welches die Größe oder Dimension einer Liste, eines Felds, eines Strings oder eines algebraischen Arguments enthält. Für ein Feld gibt SIZE eine Liste mit einem oder zwei ganzzahligen Werten zurück:

- Wenn das ursprüngliche Objekt ein Vektor ist, enthält die Liste nur eine ganze Zahl für die Anzahl der Elemente innerhalb des Vektors.
- Wenn das ursprüngliche Objekt eine Matrix ist, enthält die Liste zwei ganzzahlige Werte für die Dimension der Matrix. Die erste Zahl spezifiziert die Anzahl der Zeilen, die zweite Zahl gibt die Anzahl der Spalten an.

Beziehen Sie sich auf die Abschnitte "STRING", "LIST" und "ALGEBRA", wenn Sie Informationen zum Gebrauch von SIZE für andere Objekttypen erhalten möchten.

RDM ***Redimensionieren*** **Befehl**

Ebene 2	Ebene 1		Ebene 1
⟦ *Feld*$_1$ ⟧	⟨ *Dim* ⟩	➡	⟦ *Feld*$_2$ ⟧
' *Name* '	⟨ *Dim* ⟩	➡	

RDM ordnet die Elemente von *Feld*$_1$, welches von der Ebene 2 (oder von einer Variablen *Name*) genommen wird, neu an und gibt *Feld*$_2$ zurück. Die Dimension von *Feld*$_2$ ist in der Liste, welche sich in Ebene 1 befindet, spezifiziert. Wenn das Feld in Ebene 2 durch einen Namen spezifiziert ist, ersetzt *Feld*$_2$ das *Feld*$_1$ als neuen Inhalt der Variablen. Enthält die Liste nur eine ganze Zahl n, so ist *Feld*$_2$ ein Vektor mit n Elementen. Besitzt die Liste die Form $\{n\ m\}$, so handelt es sich bei *Feld*$_2$ um eine $n \times m$ Matrix.

Die von *Feld*$_1$ genommenen Elemente behalten dieselbe Reihenfolge in *Feld*$_2$. Wenn *Feld*$_2$ so redimensioniert wird, daß weniger Elemente als in *Feld*$_1$ vorgesehen sind, dann werden die überschüssigen Elemente von *Feld*$_1$ am Ende der Zeilenfolge verworfen. Wird *Feld*$_2$ so redimensioniert, daß mehr Elemente als in *Feld*$_1$ vorgesehen sind, so werden die zusätzlichen Elemente von *Feld*$_2$ am Ende mit Nullen ((0, 0), wenn *Feld*$_1$ komplex ist) aufgefüllt.

TRN — *Transponieren* — Befehl

Ebene 1		Ebene 1
⊏ *Matrix*$_1$ ⊐	➧	⊏ *Matrix*$_2$ ⊐
' *Name* '	➧	

TRN gibt die (konjugierte) Transposition seines Arguments zurück. Das bedeutet, eine $n \times m$ Matrix **A** in Ebene 1 (oder als Inhalt von *Name*) wird durch eine $m \times n$ Matrix $\mathbf{A}^t$ ersetzt, wobei gilt:

$$\mathbf{A}^t_{ij} = \begin{cases} \mathbf{A}_{ji} \text{ für reelle Matrizen} \\ \text{CONJ } (\mathbf{A}_{ji}) \text{ für komplexe Matrizen} \end{cases}$$

Wenn die Matrix durch *Name* spezifiziert wurde, ersetzt $\mathbf{A}^t$ die Matrix **A** in *Name*.

CON — *Konstanten-Feld* — Befehl

Ebene 2	Ebene 1		Ebene 1
⟨ *Dim* ⟩	*z*	➧	⊏ *Feld* ⊐
⊏ *Feld*$_1$ ⊐	*x*	➧	⊏ *Feld*$_2$ ⊐
⊏ *K-Feld*$_1$ ⊐	*z*	➧	⊏ *K-Feld*$_2$ ⊐
' *Name* '	*z*	➧	

CON erzeugt ein *Konstanten-Feld*—ein Feld, in welchem alle Elemente den gleichen Wert haben. Die Konstante ist entweder eine reelle oder eine komplexe Zahl, welche von Ebene 1 genommen wird. Das Ergebnisfeld ist entweder ein neues oder ein bereits existierendes Feld, dessen Elemente entsprechend dem Objekt in Ebene 2 durch einen konstanten Wert ersetzt werden.

Erzeugen eines neuen Felds. Wenn Ebene 2 eine Liste mit einer oder zwei ganzen Zahl enthält, so wird ein neues Feld in den Stack zurückgegeben. Liegt nur ein ganzzahliges *n* vor, dann ist das Ergebnis ein Konstanten-Vektor mit *n* Elementen. Besitzt die Liste die Form {*n m*}, so ist das Ergebnis eine Konstanten-Matrix mit *n* Zeilen und *m* Spalten.

Ersetzen der Elemente eines existierenden Felds. Wenn Ebene 2 einen Namen enthält, so muß dieser Name eine Benutzervariable identifizieren, welche ein Feld zum Inhalt hat. In diesem Fall werden die Elemente des Felds durch die Konstante in Ebene 1 ersetzt. Wenn die Konstante eine komplexe Zahl ist, so muß das Ausgangsfeld vom Typ komplex sein.

Wenn Ebene 2 ein Feld enthält, so wird ein Feld mit derselben Dimension zurückgegeben, wobei jedes Element durch die Konstante aus Ebene 1 ersetzt wurde. Handelt es sich bei der Konstante um eine komplexe Zahl, so muß das Ausgangsfeld vom Typ komplex sein.

IDN ***Einheitsmatrix*** **Befehl**

Ebene 1		Ebene 1
n	➧	[*R-Einheitsmatrix*]
[*Matrix*]	➧	[*Einheitsmatrix*]
' *Name* '	➧	

IDN (identity) erzeugt eine *Einheitsmatrix*—eine quadratische Matrix, deren Diagonale mit lauter Einsen besetzt ist, während alle restlichen Elemente mit Nullen besetzt sind. Die Ergebnismatrix ist in Abhängigkeit des Arguments in Ebene 1 entweder eine neue Matrix oder eine existierende, deren Elemente durch die Elemente der Einheitsmatrix ersetzt werden.

Erzeugen einer neuen Matrix. Handelt es bei dem Argument um eine reelle Zahl, so wird eine neue reelle Einheitsmatrix in den Stack zurückgegeben, wobei die Anzahl der Zeilen und Spalten dem Argument entsprechen.

Ersetzen der Elemente einer existierenden Matrix. Wenn das Argument ein Name ist, so muß dieser Name eine Benutzervariable identifizieren, welche eine quadratische Matrix zum Inhalt hat. In diesem Fall werden die Elemente der Matrix durch die Elemente der Einheitsmatrix ersetzt (komplex, wenn die Ausgangsmatrix vom Typ komplex ist).

Ist das Argument eine quadratische Matrix, dann wird eine Einheitsmatrix derselben Dimension zurückgegeben. Wenn die Ausgangsmatrix vom Typ komplex ist, dann wird ebenfalls eine komplexe Einheitsmatrix zurückgegeben, wobei die Elemente in der Diagonalen die Werte (1,0) besitzen.

RSD ***Residuum*** **Befehl**

Ebene 3	Ebene 2	Ebene 1		Ebene 1
[*Feld* **B**]	[*Matrix* **A**]	[*Feld* **Z**]	➧	[*Feld* **B**−**AZ**]

RSD berechnet das *Residuum* **B** − **AZ** der drei Felder **B**, **A** und **Z**. RSD wird normalerweise dazu benutzt, um eine Korrektur von **Z** zu berechnen, wobei **Z** als eine Näherung an die Lösung von **X** des Gleichungssystems **AX** = **B** erhalten wurde. Beziehen Sie sich für Informationen zum Gebrauch von RSD auf den Abschnitt "Erhöhen der Genauigkeit beim Lösen linearer Gleichungssysteme" am Anfang der ARRAY Beschreibung.

A, **B** und **Z** unterliegen folgenden Restriktionen:

- **A** muß eine Matrix sein.
- Die Anzahl der Spalten von **A** muß gleich der Anzahl der Elemente von **Z** (wenn **Z** ein Vektor ist) oder der Anzahl der Zeilen von **Z** sein (wenn **Z** eine Matrix ist).
- Die Anzahl der Spalten von **A** muß gleich der Anzahl der Elemente von **B** (wenn **B** ein Vektor ist) oder der Anzahl der Zeilen von **B** sein (wenn **B** eine Matrix ist).
- Beide, **B** und **Z**, müssen jeweils Vektoren oder Matrizen sein.
- **B** und **Z** müssen dieselbe Anzahl von Spalten haben, sofern es sich um Matrizen handelt.

...ARRAY

CROSS DOT DET ABS RNRM CNRM

CROSS *Kreuzprodukt* Befehl

Ebene 2	Ebene 1		Ebene 1
[*Vektor* **A**]	[*Vektor* **B**]	➧	[*Vektor* **A** × **B**]

CROSS gibt das Kreuzprodukt **A** × **B** von zwei 3-elementigen Vektoren **A** und **B** zurück, wobei gilt:

$$(\mathbf{A} \times \mathbf{B})_1 = \mathbf{A}_2\mathbf{B}_3 - \mathbf{A}_3\mathbf{B}_2$$
$$(\mathbf{A} \times \mathbf{B})_2 = \mathbf{A}_3\mathbf{B}_1 - \mathbf{A}_1\mathbf{B}_3$$
$$(\mathbf{A} \times \mathbf{B})_3 = \mathbf{A}_1\mathbf{B}_2 - \mathbf{A}_2\mathbf{B}_1$$

DOT *Skalares Produkt* Befehl

Ebene 2	Ebene 1		Ebene 1
[*Feld* **A**]	[*Feld* **B**]	➧	*x*

DOT gibt das skalare Produkt (oder auch "Punktprodukt") **A**·**B** von zwei Feldern **A** und **B** zurück. Das Skalarprodukt berechnet sich aus der Summe der Produkte, welche sich aus der Multiplikation der sich entsprechenden Elemente beider Felder ergibt. Als Beispiel:
`[1 2 3] [4 5 6] DOT` ergibt 1 × 4 + 2 × 5 + 3 × 6, oder 32.

Manchmal wird das Skalarprodukt zweier komplexer Felder auch als Summe der Produkte der konjugierten, sich entsprechenden Feldelemente, definiert. Ihr Rechner verwendet jedoch die gewöhnlichen Produkte, ohne vorherige Konjugation. Sollten Sie die andere Definition bevorzugen, so können Sie CONJ auf eines der beiden Felder anwenden, bevor Sie DOT ausführen.

DET — *Determinante* — Befehl

Ebene 1		Ebene 1
⌈ *Matrix* ⌋	➡	*Determinante*

DET gibt die Determinante des Arguments, welches aus einer quadratischen Matrix bestehen muß, zurück.

ABS — *Absoluter Betrag* — Funktion

Ebene 1		Ebene 1
z	➡	|*z*|
⌈ *Feld* ⌋	➡	‖*Feld*‖
' *Symbol* '	➡	' ABS (*Symbol*) '

ABS gibt den Absolutbetrag seines Arguments zurück. Im Falle eines Felds ermittelt ABS die Frobenius-Norm des Felds, welche als die Quadratwurzel der Summe der quadrierten absoluten Beträge aller Elemente definiert ist.

Beziehen Sie sich für den Gebrauch von ABS im Zusammenhang mit anderen Objekten auf "REAL", "COMPLEX" und "ALGEBRA".

RNRM — *Zeilensummennorm* — Befehl

Ebene 1		Ebene 1
⌈ *Feld* ⌋	➡	*Zeilensummennorm*

RNRM (*Row NoRM*) gibt die Zeilensummennorm seines Arguments zurück. Die Zeilensummennorm ist der maximale Wert (über alle Zeilen) von den Summen der absoluten Beträge aller Elemente einer Zeile. Für einen Vektor stellt die Zeilensummennorm den größten Absolutbetrag von allen Elementen dar.

...ARRAY

CNRM — Spaltensummennorm — Befehl

Ebene 1	Ebene 1
[*Feld*] ➧	*Spaltensummennorm*

CNRM (*Column NoRM*) gibt die Spaltensummennorm seines Arguments zurück. Die Spaltensummennorm ist der maximale Wert (über alle Spalten) von den Summen der absoluten Beträge aller Elemente einer Spalte. Für einen Vektor stellt die Spaltensummennorm die Summe der Absolutbeträge aller Elemente dar.

R→C **C→R** **RE** **IM** **CONJ** **NEG**

R→C — Reell nach Komplex — Befehl

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	(*x*,*y*)
[$R\text{-}Feld_1$]	[$R\text{-}Feld_2$]	➧	[*K-Feld*]

R→C kombiniert zwei reelle Zahlen (oder zwei reelle Felder) in eine komplexe Zahl (oder komplexes Feld). Das Objekt in Ebene 2 wird als reeller Anteil des Ergebnisses benutzt; das Objekt in Ebene 1 stellt den imaginären Teil des Ergebnisses dar.

Bei Feld-Argumenten stellen die Elemente des Ergebnisfelds komplexe Zahlen dar. Die Real- und Imaginärteile des Ergebnisses entsprechen den Argumenten in Ebene 2 bzw. Ebene 1. Die Felder müssen dieselbe Dimension besitzen.

C→R *Komplex in reell* Befehl

Ebene 1		Ebene 2	Ebene 1
(x,y)	➧	x	y
[K-Feld]	➧	[R-Feld$_1$]	[R-Feld$_2$]

C→R gibt in Ebene 2 und 1 den Real- bzw. Imaginärteil einer komplexen Zahl oder eines komplexen Felds.

Der Real- oder Imaginärteil eines komplexen Felds ist ein reelles Feld mit der gleichen Dimension, wobei die jeweiligen Elemente den reellen oder imaginären Teilen des jeweiligen komplexen Felds entsprechen.

RE *Reeller Teil* Funktion

Ebene 1		Ebene 1
x	➧	x
(x,y)	➧	x
[R-Feld]	➧	[R-Feld]
[K-Feld]	➧	[R-Feld]
'Symbol'	➧	'RE(Symbol)'

RE gibt den reellen Teil seines Arguments zurück. Wenn es sich bei dem Argument um ein Feld handelt, so gibt RE ein reelles Feld zurück, dessen Elemente den reellen Anteilen des Feld-Arguments entsprechen.

IM — *Imaginärer Teil* — Funktion

Ebene 1		Ebene 1
x	➧	*0*
⟨*x*, *y*⟩	➧	*y*
[*R-Feld*]	➧	[*R-Feld (Null)*]
[*K-Feld*]	➧	[*R-Feld*]
'*Symbol*'	➧	'IM⟨*Symbol*⟩'

IM gibt den imaginären Teil seines Arguments zurück. Wenn es sich bei dem Argument um ein Feld handelt, so gibt IM ein reelles Feld zurück, dessen Elemente den imaginären Anteilen des Feld-Arguments entsprechen. Enthält das Feld nur reelle Teile, so sind alle Elemente des Ergebnisfelds gleich Null.

CONJ — *Konjugiert komplex* — Analyt. Fkt.

Ebene 1		Ebene 1
x	➧	*x*
⟨*x*, *y*⟩	➧	⟨*x*, −*y*⟩
[*R-Feld*]	➧	[*R-Feld*]
[$K\text{-}Feld_1$]	➧	[$K\text{-}Feld_2$]
'*Symbol*'	➧	'CONJ⟨*Symbol*⟩'

CONJ gibt den konjugierten Wert einer komplexen Zahl bzw. eines komplexen Felds zurück. Der imaginäre Teil einer komplexen Zahl oder die Elemente eines komplexen Felds werden dabei negiert. Bei reellen Zahlen/Feldern ist der konjugiert komplexe Ausdruck gleich dem Ausgangsargument.

NEG — *Negieren* — Analyt. Fkt.

Ebene 1	Ebene 1
[*Feld*]	➧ [−*Feld*]

Bei einem Feld ist jedes Element des Ergebnisfelds im Vergleich zum entsprechenden Element des Ausgangsfelds negiert.

Ist keine Befehlszeile vorhanden, dann bewirkt das Drücken von CHS die Ausführung der Funktion NEG. Ein vollständiges Stackdiagramm für NEG finden Sie unter "Arithmetik".

BINARY

DEC	HEX	OCT	BIN	STWS	RCWS
RL	RR	RLB	RRB	R→B	B→R
SL	SR	SLB	SRB	ASR	
AND	OR	XOR	NOT		

Mit *Binärwerten* (oder *binary integers*) sind vorzeichenlose, ganze Zahlenwerte gemeint, welche vom Rechner intern als binäre Zahlen mit einer Länge von 1 bis 64 Bits dargestellt werden. Jeder dieser "Binärwerte" muß als String eingegeben werden, dem das Trennzeichen # vorangeht; die Anzeige erfolgt entsprechend.

Die Eingabe und die Anzeige von Binärwerten wird von der momentan spezifizierten *Zahlenbasis* gesteuert. Sie kann binär (Basis 2), oktal (Basis 8), dezimal (Basis 10) oder hexadezimal (Basis 16) vorgegeben werden. Wenn Sie die momentane Einstellung unter Verwendung der Tasten BIN, OCT, DEC oder HEX ändern, wird die interne Speicherung im Stack nicht verändert, sondern lediglich die Darstellung der Werte in der Anzeige (in Abhängigkeit zur neuen Basis).

Bei der Eingabe eines Binärwerts müssen die Zeichen für die momentane Basis Gültigkeit besitzen. Bei der binären Basis sind nur die Ziffern 0 und 1 erlaubt; bei oktaler Basis sind nur die Ziffern 0-7 erlaubt; bei dezimaler Basis sind die Ziffern 0-9 erlaubt, und bei hexadezimaler Basis die Ziffern 0-9 und die Buchstaben A-F. Als Voreinstellung ist die dezimale Basis spezifiziert.

Alle in einer Zeile eingegebenen Binärwerte beziehen sich auf die gleiche (momentane) Basis. Da die vier Menütasten zur Auswahl einer Basis kein ENTER durchführen, läßt sich die Basis sogar dann noch ändern, wenn Sie teilweise bereits mit der Eingabe Ihrer Werte begonnen haben. Allerdings wird die Syntax von Binärwerten in der Befehlszeile geprüft (gegen die momentane Basis), bevor die Befehlszeile ausgewertet wird. Es ist also nicht möglich, Werte für verschiedene Zahlenbasen einzugeben, indem während der Eingabe durch Drücken einer entsprechenden Menütaste die Basis geändert wird.

Die Stackanzeige von Binärwerten wird außerdem von der momentanen *Wortlänge* abhängig. Die Wortlänge kann mit dem Befehl STWS (*STore WordSize*) auf einen Wert im Bereich zwischen 1 und 64 festgesetzt werden. Wenn ein Binärwert im Stack angezeigt wird, erscheinen nur die niederwertigen Bits (bis zur jeweiligen Wortlänge), selbst wenn der Wert nicht abgeschnitten wurde. Nach der Verkleinerung der Wortlänge wird der angezeigte Wert verkürzt; eine laufende Vergrößerung bringt die "verborgenen" Bits in die Anzeige.

Der primäre Zweck der veränderlichen Wortlänge liegt in der Steuerung der Ergebnisse, welche aus den Befehlen resultieren, die Binärwerte als Argumente akzeptieren. Die Argumente werden dabei auf die spezifizierte Wortlänge abgeschnitten, und das Ergebnis wird entsprechend dieser Wortlänge zurückgegeben. Die Voreinstellung beträgt 64 Bits.

Die momentane Basis des verwendeten Zahlensystems und die Wortlänge sind über die Benutzerflags 37 bis 44 modifizierbar. Flag 37–42 entsprechen der binären Darstellung der momentanen Wortlänge minus 1 (Flag 42 ist das hochwertigste Bit). Flag 43 und 44 bestimmen die momentane Basis:

Flag 43	Flag 44	Basis
0	0	Dezimal
0	1	Binär
1	0	Oktal
1	1	Hexadezimal

Zusätzlich zu den Befehlen, welche über das BINARY-Menü zur Verfügung stehen und in den nächsten Abschnitten beschrieben werden, können die arithmetischen Funktionen +, −, * und / mit Paaren von Binärwerten (oder in Kombinationen mit reellen Zahlen) verwendet werden. Eine Anleitung dazu finden Sie unter "Arithmetik".

...BINARY

DEC HEX OCT BIN STWS RCWS

DEC ***Dezimal*** **Befehl**

DEC spezifiziert den Dezimal-Modus für Operationen mit Binärwerten. Die Binärwerte können dabei die Ziffern 0 bis 9 annehmen, wobei die Anzeige zur Basis 10 erfolgt.

DEC löscht Benutzerflag 43 und 44.

HEX ***Hexadezimal*** **Befehl**

HEX spezifiziert den Hexadezimal-Modus für Operationen mit Binärwerten. Die Binärwerte können dabei die Ziffern 0 bis 9 und die Buchstaben A bis F annehmen, wobei die Anzeige zur Basis 16 erfolgt.

HEX setzt Benutzerflag 43 und 44.

OCT ***Oktal*** **Befehl**

OCT spezifiziert den Oktal-Modus für Operationen mit Binärwerten. Die Binärwerte können dabei die Ziffern 0 bis 7 annehmen, wobei die Anzeige zur Basis 8 erfolgt.

HEX setzt Benutzerflag 43 und löscht 44.

BIN — Binär — Befehl

•

BIN spezifiziert den Binär-Modus für Operationen mit Binärwerten. Die Binärwerte können dabei die Ziffern 0 und 1 annehmen, wobei die Anzeige zur Basis 2 erfolgt.

BIN löscht Benutzerflag 43 und setzt 44.

STWS — Wortlänge speichern — Befehl

Ebene 1	
n •	

STWS (*STore WordSize*) spezifiziert n als momentane Wortlänge, wobei n eine reelle Zahl im Bereich zwischen 1 und 64 sein muß. Wenn $n > 64$ ist, wird eine Wortlänge von 64 Bits gesetzt. Wenn $n < 1$ ist, wird eine Wortlänge von 1 festgelegt. Die Benutzerflags 37–42 stellen die binäre Darstellung von $n - 1$ (Flag 42 ist dabei das hochwertigste Bit).

RCWS — Wortlänge abrufen — Befehl

	Ebene 1
• *n*	

RCWS (*ReCall WordSize*) gibt eine reelle Zahl n zurück, welche die momentane Wortlänge angibt. Die Benutzerflags 37–42 entsprechen der binären Darstellung für $n - 1$.

...BINARY

RL **RR** **RLB** **RRB** **R→B** **B→R**

Die Befehle RL und RR rotieren die einzelnen Bits von Binärwerten (in Abhängigkeit zur momentanen Wortlänge) nach links oder nach rechts. Die Befehle RLB und RRB sind gleichwertig zu RL oder RR, wenn diese acht mal wiederholt werden (was dem rotieren eines Bytes entspricht). R→B und B→R konvertieren reelle Zahlen in Binärwerte und umgekehrt.

RL ***Rotiere links*** **Befehl**

Ebene 1	Ebene 1
# n_1 ➧	# n_2

RL führt die Rotation eines binären Zahlenwerts # n_1 um 1 Bit nach links aus. Das linke Bit von # n_1 wird zum rechten Bit des Ergebnisses # n_2.

RR ***Rotiere rechts*** **Befehl**

Ebene 1	Ebene 1
# n_1 ➧	# n_2

RR führt die Rotation eines binären Zahlenwerts # n_1 um 1 Bit nach rechts aus. Das rechte Bit von # n_1 wird zum linken Bit des Ergebnisses # n_2.

RLB ***Rotiere linkes Byte*** **Befehl**

Ebene 1	Ebene 1
# n_1 ➧	# n_2

RLB führt die Rotation eines binären Zahlenwerts # n_1 um 1 Byte nach links aus. Das linke Byte von # n_1 wird zum rechten Byte des Ergebnisses # n_2.

RRB — *Rotiere rechtes Byte* — Befehl

Ebene 1	Ebene 1
# n_1 ➧	# n_2

RRB führt die Rotation eines binären Zahlenwerts # n_1 um 1 Byte nach rechts aus. Das rechte Byte von # n_1 wird zum linken Byte des Ergebnisses # n_2.

R→B — *Reel in binär* — Befehl

Ebene 1	Ebene 1
n ➧	# n

R→B konvertiert eine reelle ganze Zahl n, wobei $0 \leqslant n \leqslant$ 1.84467440737E19, in das binäre Äquivalent # n. Wenn $n < 0$ ist, erhält man # 0. Ist $n >$ 1.84467440737E19, so ergibt sich als Ergebnis # FFFFFFFFFFFFFFFF (hex).

B→R — *Binär in reell* — Befehl

Ebene 1	Ebene 1
# n ➧	n

B→R konvertiert einen Binärwert # n in seine äquivalente Darstellung als reelle Zahl n. Wenn # $n >$ # 1000000000000 (dezimal) ist, bleiben nur die 12 signifikantesten Dezimalstellen in der Mantisse des Ergebnisses erhalten.

...BINARY

SL SR SLB SRB ASR

Die Befehle SL und SR verschieben Binärwerte (auf die momentanen Wortlänge gesetzt) Bit für Bit nach links oder rechts. RLB und RRB sind gleichwertig zu den Befehlen RL oder RR, wenn diese acht mal wiederholt werden.

SL — *Nach links verschieben* — Befehl

Ebene 1	Ebene 1
# n_1 ←	# n_2

SL (*Shift Left*) führt für einen binären Zahlenwert eine Verschiebung um ein Bit nach links durch. Das linke Bit von n_1 geht verloren, wobei das rechte Bit von n_2 auf Null gesetzt wird. SL ist gleichwertig zur binären Multiplikation mit 2 (wobei der Wert auf die momentane Wortlänge abgeschnitten wird).

SR — *Nach rechts verschieben* — Befehl

Ebene 1	Ebene 1
# n_1 →	# n_2

SR (*Shift right*) führt für einen binären Zahlenwert eine Verschiebung um ein Bit nach rechts durch. Das rechte Bit von n_1 geht verloren, wobei das linke Bit von n_2 auf Null gesetzt wird. SR ist gleichwertig zur binären Division mit 2.

SLB — *Linkes Byte verschieben* — Befehl

Ebene 1	Ebene 1
# n_1 ➧	# n_2

SLB (*Shift Left Byte*) führt für einen Binärwert eine Verschiebung um ein Byte nach links durch. SLB ist gleichwertig zur Multiplikation mit # 100 (hexadeziamal), wobei das Erbebnis auf die momentane Wortlänge abgeschnitten wird.

SRB — *Rechtes Byte verschieben* — Befehl

Ebene 1	Ebene 1
# n_1 ➧	# n_2

SRB (*Shift Right Byte*) führt für einen Binärwert eine Verschiebung um ein Byte nach rechts durch. SRB ist gleichwertig zur Division mit # 100 (hexadeziamal).

ASR — *Arithmetisch nach rechts verschieben* — Befehl

Ebene 1	Ebene 1
# n_1 ➧	# n_2

ASR führt für einen Binärwert eine arithmetische Verschiebung um 1 Bit nach rechts durch. Bei einer arithmetischen Verschiebung behält das hochwertigste Bit seinen Wert, während die Verschiebung nur auf *Wortlänge* - 1 Bits stattfindet.

...BINARY

AND OR XOR NOT

Die Befehle AND, OR, XOR und NOT können auf *Flags* (reelle Zahlen oder algebraische Ausdrücke) und auf Binärwerte angewendet werden. Im ersten Fall wirken die Befehle als logische Operatoren, welche "wahr"- oder "falsch"-Flags miteinander kombinieren. Bei Binärwerten führen die Befehle logische Operationen auf die individuellen Bits der Argumente aus.

Die nachfolgenden Beschreibungen sind für die Anwendung der Befehle mit Binärwerten als Argumente gedacht. "PROGRAM TEST" erläutert deren Anwendung auf Flags.

AND *Und* **Funktion**

Ebene 2	Ebene 1		Ebene 1
# n_1	# n_2	•	# n_3

AND gibt das logische UND von zwei als Argument verwendeten Binärwerten zurück. Das Bit im Ergebnis ist durch die zugehörigen Bits der zwei Argumente entsprechend der nachfolgenden Tabelle bestimmt:

# n_1	# n_2	AND Ergebnis # n_3
0	0	0
0	1	0
1	0	0
1	1	1

OR — *Oder* — Funktion

Ebene 2	Ebene 1		Ebene 1
# n_1	# n_2	➧	# n_3

OR gibt das logische ODER von zwei als Argument verwendeten Binärwerten zurück. Das Bit im Ergebnis ist durch die zugehörigen Bits der zwei Argumente entsprechend der nachfolgenden Tabelle bestimmt:

# n_1	# n_2	OR Ergebnis # n_3
0	0	0
0	1	1
1	0	1
1	1	1

XOR — *Exklusives Oder* — Funktion

Ebene 2	Ebene 1		Ebene 1
# n_1	# n_2	➧	# n_3

XOR gibt das logische "Exklusive ODER" von zwei als Argument verwendeten Binärwerten zurück. Das Bit im Ergebnis ist durch die zugehörigen Bits der zwei Argumente entsprechend der nachfolgenden Tabelle bestimmt:

# n_1	# n_2	XOR Ergebnis # n_3
0	0	0
0	1	1
1	0	1
1	1	0

...BINARY

NOT *Nicht* Funktion

Ebene 1	Ebene 1
# n_1	# n_2

NOT gibt das Einserkomplement seiner Argumente zurück. Jedes Bit im Ergebnis entspricht dem Komplement des zugehörigen Bits in # n_1.

# n_1	NOT Ergebnis # n_2
0	1
1	0

Höhere Mathematik

Ihr Rechner ist in der Lage, eine symbolische Differentiation jedes beliebigen algebraischen Ausdrucks (innerhalb der Einschränkung durch den verfügbaren Speicherbereich) und eine numerische Integration jeder beliebigen Prozedur (algebraische Syntax) durchzuführen. Außerdem kann Ihr Rechner die symbolische Integration von Polynomen ausführen. Bei allgemeineren Ausdrücken ist der ∫ Befehl fähig, automatisch über eine Taylorreihe eine Näherung an den Ausdruck und eine anschließende symbolische Integration des resultierenden Polynoms durchzuführen.

Differentiation

∂ **Differenzieren** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
' $Symbol_1$ '	' *Name* '	➧	' $Symbol_2$ '

∂ (■ d/dx) ermittelt die Ableitung eines algebraischen Ausdrucks $Symbol_1$ nach einer spezifizierten Variablen *Name*. (*Name* kann kein lokaler Name sein.) Die Form des Ergebnisses $Symbol_2$ hängt davon ab, ob ∂ als Teil eines algebraischen Ausdrucks oder als einzelnes Objekt ausgeführt wird.

Schrittweises Differenzieren eines algebraischen Ausdrucks

Die Ableitungsfunktion ∂ wird für algebraische Ausdrücke in einer besonderen Syntax dargestellt:

' ∂*Name*‹*Symbol*› ',

wobei *Name* die Variable, nach welcher abgeleitet werden soll, und *Symbol* den zu differenzierenden Ausdruck spezifiziert.

...Höhere Mathematik

So stellt zum Beispiel `'∂X(SIN(Y))'` die Ableitung von `SIN(Y)` nach `X` dar. Bei der Auswertung des gesamten Ausdrucks wird die Differentiation ein Schritt nach vorne verlegt—das Ergebnis ist die Ableitung des Ausdrucks, multipliziert mit einem neuen Teilausdruck, welcher die Ableitung seines Arguments darstellt. Ein Beispiel soll dies verdeutlichen. Betrachten Sie die Ableitung von `SIN(Y)` nach `X` (im Bogenmaß), wobei `Y` den Wert `'X^2'` hat:

```
'∂X(SIN(Y))' EVAL ergibt 'COS(Y)*∂X(Y)'.
```

Es ist zu erkennen, daß dies der strikten Anwendung der *Kettenregel* entspricht. Diese Beschreibung über das Verhalten von *∂*, zusammen mit den allgemeinen Eigenschaften von EVAL, ist ausreichend, um die Ergebnisse von nachfolgenden Auswertungen zu verstehen:

```
EVAL ergibt 'COS(X^2)*(∂X(X)*2*X^(2-1))',

EVAL ergibt 'COS(X^2)*(2*X)'.
```

Vollständig durchgeführte Differentiation

Wird *∂* als einzelnes Objetk—d.h. in der Reihenfolge

```
'Symbol' 'Name' ∂
```

anstatt als Teil eines algebraischen Ausdrucks ausgeführt, so erfolgt eine automatische Wiederholung der Auswertung von *Symbol*, bis keine Ableitungen mehr enthalten sind. Als Teil dieses Prozesses wird—sofern die abzuleitende Variable *Name* einen Wert besitzt—in der endgültigen Form des Ausdrucks überall dieser Wert für die Variable substituiert.

Um dieses Verhalten von *∂* mit der schrittweisen Differentiation im vorangehenden Abschnitt zu vergleichen, soll erneut das Beispiel `'SIN(Y)'`, wo `Y` den Wert `'X^2'` annimmt, untersucht werden:

```
'SIN(Y)' 'X' ∂ ergibt 'COS(X^2)*(2*X)'.
```

Alle einzelnen Schritte zur Differentiation wurden mit einer Operation ausgeführt.

Die Funktion ∂ stellt aufgrund der Form des Arguments in Ebene 1 fest, ob die automatische Wiederholung der Auswertungen durchzuführen ist. Handelt es sich bei dem Argument um einen Namen, so wird eine vollständige Differentiation durchgeführt. Liegt als Argument in Ebene 1 ein algebraischer Ausdruck vor, welcher nur einen Namen enthält, so wird nur ein Differentiationsschritt ausgeführt. Normalerweise werden algebraische Ausdrücke, welche nur aus einem Namen bestehen, automatisch in Namen-Objekte konvertiert. Die besondere Syntax von ∂ erlaubt diese Ausnahme, welche als Signal zur vollständigen oder schrittweisen Differentiation verwendet wird.

Differentiation von benutzerdefinierten Funktionen

Wenn ∂ auf eine benutzerdefinierte Funktion angewendet wird:

1. Der Ausdruck, welcher aus dem Funktionsnamen und dem in Klammern eingeschlossenen Argument besteht, wird durch einen Ausdruck ersetzt, welcher die Funktion definiert.
2. Die Argumente aus dem ursprünglichen Ausdruck werden durch die lokalen Namen innerhalb der definierten Funktion ersetzt.
3. Der neue Ausdruck wird differenziert.

Als Beispiel: Definieren Sie $F(a, b) = 2a + b$:

```
« → a b '2*a+b' » 'F' STO.
```

...Höhere Mathematik

Leiten Sie dann 'F(X,X^2)' nach X ab. Die Differentiation läuft dabei automatisch wie folgt ab:

1. 'F(X,X^2)' wird ersetzt durch '2*a+b'.
2. X wird für a und 'X^2' wird für b substituiert. Der Ausdruck lautet nun '2*X+X^2'.
3. Der neue Ausdruck wird differenziert.
   - Wenn die Auswertung '∂X(F(X,X^2))' durchgeführt wird, lautet das Ergebnis '∂X(2*X)+∂X(X^2)'.
   - Bei der Ausführung von 'F(X,(X^2))' 'X' ∂ wird die Differentiation bis zum endgültigen Ergebnis '2+2*X' durchgezogen.

Benutzerdefinierte Ableitungen

Wenn ∂ auf eine HP-28S Funktion, für welche eine interne Ableitung nicht verfügbar ist, angewendet wird, so gibt ∂ eine formale Ableitung zurück—eine neue Funktion, deren Name mit "der" (*derivative*) beginnt und vom ursprünglichen Funktionsname abgeschlossen wird. So beinhaltet z.B. im HP-28S die Definition von % keine Ableitung. Wenn Sie den Ausdruck '%(X,Y)' eine Stufe nach Z ableiten, erhalten Sïe

```
'der%(X,Y,∂Z(X),∂Z(Y))'
```

Jedes Argument der % Funktion resultiert in zwei Argumenten zur der% Funnktion. In diesem Beispiel ergibt das X Argument die X und ∂Z(X) Argumente, und das Y Argument ergibt die Y und ∂Z(Y) Argumente.

Sie können mit der Differentiation fortfahren, indem Sie eine benutzerdefinierte Funktion zur Darstellung der Ableitung erzeugen. Nachstehend eine Ableitung für %:

```
« → x y dx dy '(x*dy+y*dx)/100' » 'der%' STO.
```

Mit dieser Definition erhalten Sie eine korrekte Ableitung für die %-Funktion. Als Beispiel:

'%(X,2*X)' 'X' ∂ COLCT ergibt '.04*X'.

Das Verhalten von ∂ ist ähnlich, wenn ∂ auf eine formale Benutzerfunktion (ein Name, gefolgt von in Klammern eingeschlossenen Argumenten, für welchen keine benutzerdefinierte Funktion im Speicherbereich existiert) angewendet wird: ∂ gibt eine formale Ableitung zurück, deren Name mit "der" beginnt und vom ursprünglichen Namen der Benutzerfunktion abgeschlossen wird. Zum Beispiel erhält man bei der Ableitung der formalen Benutzerfunktion
`'f(x1,x2,x3)'` nach *x*

```
'derf(x1,x2,x3,∂x(x1),∂x(x2),∂x(x3))'
```

Integration

∫ ***Integrieren*** **Befehl**

Ebene 3	Ebene 2	Ebene 1		Ebene 2	Ebene 1
' Symbol '	*' Name '*	*Grad*	➧		*' Integral '*
x	*{ Name a b }*	*Genauigkeit*	➧	*Integral*	*Fehler*
' Symbol '	*{ Name a b }*	*Genauigkeit*	➧	*Integral*	*Fehler*
≪Programm≫	*{ Name a b }*	*Genauigkeit*	➧	*Integral*	*Fehler*
≪Programm≫	*{ a b }*	*Genauigkeit*	➧	*Integral*	*Fehler*

∫ gibt entweder ein Polynom, welches ein symbolisches unbestimmtes Integral darstellt, oder zwei reelle Zahlen als bestimmtes Integral zurück. Die Art des Ergebnisses ist durch die Argumente bestimmt. Im allgemeinen erfordert ∫ drei Argumente. Ebene 3 enthält das zu integrierende Objekt; das Objekt in Ebene 2 bestimmt die Form der Integration; das Objekt in Ebene 1 spezifiziert deren Genauigkeit.

...Höhere Mathematik

Symbolische Integration

∫ ermöglicht eine begrenzte symbolische Integration. Er kann ein genaues (unbestimmtes) Integral einer ganzen rationalen Funktion (Polynom) zurückgeben. Er kann ebenso ein angenähertes Integral ermitteln, indem zur Konvertierung des Integranden in ein Polynom eine Näherung durch eine Taylorreihe verwendet wird, und das so erhaltene Polynom integriert wird.

Um ein symbolisches Integral zu erhalten, müssen die Stackargumente folgender Art sein:

3: *Integrand* (Name oder algebraischer Ausdruck)
2: *Integrationvariable* (Name)
1: *Grad des Polynoms* (reelle Zahl)

Der *Grad des Polynoms* spezifiziert den Grad der Näherung durch die Taylorreihe (oder den Grad des *Integranden*, falls dieser bereits in Polynomform vorliegt).

Numerische Integration

Um ein numerisches Integral zu erhalten, müssen folgende Informationen spezifiziert sein:

- Der Integrand.
- Die Integrationsvariable.
- Obere und untere Grenze des Integrals.
- Die Genauigkeit des Integranden bzw. der akzeptable Fehler für das Integrationsergebnis.

Verwenden einer expliziten Integrationsvariable. Bei einer numerischen Integration, in welcher die Integrationsvariable mit einem Namen-Objekt gekennzeichnet ist, wird diese als *explizite Integrationsvariable* bezeichnet. Das Namen-Objekt erscheint dabei (normalerweise) in dem als Integrand definierten Objekt. Im nächsten Abschnitt, *implizite Integrationsvariable*, erfolgt eine Beschreibung für die Integrationsweise, für welche keine Integrationsvariable benannt werden muß.

Bei der Verwendung einer expliziten Integrationsvariable müssen die relevanten Objekte wie folgt eingegeben werden:

3: *Integrand*
2: *Integrationsvariable und Grenzen*
1: *Genauigkeit*

Der Integrand ist ein Objekt, welches den zu integrierenden mathematischen Ausdruck darstellt. Er kann aus nachstehenden Elementen bestehen:

- Eine reelle Zahl, welche eine Konstante als Integrand darstellt. In diesem Fall ergibt sich der Wert des Integrals wie folgt:

  Zahl (obere Grenze − untere Grenze).

- Ein algebraischer Ausdruck.
- Ein Programm. Das Programm muß dabei den algebraischen Syntaxregeln entsprechen—d.h. keine Argumente vom Stack nehmen und eine reelle Zahl als Ergebnis zurückgeben.

Die Integrationsvariable sowie die Integrationsgrenzen müssen in einer Liste in Ebene 2 entsprechend der Form

{ Name Untergrenze Obergrenze }

enthalten sein, wobei *Name* ein Namen-Objekt darstellt und es sich bei den Grenzen um reelle Zahlen handelt.

Die *Genauigkeit* ist eine reelle Zahl, welche die Fehlertoleranz bei der durchzuführenden Integration spezifiziert. (Eigentlich wird damit festgelegt, wie groß bzw. in wieviel Intervalle der Integrationsbereich zerlegt werden soll.)

...Höhere Mathematik

Die Genauigkeit wird als relativer Fehler spezifiziert:

$$Genauigkeit \geq \left| \frac{wahrer\ Wert - berechneter\ Wert}{berechneter\ Wert} \right|$$

wobei mit *Wert* der Wert des Integranden an jedem beliebigen Punkt innerhalb des Integrationsintervalls gemeint ist. Selbst wenn Ihr Integrand bis zu 12 signifikanten Stellen genau ist, möchten Sie vielleicht einen größeren Wert als Genauigkeit verwenden, um die Integrationszeit zu verkürzen; je kleiner der Wert für die Genauigkeit, desto länger dauert der Integrationsprozeß, da der zu integrierende Bereich in mehr Teilintervalle zerlegt wird.

Die Genauigkeit des Integranden ist primär von drei Betrachtungen abhängig:

- Die Genauigkeit von empirischen Konstanten in dem Ausdruck.
- Bis zu welchem Grad der Ausdruck einen physikalischen Vorgang genau beschreiben kann.
- Das Ausmaß von Rundungsfehlern bei der internen Auswertung des Ausdrucks.

Ausdrücke wie z.B. $cos\ (x - sin\ x)$ sind rein mathematischer Natur, wobei keine empirischen Konstanten enthalten sind. In diesem Fall besteht die einzige Beschränkung für die Genauigkeit in der Häufung von Rundungsfehlern, welche aufgrund der endlichen (12-Stellen) Genauigkeit bei der numerischen Auswertung des Ausdrucks auftreten können. Sie können natürlich für die Integration solcher Ausdrücke einen größeren Genauigkeitswert als den einfachen Rundungsfehler spezifizieren, um damit die Rechenzeit zu verkürzen.

Wenn der Integrand sich auf einen konkreten physikalischen Vorgang bezieht, sind zusätzliche Überlegungen anzustellen. Sie müssen sich in dieser Situation fragen, ob die gewünschte Genauigkeit bei der Integration durch die Genauigkeit des Integranden gerechtfertigt ist. Enthält der Integrand z.B. empirische Konstanten, welche nur auf 3 Stellen genau sind, so ist es nicht sehr sinnvoll, einen kleineren Genauigkeitswert als 1E-3 anzugeben.

Weiterhin ist zu bedenken, daß fast jede Funktion, welche sich auf einen physikalischen Prozess bezieht, von Natur aus ungenau ist, da es sich nur um ein mathematisches Modell eines tatsächlichen Prozesses oder Ereignisses handelt. Das Modell ist normalerweise eine Näherung, die die Auswirkung von Faktoren, welche im Vergleich mit den Faktoren des Modells als nicht signifikant beurteilt werden, vernachlässigt.

Um die numerische Integration zu veranschaulichen, soll der Ausdruck

$$\int_1^2 exp\ x\ dx$$

mit einer Genauigkeit von .00001 berechnet werden. Für ∫ sollte der Stack folgenden Aufbau haben:

```
3: 'EXP(X)'
2: { X 1 2 }
1: .00001
```

Die numerische Integration gibt zwei Zahlenwerte in den Stack zurück. Der Wert des Integrals erscheint in Ebene 2. Der in Ebene 1 zurückgegebene Fehler stellt eine Obergrenze für den relativen Rechenfehler dar, wobei normalerweise gilt:

$$Fehler = Genauigkeit \int |Integrand|$$

Wenn der Fehler aus einer negativen Zahl besteht, ist dies als Anzeichen zu verstehen, daß eine Konvergenz der Näherung nicht erreicht werden konnte und daß das Ergebnis in Ebene 2 die zuletzt berechnete Näherung ist.

Für das Integral von `'EXP(X)'` im vorangehenden Beispiel gibt ∫ den Wert 4.67077 in Ebene 2 und den Fehler 4.7E-5 in Ebene 1 zurück.

Verwenden einer impliziten Integrationsvariablen. Der Gebrauch einer expliziten Integrationsvariablen ermöglicht Ihnen, den Integranden als gewöhnlichen algebraischen Ausdruck einzugeben. Es ist jedoch auch möglich, den Integranden in UPN Form einzugeben, wodurch sich eine beachtliche Verkürzung der Rechenzeit ergeben kann (durch die Elimination der sich ansonsten wiederholenden Auswertung des Variablennamens). Bei diesem Verfahren wird eine *implizite* Integrationsvariable verwendet. Der Stack sollte dabei wie folgt aufgebaut sein:

3: *Integrand* (Programm)
2: *Integrationsgrenzen* (Liste)
1: *Genauigkeit* (relle Zahl)

Der *Integrand* muß bei diesem Verfahren aus einem Programm bestehen, welches eine reelle Zahl vom Stack nimmt und eine reelle Zahl in den Stack zurück gibt. ∫ wertet dieses Programm für jedes Teilintervall innerhalb der Integrationsgrenzen aus. Für jede Auswertung stellt ∫ den Intervall-Wert in den Stack; das Programm nimmt diesen Wert und gibt als Ergebnis den Wert des Integranden an diesem Punkt zurück.

Die *Integrationsgrenzen* müssen in einer Liste als zwei reelle Zahlen im Format {*Untergrenze Obergrenze*} eingegeben werden. Die *Genauigkeit* spezifiziert den relativen Fehler für die Berechnung, wie er im vorangehenden Abschnitt beschrieben wird.

Um zum Beispiel das Integral

$$\int_1^2 exp\ (x)\ dx$$

mit einer Genauigkeit von .00001 auszuwerten, sind im Stack vor der Ausführung von ∫ folgende Werte bereitzustellen:

```
3: « EXP »
2: { 1 2 }
1: .00001
```

Damit wird der Wert 4.67077 und die Genauigkeit 4.7E-5, wie im vorangehenden Beispiel, wo eine explizite Integrationsvariable verwendet wurde, in den Stack zurückgegeben.

Taylorsche Reihe

TAYLR ***Taylorsche Reihe*** **Befehl**

Ebene 3	Ebene 2	Ebene 1		Ebene 1
' *Symbol*$_1$ '	' *Name* '	*n*	➧	' *Symbol*$_2$ '

TAYLR (im ALGEBRA-Menü) führt eine Näherung über eine Taylorreihe für den Ausdruck *Symbol*$_1$ bis zur n-ten Ordnung in der Variablen *Name* durch. Die Näherung wird an der Stelle *Name* = 0 (auch MacLaurin Reihe genannt) durchgeführt. Die Taylorreihe von f(x) für $x = 0$ ist wie folgt definiert:

$$\sum_{i=0}^{n} \frac{x^i}{i!} \left(\frac{\partial^i}{\partial x^i} f(x) \right) \Bigg|_{x=0}$$

...Höhere Mathematik

Verlegen des Ausgangspunktes für die Entwicklung

Wenn Sie TAYLR einfach dazu verwenden, um ein Polynom in eine wahre Potenzform zu erheben, so spielt der Ausgangspunkt für die Entwicklung der Taylorreihe keine Rolle, da das resultierende Ergebnis korrekt ist. Verwenden Sie jedoch TAYLR, um eine Näherung für eine mathematische Funktion zu erhalten, kann es erforderlich sein, den Ausgangspunkt für die Reihenentwicklung von Null zu verlegen.

Sind Sie z.B. im Verhalten der Funktion für einen bestimmten Bereich interessiert, so ist ihre TAYLR Näherung hilfreicher, wenn der Ausgangspunkt für die Entwicklung in diesen Bereich verlegt wird. Ähnliches gilt, wenn die Funktion in Null keine Ableitung besitzt; eine TAYLR Näherung wäre nicht sinnvoll, solange der Ausgangspunkt nicht von Null verlagert wird.

Hinweis

Die Ausführung von TAYLR kann zu einem bedeutungslosen Ergebnis führen, wenn der Ausdruck in Null nicht differenzierbar ist. Wenn Sie z.B Flag 59 löschen (um `Infinite Result` Fehler zu vermeiden) und

```
'√X' 'X' 2 TAYLR
```

ausführen, erhalten Sie das Ergebnis `'5.E499*X-1.25E499*X^2'`. Der Koeffizient von X ist `∂X(X^.5)`, was gleichwertig mit `.5 * X^ - .5` ist und in 5.E499 für $x = 0$ resultiert.

Obwohl TAYLR die Funktion und ihre Ableitungen an der Stelle Null auswertet, können Sie durch Veränderungen der Variablen innerhalb des Ausdrucks den Ausgangspunkt von Null verlegen. Nehmen Sie z.B. an, der Ausdruck besteht in einer Funktion von X und Sie möchten eine TAYLR Näherung für $X = 2$. Um den Ausgangspunkt der Entwicklung durch Veränderung von Variablen zu verlegen:

1. Speichern Sie `'Y+2'` in `'X'`.
2. Werten Sie die Ausgangsfunktion aus, um die Variable von X in Y zu ändern.

3. Ermitteln Sie die Taylorreihe für Y = 0.
4. Löschen Sie X (falls X noch als Variable existiert).
5. Speichern Sie `'X-2'` in `'Y'`.
6. Werten Sie die neue Funktion aus, um die Variable von Y in X ändern zu können.
7. Löschen Sie Y.

Näherungen für rationale Funktionen

Der Quotient zweier Polynome wird als *rationale Funktion* bezeichnet. Wenn sich der Zähler ohne Rest durch den Nenner teilen läßt, ist die rationale Funktion gleichwertig mit einem Polynom. Als Beispiel:

$$\frac{x^3 + 2x^2 - 5x - 6}{x^2 - x - 2} = x + 3$$

Besteht Ihr Ausdruck aus solch einer rationalen Funktion, so können Sie ihn unter Verwendung von TAYLR in eine äquivalente Polynomform konvertieren. Läßt sich der Ausdruck jedoch nicht glatt (ohne Rest) teilen, so liegt in der rationalen Funktion *kein* Polynom vor. Als Beispiel:

$$\frac{x^3 + 2x^2 - 5x - 2}{x^2 - x - 2} = x + 3 + \frac{4}{x^2 - x - 2}$$

Es gibt keine äquivalente Polynomform für eine solche rationale Funktion, Sie können aber TAYLR trotzdem zur Berechnung eines Polynoms, welches für ein kleines x (nahe Null) genau ist, verwenden. Ebenso können Sie den Bereich der größten Genauigkeit von $x = 0$ verlegen und somit eine Genauigkeit für die Näherung wählen. Für das oben stehende Beispiel wäre die TAYLR Näherung ersten Grades $2x + 1$ für $x = 0$.

...Höhere Mathematik

Lange Polynom-Divisionen. Eine weitere hilfreiche Näherung an eine rationale Funktion ist der Polynom-Quotient, welcher aus einer längeren Division resultiert. Betrachten Sie die obenstehende rechte Seite als ein Polynom mit einem Restglied. Das Polynom ist eine gute Näherung zur rationalen Funktion, wenn der Rest sehr klein ist—d.h. wenn x groß ist. Beachten Sie den Unterschied zwischen dem Polynom-Quotienten $(x + 3)$ und der TAYLR Näherung des gleichen Grades $(2x + 1)$.

Die nachfolgenden Schritte zeigen Ihnen, wie eine lange Polynom-Division mit Ihrem HP-28S durchgeführt wird. Der Prozeß ist im allgemeinen gleich mit dem Vorgehen zur langen Division von Zahlen.

1. Erzeugen Sie Ausdrücke für den Zähler und Nenner, wobei beide in Potenzform vorliegen sollten.
2. Speichern Sie den Nenner in einer Variablen 'D' (für "Divisor").
3. Speichern Sie einen Ausgangswert von Null in einer mit 'Q' benannten Variablen (für "Quotient").

Nachdem der Zähler im Stack vorliegt, fahren Sie mit den nachstehenden Schritten fort. Der Zähler dient als Erstwert für den Dividenden. Bei jeder Wiederholung der Schritte 4 bis 8 wird ein Term Q hinzugefügt und der Dividend reduziert.

4. Bringen Sie D in den Stack (in Ebene 1).
5. Dividieren Sie den Dividenden-Term höchster Ordnung (Ebene 2) durch den Divisor-Term höchster Ordnung (Ebene 1). Sie können das Ergebnis im Kopf berechnen und eintippen, oder Sie geben einen Ausdruck wie

   '*Dividend-Term*/*Divisor-Term*'

   ein und bringen ihn in Potenzform.

   Wenn z.B. der Dividend $x^3 + 2x^2 - 5x - 2$ und der Divisor $x^2 - x - 2$ lautet, so erhalten Sie als Ergebnis x; liegt als Dividend $3x^3 + x^2 - 7$ und als Divisor $2x^2 + 8x + 9$ vor, beträgt das Ergebnis $1.5x$.

   Das Ergebnis stellt einen Term des Polynom-Quotienten dar.

6. Erzeugen Sie eine Kopie des Quotienten-Terms und fügen Sie diese Kopie Q hinzu.
7. Multiplizieren Sie den Quotienten-Term und den Divisor.
8. Subtrahieren Sie das Ergebnis vom Dividenden. Das Ergebnis stellt den neuen Dividenden dar.

Ist der Grad des neuen Dividenden höher oder gleich dem des Divisors, wiederholen Sie die Schritte 4 bis 8.

Wenn der Grad des neuen Dividenden kleiner als der des Divisors ist, sind Sie fertig. Der Polynom-Quotient ist in Q gespeichert, wobei der Rest dem Ergebnis aus endgültigem Dividenden, dividiert durch den Divisor, entspricht.

CATALOG

Ihr Rechner enthält einen Katalog, unter welchem alle verfügbaren Befehle in alphabetischer Reihenfolge zusammengefaßt sind. Sie können sich diesen Katalog durch Drücken von ■ CATALOG auflisten lassen. Anhand des Katalogeintrags können Sie z.B. die richtige Schreibweise des Befehls nachsehen oder überprüfen, welche Objekttypen als Argumente für den jeweiligen Befehl verwendbar sind. Befehle, die nicht mit einen Alpha-Zeichen beginnen, erscheinen bei der Auflistung nach dem Befehl XPON (letzter Befehl, der mit einem Alpha-Zeichen beginnt).

Nachdem Sie ■ CATALOG gedrückt haben, wird die normale Anzeige des HP-28S durch die Katalog-Anzeige überlagert:

In der obersten Zeile erscheint der Befehlsname. ABORT wird aufgrund der alphabetischen Reihenfolge als erster Befehl im HP-28S Befehlskatalog angezeigt.

...CATALOG

Das Katalog-Menü erscheint in der untersten Zeile der Anzeige. Die sechs Menütasten besitzen dabei folgende Wirkungsweise:

Menütaste	Bedeutung
NEXT	Anzeigen des nächsten Katalogeintrags (Wiederholungstaste).
PREV	Anzeigen des vorangehenden Katalogeintrags (Wiederholungstaste).
SCAN	Automatisches Durchblättern des Katalogs in aufsteigender Reihenfolge, wobei jeder Befehl kurz angezeigt wird. Dabei wird das Menüfeld SCAN durch das Feld STOP ersetzt; das Drücken von STOP bricht das Durchblättern an der momentanen Stelle ab; ansonsten wird der Vorgang nach der Anzeige des letzten Befehls (→STR) automatisch beendet.
USE	Aktiviert die USAGE Anzeige (Anwendung) für den angezeigten Befehl. Daraus wird ersichtlich, welche Argumente im Stack verwendet werden.
FETCH	Abschließen der Kataloganzeige und Hinzufügen der momentanen Befehlsabkürzung an der Cursor-Position. Falls keine Befehlszeile vorhanden ist, wird eine neue begonnen.
QUIT	Abschließen der Kataloganzeige; die Befehlszeile bleibt dabei völlig unverändert.

Zusätzlich zu den über das Katalog-Menü ausführbaren Operationen gibt es die Möglichkeiten:

- Durch Drücken einer Taste auf dem linken Tastenfeld kann der erste Befehl anzeigt werden, welcher mit dem entsprechenden Zeichen beginnt.
- Wenn zu dem gedrückten Alpha-Zeichen kein zugehöriger Befehl existiert, wird ein Katalogeintrag angezeigt, welcher mit dem vorangehenden Alpha-Zeichen beginnt.

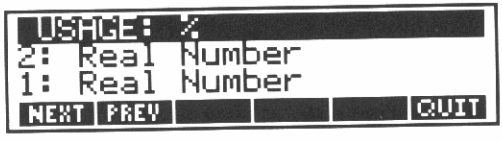
...CATALOG

- Wenn es zu dem gedrückten Sonderzeichen keinen Befehl gibt, wird als Katalogeintrag + angezeigt, was dem ersten Befehl entspricht, welcher nicht mit einem Alpha-Zeichen beginnt. (Die Taste [α LOCK] stellt den Katalogzeiger auf →STR, den letzten Katalogeintrag.)
- Das Drücken von [ON] bewirkt den Abschluß der Kataloganzeige und das Löschen der Befehlszeile.

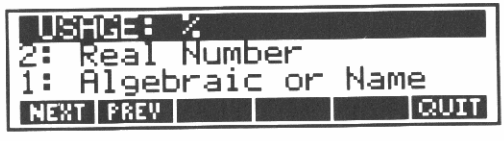
Wenn eine Taste gedrückt wird, welche während der Kataloganzeige nicht aktiviert ist, ertönt ein Akustiksignal als Hinweis.

USAGE

Das Drücken von USE aktiviert eine zweite Ebene der Katalogfunktion, welche als USAGE Anzeige (Anwendung des Befehls) bezeichnet wird. Für den % Befehl z.B. sieht die erste USAGE Anzeige wie folgt aus:



Daraus wird ersichtlich, daß für % zwei reelle Zahlen als Argumente verwendet werden. Wenn die Menütasten PREV und NEXT angezeigt werden, so stehen weitere Optionen für verwendbare Argumente zur Verfügung. Wenn Sie z.B. für % die Taste NEXT drücken, erhalten Sie folgende Anzeige:



Dies zeigt, daß % auch eine reelle Zahl in Ebene 2 und einen algebraischen Ausdruck oder einen Namen in Ebene 1 als Argumente akzeptiert. Es gibt zwei weitere Kombinationen für %, welche Sie sich durch Drücken von NEXT anzeigen lassen können. Indem Sie PREV drücken, läßt sich natürlich auch rückwärts durch die USAGE Anzeige blättern.

Sie können die Anzeige von USAGE beenden, indem Sie:

- QUIT drücken, um zur Kataloganzeige des momentanen Befehls zurückzukehren. Von hier aus ist es möglich, weitere Einträge anzeigen zu lassen, oder die Katalogfunktion durch erneutes Drücken von QUIT zu beenden.
- ON drücken, um die Katalogfunktion zu beenden. Durch Drücken von ON wird auch der Inhalt der momentanen Befehlszeile gelöscht.

COMPLEX

R→C	C→R	RE	IM	CONJ	SIGN
R→P	P→R	ABS	NEG	ARG	

Das COMPLEX-Menü (■ CMPLX) stellt Ihnen Befehle speziell zum Arbeiten mit komplexen Zahlen zur Verfügung. Objekte, welche aus *komplexen Zahlen* bestehen, werden im HP-28S als geordnete Zahlenpaare dargestellt. Die Zahlenpaare bestehen aus reellen Zahlen und sind in runden Klammern eingeschlossen; als Trennzeichen zwischen den Zahlen dient das Zeichen, welches gerade nicht als Dezimalzeichen verwendet wird (z.B. `(1,234,5,678)`. Ein Objekt, welches aus einer komplexen Zahl (x, y) besteht, kann folgende Datenpaare darstellen:

- Eine komplexe Zahl z in Rechteckskoordinaten, wobei x dem Realteil und y dem Imaginärteil von z entspricht.
- Eine komplexe Zahl z in Polarkoordinaten, wobei x dem Betrag von z und y dem Winkel entspricht.
- Die Rechteckskoordinaten eines Punktes im zweidimensionalen Raum, wobei x dem Abszissenabschnitt und y dem Ordinatenabschnitt entspricht.
- Die Polarkoordinaten eines Punktes im zweidimensionalen Raum, wobei x dem Betrag von z und y dem Winkel entspricht.

Wenn Sie mit der Anwendung von komplexen Zahlen nicht hinreichend vertraut sind, können Sie sich diese Art von Objekten auch einfach als zweidimensionale Vektoren oder Punktkoordinaten vorstellen. Die meisten der Befehle, welche sich auf komplexe Zahlen anwenden lassen, geben Ergebnisse zurück, welche für Berechnungen im herkömmlichen zweidimensionalen Raum sowie für komplexe Zahlen interpretierbar sind.

Außer dem Befehl P→R (Polar- in Rechteckskoordinaten) erwarten alle Befehle des HP-28S, welche mit Werten von komlexen Zahlen in Verbindung stehen, die Angabe der Argumente als Rechteckskoordinaten. Ebenso handelt es sich bei Befehlen, welche Ergebnisse von komplexen Zahlenberechnungen zurückgeben—außer R→P (Rechtecks- in Polarkoordinaten)—immer um Angaben als Rechteckskoordinaten.

...COMPLEX

Zusätzlich zu den nachstehend beschriebenen Befehlen gibt es noch eine Reihe anderer Befehle, welche komplexe Zahlen als Argumente akzeptieren:

- Arithmetische Funktionen +, −, *, /, INV, √, SQ, ^.
- Trigonometrische Funktionen SIN, ASIN, COS, ACOS, TAN, ATAN.
- Hyperbolische Funktionen SINH, ASINH, COSH, ACOSH, TANH, ATANH.
- Logarithmische Funktionen EXP, LN, LOG, ALOG.

R→C C→R RE IM CONJ SIGN

Weiterhin erscheinen die Befehle R→C, C→R, RE, IM und CONJ in der vierten Zeile des ARRAY-Menüs. Für die Anwendung dieser Befehle im Zusammenhang mit Feldern als Argumenten beziehen Sie sich auf Seite 126 beziehen.

R→C ***Reell in Komplex*** **Befehl**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	(*x*,*y*)
[*R-Feld*$_1$]	[*R-Feld*$_2$]	➧	[*K-Feld*]

R→C verbindet zwei reelle Zahlen x und y zu einer komplexe Zahl. x stellt den Realteil dar, während y den Imaginärteil des Ergebnisses bildet. x und y können ebenso als horizontale und vertikale Koordinaten in Bezug auf einen Punkt (x, y) in einem zweidimensionalen Raum angesehen werden.

...COMPLEX

C→R *Komplex in Reell* Befehl

Ebene 1		Ebene 2	Ebene 1
⟨ *x* , *y* ⟩	➧	*x*	*y*
⟦ *K-Feld* ⟧	➧	⟦ $R\text{-}Feld_1$ ⟧	⟦ $R\text{-}Feld_2$ ⟧

C→R teilt eine komplexe Zahl (oder ein Koordinatenpaar) in seine zwei Komponenten auf, wobei der Realteil (oder die horizontale Koordinate) in Ebene 2 und der Imaginärteil (oder die vertikale Koordinate) in Ebene 1 zurückgegeben wird.

RE *Realteil* Funktion

Ebene 1		Ebene 1
⟨ *x* , *y* ⟩	➧	*x*
' *Symbol* '	➧	' RE ⟨ *Symbol* ⟩ '
⟦ $Feld_1$ ⟧	➧	⟦ $Feld_2$ ⟧

RE gibt den Realteil x einer komplexen Zahl (x, y) zurück. x kann auch als horizontale Koordinate bzw. Abszissenabschnitt des Punktes (x, y) interpretiert werden.

IM *Imaginärteil* Funktion

Ebene 1		Ebene 1
⟨ *x* , *y* ⟩	➧	*y*
' *Symbol* '	➧	' IM ⟨ *Symbol* ⟩ '
⟦ $Feld_1$ ⟧	➧	⟦ $Feld_2$ ⟧

IM gibt den Imaginärteil y einer komplexen Zahl (x, y) zurück. y kann auch als vertikale Koordinate bzw. Ordinatenabschnitt des Punktes (x, y) interpretiert werden.

CONJ — *Konjugieren* — Analyt. Fkt.

Ebene 1		Ebene 1
x	➧	*x*
(*x*, *y*)	➧	(*x*, −*y*)
[*R-Feld*]	➧	[*R-Feld*]
[*K-Feld*$_1$]	➧	[*K-Feld*$_2$]
'*Symbol*'	➧	'CONJ(*Symbol*)'

CONJ gibt die konjugiert komplexe Zahl einer komplexen Zahl zurück. Der Imaginärteil der komplexen Zahl wurde dabei negiert.

SIGN — *Ausrichtung* — Funktion

Ebene 1		Ebene 1
z_1	➧	z_2
'*Symbol*'	➧	'SIGN(*Symbol*)'

Wird eine komplexe Zahl (x_1, y_1) als Argument für SIGN verwendet, so erhält man als Ergebnis den Einheitsvektor in der Richtung von (x_1, y_1):

$$(x_2, y_2) = \left(x_1/\sqrt{x_1^2 + y_1^2},\ y_1/\sqrt{x_1^2 + y_1^2}\right)$$

...COMPLEX

R→P **P→R** **ABS** **NEG** **ARG**

R→P — *Rechtecks- in Polarnotation* — Funktion

Ebene 1		Ebene 1
x	➧	⟨*x*,0⟩
⟨*x*,*y*⟩	➧	⟨*r*,*θ*⟩
'*Symbol*'	➧	'R→P⟨*Symbol*⟩'

R→P konvertiert eine komplexe Zahl in Rechtecksnotation (x, y) in ihre entsprechende Polarnotation (r, θ), wobei

$$r = \text{abs}\ (x, y), \quad \theta = \arg\ (x, y).$$

P→R — *Polar- in Rechtecksnotation* — Funktion

Ebene 1		Ebene 1
⟨*r*,*θ*⟩	➧	⟨*x*,*y*⟩
'*Symbol*'	➧	'P→R⟨*Symbol*⟩'

P→R konvertiert eine komplexe Zahl in Polarnotation (r, θ) in Rechtecksnotation (x, y), wobei

$$x = r \cos \theta, \quad y = r \sin \theta.$$

ABS — *Betrag* — Funktion

Ebene 1		Ebene 1
z	➧	|z|
⌊ *Feld* ⌋	➧	‖Feld‖
' *Symbol* '	➧	' ABS (*Symbol*) '

ABS (*ABSolute value*) gibt den Betrag seines Arguments zurück. Für eine komplexe Zahl (x, y) ist der Betrag als $\sqrt{(x^2 + y^2)}$ definiert.

NEG — *Negieren* — Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	$-z$
' *Symbol* '	➧	' − (*Symbol*) '
⌊ *Feld* ⌋	➧	⌊ −*Feld* ⌋

NEG gibt das negierte Ausgangsargument als Ergebnis zurück. Wenn die Befehlszeile nicht vorhanden ist, bewirkt das Drücken von [CHS] die Ausführung von NEG. Ein vollständiges Stackdiagramm für NEG finden Sie unter "Arithmetik".

ARG — *Argument* — Funktion

Ebene 1		Ebene 1
z	➧	θ
' *Symbol* '	➧	' ARG (*Symbol*) '

ARG gibt den Winkel θ der komplexen Zahl (x, y) zurück, wobei

$$\theta = \begin{cases} \arctan y/x & \text{für } x \geqslant 0, \\ \arctan y/x + \pi \text{ Vorzeichen } y & \text{für } x < 0, \text{ Bogenmaß-Modus} \\ \arctan y/x + 180 \text{ Vorzeichen } y & \text{für } x < 0, \text{ Grad-Modus} \end{cases}$$

...COMPLEX

Hauptzweige und allgemeine Lösungen

Im allgemeinen stellt die Umkehrfunktion einer Funktion eine *Relation* dar—für ein beliebiges Argument besitzt diese Relation mehr als einen Wert. Zum Beispiel hat $\cos^{-1}(z)$ für jedes Argument z unendlich viele Funktionswerte w, so daß $\cos w = z$. Für Relationen wie $\cos^{-1}$ definiert der HP-28S Funktionen wie z.B. ACOS. Diese Funktionen bestimmen den eindeutigen *Hauptwert*, der in der Teilmenge des Wertebereichs liegt, die als *Hauptzweig* der Umkehrfunktion bezeichnet wird.

Die Hauptzweige, mit denen der HP-28S arbeitet, sind in den Bereichen, in denen die Argumente der reellwertigen Umkehrfunktionen definiert sind, analytisch. Das bedeutet, der Verzweigungsschnitt liegt dort, wo die entsprechende reellwertige Umkehrfunktion nicht definiert ist. Außderdem werden die meisten wichtigen Symmetrien erhalten, wie z.B. $\text{ASIN}(-z) = -\text{ASIN}(z)$.

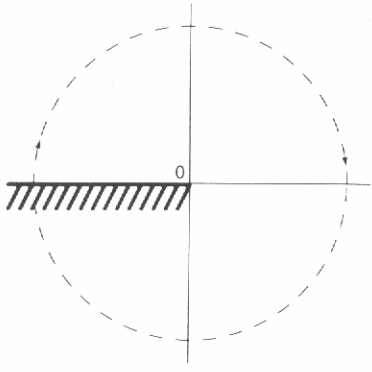
Die nachfolgenden Abbildungen zeigen die Hauptzweige für $\sqrt{}$, LN, ASIN, ACOS, ATAN und ACOSH. Anhand der Graphen für die jeweiligen Wertebereiche wird ersichtlich, wo die jeweiligen Schnitte auftreten: Die durchgezogenen roten oder schwarzen Linien sind auf der einen Seite des Schnittes, während die schraffierten roten bzw. schwarzen Bereiche die andere Seite bilden. Die Abbildungen für die Hauptzweige zeigen, wo jede Seite des Schnittes unter der Funktion angeordnet ist. Die zusätzlichen gestrichelten Linien in den Abbildungen sollen zur besseren Verständlichkeit der Funktion dienen.

Weiterhin sind die allgemeinen Lösungen enthalten, welche von ISOL zurückgegeben werden (unter der Annahme, daß Flag 34, Hauptwert, gelöscht ist und als Winkelmodus Bogenmaß gewählt wurde). Jede allgemeine Lösung besteht aus einem Ausdruck, welcher die multiplen Werte der Umkehrfunktion darstellt.

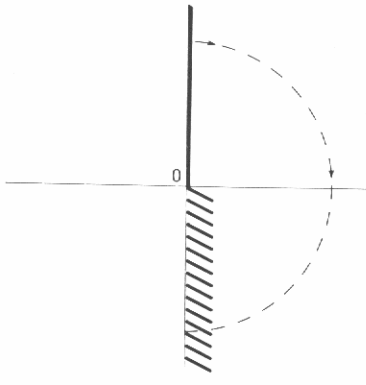
Die Funktionen LOG, ^, ASINH und ATANH beziehen sich sehr nahe auf die abgebildeten Funktionen. Sie können die Hauptwerte für LOG, ^, ASINH und ATANH anhand der Abbildungen bestimmen. Die allgemeinen Lösungen dieser Funktionen sind ebenfalls gegeben.

Hauptzweig für $\sqrt{Z}$

Wertebereich: `Z = (x,y)`



Hauptwert: `W = (u,v) = ⍎(x,y)`

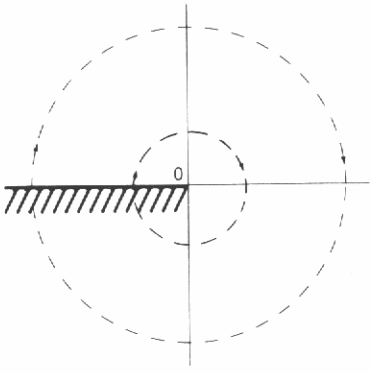


Allgemeine Lösung: `'SQ(W)=Z' 'W' ISOL` gibt `'≤1*⍎Z'` zurück.

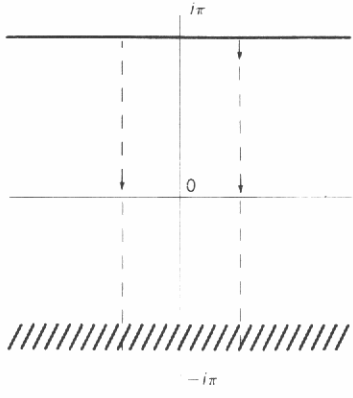
...COMPLEX

Hauptzweig für LN(Z)

Wertebereich: Z = (x,y)



Hauptwert: W = (u,v) = LN(x,y)



Allgemeine Lösung: 'EXP(W)=Z' 'W' ISOL gibt 'LN(Z)+2*π*i*n1' zurück.

Hauptzweig für LOG(Z)

Sie können den Hauptzweig für LOG aus den Abbildungen für LN (auf der vorangehenden Seite) ableiten; es gilt die Beziehung log (*z*) = ln (*z*)/ln (10).

Allgemeine Lösung: `'ALOG(W)=Z' 'W' ISOL` ergibt
`'LOG(Z)+2*π*i*n1/2.30258509299'`

Hauptzweig für U^Z

Sie können den Hauptzweig für komplexe Potenzen aus den Abbildungen für LN (auf der vorangehenden Seite) ableiten; es gilt die Beziehung $u^z = \exp(\ln(u)\, z)$.

Hauptzweig für ASINH(Z)

Sie können den Hauptzweig für ASINH aus den Abbildungen für ASIN (auf der folgenden Seite) ableiten; es gilt die Beziehung asinh $z = -i$ asin iz.

Allgemeine Lösung: `'SINH(W)=Z' 'W' ISOL` ergibt
`'ASINH(Z)+2*π*i*n1'`

Hauptzweig für ATANH(Z)

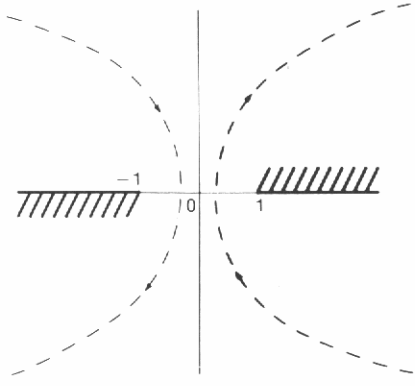
Sie können den Hauptzweig für ATANH aus den Abbildungen für ATAN (auf Seite 172) ableiten; es gilt die Beziehung
atanh $z = -i$ atan iz.

Allgemeine Lösung: `'TANH(W)=Z' 'W' ISOL` ergibt
`'ATANH(Z)+π*i*n1'`

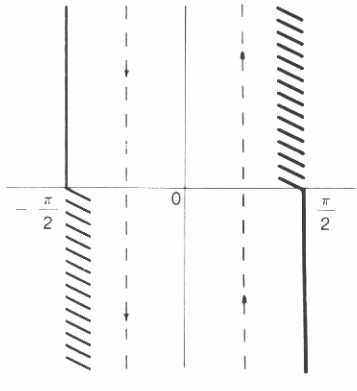
...COMPLEX

Hauptzweig für ASIN(Z)

Wertebereich: Z = (x,y)



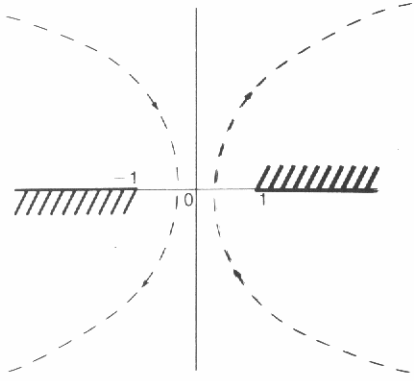
Hauptwert: W = (u,v) = ASIN(x,y)



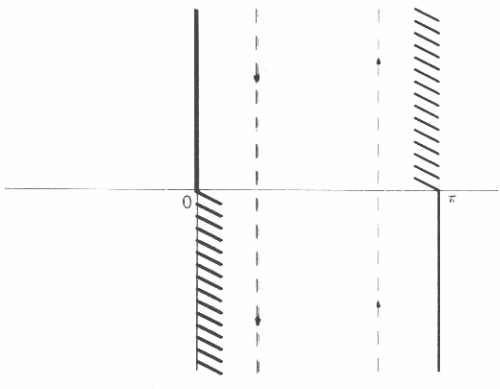
Allgemeine Lösung: 'SIN(W)=Z' 'W' ISOL ergibt 'ASIN(Z)*(-1)^n1+π*n1'.

Hauptzweig für ACOS(Z)

Wertebereich: Z = (x,y)



Hauptwert: W = (u,v) = ACOS(x,y)

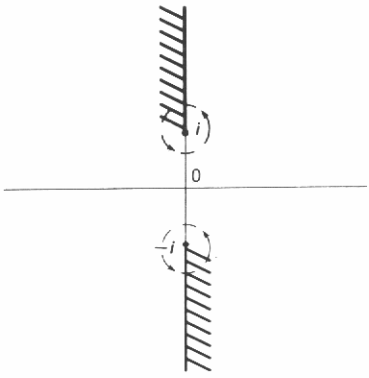


Allgemeine Lösung: 'COS(W)=Z' 'W' ISOL ergibt
's1*ACOS(Z)+2*π*n1'

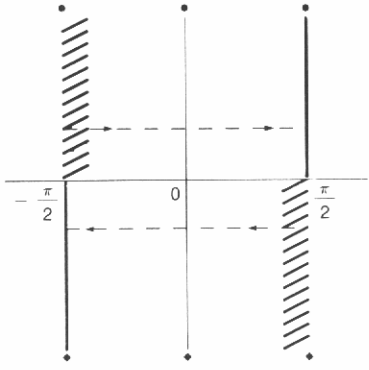
...COMPLEX

Hauptzweig für ATAN(Z)

Wertebereich: Z = (x,y)



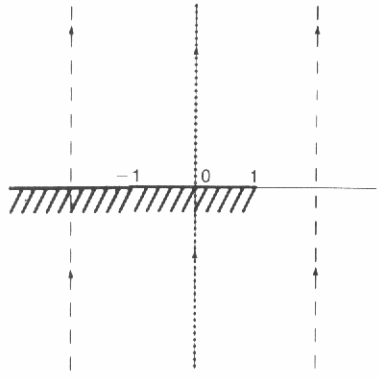
Hauptwert: W = (u,v) = ATAN(x,y)



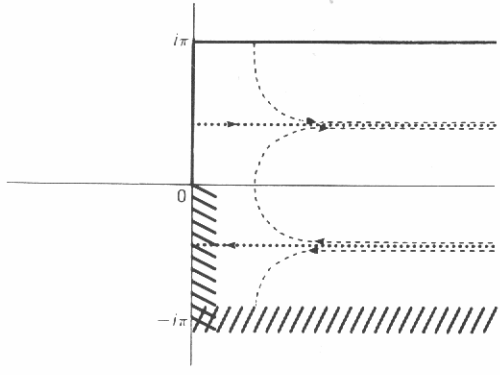
Allgemeine Lösung: 'TAN(W)=Z' 'W' ISOL ergibt 'ATAN(Z)+π*n1'

Hauptzweig für ACOSH(Z)

Wertebereich: Z = (x,y)



Hauptwert: W = (u,v) = ACOSH(x,y)



Allgemeine Lösung: 'COSH(W)=Z' 'W' ISOL ergibt
's1*ACOSH(Z)+2*π*i*n1'

LIST

→LIST	LIST→	PUT	GET	PUTI	GETI
SUB	SIZE				

Eine *Liste* ist eine geordnete Sammlung von willkürlichen Objekten, die selbst ein Objekt darstellt und daher in den Stack eingegeben oder als Variable gespeichert werden kann. Die Objekte innerhalb der Liste werden als *Elemente* bezeichnet und sind von links nach rechts (von 1 bis *n*) durchnumeriert. Die Befehle im LIST-Menü erlauben Ihnen, Listen zu erzeugen und zu ändern, wobei der Zugriff auf die einzelnen Objekte in der Liste möglich ist.

In Ergänzung zu den Befehlen im LIST-Menü können Sie die Tastenfeld-Funktion + zum Verknüpfen zweier Listen benutzen.

+ ***Addieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
{ $Liste_1$ }	{ $Liste_2$ }	➧	{ $Liste_1$ $Liste_2$ }

+ verkettet zwei Listen. Dies bedeutet, daß zwei Listen vom Stack genommen werden und eine einzelne Liste, welche alle Objekte der zwei ursprünglichen enthält, in den Stack gestellt wird.

Ein vollständiges Stackdiagramm für + finden Sie unter "Arithmetik".

→LIST LIST→ PUT GET PUTI GETI

→LIST ***Stack in Liste*** **Befehl**

Ebene *n*+1 ... Ebene 2	Ebene 1		Ebene 1
$Objekt_1$ ... $Objekt_n$	*n*	➧	{ $Objekt_1$ ... $Objekt_n$ }

→LIST nimmt eine ganze Zahl *n* aus Ebene 1 sowie *n* zusätzliche Objekte von Ebene 2 bis *n* + 1, und gibt eine Liste zurück, welche die *n* Objekte enthält.

Auf →LIST kann auch über das STACK-Menü zugegriffen werden.

LIST→ *Liste in Stack* Befehl

Ebene 1		Ebene n+1 ... Ebene 2	Ebene 1
{ $Objekt_1$ ... $Objekt_n$ }	➧	$Objekt_1$ ... $Objekt_n$	*n*

LIST→ nimmt eine Liste mit *n* Objekten vom Stack und gibt die Objekte in die einzelnen Ebenen 2 bis *n* + 1 zurück. Die Zahl *n* erscheint in Ebene 1.

Auf LIST→ kann auch über das STACK-Menü zugegriffen werden.

PUT *Element übernehmen* Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
[$Feld_1$]	{ *Index* }	*x*	➧	[$Feld_2$]
' *Name* '	{ *Index* }	*x*	➧	
[$K\text{-}Feld_1$]	{ *Index* }	*z*	➧	[$K\text{-}Feld_2$]
' *Name* '	{ *Index* }	*z*	➧	
{ $Liste_1$ }	*n*	*Objekt*	➧	{ $Liste_2$ }
' *Name* '	*n*	*Objekt*	➧	

...LIST

PUT ist ein allgemeines Verfahren, um ein Element in einem Feld oder ein Objekt in einer Liste zu speichern. Dieser Abschnitt beschreibt die Verwendung mit einer Liste; der Gebrauch mit einem Feld wird unter "ARRAY" erläutert.

PUT nimmt 3 Argumente vom Stack. Ebene 1 muß das Objekt enthalten, welches Sie in die Liste übernehmen möchten. Ebene 2 enthält den Index des Listenelements, welches ersetzt werden soll. Ebene 3 kann entweder eine Liste oder einen Namen enthalten:

- Enthält Ebene 3 eine Liste, so gibt PUT diese Liste in den Stack zurück, wobei das n-te Element durch das Objekt von Ebene 1 ersetzt wurde.
- Enthält Ebene 3 einen Namen, so ersetzt PUT das n-te Element in der Liste, die über die Variable spezifiziert ist, durch das Objekt in Ebene 1. Die Variable kann keine lokale Variable sein.

GET ist die umgekehrte Operation zu PUT.

GET ***Element holen*** **Befehl**

Ebene 2	Ebene 1		Ebene 1
[*Feld*]	{ *Index* }	➧	*z*
' *Name* '	{ *Index* }	➧	*z*
{ *Liste* }	*n*	➧	*Objekt*
' *Name* '	*n*	➧	*Objekt*

GET ist ein allgemeines Verfahren, um ein Element aus einem Feld oder einer Liste zu holen. In diesem Abschnitt wird die Verwendung mit einer Liste beschrieben. Der Gebrauch im Zusammenhang mit einem Feld ist unter "ARRAY" erläutert.

GET nimmt zwei Argumente vom Stack. Ebene 1 sollte einen Index enthalten, der das abzurufende Listenelement spezifiziert. Ebene 2 kann entweder eine Liste oder einen Namen enthalten:

- Bei einer Liste holt GET das *n*-te Element aus der Liste in den Stack.
- Enthält Ebene 2 einen Namen, so holt GET das *n*-te Element der unter *Name* gespeicherten Liste in den Stack zurück.

PUT ist die umgekehrte Operation zu GET.

PUTI ***Übernehmen und Index erhöhen*** **Befehl**

Ebene 3	Ebene 2	Ebene 1		Ebene 2	Ebene 1
[$Feld_1$]	{ $Index_1$ }	*x*	➧	[$Feld_2$]	{ $Index_2$ }
' *Name* '	{ $Index_1$ }	*x*	➧	' *Name* '	{ $Index_2$ }
[$K\text{-}Feld_1$]	{ $Index_1$ }	*z*	➧	[$K\text{-}Feld_2$]	{ $Index_2$ }
' *Name* '	{ $Index_1$ }	*z*	➧	' *Name* '	{ $Index_2$ }
{ $Liste_1$ }	n_1	*Objekt*	➧	{ $Liste_2$ }	n_2
' *Name* '	n_1	*Objekt*	➧	' *Name* '	n_2

PUTI (*PUT and Increment*) ist ein allgemeines Verfahren, um ein Element in ein Feld oder ein Objekt in eine Liste zu schreiben. In diesem Abschnitt wird die Verwendung mit einer Liste beschrieben. Der Gebrauch im Zusammenhang mit einem Feld ist unter "ARRAY" erläutert.

...LIST

PUTI übernimmt ein Objekt in eine Liste in der gleichen Art und Weise wie PUT, aber gleichzeitig wird die Liste in den Stack übernommen und der Index der Listenelemente ist um 1 erhöht. Damit bleibt der Stack immer für die nächste Eingabe eines neuen Objekts bereit; führen Sie nach dem Eintippen PUTI erneut aus, um das Objekt als nächstes Element in der Liste zu speichern. Wenn der Index das Maximum für die vorliegende Liste erreicht hat, gibt PUTI den Index 1 zurück und startet den Eingabeprozeß erneut vom Anfang der Liste.

PUTI nimmt drei Argumente vom Stack. Ebene 1 muß das Objekt enthalten, welches Sie in die Liste schreiben möchten. Ebene 2 enthält den Index des Listenelements, welches ersetzt werden soll. Ebene 3 kann entweder eine Liste oder einen Namen enthalten:

- Wenn eine Liste vorliegt, gibt PUTI diese Liste in Ebene 2 zurück, wobei das n-te Element durch das Objekt von Ebene 1 ersetzt wurde; in Ebene 1 wird der nächste Index ($n_1 + 1$, oder 1) zurückgegeben.
- Enthält Ebene 3 einen Namen, so schreibt PUTI das Objekt als n-tes Element in die unter dem Variablennamen gespeicherte Liste. Die Variable kann keine lokale Variable sein. Der Name wird dabei in Ebene 2 zurückgegeben und der nächste Index ($n_1 + 1$, oder 1) wird in Ebene 1 gestellt.

GETI ist die umgekehrte Operation zu PUTI.

GETI ***Holen und Index erhöhen*** **Befehl**

Ebene 2	Ebene 1		Ebene 3	Ebene 2	Ebene 1
[*Feld*]	{ $Index_1$ }	➧	[*Feld*]	{ $Index_2$ }	z
' *Name* '	{ $Index_1$ }	➧	' *Name* '	{ $Index_2$ }	z
{ *Liste* }	n_1	➧	{ *Liste* }	n_2	*Objekt*
' *Name* '	n_1	➧	' *Name* '	n_2	*Objekt*

GETI (*GET and Increment*) ist ein allgemeines Verfahren, um ein Element aus einem Feld oder einer Liste zu holen. Der Gebrauch von GETI im Zusammenhang mit Feldern ist unter "ARRAY" beschrieben.

GETI ruft ein Element aus einer Liste in der gleichen Art und Weise zurück, wie es durch GET erfolgt. Als zusätzliche Fähigkeit jedoch läßt GETI die Liste im Stack und erhöht den Index für die Listenelemente um 1. Damit wird das fortlaufende Abrufen der Elemente aus dem selben Feld ermöglicht. Wenn der Index das Maximum für die vorliegende Liste erreicht hat, gibt GETI den Index 1 zurück und beginnt erneut mit dem Abrufen des ersten Elements.

GETI nimmt zwei Argumente vom Stack. Ebene 1 sollte dabei einen Index enthalten, welcher das abzurufende Listenelement spezifiziert. Ebene 2 kann eine Liste oder einen Namen enthalten:

- Für eine Liste holt GETI das n-te Element aus der Liste in Ebene 1. GETI holt außerdem die Liste in Ebene 3 und den Index des nächsten Elements ($n_1 + 1$, oder 1) in Ebene 2.
- Wenn Ebene 2 einen Namen enthält, holt GETI das n-te Element der unter der Variablen *Name* gespeicherten Liste in Ebene 1. Weiterhin wird *Name* in Ebene 3 und der Index des nächsten Elements ($n_1 + 1$ oder 1) in Ebene 2 geholt.

PUTI ist die umgekehrte Operation zu GETI.

...LIST

SUB SIZE

SUB — Teilmenge — Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
"*String*$_1$"	n_1	n_2	➡	"*String*$_2$"
{ *Liste*$_1$ }	n_1	n_2	➡	{ *Liste*$_2$ }

SUB (*SUBset*) nimmt eine Liste und zwei Indizes n_1 und n_2 vom Stack und gibt eine neue Liste der Objekte zurück, welche den Elementen n_1 bis n_2 der ursprünglichen Liste entsprechen. Ist $n_2 < n_1$, so gibt SUB eine leere Liste zurück.

SUB arbeitet in einer ähnlichen Art und Weise für Strings. Beziehen Sie sich dazu auf "STRING".

SIZE — Größe — Befehl

Ebene 1		Ebene 1
"*String*"	➡	*n*
{ *Liste* }	➡	*n*
[*Feld*]	➡	{ *Liste* }
' *Symbol* '	➡	*n*

SIZE gibt ein Objekt zurück, welches die Größe oder Dimension einer Liste, eines Felds, eines Strings oder eines algebraischen Arguments enthält. Für eine Liste gibt SIZE die Anzahl der in der Liste enthaltenen Elemente zurück.

Beziehen Sie sich auf "ALGEBRA", "ARRAY" und "STRING" für die Anwendungsfälle für algebraische Argumente, Felder und Strings.

LOGS

LOG	ALOG	LN	EXP	LNP1	EXPM
SINH	ASINH	COSH	ACOSH	TANH	ATANH

Das LOGS-Menü enthält exponentielle, logarithmische und hyperbolische Funktionen. Jede dieser Funktionen akzeptiert reelle und algebraische Argumente; bis auf LNP1 und EXPM akzeptieren alle komplexe Argumente.

LOG ALOG LN EXP LNP1 EXPM

LOG *Dekadischer Logarithmus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	log z
'*Symbol*'	➧	'LOG(*Symbol*)'

LOG gibt den dekadischen Logarithmus (Basis 10) seines Arguments zurück.

Wenn als Argument 0 oder (0, 0) spezifiziert wurde, ergibt sich die `Infinite Result` Bedingung.

ALOG *Dekadischer Antilogarithmus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	10^z
'*Symbol*'	➧	'ALOG(*Symbol*)'

ALOG gibt den dekadischen Antilogarithmus (Basis 10) seines Arguments zurück—d.h. 10 potenziert mit dem Argument.

...LOGS

Für komplexe Argumente gilt:

$$\text{alog}\,(x, y) = \exp cx \cos cy + i \exp cx \sin cy,$$

wobei $c = \ln 10$. (Für die Berechnung wird das Bogenmaß verwendet).

LN — *Natürlicher Logarithmus* — Analyt. Fkt.

	Ebene 1		Ebene 1
	z	➧	ln z
	'Symbol'	➧	'LN⟨Symbol⟩'

LN gibt den natürlichen Logarithmus (Basis e) seines Arguments zurück.

Wenn als Argument 0 oder (0, 0) spezifiziert wurde, ergibt sich die `Infinite Result` Bedingung.

EXP — *Natürlicher Antilogarithmus* — Analyt. Fkt.

	Ebene 1		Ebene 1
	z	➧	exp z
	'Symbol'	➧	'EXP⟨Symbol⟩'

EXP gibt die Exponentialfunktion bzw. den natürlichen Antilogarithmus (Basis e) seines Arguments zurück—d.h. e potenziert mit dem Argument. EXP gibt ein genaueres Ergebnis als `e^` zurück, da EXP einen besonderen Algorithmus zur Berechnung des Funktionswerts verwendet.

Für komplexe Argumente gilt:

$$\exp\,(x, y) = \exp x \cos y + i \exp x \sin y.$$

(Für die Berechnung wird das Bogenmaß verwendet).

LNP1 — *LN von 1+x* — Analyt. Fkt.

Ebene 1		Ebene 1
x	➧	ln (1+*x*)
'*Symbol*'	➧	'LNP1(*Symbol*)'

LNP1 gibt $\ln(1 + x)$ zurück, wobei x dem reellwertigen Argument entspricht. LNP1 ist hauptsächlich bei der Bestimmung des natürlichen Logarithmus von Zahlenwerten nahe 1 hilfreich. LNP1 liefert ein genaueres Ergebnis von $\ln(1 + x)$, für x nahe Null, als unter Verwendung von LN erreicht werden kann.

Argumente kleiner als 1 verursachen einen `Undefined Result` Fehler.

EXPM — *Exponentiell minus 1* — Analyt. Fkt.

Ebene 1		Ebene 1
x	➧	exp (*x*)−1
'*Symbol*'	➧	'EXPM(*Symbol*)'

EXPM gibt $e^x - 1$ zurück, wobei x dem reellwertigen Argument entspricht. EXPM ist hauptsächlich bei der Bestimmung von e^X für Werte nahe Null hilfreich. EXPM liefert ein genaueres Ergebnis von $e^x - 1$, für x nahe Null, als unter Verwendung von EXP erreicht werden kann.

...LOGS

SINH ASINH COSH ACOSH TANH ATANH

Dies sind die hyperbolischen Funktionen und ihre Umkehrfunktionen.

SINH *Sinus hyperbolicus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	sinh *z*
'*Symbol*'	➧	'SINH(*Symbol*)'

SINH gibt den Sinus hyperbolicus seines Arguments zurück.

ASINH *Inverser Sinus hyperbolicus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	asinh *z*
'*Symbol*'	➧	'ASINH(*Symbol*)'

ASINH gibt den inversen Sinus hyperbolicus seines Arguments zurück. Bei reellen Argumenten $|x| > 1$ gibt ASINH das komplexe Ergebnis für das Argument $(x, 0)$ zurück.

COSH *Cosinus hyperbolicus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	cosh z
'*Symbol*'	➧	'COSH(*Symbol*)'

COSH gibt den Cosinus hyperbolicus seines Arguments zurück.

ACOSH *Inverser Cosinus hyperbolicus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	acosh z
'*Symbol*'	➧	'ACOSH(*Symbol*)'

ACOSH gibt den inversen Cosinus hyperbolicus seines Arguments zurück. Bei reellen Argumenten $|x| < 1$ gibt ACOSH das komplexe Ergebnis zurück, welches durch das Argument $(x, 0)$ erreicht wird.

TANH *Tangens hyperbolicus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	tanh z
'*Symbol*'	➧	'TANH(*Symbol*)'

TANH gibt den Tangens hyperbolicus seines Arguments zurück.

...LOGS

ATANH *Inverser Tangens hyperbolicus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	atanh z
'*Symbol*'	➧	'ATANH(*Symbol*)'

ATANH gibt den inversen Tangens hyperbolicus seines Arguments zurück. Bei reellen Argumenten $|x| > 1$ gibt ATANH das komplexe Ergebnis zurück, welches durch das Argument $(x, 0)$ erreicht wird.

Bei einem reellen Argument $x = \pm 1$ tritt die `Infinite Result` Bedingung auf. Wenn Flag 59 gelöscht ist, entspricht das Vorzeichen des Ergebnisses (MAXR) dem des Arguments.

MODE

STD	FIX	SCI	ENG	DEG	RAD
+CMD	−CMD	+LAST	−LAST	+UND	−UND
+ML	−ML	RDX.	RDX,	PRMD	

Im MODE-Menü sind alle Menütasten zusammengefaßt, über welche die verschiedenen *Rechnermodi* gesteuert werden: Anzeigeformat, Winkel, Rücksicherung, Dezimalzeichen und Mehrzeilen-Anzeige.

Die Menüfelder in diesem Menü wirken außerdem als Indikatoren zur Kennzeichnung des momentan spezifizierten Modus. Jeder Modus hat zwei oder mehrere Tasten, über welche die entsprechende Auswahl stattfindet. Eines der Menüfelder erscheint immer in normaler Darstellung (dunkle Zeichen auf hellem Hintergrund) und zeigt damit die momentane Einstellung an. So wird z.B. der Winkelmodus durch die Befehle DEG und RAD gesetzt. Wenn als Winkelmodus *Grad* spezifiziert wurde, so erscheint das DEG Feld in normaler Darstellung und das RAD Menüfeld invers. Nach dem Drücken von RAD ändert sich das RAD Feld in normale Darstellung (und das DEG Feld in invers), um anzuzeigen, daß *Bogenmaß* als Winkelmodus spezifiziert wurde.

Im unmittelbaren Eingabemodus werden alle MODE Befehle, außer FIX, SCI und ENG (welche Argumente erfordern), ohne Drücken von ENTER ausgeführt; die Befehlszeile bleibt dadurch unverändert.

STD FIX SCI ENG DEG RAD

Über diese Funktionen kann das Anzeigeformat und der Winkelmodus spezifiziert werden.

Die Funktionen für das Anzeigeformat (STD, FIX, SCI und ENG) steuern die Anzeige von Gleitkommazahlen, wie sie in jeder Form von Objekten in der Anzeige des Stacks erscheinen. In algebraischen Ausdrücken werden Gleitkommazahlen im momentanen Format und ganzzahlige Werte immer im STD Format angezeigt.

...MODE

Der momentane Anzeigemodus ist über Flag 49 und 50 verschlüsselt. Die Ausführung einer Funktion für das Anzeigeformat ändert den Status dieser Flags. Umgekehrt beeinflußt das Setzen und Löschen der Flags den Anzeigemodus. Im einzelnen handelt es sich dabei um folgende Zusammenhänge:

Modus	Flag 49	Flag 50
Standard	0	0
Fest	1	0
Wissenschaftlich	0	1
Technisch	1	1

Die Flags 53–56 verschlüsseln (binär) die Anzahl der Dezimalstellen von 0 bis 11. Flag 56 ist das hochwertigste Bit.

STD ***Standard*** **Befehl**

STD setzt das Anzeigeformat auf *Standardformat*. Standardformat (ANSI Minimal BASIC Standard X3J2) bewirkt beim Anzeigen oder Drucken die nachstehenden Ergebnisse:

- Zahlen, welche exakt als ganze Zahlen mit 12 oder weniger Stellen dargestellt werden können, werden ohne Dezimaltrennzeichen oder Exponent angezeigt. "Null" erscheint hierbei als `0`.
- Zahlen, welche exakt mit 12 oder weniger Stellen—jedoch nicht als ganze Zahlen—dargestellt werden können, werden mit Dezimaltrennzeichen, aber ohne Exponent, angezeigt. Führende Nullen des ganzzahligen Teils und nachlaufende Nullen des Dezimalteils werden vernachlässigt.

- Alle anderen Zahlen werden in diesem Format angezeigt:

  (Vorzeichen) Mantisse E (Vorzeichen) Exponent

  wobei der Betrag der Mantisse im Wertebereich $1 \leqslant x < 10$ liegt und der Exponent 1 bis 3 Stellen annimmt. Nachlaufende Nullen der Mantisse und führende Nullen des Exponenten werden vernachlässigt.

Die nachstehende Tabelle liefert Beispiele für die Anzeige von Zahlen im Standardformat:

Zahl	Anzeige	Mit 12 Stellen darstellbar?
10^{11}	100000000000	Ja (ganzzahlig)
10^{12}	1.E12	Nein
10^{-12}	.000000000001	Ja
1.2×10^{-11}	.000000000012	Ja
1.23×10^{-11}	1.23E-11	Nein
12.345	12.345	Ja

FIX

Fest **Befehl**

Ebene 1	
n ■	

FIX spezifiziert als Anzeigeformat das *Festkommaformat*, wobei die Anzahl der angezeigten Dezimalstellen (0 bis 11) durch ein entsprechendes Argument festgelegt wird. Dazu wird der gerundete Wert des Arguments verwendet. Ist der Wert größer als 11, so wird 11 verwendet; ist der Wert kleiner als 0, so wird 0 verwendet.

...MODE

Im Festkommaformat erscheinen angezeigte oder gedruckte Zahlen in der Form

(Vorzeichen) Mantisse

Die Mantisse wird auf *n* Stellen rechts vom Dezimalzeichen aufgerundet, wobei *n* die Anzahl der Dezimalstellen spezifiziert. Wenn das Festkommaformat vorgegeben wurde, schaltet der HP-28S in einem der beiden Fällen automatisch auf das wissenschaftliche Anzeigeformat um:

- Die anzuzeigende Zahl hat mehr als 12 Stellen.
- Ein Wert ungleich Null, auf *n* Dezimalstellen gerundet, würde im Festkommaformat als Null angezeigt werden.

SCI ***Wissenschaftlich*** **Befehl**

Ebene 1	
n	➧

SCI (*SCIentific*) spezifiziert für die Zahlenanzeige das *wissenschaftliche Anzeigeformat*. Das Argument gibt dabei die Anzahl der signifikanten Stellen (im Bereich 0 bis 11) an. Es wird der gerundete Wert des Arguments verwendet. Ist der Wert größer als 11, so wird 11 verwendet; ist der Wert kleiner als 0, so wird 0 verwendet.

Im wissenschaftlichen Anzeigeformat werden Zahlen in der wissenschaftlichen Notation bis zu $n + 1$ signifikanten Stellen angezeigt; n (das Argument für SCI) spezifiziert dabei die Zahl der angezeigten Stellen. Ein Wert erscheint im Format

```
(Vorzeichen) Mantisse E (Vorzeichen) Exponent
```

wobei $1 \leq Mantisse < 10$.

ENG — *Technisch* — Befehl

Ebene 1	
n ●	

ENG (*ENGineering*) spezifiziert für die Zahlenanzeige das *technische Anzeigeformat*. Das Argument gibt dabei die Anzahl der angezeigten signifikanten Stellen (im Bereich 0 bis 11) an. Es wird der gerundete Wert des Arguments verwendet. Ist der Wert größer als 11, so wird 11 verwendet; ist der Wert kleiner als 0, so wird 0 verwendet.

Im technischen Anzeigeformat erscheinen angezeigte oder gedruckte Zahlen im Format

```
(Vorzeichen) Mantisse E (Vorzeichen) Exponent
```

wobei $1 \leqslant$ *Mantisse* < 1000 und der Exponent ein mehrfaches von 3 ist. Die Anzahl der angezeigten signifikanten Stellen ist um eins größer als durch das Argument spezifiziert wurde. Wenn der Exponent eines angezeigten Werts größer als -499 ist, dann wird die Zahl im wissenschaftlichen Format angezeigt.

DEG — *Grad* — Befehl

●

DEG (*DEGgrees*) spezifiziert *Grad* als momentanen Winkelmodus. In diesem Modus gilt:

Reelle Argumente: Funktionen, welche als Argumente reellwertige Winkel verwenden, interpretieren diese Winkel in Grad (Altgrad). Von komplexen Argumenten für SIN, COS und TAN wird immer angenommen, daß sie im Bogenmaß ausgedrückt sind.

Reelle Ergebnisse: Funktionen, welche als Ergebnis reellwertige Winkel berechnen, drücken diese Ergebnisse in Grad aus: ASIN, ACOS, ATAN, ARG und R→P. Komplexe Ergebnisse von ASIN oder ACOS, für Argumente außerhalb des Wertebereichs $x \leqslant 1$, werden immer im Bogenmaß ausgedrückt.

Die Ausführung von DEG schaltet den (2π) Indikator aus und löscht den Benutzerflag 60.

RAD ***Bogenmaß*** **Befehl**

RAD (*RADians*) spezifiziert das *Bogenmaß* als momentanen Winkelmodus. In diesem Modus gilt:

Reelle Argumente: Funktionen, welche als Argumente reellwertige Winkel verwenden, interpretieren diese Winkel im Bogenmaß. Von komplexen Argumenten für SIN, COS und TAN wird immer angenommen, daß sie im Bogenmaß ausgedrückt sind.

Relle Ergebnisse: Funktionen, welche als Ergebnis reellwertige Winkel berechnen, drücken diese Ergebnisse im Bogenmaß aus: ASIN, ACOS, ATAN, ARG und R→P. Komplexe Ergebnisse von ASIN oder ACOS, für Argumente außerhalb des Wertebereichs $x \leqslant 1$, werden immer im Bogenmaß ausgedrückt.

Die Ausführung von RAD schaltet den (2π) Indikator ein und setzt den Benutzerflag 60.

+CMD −CMD +LAST −LAST +UND −UND

Diese sechs Menütasten steuern die 3 Verfahren zur Rücksicherung COMMAND, LAST und UNDO. "+" und "−" in den Menüfeldern bedeuten dabei das Aktivieren bzw. Desaktivieren des jeweiligen Verfahrens.

Von diesen Operationen kann keine innerhalb von Programmen verwendet werden; diese Menütasten bewirken immer eine unmittelbare Ausführung. LAST kann aktiviert oder desaktiviert werden, indem Flag 31 gesetzt oder gelöscht wird.

COMMAND aktiviert/desaktiviert

Operation	Bedeutung
+CMD	Aktiviert COMMAND. Vorangegangene Befehlszeilen können nun durch Drücken von ■ COMMAND wieder erhalten werden.
-CMD	Desaktiviert COMMAND und gibt den Speicherbereich, in welchem vorangegangene Befehlszeilen gespeichert wurden, wieder frei. Das Drücken von ■ COMMAND verursacht jetzt einen `COMMAND Stack Disabled` Fehler.

LAST aktiviert/desaktiviert

Operation	Bedeutung
+LAST	Aktiviert LAST. Befehle, welche bis zu 3 Argumente verwenden, sichern nun diese Argumente, um mit LAST zurückgerufen werden zu können.
-LAST	Deskativiert LAST und gibt den zum Sichern der Argumente verwendeten Speicherbereich wieder frei. Die Ausführung von LAST verursacht jetzt einen `LAST Disabled` Fehler.

...MODE

UNDO aktiviert/desaktiviert

Operation	Bedeutung
+UND	Aktiviert UNDO. Dadurch wird der Stackinhalt nach jeder Ausführung von ENTER gesichert und das Drücken von ■ UNDO bewirkt den Austausch des momentanen Stackinhalts mit dem zuvor gesicherten Stack.
-UND	Desaktiviert UNDO und gibt den vom gesicherten Stack belegten Speicherplatz wieder frei. Das Drücken von ■ UNDO verursacht nun einen `UNDO Disabled` Fehler.
	Die Auswirkungen von +UND und -UND sind von "lokaler Art" für das momentan unterbrochene Programm. Dies bedeutet, daß bei einem unterbrochenen Programm die Ausführung von +UND und -UND die mögliche Stacksicherung nur ändert, während die Programmausführung ausgesetzt ist.

+ML −ML RDX. RDX, PRMD

Mehrfachzeile aktiviert/desaktiviert

Objekte im Stack werden in einem der beiden allgemeinen Anzeigeformate angezeigt: im Mehrzeilen- oder Kompakt-Modus. Ab Ebene 2 und höher werden Objekte immer in kompaktem Format angezeigt, d.h., es wird nur eine Zeile zur Anzeige des Objekts verwendet (wenn zu viel Zeichen enthalten sind, erscheint eine Ellipse "..." am rechten Anzeigerand).

Sie haben die Wahl, über die Menütasten +ML und -ML die Objekte in Ebene 1 im Mehrzeilen- (+ML) oder Kompakt-Modus (-ML) anzuzeigen. Für den Mehrzeilen-Modus gilt:

- Matrizen erscheinen zeilenweise, d.h. in jeder Zeile werden so viele Elemente wie möglich angezeigt.
- Prozeduren und Listen werden mit eingefügten neuen Zeilen angezeigt, so daß deren gesamte Definitionen sichtbar sind. Zusätzliche Prozeduren, Listen und Matrizen, welche in diesen Definitionen enthalten sind, werden ebenso im Mehrzeilen-Modus angezeigt.
- Zahlen, komplexe Zahlen, Vektoren, Namen und Strings werden zwar nicht über mehrere Anzeigezeilen abgetrennt, können jedoch trotzdem abgeschnitten in der Anzeige erscheinen. "Versteckte" Zeichen werden wie im kompakten Anzeigemodus durch eine Ellipse "..." am rechten Anzeigerand gekennzeichnet.

Wenn für die vollständige Anzeige des Objekts mehr Anzeigezeilen erforderlich sind, als momentan zur Verfügung stehen (vier mit dem Cursor-Menü, weniger eine für ein beliebiges anderes Menü, weniger eine—oder mehrere—falls eine Befehlszeile aktiviert wurde), dann können Sie sich über die Tastenfolgen ■ VIEW↑ oder ■ VIEW↓ weitere Zeilen anzeigen lassen.

Die momentane Wahl für den Mehrzeilen- bzw. Kompakt-Modus wird außerdem durch den Flag 45 dargestellt. Der Kompakt-Modus entspricht dem gelöschten Flag 45.

Operation	Bedeutung
+ML	Wählt den Mehrzeilen-Modus für Objekte in Ebene 1 und setzt Flag 45. Durch die Aktivierung dieses Modus werden zur Anzeige von Objekten in Ebene 1 eine oder mehrere Zeilen verwendet. Die Druckausgabe von Objekten in Ebene 1, welche im Protokoll-Modus an den Drucker HP-82240A gesendet werden, erfolgt ebenfalls im Mehrzeilen-Modus.
-ML	Wählt den Kompakt-Modus für Objekte in Ebene 1 und löscht Flag 45. Die Druckausgabe von Objekten in Ebene 1, welche im Protokoll-Modus an den Drucker HP-82240A gesendet werden, erfolgt ebenfalls im Kompakt-Modus.

...MODE

Punkt/Komma als Dezimalzeichen

RDX. und RDX, ermöglichen Ihnen, das von Ihnen bevorzugte Zeichen zur Trennung des ganzzahligen Teils vom Dezimalteil einer Zahl zu wählen. Sie haben die Wahl zwischen Punkt "." oder Komma ","; das Zeichen, welches nicht als Dezimalzeichen verwendet wird, kann im Tausch mit dem Leerzeichen bei der Eingabe von Objekten zu deren Trennung verwendet werden (außer in algebraischen Objekten, bei welchen dieses Zeichen zur Trennung der Argumente von Funktionen mit mehreren Argumenten dient).

Operation	Bedeutung
RDX.	Spezifiziert einen Punkt (Dezimalpunkt) als Trennzeichen zwischen ganzzahligem Teil und Dezimalteil einer Zahl und löscht Benutzerflag 48. In diesem Modus kann das Komma innerhalb der Befehlszeile im Austausch mit dem Leerzeichen verwendet werden (außer in algebraischen Objekten).
RDX,	Spezifiziert ein Komma (Dezimalkomma) als Trennzeichen zwischen ganzzahligem Teil und Dezimalteil einer Zahl und setzt Benutzerflag 48. In diesem Modus kann der Punkt innerhalb der Befehlszeile im Tausch mit dem Leerzeichen verwendet werden (außer in algebraischen Objekten).

PRMD	*Modi drucken*	Befehl
	•	

PRMD (*PRint MoDes*) zeigt und druckt eine Auflistung der momentanen Modi des HP-28S. Die Auflistung beinhaltet den gewählten Modus für das Anzeigeformat von Zahlen, den allgemeinen Anzeigemodus (mehrzeilen oder kompakt), den Winkel, die Basis für Binärwerte und das Dezimalzeichen; außerdem wird gelistet, ob die Rücksicherungsmöglichkeiten UNDO, COMMAND und LAST aktiviert oder desaktiviert sind. Eine typische Auflistung könnte etwa wie folgt aussehen:

PLOT

STEQ	**RCEQ**	**PMIN**	**PMAX**	**INDEP**	**DRAW**
PPAR	**RES**	**AXES**	**CENTR**	**∗W**	**∗H**
STOΣ	**RCLΣ**	**COLΣ**	**SCLΣ**	**DRWΣ**	
CLLCD	**DISP**	**PIXEL**	**DRAX**	**CLMF**	**PRLCD**

Über die Befehle im PLOT-Menü (plotten/zeichnen) haben Sie die Möglichkeit, spezielle Anzeigen zu erzeugen, welche den Stack und die gewöhnliche Menüanzeige überlagern. Sie können den Verlauf einer mathematischen Funktion abbilden, ein statistisches Streudiagramm oder Daten während eines Programmablaufs anzeigen sowie Informationen anhand einer graphischen Abbildung digital darstellen.

Die Anzeige

Die LCD-Anzeige (*Liquid-Crystal Display*) des HP-28S besteht aus einem Feld von 32 Zeilen mit 137 *Pixels* (Punkte), organisiert als vier Zeilen mit je 23 Zeichen. Ein Zeichenfeld ist sechs Pixel breit und 8 Pixel hoch, mit Ausnahme der Felder am rechten Anzeigerand, welche nur fünf Pixel breit sind. Normalerweise sind die angezeigten Zeichen selbst 5 Pixel breit, womit eine Leerspalte von einem Pixel zwischen den Zeichen möglich ist.

In der Standardanzeige erscheinen Menüfelder, die Befehlszeile und der Stack. Sie können temporär diese Standardanzeige auf folgende Weise ersetzen:

- **Löschen der Anzeige:** Der Befehl CLLCD überschreibt die gesamte Anzeige mit Leerzeichen (außer den Indikatoren).
- **Anzeigen von Meldungen:** Unter Verwendung von DISP können Sie Objekte in einer oder mehreren der vier Zeilen anzeigen.
- **Anzeigen von Graphen:** Die Befehle PIXEL, DRAW und DRWΣ erlauben Ihnen, individuelle Datenpunkte, mathematische Funktionen und die entsprechenden Punkte einer Statistikmatrix abzubilden. DRAX bietet die Möglichkeit, Achsen innerhalb einer Abbildung zu zeichnen.

Wenn Sie einen dieser Befehle ausführen, wird automatisch ein Systemmeldungsflag gesetzt, um das normale Menü und den Stack vor dem Überschreiben durch Ihre spezielle Anzeige zu schützen. Das Drücken einer beliebigen Taste nach Abschluß aller aktiven Prozeduren (der "Busy"-Indikator ((•)) ist nicht mehr sichtbar) bewirkt das Löschen des Meldungsflags und die Rücksicherung der normalen Anzeige. Der Befehl CLMF kann innerhalb eines Programms zum Löschen des Meldungsflags benutzt werden.

Es gibt auch Tasten, welche eine besondere Anzeige erzeugen und das Tastenfeld temporär neu definieren:

- ■ CATALOG und ■ UNITS erzeugen Kataloganzeigen und bieten entsprechende Menü- und Buchstabentasten zur Steuerung der Kataloge an. ■ CATALOG und ■ UNITS sind unter "CATALOG" bzw. "Konvertierung von Einheiten" beschrieben.
- DRAW und DRWΣ führen die Befehle DRAW und DRWΣ aus, um den Graphen einer mathematischen Funktion oder ein statistisches Streudiagramm zu erzeugen. Wird DRAW oder DRWΣ durch Drücken der jeweiligen Menütaste ausgeführt, während die Abbildungen in der Anzeige sichtbar sind, so können zusätzlich die Menütasten zur Cursor-Steuerung verwendet werden; damit können Sie Koordinaten digitalisieren und in den Stack übertragen.

Anzeigepositionen

Um Strings anzeigen zu können, wird die Anzeige so behandelt, als wären vier Zeilen mit Zeichen möglich. Die oberste Zeile ist Zeile 1, die nächste Zeile ist Zeile 2, usw. Jede Zeile muß als einzelne Einheit angezeigt werden; Sie können einzelne Zeichen nicht an willkürlichen Positionen anzeigen. DISP bringt ein Objekt in die Anzeige, wobei das Objekt als ein String dargestellt wird, äquivalent zu der Stackanzeige im Mehrzeilen-Modus.

...PLOT

Zur Anzeige graphischer Daten wird die Anzeige als Gitter mit 32 × 137 *Pixel* (Punkte) behandelt. Ein Pixel ist durch seine *Koordinaten* spezifiziert, eine komplexe Zahl mit einem geordneten Koordinatenpaar (x, y). x stellt dabei die horizontale Koordinate und y die vertikale Koordinate dar. (In dieser Diskussion werden die Buchstaben x und y zur Andeutung der Horizontalen und Vertikalen benutzt, Sie können jedoch jeden beliebigen Variablennamen beim Erzeugen von Graphiken mit dem HP-28S verwenden.)

Der Maßstab von Koordinaten zu Pixel wird durch die Koordinaten für die Eckpunkte P_{max} und P_{min}, welche über die Befehle PMAX und PMIN festgelegt werden, gesetzt. P_{max} stellt den obersten rechten Pixel in der Anzeige dar; seine Koordinaten sind (x_{max}, y_{max}). P_{min} (x_{min}, y_{min}) ist der unterste linke Pixel. Die Voreinstellungen dieser Koordinaten sind $P_{max} = (6.8, 1.6)$ und $P_{min} = (-6.8, -1.5)$. Die Koordinaten für das Zentrum eines bestimmten Pixels lauten

$$x = n_x\, w_x + x_{min}$$

$$y = n_y\, w_y + y_{min}$$

wobei n_x der horizontalen Pixelnummer und n_y der vertikalen Pixelnummer entspricht (P_{min} hat $n_x = 0$ und $n_y = 0$; P_{max} hat $n_x = 136$, $n_y = 31$). w_x und w_y sind horizontale und vertikale Pixelbreiten:

$$w_x = (x_{max} - x_{min})/136.$$

$$w_y = (y_{max} - y_{min})/31.$$

Der Pixel mit $n_x = 68$ und $n_y = 15$ ist als *zentraler* Pixel definiert. Mit der Voreinstellung für P_{max} und P_{min} besitzt der zentrale Pixel die Koordinaten $(0, 0)$.

Abbildungen mathematischer Funktionen

Die Abbildung einer mathematischen Funktion basiert auf den Werten einer Prozedur, welche in der Variablen EQ (die gleiche, die vom Gleichungslöser benutzt wird) als Funktion einer spezifizierten *unabhängigen Variablen* gespeichert ist. Die Prozedur wird für jeden der $137/r$ Werte der unabhängigen Variablen im Bereich von x_{min} bis x_{max} vollständig ausgewertet; r (*resoluton*) stellt dabei die *Auflösung* der Abbildung dar. Für jedes Koordinatenpaar (*unabhängige Variable, Prozedurwert*) wird dem Graphen ein Pixel hinzugefügt—solange sich der Prozedurwert innerhalb des Bereichs y_{min} und y_{max} bewegt. Die Koordinatenachsen der Abbildung sind mit Markierungen im Abstand von jeweils 10 Pixel versehen.

Der eigentliche Graph wird durch den Befehl DRAW erzeugt. Wenn Sie DRAW direkt durch Drücken der Menütaste DRAW ausführen, haben Sie die Möglichkeit, über die Cursorposition Daten in digitaler Form aus der Abbildung zu übernehmen.

Eine Funktionsabbildung erzeugt eine oder zwei Kurven, in Abhängigkeit von der Definition der EQ Prodzedur:

- Wenn EQ einen algebraischen Ausdruck ohne Gleichheitszeichen enthält, dann zeichnet DRAW eine einzelne Kurve, entsprechend dem zugehörigen Wert des Ausdrucks für den Wert der unabhängigen Variablen innerhalb des Zeichenbereichs.
- Enthält EQ eine Gleichung, dann zeichnet DRAW zwei Kurven, eine für jede Seite der Gleichung. Beachten Sie, daß die Schnittpunkte der zwei Kurven an den Punkten der unabhängigen Variablen auftreten, welche den über den Gleichungslöser ermittelten Nullstellen entsprechen.
- Wenn in EQ ein Programm gespeichert ist, so wird dieses als algebraischer Ausdruck interpretiert und es wird nur eine Kurve gezeichnet. Dabei wird vorausgesetzt, daß das Programm der Syntax für einen algebraischen Ausdruck folgt: Es darf kein Argument vom Stack genommen werden und es muß ein Objekt in den Stack zurückgegeben werden.

...PLOT

Die allgemeine Vorgehensweise zum Abbilden einer Funktion ist nachstehend zusammengefaßt. Für Details sollten Sie sich auf die Beschreibung der einzelnen Befehle beziehen.

1. Speichern Sie die zu zeichnende Prozedur unter Verwendung von STEQ in EQ.
2. Wählen Sie die unabhängige Variable mit Hilfe von INDEP.
3. Wählen Sie den Zeichenbereich. Dies ist mit den Funktionen PMIN, PMAX, CENTR, *H und *W möglich.
4. Spezifizieren Sie mit AXES den Schnittpunkt der Achsen.
5. Wählen Sie unter Verwendung von RES die Auflösung der graphischen Darstellung.
6. Führen Sie DRAW aus.

Jeder der Schritte 1–5 kann weggelassen werden, was die Verwendung der momentanen Werte zur Folge hat.

Statistische Streudiagramme

Die Anzeige eines statistischen *Streudiagramms* entspricht der Abbildung von individuellen Punkten, welche momentan im Statistik-Feld ΣDAT gespeichert sind. Sie können als Koordinatenwerte für Abszisse und Ordinate jede beliebige Spalte des Feldes wählen. Es wird dann für jeden Datenpunkt der Matrix ein Punkt in der Anzeige abgebildet.

Die allgemeine Vorgehensweise zum Zeichnen eines Streudiagramms ist nachstehend zusammengefaßt. Für Details sollten Sie sich auf die Beschreibung der einzelnen Befehle beziehen.

1. Speichern Sie die zu zeichnenden Daten unter Verwendung von STOΣ in ΣDAT.
2. Wählen Sie die Koordinatenspalten für Abszisse und Ordinate mit Hilfe von COLΣ.
3. Legen Sie den Zeichenbereich unter Verwendung von PMIN, PMAX, CENTR, *H und *W fest. Mit SCLΣ läßt sich eine automatische Skalierung durchführen.

4. Spezifizieren Sie mit Hilfe von AXES den Achsenschnittpunkt.
5. Führen Sie DRWΣ aus.

Jeder der Schritte 1–4 kann weggelassen werden, was zur Verwendung der momentanen Werte führt.

Interaktive Abbildungen

Wenn Sie DRAW oder DRWΣ durch Drücken der korrespondierenden Menütaste ausführen, begibt sich der Rechner in einen interaktiven Zeichenmodus, welcher Ihnen ermöglicht, digitale Informationen aus der Graphik abzurufen. Nach dem Start einer interaktiven Abbildung ist der Ablauf wie folgt:

1. Die Anzeige wird gelöscht.
2. Entweder erfolgt die Ausführung von DRAW oder DRWΣ (um die entspechende Graphik abzubilden). Wenn Sie [ON] drücken, bevor der Zeichenvorgang abgeschlossen ist, werden keine Punkte mehr gezeichnet und es beginnt der interaktive Modus.
3. In der Mitte der Anzeige erscheint der Cursor in Form eines kleinen Kreuzes (+). (Wenn der Ursprung des Achsenkreuzes im Zentrum liegt, ist der Cursor erst sichtbar, nachdem er bewegt wird.)
4. Die Menütasten werden als Cursor- bzw. Digitalisierungstasten aktiviert:
   - Die zwei linken Tasten übertragen die Koordinaten des Cursors in den Stack, ohne dabei die Anzeige der Graphik zu unterbrechen. [INS] schreibt die Koordinaten in komplexer Form *(x, y)* in den Stack, [DEL] gibt sie als zwei reelle Zahlen (*x* in Ebene 2, *y* in Ebene 1) in den Stack.
   - Die vier rechten Tasten wirken wie die normalen Tasten zur Cursor-Steuerung, d.h. sie bewegen den Cursor nach oben, unten, rechts und links (wenn zuerst ■ gedrückt wird, springt der Cursor an den entsprechenden Anzeigerand).

Der interaktive Graphikmodus bleibt so lange aktiviert, bis [ON] gedrückt wird.

Es können beliebig viele Punkte während des interaktiven Graphikmodus digitalisiert werden, indem [INS], [DEL] und die Cursortasten wiederholt benutzt werden.

...PLOT

Abbildungsparameter

Die Skalierungsfaktoren, welche zur Konvertierung eines Koordinatenpaares in einen Zeichenpunkt (und umgekehrt) notwendig sind, werden in einer Liste unter der Variable PPAR (*Plot PARameter*) gespeichert. In der nachfolgenden Beschreibung sind diese Parameter als *Abbildungsparameter* bezeichnet; dabei handelt es sich im einzelnen um:

Parameter	Bedeutung
P_{min}	Eine Zahl in komplexer Form, welche die Koordinaten des untersten linken Punktes darstellt. Gesetzt durch PMIN, CENTR, *H, *W und SCLΣ.
P_{max}	Eine Zahl in komplexer Form, welche die Koordinaten des obersten rechten Punktes darstellt. Gesetzt durch PMAX, CENTR, *H, *W und SCLΣ.
Unabhängige Variable	Der Variablenname, welcher bei der Abbildung einer mathematischen Funktion der Abszisse entspricht. Gesetzt durch INDEP.
Auflösung	Eine reelle, positive ganze Zahl, welche den Abstand zwischen den einzelnen Punkten in einer Abbildung spezifiziert. Gesetzt durch RES.
P_{Achsen}	Eine Zahl in komplexer Form, welche den Ursprung des Koordinatensystems darstellt. Gesetzt durch AXES.

STEQ RCEQ PMIN PMAX INDEP DRAW

Dieser Befehlssatz erlaubt Ihnen, eine Prozedur für die Abbildung einer Funktion zu wählen, die primären Abbildungsparameter zu spezifizieren, und die Prozedur als Graphik anzuzeigen.

STEQ — *Gleichung speichern* — Befehl

Ebene 1	
Objekt ➡	

STEQ (*STore EQuation*) nimmt ein Objekt vom Stack und speichert dieses in der Variablen EQ ("EQuation"). Dieser Befehl ist gleichwertig mit `'EQ' STO`.

EQ wird dazu verwendet, eine Prozedur (die momentane Gleichung) zu speichern; die Prozedur wird vom Gleichungslöser und von DRAW als implizites Argument benutzt, womit das Argument für STEQ normalerweise eine Prozedur sein sollte.

RCEQ — *Gleichung zurückrufen* — Befehl

	Ebene 1
➡ *Objekt*	

RCEQ (*ReCall EQuation*) gibt den Inhalt der Variablen EQ in den Stack zurück. Dieser Befehl ist gleichwertig mit `'EQ' RCL`.

PMIN — *Minimum zeichnen* — Befehl

Ebene 1	
(x,y) ➡	

PMIN (*Plot MINima*) setzt die Koordinaten des untersten linken Punkts der Anzeige als Punkt (x, y). Die Zahl in der komplexen Form (x, y) wird als erster Listeneintrag unter der Variablen PPAR gespeichert.

...PLOT

PMAX ***Maximum zeichnen*** **Befehl**

Ebene 1	
(x,y) ➡	

PMAX (*Plot MAXima*) setzt die Koordinaten des obersten rechten Punkts der Anzeige als Punkt (x, y). Die Zahl in der komplexen Form (x, y) wird als zweiter Listeneintrag unter der Variablen PPAR gespeichert.

INDEP ***Unabhängige Variable*** **Befehl**

Ebene 1	
'Name' ➡	

INDEP (*INDEPendent*) nimmt einen Namen vom Stack und speichert diesen als unabhängige Variable. Die Speicherung erfolgt als dritter Listeneintrag unter der Variablen PPAR. Bei einer aufeinanderfolgenden Ausführung von DRAW wird der Name als die unabhängige Variable, korrespondierend zur Abszisse, verwendet.

DRAW ***Zeichnen*** **Befehl**

➡

DRAW erzeugt die eigentliche Abbildung der mathematischen Funktion in der Anzeige des Rechners. Wenn Sie DRAW durch Drücken von DRAW ausführen, wird eine interaktive Abbildung erzeugt, welche unter "Interaktive Abbildungen" auf Seite 203 beschrieben ist.

DRAW führt automatisch DRAX aus, was die Zeichnung der Koordinatenachsen zur Folge hat; danach werden eine oder zwei Kurven gezeichnet, welche die Werte der momentanen Gleichung an jedem der $137/r$ Werte der unabhängigen Variablen darstellen. Mit "momentaner Gleichung" ist die unter der Variablen EQ gespeicherte Prozedur gemeint.

Wenn EQ eine algebraische Gleichung enthält, wird für jede Seite der Gleichung eine separate Kurve gezeichnet. Entspricht die momentane Gleichung einem algebraischen Ausdruck oder einem Programm, so entsteht nur eine Kurve.

Die Auflösung r (*resolution*) bestimmt die Anzahl der gezeichneten Punkte. $r = 1$ bedeutet, daß für jede Spalte von Anzeigepunkten (pixel) ein Punkt gezeichnet wird; $r = 2$ entspricht jeder zweiten Spalte, usw. r wird durch den Befehl RES festgelegt. Als Voreinstellung dient der Wert 1; wenn die Zeit zum Erzeugen der Kurven verkürzt werden soll, können größere Werte von r verwendet werden.

DRAW prüft die momentane Gleichung, um zu sehen, ob wenigstens einmal Bezug (direkt oder indirekt) auf die unabhängige Variable genommen wird. Wenn diese nie gewählt wurde, benutzt DRAW die erste Variable der momentanen Gleichung und speichert sie in PPAR. Wenn auf die unabhängige Variable in der momentanen Gleichung nicht Bezug genommen wird, erscheint kurzzeitig die Meldung

$Name_1$ `Not In Equation`
`Using` $Name_2$

bevor die Anzeige gelöscht wird und die Graphik in der Anzeige erscheint. Dabei bedeutet $Name_1$ die unabhängige Variable, gespeichert in PPAR, und $Name_2$ entspricht der ersten Variablen der Gleichung. Wenn die momentane Gleichung keine Variable enthält, wird in der zweiten Zeile der Meldung der Hinweis `Constant Equation` (Konstantengleichung) ausgegeben. Die unabhängige Variable in PPAR ist in diesem Fall eine Konstante.

PPAR RES AXES CENTR *W *H

Diese Befehle bieten Alternativen, um Abbildungsparameter zu spezifizieren.

...PLOT

PPAR — *Abbildungsparameter abrufen* — Operation

	Ebene 1
	➧ { *Abbildungsparameter* }

Durch PPAR (*Plot PARameter*) besteht eine bequeme Möglichkeit, die momentanen Abbildungsparameter anzusehen.

PPAR ist eine Variable, die eine Liste der Abbildungsparameter in der Form

$\{ (x_{min}\ y_{min})\ (x_{max}\ y_{max})$ *unabhängige Var. Auflösung* $(x_{Achse}\ y_{Achse}) \}$

enthält. Das Drücken von PPAR gibt die Liste in den Stack zurück. Der Inhalt der Liste ist unter "Abbildungsparameter" auf Seite 204 beschrieben.

RES — *Auflösung* — Befehl

Ebene 1	
n	→

RES (*RESolution*) legt für die Auflösung der jeweiligen Abbildung den Wert *n* fest. *n* wird als vierter Listeneintrag unter der Variablen PPAR gespeichert. *n* bestimmt die Anzahl der Zeichenpunkte: $n = 1$ bedeutet, daß für jede Spalte der Anzeigepunkte (Pixel) ein Punkt gezeichnet wird; $n = 2$ bedeutet ein Zeichenpunkt in jeder zweiten Spalte, usw. Der Vorgabewert für *n* beträgt 1; die Verwendung eines größeren Werts für *n* ist zur Verkürzung des Zeichenvorgangs sinnvoll.

AXES — *Koordinatenursprung* — Befehl

Ebene 1	
⟨x,y⟩	➧

AXES legt die Koordinaten für den Ursprung des Koordinatenkreuzes (durch die Befehle DRAX, DRAW oder DRWΣ gezeichnet) auf den Punkt (x, y) fest. Die Zahl in der komlexen Form (x, y) wird als fünfter und letzter Listeneintrag in PPAR gespeichert. Als Voreinstellung für AXES sind die Koordinaten (0, 0) definiert.

CENTR — *Zentrieren* — Befehl

Ebene 1	
‹x,y› ➧	

CENTR (*CENTeR*) modifiziert die Abbildungsparameter in der Weise, daß der durch das Argument (x, y) dargestellte Punkt mit dem zentralen Anzeigepunkt ($n_x = 68$, $n_y = 15$) korrespondiert. Die Höhe und Breite der Abbildung werden dabei nicht verändert. P_{max} und P_{min} werden durch P_{max}' und P_{min}' ersetzt, wobei:

$$x_{max}' = x + 1/2\,(x_{max} - x_{min}),\ y_{max}' = y + 16/31\,(y_{max} - y_{min})$$

$$x_{min}' = x - 1/2\,(x_{max} - x_{min}),\ y_{min}' = y - 15/31\,(y_{max} - y_{min})$$

*W — *Breite multiplizieren* — Befehl

Ebene 1	
x ➧	

*W (*multiply Width*) modifiziert die Abbildungsparameter so, als ob x_{min} und x_{max} mit dem Wert x multipliziert wurden.

$$x_{min}' = x \times x_{min}$$

$$x_{max}' = x \times x_{max}$$

...PLOT

*H — *Höhe multiplizieren* — Befehl

Ebene 1	
x ➧	

*H (*mulitply Height*) modifiziert die Abbildungsparameter so, als ob y_{min} und y_{max} mit dem Wert x multipliziert wurden.

$$y_{min}' = x \times y_{min}$$

$$y_{max}' = x \times y_{max}$$

STOΣ RCLΣ COLΣ SCLΣ DRWΣ

Diese Gruppe von Befehlen ermöglicht Ihnen, Streudiagramme aus dem Statistikbereich zu erzeugen. Sehen Sie unter "STAT" nach einer Beschreibung der Fähigkeiten des HP-28S für allgemeine Statistikanwendungen.

STOΣ — *Sigma speichern* — Befehl

Ebene 1	
⌊ *R-Feld* ⌋ ➧	

STOΣ nimmt ein reelles Feld vom Stack und speichert es in der Variablen ΣDAT. Die Ausführung von STOΣ ist zur Ausführung des Befehls `'ΣDAT' STO` äquivalent. Das gespeicherte Feld wird zur momentanen Statistikmatrix.

RCLΣ — *Sigma zurückrufen* — Befehl

	Ebene 1
➧ *Objekt*	

RCLΣ gibt den Inhalt der Variablen ΣDAT in den Stack zurück. RCLΣ ist äquivalent zu dem Befehl `'ΣDAT' RCL`.

COLΣ — *Sigma Spalten* — Befehl

Ebene 2	Ebene 1	
n_1	n_2	↵

COLΣ nimmt zwei reelle ganze Zahlen, n_1 und n_2, vom Stack und speichert sie als die ersten zwei Listeneinträge unter der Variablen ΣPAR. Die Zahlen identifizieren die Spalten in der momentanen Statistik-Matrix ΣDAT und werden von Statistik-Befehlen, welche mit Spaltenpaaren arbeiten, verwendet. Beziehen Sie sich für Details über ΣPAR auf "STAT".

n_1 legt die Spalte fest, welche sich auf die unabhängige Variable für LR oder die horizontale Koordinate für DRWΣ bzw. SCLΣ bezieht. n_2 legt die abhängig Variable oder die vertikale Koordinate bezieht. Für CORR und COV ist die Reihenfolge der zwei Spalten unrelevant.

Wenn ein Befehl, welcher zwei Spalten als Argumente erwartet, ausgeführt wird und ΣPAR existiert noch nicht, so wird sie automatisch mit den Vorgabewerten $n_1 = 1$ und $n_2 = 2$ erzeugt.

SCLΣ — *Sigma skalieren* — Befehl

↵

SCLΣ (*SCaLe* sigma) bewirkt eine automatische Skalierung der Abbildungsparameter in PPAR in der Weise, daß ein nachfolgendes Streudiagramm genau in die Anzeige eingepaßt wird. Dazu werden die horizontalen Koordinaten von P_{max} und P_{min} als Minimum- und Maximum-Koordinatenwerte gesetzt, entsprechend den in der Spalte für die unabhängigen Werte in der Statistikmatrix gespeicherten Daten. In gleicher Weise werden die vertikalen Koordinaten von P_{max} und P_{min} anhand der Spalte für die abhängigen Werte gesetzt. Die unabhängigen und abhängigen Spaltenwerte sind unter der Variablen ΣPAR gespeichert.

...PLOT

DRWΣ ***Sigma zeichnen*** **Befehl**

DRWΣ (*DRaW Σ*) führt automatisch DRAX aus und erzeugt danach ein Streudiagramm anhand der Koordinatenpaare, welche in den Spalten für abhängige und unabhängige Werte in der Statistikmatrix ΣDAT gespeichert sind. Wenn Sie DRWΣ durch Drücken von DRWΣ ausführen, dann wird eine interaktive Abbildung erzeugt, wie auf Seite 203 beschrieben.

Die Spalten der unabhängigen und abhängigen Werte sind in der Variablen ΣPAR (Voreinstellung 1 und 2) spezifiziert. DRWΣ zeichnet für jeden Datenpunkt der Statistikmatrix einen Punkt. Dabei ergibt sich jeder Abszissenwert aus der Spalte der unabhängigen Daten, und der Ordinatenwert ergibt sich aus der Spalte mit den abhängigen Daten.

CLLCD DISP PIXEL DRAX CLMF PRLCD

Diese Befehle ermöglichen Ihnen, spezielle Anzeigen zu erzeugen und eine Kopie der Anzeige auf dem Drucker HP-82440A auszudrucken.

CLLCD ***LCD löschen*** **Befehl**

CLLCD (*CLear LCD*) bewirkt das Löschen der Rechneranzeige (außer den Indikatoren) und das Setzen des Systemmeldungsflags.

DISP — *Anzeigen* — Befehl

Ebene 2	Ebene 1	
Objekt	*n*	➧

DISP (*DISPlay*) zeigt ein *Objekt* in der *n*-ten Zeile der Anzeige an. $n = 1$ bedeutet hier die oberste Anzeigezeile, während mit $n = 4$ die unterste Zeile spezifiziert wird. DISP setzt den Systemmeldungsflag, um die normale Stackanzeige zu unterdrücken.

Ein durch DISP angezeigtes Objekt erscheint in der selben Form, wie wenn das Objekt in Ebene 1 im Mehrzeilen-Anzeigeformat erscheinen würde (außer bei Strings, welche ohne die Abgrenzungszeichen " angezeigt werden, um die Anzeige von Meldungen zu ermöglichen). Wenn das Objekt zur Anzeige mehr als eine Zeile benötigt, dann beginnt die Anzeige in Zeile *n* und setzt sich bis zum Ende des Objekts nach unten fort.

PIXEL — *Pixel* — Befehl

Ebene 1	
(*x*,*y*)	➧

PIXEL zeigt den Punkt an, welcher durch die Koordinaten in der komplexen Form (x, y) spezifiziert ist, und setzt den Systemmeldungsflag.

DRAX — *Achsen zeichnen* — Befehl

➧

...PLOT

DRAX bildet ein Koordinatenkreuz in der Anzeige ab und setzt den Systemmeldungsflag. Der Ursprung des Kreuzes liegt in dem Punkt P_{Achsen}, welcher unter der Variablen PPAR spezifiziert wurde. Außerdem werden im Abstand von jeweils 10 Pixel Strichmarkierungen auf den Koordinatenachsen angebracht.

CLMF ***Meldungsflag löschen*** **Befehl**

CLMF (*CLear Message Flag*) löscht den internen Meldungsflag, der von CLLCD, DISP, PIXEL, DRAX, DRAW und DRWΣ gesetzt wird. Wird der Befehl CLMF in einem Programm nach dem letzten Auftreten von einem dieser Befehle verwendet, so bewirkt dies die Wiederanzeige des normalen Stacks, nachdem die Programmausführung abgeschlossen wurde.

PRLCD ***LCD drucken*** **Befehl**

PRLCD (*PRint LCD*) liefert Ihnen die Möglichkeit, eine Kopie von Streudiagrammen oder von Abbildungen mathematischer Funktionen auf einem Drucker auszugeben. Da PRLCD nur eine Kopie der momentanen Anzeige erstellt, müssen Sie PRLCD und DRAW (oder DRWΣ) in der gleichen Befehlszeile mit eingeben. Als Beispiel wird bei der Auswertung von

CLLCD DRAW PRLCD ENTER

die Anzeige zuerst gelöscht, danach die momentane Gleichung abgebildet und dann eine Kopie der Anzeige auf den Drucker ausgegeben.

PRINT

PR1	PRST	PRVAR	PRLCD	TRACE	NORM
PRSTC	PRUSR	PRMD	CR		

Der HP-28S überträgt über einen Infrarot-Sender Text und Graphiken an den Drucker HP 82240A. Die Diode des Senders ist am oberen Rand des rechten Rechnergehäuses angebracht. Vor dem Drucken sollten Sie sicherstellen, daß der Übertragungsweg für die Infrarotstrahlen nicht beeinträchtigt ist und der Drucker die gesendeten Daten auch empfangen kann. Die Betriebsweise des Druckers selbst ist in dessen Betriebsanleitung dokumentiert.

Sie können mittels den Druckbefehlen Objekte, Variableninhalte, Stackebenen, Graphiken, usw. drucken. Zusätzlich läßt sich der TRACE-Modus (eine Art Protokoll) wählen, wodurch automatisch jede Operation bzw. Berechnung über den Drucker protokolliert wird.

Bei jeder Zeichenübertragung via Infrarotdiode erscheint der Druck-Indikator in der Anzeige des HP-28S. Der Rechner kann dabei nicht feststellen, ob tatsächlich gedruckt wird, da die Übertragung nur in einer Richtung erfolgt. Stellen Sie sicher, daß der TRACE-Modus nicht aktiviert ist—ansonsten dauert die Ausführung der Tastenfeld-Operationen, bedingt durch die ständige Übertragung der Infrarotsignale, etwas länger.

Druckformate

Mehrzeilen-Objekte können in kompaktem Format oder im Mehrzeilen-Druckformat gedruckt werden. Das kompakte Druckformat ist mit dem kompakten Anzeigeformat identisch; das Mehrzeilen-Druckformat ist dem mehrzeiligen Anzeigeformat ähnlich, außer daß die folgenden Objekte vollständig gedruckt werden:

- Strings und Namen mit mehr als 23 Zeichen werden auf der nachfolgenden Druckzeile ausgegeben.

- Real- und Imaginärteil von komlexen Zahlen werden als getrennte Zeilen ausgegeben, wenn deren Zahlenwerte nicht in eine Zeile passen.
- Beim Ausdrucken von Feldern erscheint der Index vor dem jeweiligen Element. So geht dem ersten Element z.B. die Angabe `1 , 1 :` voraus.

Wenn Sie den TRACE-Modus spezifiziert haben, ist das Druckformat vom Anzeigeformat (Mehrzeilen-Modus aktiviert oder desaktiviert) abhängig. Der Druckbefehl PRSTC gibt die Daten im kompakten Format aus, alle anderen Druckbefehle verwenden das Mehrzeilen-Format.

Druckgeschwindigkeit

Die Druckgeschwindigkeit eines batteriebetriebenen Druckers nimmt mit zunehmender Batterieentladung ab. Der Rechner dosiert normalerweise die übertragene Datenmenge, um der langsameren Druckgeschwindigkeit des Druckers zu entsprechen.

Wenn der Drucker über ein Ladegerät betrieben wird, kann mit einer höheren Druckgeschwindigkeit gearbeitet werden. Die dafür notwendige erhöhte Übertragungsgeschwindigkeit wird durch Setzen von Flag 52 erreicht. Falls später der Drucker wieder batteriebetrieben wird, muß dieser Flag wieder gelöscht werden.

Flag 52 darf nur gesetzt werden, wenn der Drucker über das Ladegerät betrieben wird. Obwohl der Drucker mit einen neuen Batteriesatz mit einer höheren Druckgeschwindigkeit betrieben werden kann, besteht das Risiko, daß mit abnehmender Batteriekapazität der Druckvorgang so verlangsamt wird, daß letztlich vom Rechner gesendete Daten verloren gehen. Damit könnte der Ausdruck verfälscht oder die Druckerkonfiguration verändert werden.

Konfigurieren des Druckers

Über besondere Zeichensequenzen (*escape sequences*) lassen sich verschiedene Druckmodi spezifizieren. Eine Escape-Sequenz besteht aus dem Escape-Zeichen (Zeichen 27), welchem ein weiteres Zeichen folgt. Wenn der Drucker eine Escape-Sequenz empfängt, schaltet er in den Auswahlmodus um. Die Escape-Sequenz selbst wird nicht gedruckt. Über nachfolgende Sequenzen lassen sich die jeweiligen Druckmodi für den Drucker HP 82240A spezifizieren:

Druckmodus	Escape-Sequenz
Graphikspalte drucken	27 001...166
Keine Unterstreichung*	27 250
Unterstreichung	27 251
Einfache Druckbreite*	27 252
Doppelte Druckbreite	27 253
Selbsttest	27 254
Reset	27 255

* Voreinstellung

Sie können CHR und + verwenden, um die Escape-Sequenzen zu erzeugen; mit PR1 werden sie an den Drucker gesendet. Zum Beispiel läßt sich der Ausdruck `Underline` mit der nachstehenden Sequenz erhalten:

```
27 CHR 251 CHR + "Under" + 27 CHR + 250 CHR +
 "line" + PR1
```

...PRINT

PR1 PRST PRVAR PRLCD TRACE NORM

PR1 *Ebene 1 drucken* **Befehl**

Ebene 1	Ebene 1
Objekt	*Objekt*

PR1 druckt den Inhalt von Ebene 1 im Mehrzeilen-Druckformat. Alle Objekte, außer Strings, werden mit ihren zugehörigen Begrenzungszeichen gedruckt. Strings erscheinen ohne die führenden und abschließenden ". Wenn Ebene 1 keine Daten enthält, wird die Meldung `[Empty Stack]` (leerer Stack) gedruckt.

Drucken eines Textstrings

Sie können jede beliebige Zeichenfolge drucken, indem Sie die gewünschten Zeichen in einem String zusammenfassen, diesen String in Ebene 1 anzeigen und dann PR1 ausführen. Nach dem Drucken der Zeichen bleibt der Druckkopf am rechten Ende der Druckzeile stehen. Nachfolgende Ausdrucke beginnen in der nächsten Zeile.

Drucken eines Graphikstrings

Sie können Graphiken über den Drucker ausgeben, indem Sie ein String-Objekt drucken, welches mit dem Escape-Zeichen "27" beginnt und dem eine Zahl n (im Bereich 1 bis 166) folgt. Diese Escape-Sequenz weist den Drucker an, die nächsten n Zeichen ($n \leq 166$) als *Graphikcode* zu interpretieren, wobei jedes Zeichen eine Graphikspalte spezifiziert. Beziehen Sie sich für detaillierte Informationen über Graphikcodes auf das Bedienungshandbuch des Druckers.

Nach dem Drucken der Graphikzeichen bleibt der Druckkopf am rechten Ende der Druckzeile stehen. Nach dem Einschalten des Druckers ist zuerst das Drucken von Text bzw. die Ausführung von CR erforderlich, bevor Graphiken gedruckt werden können.

Sammeln von Daten im Druckpuffer

Sie können jede beliebige Kombination von Text, Graphiken und Objekten in einer einzelnen Druckzeile ausgeben, indem Sie die Daten im Drucker anhäufen. Der Drucker speichert die Daten in einem separaten Teil seines Speichers, dem *Druckpuffer*.

Normalerweise schließt jeder Druckbefehl die Datenübertragung mit einem CR Zeichen (*Carriage Right*) ab. Wenn der Drucker das CR empfängt, druckt er die Daten, welche sich im Druckpuffer befinden, und läßt den Druckkopf am rechten Zeilenende stehen.

Die automatische Übertragung von CR kann durch das Setzen des Flags 33 verhindert werden. Die folgenden Druckbefehle senden dann die Daten ohne CR zum Drucker. Sämtliche Daten werden im Druckpuffer gesammelt und erst aufgrund Ihrer Anweisung gedruckt. Beachten Sie die nachstehenden Regeln, wenn Sie Flag 33 gesetzt haben:

- Senden Sie CR (Zeichen 4) oder "Neue Zeile" (Zeichen 10), oder führen Sie den Befehl CR aus, wenn Sie den Drucker zum Ausdrucken der gesammelten Daten veranlassen möchten.
- Übertragen Sie nicht mehr als 200 Zeichen, ohne daß der Drucker zwischendurch Daten ausdruckt. Ansonsten überfüllt sich der Druckpuffer und Daten gehen verloren.
- Sehen Sie etwas Zeit für den eigentlichen Druck vor, bevor Sie weitere Zeichen an den Drucker übertragen. Der Drucker benötigt etwa 1,8 Sekunden für die Ausgabe einer Druckzeile.
- Löschen Sie Flag 33, nachdem Sie im Druckpuffer keine Daten mehr sammeln möchten. Damit kehren Sie zur normalen Funktionsweise der Druckbefehle zurück.

PRST **Stack *drucken*** **Befehl**

... **Ebene 1**	... **Ebene 1**
... *Objekt*	... *Objekt*

...PRINT

PRST (*PRint STack*) druckt alle Objekte des Stacks, beginnend mit dem Objekt in der höchsten Ebene. Alle Objekte werden im Mehrzeilen-Druckformat ausgegeben.

PRVAR ***Variable drucken*** **Befehl**

Ebene 1	
' *Name* ' ➧	

PRVAR (*PRint VARiable*) druckt das unter der Variablen *Name* gespeicherte Objekt (im Mehrzeilen-Druckformat).

PRLCD ***LCD drucken*** **Befehl**

➧

PRLCD (*PRint LCD*) druckt ein Punkt-für-Punkt Bild der momentanen HP-28S Anzeige (ohne Indikatoren).

Die Breite eines gedruckten Objekts ist etwas schmäler, wenn PRLCD anstatt eines Befehls wie z.B. PR1 verwendet wird. Der Unterschied resultiert aus dem verschiedenen Zeichenabstand, welcher bei den beiden Befehlen zur Anwendung kommt. In der Anzeige ist eine einfache Leerspalte zwischen den Zeichen und PRLCD druckt diese Leerspalte. Druckbefehle wie PR1 drucken dagegen zwei Leerspalten zwischen benachbarten Zeichen.

TRACE-Modus: TRACE, NORM

Sie können ein fortlaufendes Protokoll Ihrer Berechnungen erstellen, indem Sie den TRACE-Modus wählen. Wann immer Sie ENTER ausführen, entweder durch Drücken von ENTER oder einer unmittelbaren Ausführungstaste, wird der Inhalt der Befehlszeile, der unmittelbare Ausführungsbefehl und das Ergebnis von Ebene 1 gedruckt.

Um den TRACE-Modus zu aktivieren, drücken Sie TRACE. Danach erscheint das TRACE Menüfeld mit dunklen Zeichen auf hellen Hintergrund, womit angezeigt werden soll, daß der TRACE-Modus aktiviert ist. Der TRACE-Modus läßt sich über ein Programm aktivieren, indem Flag 32 gesetzt wird.

Um TRACE wieder auszuschalten, drücken Sie NORM. Danach erscheint das NORM Menüfeld mit dunklen Zeichen auf hellem Hintergrund, womit angezeigt werden soll, daß TRACE desaktiviert ist. Der TRACE-Modus läßt sich über ein Programm desaktivieren, indem Flag 32 gelöscht wird.

Das Druckformat für das Objekt in Ebene 1 hängt davon ab, ob das Mehrzeilen-Anzeigeformat aktiviert ist oder nicht (Flag 45 ist gesetzt oder gelöscht). Bei aktiviertem Mehrzeilen-Anzeigeformat (Flag 45 gesetzt) wird das Objekt im entsprechendem Druckformat gedruckt.

PRSTC PRUSR PRMD CR

PRSTC ***Stack drucken (kompakt)*** **Befehl**

... **Ebene 1**	... **Ebene 1**
... *Objekt*	... *Objekt*

PRSTC (*PRint STack Compact*) dient zum Drucken aller Objekte des Stacks, beginnend mit dem Objekt in der höchsten Ebene. Der Druck erfolgt hierbei im kompakten Format.

...PRINT

PRUSR *Benutzervariable drucken* Befehl

PRUSR (*PRint USeR variables*) druckt die Namen der momentanen Benutzervariablen. Die Reihenfolge der Namen entspricht der im USER-Menü. Wenn es keine Benutzervariable gibt, verursacht PRUSR den Ausdruck `No User Variables`.

PRMD *Modi drucken* Befehl

PRMD (*PRint MoDes*) zeigt und druckt die momentane Auswahl für Zahlen-Anzeigeformat, Basis für binäre Werte, Winkelmodus, Dezimalzeichen sowie die Spezifikationen für UNDO, COMMAND, LAST und Mehrzeilen-Anzeigeformat.

CR *Carriage Right* Befehl

CR druckt den Inhalt (sofern nicht leer) des Druckpuffers.

Programme

Ein *Programm* besteht aus einem Prozedur-Objekt und wird von den Begrenzungszeichen « » eingeschlossen. Es kann eine Reihenfolge von Befehlen, Objekten und *Programmstrukturen* enthalten, welche bei der Auswertung des Programms ausgeführt werden. Bestimmte Programmstrukturen, wie z.B. die unter "PROGRAM BRANCH" beschriebenen oder die zur Spezifikation von lokalen Namen, müssen einer speziellen Syntax folgen. Ansonsten ist jedoch der Inhalt eines Programms viel flexibler als der von algebraischen Objekten (der zweite Prozedur-Typ).

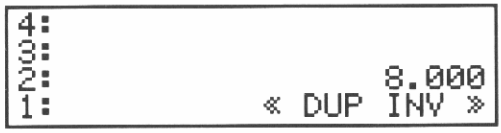
Ein Programm ist, in einfachen Worten, eine Befehlszeile, deren Ausführung aufgeschoben ist. Jede Befehlszeile kann in ein Programm umgewandelt werden, indem ein « am Zeilenanfang eingefügt wird. Nach dem Drücken von [ENTER] wird die Befehlszeile als Programm in den Stack übernommen. Die individuellen Objekte des Programms werden dabei erst bei der Auswertung des Programms ausgeführt.

Durch die Umwandlung der Befehlszeile in ein Programm läßt sich nicht nur die Auswertung aufschieben; die Ausführung selbst kann beliebig oft wiederholt werden. Jede gewünschte Anzahl von Programmkopien kann im Stack, unter Verwendung der gewöhnlichen Stack-Befehle, erzeugt werden. Ein Programm läßt sich auch unter einer Variablen speichern, wodurch die Ausführung durch den Aufruf der Variablen—oder durch Drücken der entsprechenden USER-Menütaste—gestartet werden kann. Nachdem ein Programm erst einmal in einer Variablen gespeichert ist, kann es nicht mehr wesentlich von einem Befehl unterschieden werden. (Im Prinzip sind die Befehle selbst nur Programme, welche im ROM anstatt im RAM gespeichert sind.) Durch die Verwendung von Programmen im HP-28C erweitern Sie dessen eigentlichen Befehlssatz.

...Programme

Auswertung von Programm-Objekten

Beim Auswerten eines Programms wird jedes Objekt in den Stack übernommen und danach ausgewertet, sofern es sich bei dem Objekt um einen Befehl oder einen nicht in Anführungszeichen stehenden Namen handelt. Enthält der Stack z.B.:



so erhält man nach dem Drücken von EVAL:

Beim Auswerten von DUP wurde 8.000 in Ebene 2 kopiert, wonach INV ausgewertet wurde, wodurch 8.000 in Ebene 1 durch seinen Kehrwert ersetzt wurde.

Einfache und umfangreichere Programme

Die einfachste Art eines Programms besteht in einer einzelnen Folge von Objekten, welche sequentiell ausgeführt werden, ohne daß dabei eine Schleife durchlaufen oder eine Unterbrechung vorkommt. So multipliziert z.B. das Programm « 5 * 2 + » eine Zahl in Ebene 1 mit dem Faktor 5 und addiert anschließend 2.

Wenn dies eine Operation wäre, welche Sie häufig anwenden, so könnten Sie dieses Programm unter einem Variablennamen speichern und danach beliebig oft über das USER-Menü wieder aufrufen bzw. ausführen.

Ein Programm läßt sich durch nachstehende Möglichkeiten anspruchsvoller entwerfen:

Bedingte Verzweigungen Durch den Gebrauch der Verzweigungsstrukturen IF...THEN...END oder IF...THEN... ELSE...END (die korrespondierenden Befehle sind IFT und IFTE) kann der Ablauf des Programms in Abhängigkeit von bestimmten Ergebnissen gesteuert werden.

Schleifen: Sie können die Ausführung des Programms (oder Teile des Programms) eine bestimmte oder unbestimmte Anzahl wiederholen, indem Sie die Programmschleifen FOR...NEXT, START...NEXT, DO...UNTIL...END und WHILE...REPEAT...END benutzen.

Fehlerabhandlung: Durch den Gebrauch der Strukturen IFERR...THEN...END oder IFERR... THEN...ELSE...END können Sie den Programmablauf so steuern, daß bei erwarteten oder unerwarteten Fehlern eine entsprechende Routine ausgeführt wird.

Anhalten der Programmausführung: Über den Befehl HALT wird Ihnen ermöglicht, die Ausführung an einem vorgegebenen Punkt anzuhalten, um z.B. Benutzereingaben oder andere Operationen durchzuführen. Die Programmausführung wird durch ■ CONT oder SST fortgesetzt.

Programme innerhalb von Programmen: Genauso wie Sie die Auswertung einer Befehlszeile durch das Einschließen ihres Inhalts in « » verschieben können, lassen sich Programm-Objekte durch die Abgrenzung mit « » innerhalb von anderen Programmen verwenden. Wenn bei der Ausführung des "äußeren" Programms das "innere" Programm erkannt wird, dann wird anstatt der Auswertung des inneren Programms dieses in den Stack übernommen. Danach kann es mit EVAL oder einem anderen Befehl, welcher ein Programm als Argument verwendet, ausgewertet werden.

...Programme

Mit zunehmender Länge und Komplexität eines Programms kann es vorkommen, daß das Ansehen des Codes in der Rechneranzeige nicht sehr angenehm ist, oder daß das Programm zu groß ist, um eingegeben werden zu können. Aus diesen Gründen (und um zu einer ordentlichen Programmierweise anzuhalten) wird empfohlen, daß Sie längere Programme in mehrere kürzere Programme aufteilen. So kann z.B. das Programm « A B C D » in « AB CD » umgeschrieben werden, wobei AB das Programm « A B » und CD das Programm « C D » darstellt.

Der Teilungsprozeß eines langen Programms in eine Reihe von kleineren Programme macht es relativ einfach, das Programm zu "debuggen", d.h. Fehler aufzusuchen und zu beseitigen. Jede kleinere Programmeinheit kann unabhängig von den anderen Programmen getestet werden, um sicherzustellen, daß die richtige Anzahl und Art von Argumenten vom Stack genommen wird und korrekte Werte in den Stack zurückgegeben werden. Es ist danach relativ einfach, die "Unterprogramme" zu einem Hauptprogramm zusammenzuschließen, welches aus den Namen (ohne Anführungszeichen) der Unterprogramme besteht.

Lokale Variablen und Namen

Eine *lokale Variable* ist die Kombination eines Objekts und eines *lokalen Namens*, die zusammen in einem Teil des Speicherbereichs gespeichert werden, welcher temporär für die Ausführungszeit einer Prozedur reserviert wurde. Nach dem Abschluß einer Prozedurausführung wird automatisch jede lokale Variable, die mit der Prozedur in Zusammenhang stand, gelöscht.

Lokale Namen sind Objekte, die zur Benennung von lokalen Variablen dienen; sie unterliegen denselben Namensrestriktionen wie für gewöhnliche Namen. Lokale Variable können innerhalb ihrer definierenden Prozedur beinahe wie gewöhnliche Namen verwendet werden. Allerdings gibt es mehrere wichtige Unterscheidungen:

- Bei der Auswertung von lokalen Namen geben diese das Objekt, welches unter der zugehörigen lokalen Variablen gespeichert ist, unausgewertet zurück. Sie werten im Gegensatz zu gewöhnlichen Namen nicht automatisch Namen oder Programme aus, die unter ihren lokalen Variablen gespeichert sind.
- Ein in Anführungszeichen stehender lokaler Name kann nicht als Argument für ■ VISIT oder für einen dieser Befehle verwendet werden: CON, IDN, PRVAR, PURGE, PUT, PUTI, RDM, SCONJ, SINV, SNEG, STO+, STO-, STO*, STO/, TAYLR oder TRN.
- Lokale Variable erscheinen nicht im Variablen-Menü des Gleichungslösers.

Wenn Sie eine normale Variable mit dem gleichen Namen wie eine lokale Variable haben, wird beim Gebrauch des gemeinsamen Namens innerhalb der entsprechenden Prozedur nur auf die lokale Variable Bezug genommen; die normale Variable bleibt unverändert. Ähnlich verhält es sich, wenn eine lokale Variablenstruktur innerhalb einer anderen eingebaut ist: Die lokalen Namen der ersten (äußeren) Struktur können in der zweiten (inneren) Struktur verwendet werden.

Es ist für lokale Namen möglich, im Stack oder in Prozeduren und Listen zu bleiben, auch wenn ihre zugehörige lokale Variable bereits gelöscht wurde. So läßt z.B. die Operation `1 → x « 'x' »` ENTER den lokalen Namen `'x'` im Stack. Wenn Sie versuchen, den lokalen Namen auszuwerten oder als Argument für STO, RCL oder PURGE zu verwenden, erscheint die Fehlermeldung `Undefined Local Name` in der Anzeige.

Um jeder möglichen Verwechslung zwischen gewöhnlichen Namen und lokalen Namen vorzubeugen, wird empfohlen, daß Sie sich eine spezielle Konvention bei der Vergabe von lokalen Namen zugrunde legen. Eine denkbare Konvention, die z.B. in diesem Handbuch verwendet wurde, ist die Verwendung von Kleinbuchstaben für lokale Variable (erscheinen nie in Menüfeldern) und Großbuchstaben für normale Variable.

...Programme

Erzeugen von lokalen Variablen

Bei der Verwendung von Programmstrukturen werden lokale Variable erzeugt. Dieser Abschnitt beschreibt zwei *lokale Variablenstrukturen*, welche die primären Mittel zur Erzeugung lokaler Variablen bilden. Es gibt auch zwei Programmverzweigungsstrukturen, FOR...NEXT und FOR...STEP, welche endliche Schleifendurchgänge definieren und wobei der Schleifenindex aus einer lokalen Variablen besteht. Diese Programmverzweigungsstrukturen sind unter "PROGRAM BRANCH" beschrieben.

Die Strukturen für lokale Variable haben die Form:

→ $Name_1$ $Name_2$...« *Program* »

→ $Name_1$ $Name_2$...' *algebr. Ausdruck* '

Der Befehl → bewirkt den Beginn einer lokalen Variablenstruktur. (Das Zeichen → wird durch ■U auf dem linken Tastenfeld erzeugt. Im vorliegenden Anwendungsfall ist → ein selbständiger Befehl, dem ein Leerzeichen folgt.) Die Namen bestimmen den lokalen Namen, für welche eine lokale Variable erzeugt wird. Das Programm oder der algebraische Ausdruck wird als *definierende Prozedur* der lokalen Variablenstruktur bezeichnet. Ihr ursprüngliches Begrenzungszeichen, « oder ', beendet die Folge der lokalen Namen.

Wenn → ausgewertet wird, nimmt er für jeden lokalen Namen ein Objekt vom Stack und speichert jedes Objekt in einer lokalen Variablen mit dem entsprechenden Namen. Die Objekte werden mit den lokalen Namen in der Reihenfolge der Eingabe abgestimmt, d.h. die Eingabe

```
1 2 3 4 5 → a b c d e
```

ordnet die Zahl 1 der lokalen Variablen a, 2 der Variablen b, 3 c, 4 d und 5 e zu. (Da es sich hier um lokale Variable handelt, gibt es keinen Konflikt mit der symbolischen Konstanten e.)

Nachdem die lokalen Variablen erzeugt und die Werte zugeordnet sind, erfolgt die Auswertung der Prozedur, welche der Liste folgt. Innerhalb dieser Prozedur können die lokalen Variablennamen wie normale Namen verwendet werden (außer den oben erwähnten Einschränkungen). Beim Abschluß der Prozedurausführung werden die lokalen Variablen automatisch gelöscht.

Als Beispiel: Es sollen drei Zahlenwerte vom Stack genommen werden, wobei die erste Zahl (Ebene 3) mit 4, die zweite (Ebene 2) mit 3, und die dritte (Ebene 1) mit 2 multipliziert und anschließend die Produkte addiert werden sollen. Ein einfaches Programm für diese Aufgabenstellung wäre:

```
« 2 * SWAP 3 * + SWAP 4 * + »
```

Unter Verwendung von lokalen Variablen würde man dieses Programm erhalten:

```
« → a b c « a 4 * b 3 * + c 2 * + » »
```

Die Verwendung von lokalen Variablen hat die SWAP Operation eliminiert. In diesem einfachen Fall ist der Gebrauch von lokalen Variablen ohne große Bedeutung; mit zunehmender Komplexität jedoch helfen Ihnen lokale Variable dabei, ein Programm in einer einfacheren und weniger fehlerbehafteten Weise zu schreiben, als wenn Sie versuchen, alles über den Stack abzuwickeln.

Das vorangegegangene Beispiel läßt sich außerdem leicht in die algebraische Form überführen. Die Prozedur ergibt sich dann wie folgt:

```
« → a b c '4*a+3*b+2*c' »
```

was zum selben Ergebnis führt.

...Programme

Benutzerdefinierte Funktionen

Der Befehl → kann, sofern er in einer besonderen Syntax verwendet wird, zum Erzeugen neuer algebraischer Funktionen benutzt werden. Eine algebraische Funktion ist ein Befehl, welcher innerhalb von Definitionen für algebraische Ausdrücke verwendet werden kann. In diesen Definitionen nimmt die Funktion ihre Argumente aus einer Sequenz, die in Klammern eingeschlossen dem Funktionsnamen folgt. Der Befehl SIN zum Beispiel ist eine typische algebraische Funktion, welche ein Argument verwendet. Innerhalb eines algebraischen Ausdrucks kommt sie in der Form `'SIN(X)'` zur Anwendung, wobei `X` ihr Argument darstellt.

Eine *benutzerdefinierte Funktion* mit *n* Argumenten wird durch ein Programm in der folgenden Syntax definiert:

« → $Name_1$ $Name_2$ ... $Name_n$ '*Ausdruck*' »

wobei $Name_1$ $Name_2$ ... $Name_n$ aus einer Folge von *n* lokalen Variablen besteht. *Ausdruck* ist ein algebraischer Ausdruck, der die lokalen Variablennamen enthält und die mathematische Definition der Funktion darstellt. Im Programm dürfen dem → keine Objekte vorangehen, und '*Ausdruck*' dürfen keine Objekte folgen.

Betrachten Sie als Beispiel die algebraische Form der Prozedur, welche im vorangehenden Abschnitt definiert ist:

```
« → a b c '4*a+3*b+2*c' »
```

Sie nimmt drei Argumente, multipliziert diese mit 4, 3 und 2 und addiert die einzelnen Produkte. Da → nichts vorangeht und dem algebraischen Ausdruck nichts folgt, stellt dieses Programm eine benutzerdefinierte Funktion dar. Nehmen Sie an, daß Sie diese Funktion—durch Speichern des Programms in der Variablen XYZ—als XYZ benennen möchten:

```
« → a b c '4*a+3*b+2*c' » 'XYZ' STO.
```

In UPN Syntax läßt sich `1 2 3 XYZ` ausführen, wodurch man 16 ($4 \times 1 + 3 \times 2 + 2 \times 3$) als Ergebnis erhält. Sie können aber auch die algebraische Syntax verwenden: `'XYZ(1,2,3)' EVAL` ergibt ebenfalls 16. Sie sind nicht auf numerische Argumente eingeschränkt; jedes der Argumente von XYZ kann ein algebraischer Ausdruck sein. XYZ selbst kann in jedem anderen algebraischen Ausdruck verwendet werden.

PROGRAM BRANCH

IF	**IFERR**	**THEN**	**ELSE**	**END**	
START	**FOR**	**NEXT**	**STEP**	**IFT**	**IFTE**
DO	**UNTIL**	**END**	**WHILE**	**REPEAT**	**END**

Das PROGRAM BRANCH Menü (■ BRANCH) enthält Befehle, die für Abfragen und Schleifen innerhalb eines Programms dienen. Die Befehle dürfen nur in bestimmen Kombinationen, *Programmstrukturen* genannt, vorkommen. Programmverzweigungsstrukturen können in 4 Kategorien aufgeteilt werden: Abfragen, Fehlerabhandlungen, bestimmte und unbestimmte Schleifen.

Im nachfolgenden ist mit *Anweisung* eine beliebige Anweisung in einem Programm gemeint.

1. Abfragestrukturen:
   - IF *Test-Anweisung* THEN *Wahr-Anweisung* END. Wenn die *Test-Anweisung* erfüllt ist, dann führe die *Wahr-Anweisung* aus. (IFT ist die Einzelbefehlsform dieser Struktur.)
   - IF *Test-Anweisung* THEN *Wahr-Anweisung* ELSE *Falsch-Anweisung* END. Wenn die *Test-Anweisung* erfüllt ist, dann führe die *Wahr-Anweisung* aus; ansonsten führe die *Falsch-Anweisung* aus. (IFTE ist die Einzelbefehlsform dieser Struktur.)
2. Fehlerbehandlungsstrukturen:
   - IFERR *Prüf-Anweisung* THEN *Fehler-Anweisung* END. Wenn während der Ausführung von *Prüf-Anweisung* ein Fehler auftritt, dann führe *Fehler-Anweisung* aus.
   - IFERR *Prüf-Anweisung* THEN *Fehler-Anweisung* ELSE *normale Anweisung* END. Wenn während der Ausführung von *Prüf-Anweisung* ein Fehler auftritt, dann führe *Fehler-Anweisung* aus; ansonsten führe die *normale Anweisung* aus.

3. Bestimmte Schleifenstrukturen:
   - *Anfang Ende* START *Schleifen-Anweisung* NEXT. Führe die *Schleifen-Anweisung* für jeden Wert des Schleifenzählers einmal aus; der Schleifenzähler wird beim Durchlaufen von *Anfang* bis *Ende* um eins erhöht.
   - *Anfang Ende* START *Schleifen-Anweisung Schrittweite* STEP. Führe die *Schleifen-Anweisung* für jeden Wert des Schleifenzählers einmal aus; der Schleifenzähler wird beim Durchlaufen von *Anfang* bis *Ende* um die *Schrittweite* erhöht.
   - *Anfang Ende* FOR *Name Schleifen-Anweisung* NEXT. Führe die *Schleifen-Anweisung* für jeden Wert der lokalen Variablen *Name*, die als Schleifenzähler verwendet wird, einmal aus; der Wert der Variablen wird beim Durchlaufen von *Anfang* bis *Ende* um eins erhöht.
   - *Anfang Ende* FOR *Name Schleifen-Anweisung Schrittweite* STEP. Führe die *Schleifen-Anweisung* für jeden Wert der lokalen Variablen *Name*, die als Schleifenzähler verwendet wird, einmal aus; der Wert der Variablen wird beim Durchlaufen von *Anfang* bis *Ende* um die *Schrittweite* erhöht.
4. Unbestimmte Schleifenstrukturen:
   - DO *Schleifen-Anweisung* UNTIL *Test-Anweisung* END. Führe die *Schleifen-Anweisung* wiederholt aus, bis die *Test-Anweisung* erfüllt ist.
   - WHILE *Test-Anweisung* REPEAT *Schleifen-Anweisung* END. Führe die *Schleifen-Anweisung* wiederholt aus, solange die *Test-Anweisung* erfüllt ist.

Diese Strukturen werden nach den zwei folgenden Einführungsabschnitten detailliert beschrieben.

...PROGRAM BRANCH

Tests und Flags

Jede Programmstruktur (außer bestimmte Schleifen) führen eine Verzweigung durch, die auf der Auswertung einer *Testanweisung* basiert. Eine Testanweisung ist eine beliebige Programmsequenz, welche nach der Auswertung ein *Flag* zurückgibt. Ein Flag wird durch eine reelle Zahl dargestellt, die normalerweise den Wert Null oder Eins annimmt. Wenn der Flag den Wert 0 hat, so ist dies äquivalent mit der Bedeutung "falsch" oder "gelöscht"; für jeden anderen Wert ist es gleichbedeutend mit "wahr" oder "gesetzt".

Jede Verzweigung wird dadurch entschieden, indem ein Flag vom Stack genommen und getestet wird. Wenn z.B. bei einer "IF *Testanweisung* THEN *Wahr-Anweisung* END" Struktur die Auswertung der *Testanweisung* zu einem Ergebnis ungleich Null führt, dann wird die *Wahr-Anweisung* ausgewertet. Ergibt sich aus der Testanweisung 0 in Ebene 1, so wird die Ausführung des Programms bis nach END übersprungen.

Ein *Testbefehl* gibt explizit ein Flag mit dem Wert 0 oder 1 zurück. So testet z.B. der Befehl $<$ zwei reelle Zahlen (oder Binärwerte oder Strings), um festzustellen, ob der Ausdruck in Ebene 2 kleiner als der in Ebene 1 ist. Wenn dies der Fall ist, gibt $<$ den Flag mit 1 zurück; ansonsten wird 0 zurückgegeben. Die anderen Testbefehle sind $>$, $\leq$, $\geq$, $==$, $\neq$, FS?, FC?, FS?C und FC?C, welche alle in "PROGRAM TEST" beschrieben sind.

Ersetzen von GOTO

Anwender, die bereits mit anderen Rechner-Programmiersprachen wie z.B. RPN Language (von anderen HP Rechnern) oder BASIC vertraut sind, haben wahrscheinlich das Fehlen einer einfachen GOTO Anweisung für den HP-28S bemerkt. GOTO wird häufig bei Verzweigungen benutzt, und um durch Wiederholung von Programmschritten die Programmgröße möglichst klein zu halten. Es soll noch untersucht werden, wie GOTO in HP-41 RPN und BASIC verwendet wird, und wie man gleichwertige Ergebnisse mit dem HP-28S erhalten kann.

...PROGRAM BRANCH

- Anwendung von GOTO Anweisungen bei bedingten Verzweigungen: Das nachstehende Programm z.B. führt die Sequenz ABC DEF aus, wenn die Zahl im X-Register (oder Variable) positiv ist; ansonsten wird die Sequenz GHI JKL ausgeführt.

HP-41 RPN	BASIC
01 X>0? 02 GTO 01 03 GHI 04 JKL 05 GTO 02 06 LBL 01 07 ABC 08 DEF 09 LBL 02 ⋮	10 IF X>0 THEN GOTO 50 20 GHI 30 JKL 40 GOTO 70 50 ABC 60 DEF ⋮

Hier die äquivalente Darstellung im HP-28S:

```
IF 0 > THEN ABC DEF ELSE GHI JKL END
```

- Anwendung von GOTO Anweisungen zur Wiederholung von Programmschritten und zur Minimierung der Programmmgröße: Beide nachstehende Programme enthalten die Sequenz MNO PQR STU, die in zwei Programmzweigen vorkommt.

HP-41 RPN	BASIC
01 ABC 02 DEF 03 GTO 01 ⋮ 10 GHI 11 JKL 12 GTO 01 ⋮ 20 LBL 01 21 MNO 22 PQR 23 STU ⋮	10 ABC 20 DEF 30 GOTO 200 ⋮ 100 GHI 110 JKL 120 GOTO 200 ⋮ 200 MNO 210 PQR 220 STU ⋮

...PROGRAM BRANCH

Im HP-28S würde die gemeinsame Sequenz MNO PQR STU... als separates Programm gespeichert werden:

```
« MNO PQR STU ... » 'COMMON' STO
```

Danach würde jeder Programmzweig das "Unterprogramm" COMMON aufrufen:

```
... ABC DEF COMMON ... GHI JKL COMMON ...
```

Der Vorteil bei der Programmierung des HP-28S liegt darin, daß jedes Programm nur einen Eingang und einen Ausgang besitzt. Dadurch wird das Schreiben eines Programms sowie das anschließende Testen relativ einfach. Wenn Sie die Einzelprogramme zu einem Hauptprogramm zusammenbauen, müssen Sie lediglich sicherstellen, daß die Unterprogramme in der von Ihnen vorgesehenen Weise zusammenarbeiten.

IF IFERR THEN ELSE END

Diese Befehle können in verschiedenen Kombinationen für bedingte Verzweigungs- und Fehlerabhandlungsstrukturen verwendet werden.

IF *Test-Anweisung* THEN *Wahr-Anweisung* END. Der Befehl THEN nimmt ein Flag vom Stack und überprüft dieses auf seinen Inhalt. Ist dieser wahr (ungleich Null), so wird die Wahr-Anweisung ausgewertet und die Ausführung des Programms nach der END Anweisung fortgesetzt. Wenn der Inhalt falsch ist (gleich Null), so wird die Ausführung des Programms bis nach END übersprungen und dann fortgesetzt. (Beachten Sie, daß eigentlich nur THEN den Flag tatsächlich benutzt—die Position von IF ist willkürlich, solange es THEN vorangestellt ist. *Test-Anweisung* IF THEN bewirkt das gleiche wie IF *Test-Anweisung* THEN.) Zum Beispiel gibt

```
IF X 0 > THEN "Positive" END
```

den String `"Positive"` zurück, wenn X eine positive reelle Zahl enthält.

IF *Test-Anweisung* THEN *Wahr-Anweisung* ELSE *Falsch-Anweisung* END. Der Befehl THEN nimmt ein Flag vom Stack und überprüft dieses auf seinen Inhalt. Ist dieser wahr (ungleich Null), so wird die Wahr-Anweisung ausgewertet und die Ausführung des Programms nach der END Anweisung fortgesetzt. Wenn der Inhalt falsch ist (gleich Null), so wird die Falsch-Anweisung ausgeführt und die Ausführung des Programms nach der END Anweisung fortgesetzt. (Beachten Sie, daß eigentlich nur THEN den Flag tatsächlich benutzt—die Position von IF ist willkürlich, solange es THEN vorangestellt ist. *Test-Anweisung* IF THEN bewirkt das gleiche wie IF *Test-Anweisung* THEN.) Zum Beispiel gibt

```
IF X 0 ≥ THEN "Positive" ELSE "Negative" END
```

den String `"Positive"` zurück, wenn X eine nicht negative reelle Zahl enthält; ansonsten wird der String `"Negative"` zurückgegeben.

IFERR *Prüf-Anweisung* THEN *Fehler-Anweisung* END. Diese Struktur wertet die *Fehler-Anweisung* aus, wenn während der Ausführung der *Prüf-Anweisung* ein Fehler auftritt.

Nach der Auswertung der *Prüf-Anweisung* werden die nachfolgenden Elemente der Anweisung normal ausgeführt, bis ein Fehler auftritt. Im Fehlerfall setzt das Programm mit der Ausführung der *Fehler-Anweisung* fort. (Die dazwischen liegenden Elemente werden übersprungen bzw. ignoriert.) Zum Beispiel addiert

```
IFERR WHILE 1 REPEAT + END THEN "OK" 1 DISP END
```

alle Zahlenwerte im Stack. Die + Funktion wird solange wiederholt, bis ein Fehler auftritt (was durch einen leeren Stack oder durch nicht zusammenpassende Objekttypen verursacht werden kann). Die *Fehler-Anweisung* bewirkt danach die Anzeige von `OK`.

Wenn Sie Fehler-Anweisungen definieren, sollten Sie daran denken, daß im Fehlerfall der Status des Stacks davon abhängig sein kann, ob LAST aktiviert ist oder nicht. Bei der Aktivierung von LAST geben Befehle, die einen Fehler verursachen, ihre Argumente in den Stack zurück; ansonsten sind die Argumente verloren.

...PROGRAM BRANCH

IFERR *Prüf-Anweisung* THEN *Fehler-Anweisung* ELSE *Normal-Anweisung* END. Diese Struktur erlaubt Ihnen die Spezifikation einer *Fehler-Anweisung*, wenn während der Ausführung einer *Prüfanweisung* ein Fehler festgestellt wird; außerdem ist für den fehlerfreien Ablauf die Angabe einer *Normal-Anweisung* möglich.

Wenn die *Prüf-Anweisung* ausgewertet wird, werden die nachfolgenden Elemente der Anweisung normal ausgeführt, solange kein Fehler auftritt.

- Im Fehlerfall wird der Rest der *Prüf-Anweisung* verworfen und es wird die *Fehler-Anweisung* ausgeführt.
- Wenn kein Fehler auftritt, folgt der Auswertung von *Prüf-Anweisung* die Auswertung von *Normal-Anweisung*.

In beiden Fällen wird die Programmausführung bis nach END fortgesetzt.

START FOR NEXT STEP IFT IFTE

***Anfang Ende* START *Schleifen-Anweisung* NEXT.** Der Befehl START nimmt zwei reelle Zahlen, *Anfang* und *Ende*, vom Stack und speichert diese als Anfangs- und Endwerte für einen Schleifenzähler. Danach wird eine Reihe von Objekten, die *Schleifen-Anweisung*, ausgewertet. Der Befehl NEXT erhöht den Schleifenzähler um 1; wenn der Schleifenzähler kleiner oder gleich *Ende* ist, wird die *Schleifen-Anweisung* nochmals ausgewertet. Dieser Prozeß wird solange wiederholt, bis der Schleifenzähler *Ende* überschreitet, wonach die Ausführung mit der NEXT folgenden Anweisung fortgesetzt wird. Zum Beispiel wird mit

```
1 10 START XYZ NEXT
```

XYZ 10 mal ausgeführt.

***Anfang Ende* START *Schleifen-Anweisung Schrittweite* STEP.** Diese Struktur ist ähnlich zu START...NEXT, außer daß STEP den Schleifenzähler um einen variablen Betrag erhöht, während mit NEXT der Zähler immer um 1 erhöht wird.

Der Befehl START nimmt zwei reelle Zahlen, *Anfang* und *Ende*, vom Stack und speichert diese als Anfangs- und Endwerte für einen Schleifenzähler. Danach wird eine Reihe von Objekten, die *Schleifen-Anweisung*, ausgewertet. Der Befehl STEP erhöht den Schleifenzähler um die reelle Zahl *Schrittweite*, die von der Ebene 1 des Stacks genommen wird.

Wenn *Schrittweite* positiv und der Schleifenzähler kleiner oder gleich *Ende* ist, dann wird die *Schleifen-Anweisung* erneut ausgeführt. Dieser Prozeß wird solange wiederholt, bis der Schleifenzähler *Ende* überschreitet, wonach die Ausführung mit der STEP folgenden Anweisung fortgesetzt wird.

Wenn *Schrittweite* negativ und der Schleifenzähler größer oder gleich *Ende* ist, dann wird die *Schleifen-Anweisung* erneut ausgeführt. Dieser Prozess wird solange wiederholt, bis der Schleifenzähler *Ende* unterschreitet, wonach die Ausführung mit der STEP folgenden Anweisung fortgesetzt wird. Zum Beispiel führt

```
10 1 START XYZ -2 STEP
```

XYZ 5 mal aus.

***Anfang Ende* FOR *Name Schleifen-Anweisung* NEXT.** Diese Struktur stellt eine bestimmte Schleife dar, in welcher der Schleifenzähler *Name* eine innerhalb der Schleife auswertbare lokale Variable ist. (Der auf FOR folgende Name ist ohne Anführungszeichen einzugeben.) Der Ablauf im Detail:

1. FOR nimmt zwei reelle Zahlen, *Anfang* und *Ende*, vom Stack. Er erzeugt die lokale Variable *Name* und speichert in ihr *Anfang* als Ausgangswert für *Name*.
2. Die Objekte unter *Schleifen-Anweisung* werden ausgewertet. Wenn *Name* innerhalb der Sequenz ausgewertet wird, dann ergibt *Name* den momentanen Wert des Schleifenzählers.
3. NEXT erhöht den Schleifenzähler um 1. Wenn danach sein Wert den Inhalt von *Ende* überschreitet, wird die Ausführung mit dem auf NEXT folgenden Objekt fortgesetzt und die lokale Variable *Name* wird gelöscht; ansonsten werden Schritt 2 und 3 wiederholt.

...PROGRAM BRANCH

Zum Beispiel bewirkt das Programm

```
1 5 FOR x x SQ NEXT
```

das Speichern der Quadrate von 1 bis 5 im Stack.

***Anfang Ende* FOR *Name Schleifenzähler Schrittweite* STEP.** Diese Struktur stellt eine bestimmte Schleife dar, in welcher der Schleifenzähler *Name* eine innerhalb der Schleife auswertbare lokale Variable ist. (Der auf FOR folgende Name ist ohne Anführungszeichen einzugeben.) Sie ist ähnlich zu FOR...NEXT, außer daß der Schleifenzähler durch einen variablen Betrag erhöht wird. Der Ablauf im Detail:

1. FOR nimmt zwei reelle Zahlen, *Anfang* und *Ende*, vom Stack. Er erzeugt die lokale Variable *Name* und speichert in ihr *Anfang* als Ausgangswert für *Name*.
2. Die Objekte unter *Schleifen-Anweisung* werden ausgewertet. Wenn *Name* innerhalb der Sequenz ausgewertet wird, dann ergibt *Name* den momentanen Wert des Schleifenzählers.
3. STEP nimmt die reelle Zahl *Schrittweite* vom Stack und erhöht den Schleifenzähler um den Inhalt von *Schrittweite*. Wenn danach der Schleifenzähler den Inhalt von *Ende* überschreitet (sofern *Schrittweite* $>$ 0) oder unterschreitet (sofern *Schrittweite* $<$ 0), dann wird die Ausführung mit dem auf STEP folgenden Objekt fortgesetzt und die lokale Variable *Name* wird gelöscht; ansonsten werden Schritt 2 und 3 wiederholt.

Zum Beispiel speichert das Programm

```
1 11 FOR x x SQ 2 STEP
```

die Quadrate der Zahlen 1, 3, 5, 7, 9 und 11 im Stack.

IFT *If-Then* Befehl

Ebene 2	Ebene 1	
Flag	*Objekt*	➧

IFT ist die Einzelbefehlsform von IF...THEN...END. IFT nimmt ein Flag von Ebene 2 und ein beliebiges Objekt von Ebene 1. Wenn der Flag wahr ist (ungleich Null), dann wird das Objekt ausgewertet; ist der Flag falsch (gleich Null), so wird das Objekt verworfen. Zum Beispiel läßt

```
X 0 > "Positive" IFT
```

den String `"Positive"` in Ebene 1, wenn X eine positive reelle Zahl enthält.

IFTE *If-Then-Else* Funktion

Ebene 3	Ebene 2	Ebene 1	
Flag	*Wahr-Objekt*	*Falsch-Objekt*	➧

IFTE ist die Einzelbefehlsform von IF...THEN...ELSE...END. IFTE nimmt ein Flag von Ebene 3 und zwei beliebige Argumente von Ebene 1 und 2. Wenn der Flag wahr ist (d.h. der Inhalt ist ungleich Null), dann wird das *Falsch-Objekt* verworfen und es wird das *Wahr-Objekt* ausgewertet. Wenn der Flag falsch ist (Inhalt gleich Null), so wird das *Wahr-Objekt* verworfen und es wird das *Falsch-Objekt* ausgewertet. Zum Beispiel läßt das Programm

```
X 0 ≥ "Positive" "Negative" IFTE
```

den String `"Positive"` im Stack, wenn X eine nicht negative reelle Zahl enthält; im umgekehrten Fall bleibt der String `"Negative"` im Stack.

IFTE kann unter Beachtung folgender Syntax auch in algebraischen Ausdrücken zur Anwendung kommen:

'IFTE(*Test-Ausdruck*, *Wahr-Ausdruck*, *Falsch-Ausdruck*)'

...PROGRAM BRANCH

Bei der Auswertung eines algebraischen Ausdrucks, der IFTE enthält, wird sein erstes Argument *Test-Ausdruck* als Flag interpretiert. Ein Wert ungleich Null führt zur Auswertung des *Wahr-Ausdrucks*, während der Wert Null die Ausführung des *Falsch-Ausdrucks* zur Folge hat. Zum Beispiel gibt der Ausdruck

```
'IFTE(X≠0,SIN(X)/X,1)'
```

den Wert für $\sin(x)/x$, auch für $x = 0$, was normalerweise die Fehlermeldung `Infinite Result` hervorruft.

DO UNTIL END WHILE REPEAT END

DO *Schleifen-Anweisung* UNTIL *Test-Anweisung* END. Diese Struktur führt die *Schleifen-Anweisung* und *Test-Anweisung* wiederholt aus, bis der von *Test-Anweisung* zurückgegebene Flag den Inhalt "wahr" annimmt (ungleich Null). Als Beispiel:

```
DO X INCX X - UNTIL .0001 < END.
```

Hierbei stellt INCX ein Programm dar, welches die Variable X um einen kleinen Betrag erhöht. INCX wird solange wiederholt ausgeführt, bis sich als Änderung für X ein kleinerer Wert als .0001 ergibt.

WHILE *Test-Anweisung* REPEAT *Schleifen-Anweisung* END. Diese Struktur führt die *Test-Anweisung* und *Schleifen-Anweisung* wiederholt aus, bis der von *Test-Anweisung* zurückgegebene Flag den Inhalt "wahr" annimmt (ungleich Null). Wird ein Wert gleich Null zurückgegeben, so wird die *Schleifen-Anweisung* übersprungen und die Ausführung nach END fortgesetzt. Die *Test-Anweisung* gibt eine reelle Zahl zurück, die von REPEAT als Flag getestet wird. Als Beispiel:

```
WHILE STRING "P" POS REPEAT REMOVEP END.
```

Hierbei stellt REMOVEP ein Programm dar, welches das Zeichen `P` aus der Variablen STRING entfernt. Die Sequenz wird solange wiederholt, bis das Zeichen `P` nicht mehr in dem String vorkommt.

PROGRAM CONTROL

SST	**HALT**	**ABORT**	**KILL**	**WAIT**	**KEY**
BEEP	**CLLCD**	**DISP**	**CLMF**	**ERRN**	**ERRM**

Das PROGRAM CONTROL Menü (■ CTRL) enthält Befehle, die es Ihnen ermöglichen, die Programmausführung auszusetzen oder interaktive Operationen während der Programmausführung auszuführen.

Anhalten der Programmausführung

Die Auswertung eines Programms bewirkt normalerweise die fortlaufende Ausführung der im Programm enthaltenen Objekte bis zum Ende des Programms. Die Befehle im PROGRAM CONTROL Menü gestatten Ihnen, den Programmablauf zu einem anderen Zeitpunkt als am Progammende zu unterbrechen oder abzubrechen:

Befehl	Bedeutung
HALT	Unterbricht die Programmausführung und gestattet die spätere Fortsetzung des Ablaufs.
ABORT	Bricht die Programmausführung ab, ohne daß das Programm später fortgesetzt werden kann.
KILL	Bricht die Programmausführung ab und beendet außerdem alle anderen unterbrochenen Programme.
WAIT	Unterbricht die Programmausführung, die dann nach Ablauf einer spezifizierten Zeitspanne automatisch wieder aufgenommen wird.

Ein *unterbrochenes* Programm ist ein Programm, dessen Ausführung in der Weise angehalten wurde, daß eine spätere Fortsetzung (Wiederaufnahme der Ausführung) an der gleichen Stelle möglich ist. Während einer Programmunterbrechung können Sie jede beliebige HP-28S Operation (außer Systemstopp, Speicher-Reset und KILL Befehl)—z.B. Dateneingaben, Ansehen von Ergebnissen, Ausführung anderer Programme, usw.—durchführen und danach die Programmausführung wieder aufnehmen.

...PROGRAM CONTROL

Der ○ Indikator zeigt an, daß ein oder mehrere Programme unterbrochen sind.

Der HALT Befehl bewirkt die Unterbrechung des Programmablaufs an der Stelle, wo HALT im Programm codiert ist. Um die Ausführung wieder aufzunehmen, können Sie:

- ■ CONT (*CONTinue*/fortsetzen) drücken, um mit dem auf HALT folgenden Objekt den Programmablauf fortzusetzen. Sie können HALT in Verbindung mit ■ CONT innerhalb eines Programms verwenden, wenn Sie z.B. die Programmausführung zum Zweck der Dateneingabe durch den Benutzer unterbrechen und danach fortsetzen möchten.
- SST (*Single-STep*/Einzelschritt—im PROGRAM CONTROL Menü) drücken, um mit dem auf HALT folgenden Objekt den Programmablauf fortzusetzen. Der wiederholte Gebrauch von SST setzt den Programmablauf Schritt für Schritt fort. Damit steht Ihnen ein leistungsstarkes "debugging" Mittel (Fehleraufspürung und -beseitigung) zur Verfügung. Sie können damit den Stack oder jeden anderen Rechnerstatus nach der Ausführung jedes einzelnen Programmschritts ansehen.

Wenn Sie keine der beiden Möglichkeiten wählen, bleibt das Programm solange unterbrochen, bis Sie KILL oder einen Systemstopp ausführen; damit wird jedes unterbrochene Programm gelöscht.

Sie können auch unterbrochene Programme "verschachteln", d.h., Sie können ein Programm, welches HALT enthält, ausführen, während die Ausführung eines anderen Programms bereits unterbrochen wurde. Wenn Sie das zweite Programm fortsetzen (■ CONT), wird die Ausführung nach Abschluß des Programms erneut angehalten. Danach ist erneut das Drücken von ■ CONT möglich, um die Fortsetzung des ersten Programms zu erreichen.

Während die Ausführung eines Programms unterbrochen ist, wirken die mit UNDO verbundenen Rücksicherungsmaßnahmen des Stacks "lokal" auf das Programm. Beziehen Sie sich für weitere Informationen über den Gebrauch von UNDO (in Zusammenhang mit unterbrochenen Programmen) auf die Beschreibung unter "MODE".

SST HALT ABORT KILL WAIT KEY

Einzelschrittanweisung

SST (*Single STep*) führt den "nächsten Schritt" eines unterbrochenen Programms aus. "Nächster Schritt" bedeutet in diesem Zusammenhang den nächsten Befehl bzw. das nächste Objekt—im Programmablauf—welches dem zuletzt ausgewerteten Befehl/Objekt folgt.

Wenn Sie SST drücken, wird der als nächstes auszuführende Befehl kurz in inverser Darstellung angezeigt und danach ausgeführt. Nach jedem Schritt wird der Stack und die Menüfelder in der normalen Weise angezeigt. Zwischen den einzelnen Schritten können Sie Rechneroperationen ausführen, ohne daß dies einen Einfluß auf das unterbrochene Programm hat. Natürlich sollten Sie bei einer Veränderung des Stacks sicherstellen, daß vor der Fortsetzung des Programms die zutreffenden Objekte im Stack gespeichert sind.

Für jede beliebige Programmschleife, die mit Hilfe von FOR...NEXT, START...NEXT, DO...UNTIL...END oder WHILE...REPEAT...END definiert wurde, erscheint der Ausgangsbefehl (FOR, START, DO oder WHILE) als Schritt nur beim ersten Durchlauf der Schleife. Bei den nachfolgenden Iterationen beginnt jede Schleife mit dem ersten Befehl/Objekt, der dem Ausgangsbefehl folgt.

Wenn Sie ein Objekt mittels SST auswerten möchten und es tritt ein Fehler auf, so bringt Sie die Einzelschrittanweisung nicht weiter. Damit haben Sie die Möglichkeit, die Fehlerursache zu beseitigen und danach SST erneut auszuführen.

Besteht der nächste Schritt aus IFERR, so bewirkt das Drücken von SST die vollständige Ausführung der IFERR...THEN...END oder IFERR...THEN...ELSE...END Struktur. Um die Schritte der Struktur einzeln zu durchlaufen, ist der Befehl HALT in die Anweisungen einzubauen.

Wird → angezeigt, so ist der Ablauf nach Drücken von SST ähnlich; es erfolgt die vollständige Ausführung der → $Name_1$ $Name_2$ ... $Name_n$ Struktur in einem Schritt. Wenn den lokalen Namen ein algebraischer Ausdruck folgt, so wird dieser im gleichen Schritt mit ausgewertet.

HALT — *Programm unterbrechen* — Befehl

HALT bewirkt die Unterbrechung der Programmausführung an der Stelle, wo sich der HALT Befehl im Programmcode befindet. HALT:

1. Schaltet den O Indikator ein.
2. Weist einen Speicherbereich für den temporär gespeicherten Stack zu, sofern UNDO aktiviert ist.
3. Gibt die Rechnersteuerung für normale Operationen an das Tastenfeld zurück.

Programme, deren Ausführung durch ▮ CONT oder SST wieder aufgenommen wurde, setzen mit der auf HALT folgenden Anweisung ihren Ablauf fort.

ABORT — *Programm abbrechen* — Befehl

ABORT beendet die Ausführung eines Programms an der Stelle, wo der ABORT Befehl sich in der Programmdefinition befindet. Eine Wiederaufnahme der Programmausführung ist nicht möglich.

KILL — *Unterbrochene Programme abbrechen* — Befehl

➡

KILL bricht das momentane sowie alle unterbrochenen Programme ab. Die Programmausführung eines abgebrochenen Programms kann nicht wieder aufgenommen werden.

WAIT — *Warten* — Befehl

Ebene 1	
x ➡	

WAIT unterbricht die Programmausführung für *x* Sekunden.

KEY — *Taste* — Befehl

	Ebene 2	Ebene 1
➡		0
➡	*"String"*	1

KEY gibt einen String zurück, der die älteste Taste, welche momentan im Tastenfeld-Puffer gespeichert wird, darstellt; dabei wird diese Taste aus dem Puffer gelöscht. Wenn der Puffer leer ist, gibt KEY ein Falsch-Flag zurück. Enthält der Tastenfeld-Puffer eine oder mehrere Tasten, löscht KEY die älteste Taste aus dem Puffer und gibt ein Wahr-Flag in Ebene 1 sowie einen String in Ebene 2 zurück. Der String "nennt" die Taste, welche aus dem Puffer gelöscht wurde.

Der Tastenfeld-Puffer des HP-28S kann bis zu 15 Tasten, welche gedrückt und noch nicht ausgeführt wurden, aufnehmen. Wenn KEY eine Taste im Puffer löscht, wird diese in einen lesbaren String konvertiert. Der String enthält die Zeichen auf der Tastenoberseite, außer für:

...PROGRAM CONTROL

Taste	String
SPACE	" "
LC	"1"
INS	"INS"
DEL	"DEL"
▲	"UP"
▼	"DOWN"
◀	"LEFT"
▶	"RIGHT"
◀✥▶	"CURSOR"
◀	"BACK"

ON behält seine Bedeutung als ATTN Taste und unterbricht die momentane Programmausführung.

Die Wirkungsweise von KEY kann durch das nachfolgende Programm veranschaulicht werden:

```
« DO UNTIL KEY END "Y" SAME ».
```

Bei der Ausführung dieses Programms wird durch Drücken von Y die Zahl 1 (wahr) in Ebene 1 zurückgegeben, während das Drücken jeder anderen Taste die Zahl 0 (falsch) in Ebene 1 bringt.

BEEP CLLCD DISP CLMF ERRN ERRM

BEEP — *Akustiksignal* — Befehl

Ebene 2	Ebene 1	
Frequenz	*Dauer*	➧

BEEP bewirkt die Ausgabe eines Akustiksignals unter der spezifizierten *Frequenz* und *Dauer*. Die Frequenz wird in Hertz (auf eine ganze Zahl aufgerundet) angegeben, während sich die Zeitdauer auf Sekunden bezieht.

Die Frequenz des Akustiksignals unterliegt der Auflösung des eingebauten Tongenerators. Die maximale Frequenz ist etwa 4400 Hz, die maximale Zeitdauer ist 1048.575 Sekunden (# FFFFF msec). Größer spezifizierte Argumente werden automatisch auf diese Maximalwerte gesetzt.

Wenn Flag 51 gesetzt ist (desaktivieren des Akustiksignals), hat die Ausführung von BEEP keine Auswirkung.

CLLCD — *Anzeige löschen* — Befehl

➧

CLLCD (*CLear LCD*) löscht die Anzeige (außer den Indikatoren) und setzt den Systemmeldungsflag, um die normale Stack- und Menüanzeige zu unterdrücken.

...PROGRAM CONTROL

DISP — *Anzeigen* — Befehl

Ebene 2	Ebene 1	
Objekt	*n*	➧

DISP (*DISPlay*) dient zum Anzeigen von *Objekt* in der *n*-ten Zeile der Anzeige (*n* ist eine reelle ganze Zahl). $n = 1$ bedeutet die oberste Anzeigezeile, $n = 4$ ist die unterste Zeile. DISP setzt den Systemmeldungsflag, um die normale Stackanzeige zu unterdrücken.

Mit DISP wird ein Objekt in der gleichen Form angezeigt wie durch die normale Darstellung in Ebene 1 bei Verwendung des Mehrzeilen-Anzeigeformats. Lediglich Strings werden abweichend von der Normalform angezeigt, da die Begrenzungszeichen " wegfallen. Wenn das Objekt zur Anzeige mehr als eine Zeile benötigt, erscheint der Anfang des Objekts in Zeile *n* und setzt sich über die nachfolgenden Zeilen fort, bis entweder das Ende des Objekts oder das untere Ende der Anzeige erreicht wird.

CLMF — *Meldungsflag löschen* — Befehl

➧	

CLMF (*CLear Message Flag*) löscht den internen Meldungsflag, der von CLLCD, DISP, PIXEL, DRAX, DRAW und DRWΣ gesetzt wird. Wird der Befehl CLMF in einem Programm nach dem letzten Auftreten von einem dieser Befehle verwendet, so bewirkt dies die Wiederanzeige des normalen Stacks, nachdem die Programmausführung abgeschlossen wurde.

ERRN *Fehlernummer* Befehl

	Ebene 1
	➧ # *n*

ERRN (*ERRor Number*) gibt einen Binärwert zurück, welcher der Fehlernummer des zuletzt aufgetretenen Fehlers entspricht. Eine Tabelle über Fehler, Fehlermeldungen und Fehlernummern des Rechners ist in Anhang A enthalten.

ERRM *Fehlermeldung* Befehl

	Ebene 1
	➧ "*Fehlermeldung*"

ERRM (*ERRor Message*) gibt einen String zurück, der die Fehlermeldung des zuletzt aufgetretenen Fehlers enthält. Eine Tabelle über Fehler, Fehlermeldungen und Fehlernummern des Rechners ist in Anhang A enthalten.

PROGRAM TEST

SF	**CF**	**FS?**	**FC?**	**FS?C**	**FC?C**
AND	**OR**	**XOR**	**NOT**	**SAME**	==
STOF	**RCLF**	**TYPE**			

Das PROGRAM TEST Menü (■ TEST) enthält neben Befehlen für logische Verknüpfungen Befehle, mit welchen Sie Flags testen und ändern können.

Die Testbefehle geben ein *Flag* als Ergebnis eines Vergleichs zwischen zwei Argumenten oder eines Benutzerflag-Tests zurück. Die Vergleichsoperatoren ≠, <, >, ≤ und ≥ sind auf dem linken Tastenfeld als Zeichen enthalten. Die restlichen Testbefehle FS?, FC?, FS?C, FC?C, SAME und == sind über das TEST-Menü abrufbar. Außerdem enthält das TEST-Menü die logischen Operatoren AND, OR, XOR und NOT, mit welchen Sie die Werte von Flags kombinieren können. Beachten Sie, daß die Funktion "=" *kein* Vergleichsoperator ist; sie definiert eine Gleichung. Beide, "==" und SAME, testen die Gleichwertigkeit von Objekten.

Tastenfeld-Funktionen

≠ ***Ungleich*** **Funktion**

Ebene 2	Ebene 1		Ebene 1
Objekt$_1$	*Objekt*$_2$	➧	*Flag*
z	'*Symbol*'	➧	'*z*≠*Symbol*'
'*Symbol*'	*z*	➧	'*Symbol*≠*z*'
'*Symbol*$_1$'	'*Symbol*$_2$'	➧	'*Symbol*$_1$≠*Symbol*$_2$'

≠ nimmt ein Objekt von Ebene 1 und Ebene 2:

- Wenn keines der Objekte ein algebraischer Ausdruck oder ein Name ist, wird in dem Fall ein Falsch-Flag (0) zurückgegeben, wenn beide Objekte vom selben Typ sind und den gleichen Wert haben; ansonsten wird ein Wahr-Flag (1) zurückgegeben. Von Listen und Programmen wird angenommen, daß sie den gleichen Wert haben, wenn ihre Objekte identisch sind.
- Wenn eines der Objekte ein algebraischer Ausdruck oder ein Name ist und das andere stellt eine Zahl, einen Namen oder einen algebraischen Ausdruck dar, dann gibt ≠ einen symbolischen Vergleichsausdruck in der Form '$Symbol_1 \neq Symbol_2$' zurück, wobei $Symbol_1$ das Objekt in Ebene 2 und $Symbol_2$ das in Ebene 1 darstellt. Das Ergebnis kann mit EVAL oder →NUM ausgewertet werden, um ein Flag zurückzugeben.

< **Kleiner als** **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
# n_1	# n_2	➧	*Flag*
"$String_1$"	"$String_2$"	➧	*Flag*
x	'*Symbol*'	➧	'*x*<*Symbol*'
'*Symbol*'	*x*	➧	'*Symbol*<*x*'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$<$Symbol_2$'

> **Größer als** **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
# n_1	# n_2	➧	*Flag*
"$String_1$"	"$String_2$"	➧	*Flag*
x	'*Symbol*'	➧	'*x*>*Symbol*'
'*Symbol*'	*x*	➧	'*Symbol*>*x*'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$>$Symbol_2$'

...PROGRAM TEST

≤ **Kleiner oder gleich** **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
# n_1	# n_2	➧	*Flag*
"$String_1$"	"$String_2$"	➧	*Flag*
x	'*Symbol*'	➧	'$x \leq Symbol$'
'*Symbol*'	*x*	➧	'$Symbol \leq x$'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1 \leq Symbol_2$'

≥ **Größer oder gleich** **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
# n_1	# n_2	➧	*Flag*
"$String_1$"	"$String_2$"	➧	*Flag*
x	'*Symbol*'	➧	'$x \geq Symbol$'
'*Symbol*'	*x*	➧	'$Symbol \geq x$'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1 \geq Symbol_2$'

Die nachstehende Beschreibung bezieht sich auf die vier vorangehenden Stackdiagramme.

Jeder der 4 Befehle $<$, $>$, $\leq$ und $\geq$ nimmt zwei Objekte vom Stack, wendet auf diese den entsprechenden logischen Vergleich an und gibt ein Flag zurück, der das Ergebnis des Vergleichs enthält. Die logische Reihenfolge für den Vergleich ist *Ebene 2 Test Ebene 1*, wobei *Test* einen der vier Operatoren darstellt. Wenn z.B. Ebene 2 eine reelle Zahl x und Ebene 1 eine reelle Zahl y enthält, dann gibt $<$ ein Wahr-Flag (1) zurück, falls x kleiner als y ist, ansonsten wird ein Falsch-Flag (0) zurückgegeben.

$<$, $>$, $\leqslant$ und $\geqslant$ lassen sich auf weniger Objekttypen als $\neq$, $==$ oder SAME anwenden, da sie eine Reihenordnung implizieren:

- Für reelle Zahlen und Binärwerte ist "kleiner als" im Sinne von "numerisch kleiner" gemeint (1 ist kleiner als 2). Bei reellen Zahlen bedeutet "kleiner als" außerdem "mehr negativ" (-2 ist kleiner als -1).
- Für Strings ist "kleiner als" im Sinne von "alphabetisch vorangehend" gemeint ("ABC" ist kleiner als "DEF"; "AAA" ist kleiner als "AAB"; "A" ist kleiner als "AA"). Prinzipiell werden Buchstaben entsprechend ihrem Zeichencode geordnet. Beachten Sie dabei, daß demzufolge "B" kleiner als "a" ist, da der Zeichencode für "B" 66 ist, während "a" als 97 codiert wird.

SF CF FS? FC? FS?C FC?C

Diese Befehlsgruppe setzt, löscht und testet die 64 Benutzerflags. In diesem Zusammenhang ist "setzen" im Sinne von "wahr machen" oder "den Wert 1 zuweisen" gemeint, während "löschen" gleichbedeutend mit "falsch machen" oder "den Wert 0 zuweisen" ist.

SF — *Flag setzen* — Befehl

Ebene 1	
n	➧

SF (*Set Flag*) setzt den Benutzerflag, der durch das reelle, ganzzahlige Argument *n* spezifiziert wurde; dabei gilt $1 \leqslant n \leqslant 64$.

CF — *Flag löschen* — Befehl

Ebene 1	
n	➧

CF (*Clear Flag*) löscht den Benutzerflag, der durch das reelle, ganzzahlige Argument *n* spezifiziert wurde; dabei gilt $1 \leqslant n \leqslant 64$.

...PROGRAM TEST

FS? ***Flag gesetzt?*** **Befehl?**

Ebene 1	Ebene 1
n	*Flag*

FS? (*Flag Set*) testet den Benutzerflag, der durch das reelle, ganzzahlige Argument *n* spezifiziert wurde; dabei gilt $1 \leqslant n \leqslant 64$. Ist der Flag gesetzt, so gibt FS? ein Wahr-Flag (1) zurück, ansonsten ein Falsch-Flag (0).

FC? ***Flag gelöscht?*** **Befehl**

Ebene 1	Ebene 1
n	*Flag*

FC? (*Flag Clear*) testet den Benutzerflag, der durch das reelle, ganzzahlige Argument *n* spezifiziert wurde; dabei gilt $1 \leqslant n \leqslant 64$. Ist der Flag gelöscht, so gibt FC? ein Wahr-Flag (1) zurück, ansonsten ein Falsch-Flag (0).

FS?C ***Flag gelöscht? Löschen*** **Befehl**

Ebene 1	Ebene 1
n	*Flag*

FS?C (*Flag Set? Clear*) testet und löscht danach den Benutzerflag, der durch das reelle, ganzzahlige Argument *n* spezifiziert wurde; dabei gilt $1 \leqslant n \leqslant 64$. Ist der Flag gesetzt, so gibt FS?C ein Wahr-Flag (1) zurück, ansonsten ein Falsch-Flag (0).

FC?C — *Flag gelöscht? Löschen* — Befehl

Ebene 1		Ebene 1
n	➧	*Flag*

FC?C (*Flag Clear? Clear*) testet und löscht danach den Benutzerflag, der durch das reelle, ganzzahlige Argument *n* spezifiziert wurde; dabei gilt $1 \leqslant n \leqslant 64$. Ist der Flag gelöscht, so gibt FC?C ein Wahr-Flag (1) zurück, ansonsten ein Falsch-Flag (0).

AND OR XOR NOT SAME ==

Die Befehle AND, OR, XOR und NOT können nicht auf Flags (reelle Zahlen oder algebraische Ausdrücke) und Binärwerte angewendet werden. Im vorangehenden Fall wirken die Befehle als logische Operatoren, die wahre und falsche Wahrheitswerte in Ergebnisflags kombinieren. Bei Binärwerten führen die Befehle logische Kombinationen der individuellen Bits der Argumente durch.

Die nachfolgende Beschreibung bezieht sich auf den Gebrauch der Befehle für reelle Argumente (Flags). Unter "BINARY" ist die Anwendung der Befehle im Zusammenhang mit Binärwerten beschrieben.

AND, OR, XOR und NOT sind für algebraische Objekte zulässig. AND und NOT haben Vorrang vor OR oder XOR. AND, OR und XOR werden innerhalb algebraischen Ausdrücken als *zwischengestellte* Operatoren angezeigt:

```
'X AND Y'  '5+X XOR Z AND Y'
```

NOT erscheint als *vorangestellter* Operator:

```
'NOT X'  'Z+NOT (A AND B)'
```

Versichern Sie sich, daß Sie bei der Eingabe der Befehle in dieser Form die Befehle durch ein Leerzeichen von anderen Befehlen oder Objekten trennen. Sie können aber auch in der Befehlszeile die Befehle in vorangestellter Form eingeben:

```
'AND(X,Y)'  'AND(XOR(X,Z),Y)'
```

...PROGRAM TEST

AND *Und* **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
x	'*Symbol*'	➧	'*x* AND *Symbol*'
'*Symbol*'	*x*	➧	'*Symbol* AND *x*'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$ AND $Symbol_2$'

AND gibt ein Flag zurück, der dem logischen AND von zwei Flags entspricht:

Erstes Argument x	Zweites Argument y	AND Ergebnis
wahr	wahr	wahr
wahr	falsch	falsch
falsch	wahr	falsch
falsch	falsch	falsch

Wenn eines der Argumente oder beide algebraische Ausdrücke sind, dann ist das Ergebnis wieder ein algebraischer Ausdruck der Form '$Symbol_1$ AND $Symbol_2$', wobei $Symbol_1$ und $Symbol_2$ die Argumente darstellen.

OR *Oder* **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
x	'*Symbol*'	➧	'*x* OR *Symbol*'
'*Symbol*'	*x*	➧	'*Symbol* OR *x*'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$ OR $Symbol_2$'

OR gibt ein Flag zurück, der dem logischen OR von zwei Flags entspricht:

Erstes Argument x	Zweites Argument y	OR Ergebnis
wahr	wahr	wahr
wahr	falsch	wahr
falsch	wahr	wahr
falsch	falsch	falsch

Wenn eines der Argumente oder beide algebraische Ausdrücke sind, dann ist das Ergebnis wieder ein algebraischer Ausdruck der Form '$Symbol_1$ OR $Symbol_2$', wobei $Symbol_1$ und $Symbol_2$ die Argumente darstellen.

XOR *Exklusives Oder* Funktion

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*Flag*
x	'*Symbol*'	➧	'*x* XOR *Symbol*'
'*Symbol*'	*x*	➧	'*Symbol* XOR *x*'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$ XOR $Symbol_2$'

XOR gibt ein Flag zurück, der dem logischen exklusiven OR von zwei Flags entspricht:

Erstes Argument x	Zweites Argument y	XOR Ergebnis
wahr	wahr	falsch
wahr	falsch	wahr
falsch	wahr	wahr
falsch	falsch	falsch

Wenn eines der Argumente oder beide algebraische Ausdrücke sind, dann ist das Ergebnis wieder ein algebraischer Ausdruck der Form '*Symbol*$_1$ XOR *Symbol*$_2$', wobei *Symbol*$_1$ und *Symbol*$_2$ die Argumente darstellen.

NOT — *Nicht* — Funktion

Ebene 1		Ebene 1
x	➧	*Flag*
'*Symbol*'	➧	'NOT *Symbol*'

NOT gibt ein Flag zurück, der die logische Umkehrung eines Flags ist:

Argument x	NOT Ergebnis
wahr	falsch
falsch	wahr

Wenn das Argument ein algebraischer Ausdruck ist, dann ist das Ergebnis ein algebraischer Ausdruck der Form 'NOT Symbol', wobei *Symbol* das Argument darstellt.

SAME — *Identisch* — Befehl

Ebene 2	Ebene 1		Ebene 1
Objekt$_1$	*Objekt*$_2$	➧	*Flag*

SAME nimmt zwei Objekte des gleichen Typs aus Ebene 1 und 2 und gibt ein Wahr-Flag (1) zurück, wenn die beiden Objekte identisch sind; ansonsten wird ein Falsch-Flag (0) zurückgegeben.

SAME hat für alle Objekte (außer algebraische Ausdrücke und Namen) die gleiche Wirkungsweise wie ==. == gibt einen symbolischen (algebraischen) Flag für diese Art von Objekten zurück.

SAME gibt einen reellwertigen Flag für alle Objekttypen zurück und darf nicht in algebraischen Ausdrücken verwendet werden.

== **Gleich** **Funktion**

Ebene 2	Ebene 1		Ebene 1
$Objekt_1$	$Objekt_2$	➧	*Flag*
z	'*Symbol*'	➧	'z==*Symbol*'
'*Symbol*'	*z*	➧	'*Symbol*==z'
'$Symbol_1$'	'$Symbol_2$'	➧	'$Symbol_1$==$Symbol_2$'

== nimmt ein Objekt aus Ebene 1 und 2:

- Wenn keines der Objekte ein algebraischer Ausdruck oder ein Name ist, wird in dem Fall ein Wahr-Flag (1) zurückgegeben, wenn beide Objekte vom selben Typ sind und den gleichen Wert haben; ansonsten wird ein Falsch-Flag (0) zurückgegeben. Von Listen und Programmen wird angenommen, daß sie den gleichen Wert haben, wenn ihre Objekte identisch sind.
- Wenn eines der Objekte ein algebraischer Ausdruck oder ein Name ist und das andere stellt eine Zahl, einen Namen oder einen algebraischen Ausdruck dar, dann gibt == einen symbolischen Vergleichsausdruck in der Form '$Symbol_1$==$Symbol_2$' zurück, wobei $Symbol_1$ das Objekt in Ebene 2 und $Symbol_2$ das in Ebene 1 darstellt. Das Ergebnis kann mit EVAL oder →NUM ausgewertet werden, um ein Flag zurückzugeben.

Für die Abfrage auf Gleichheit wird die Funktion "==" anstatt "=" verwendet, um zwischen einem logischen Vergleich (==) und einer Gleichung (=) zu unterscheiden.

STOF RCLF TYPE

STOF *Flags speichern* Befehl

Ebene 1	
# *n* ➧	

STOF (*STOre Flags*) setzt die Status der 64 Benutzerflags so, daß sie mit den Bits des Binärwerts # *n* übereinstimmen. Ein Bit mit dem Inhalt 1 setzt den entsprechenden Flag, ein Bit mit 0 löscht den Flag. Das erste (niederwertigste) Bit von # *n* entspricht Flag 1; das 64. (hochwertigste) Bit entspricht Flag 64.

Wenn # *n* weniger als 64 Bits enthält, dann wird von den nicht spezifizierten hochwertigen Bits angenommen, daß sie den Inhalt 0 haben.

RCLF *Flags zurückrufen* Befehl

	Ebene 1
➧ # *n*	

RCLF (*ReCaLl Flags*) gibt einen Binärwert # *n* von 64 Bits zurück, die die Status der 64 Benutzerflags darstellen. Flag 1 entspricht dem ersten (niederwertigsten) Bit des Binärwerts; Flag 64 wird durch das 64. (hochwertigste) Bit dargestellt.

Sie können die Status aller Benutzerflags unter Verwendung von RCLF sichern und diese später mit STOF zurückspeichern. Denken Sie daran, daß die momentane Wortlänge 64 Bits betragen muß (Voreinstellung), um alle Flags zu sichern und zurückzuspeichern. Wenn die momentane Wortlänge z.B. 32 Bits ist, gibt RCLF einen Binärwert mit nur 32 Bits zurück; beim Zurückspeichern mittels STOF werden nur die Flags 1 bis 32 korrekt berücksichtigt und die Flags 33 bis 64 werden gelöscht.

Nach einem Speicher-Reset gibt RCLF den Binärwert # 4001FFC40000000 (hexadezimal) zurückgeben, welcher der Voreinstellung für die 64 Benutzerflags entspricht.

TYPE *Typ* **Befehl**

Ebene 1	Ebene 1
Objekt	*n*

Der Befehl TYPE gibt eine reelle ganze Zahl zurück, welche den Typ eines Objekts in Ebene 1 wiedergibt. Die Objekttypen und ihre zugehörige Zahl sind in der nachstehenden Tabelle beschrieben.

Objekttypen und TYPE Zahlen

Objekt	TYPE
Reelle Zahl	0
Komplexe Zahl	1
String	2
Reell (Vektor oder Matrix)	3
Komplex (Vektor oder Matrix)	4
Liste	5
Name	6
Lokaler Name	7
Programm	8
Algebraischer Ausdruck	9
Binärwert	10

REAL

NEG	FACT	RAND	RDZ	MAXR	MINR
ABS	SIGN	MANT	XPON		
IP	FP	FLOOR	CEIL	RND	
MAX	MIN	MOD	%T		

Im HP-28S besteht eine *reelle Zahl* aus einer Gleitkommazahl, welche sich aus einer 12-stelligen Mantisse und einem 3-stelligen Exponenten im Bereich zwischen −499 und +499 zusammensetzt. Reelle Zahlen werden als Zahlenkette eingegeben und angezeigt, ohne Begrenzungszeichen oder dazwischenliegenden Leerzeichen. Numerische Zeichen beinhalten die Ziffern 0 bis 9, +, −, ein Dezimaltrennzeichen ("." oder "," in Abhängigkeit vom momentanen Modus) und den Buchstaben E, der den Beginn der Exponentendarstellung anzeigt. Das allgemeine Format für eine reelle Zahl ist

(Vorzeichen) Mantisse E *(Vorzeichen) Exponent*

Wenn Sie eine reelle Zahl eingeben, ist das Format wie folgt:

- Das *Vorzeichen der Mantisse* kann ein +, ein −, oder weggelassen werden (impliziert +).
- Die *Mantisse* kann jede Anzahl von Ziffern sein, mit einem Dezimalzeichen an einer beliebigen Stelle. Wennn Sie mehr als 12 Ziffern eintippen, wird die Mantisse auf 12 Stellen gerundet. Führende Nullen werden ignoriert, sofern ihnen Mantissenziffern ungleich Null folgen.
- Der Exponent ist optional; wenn Sie einen Exponenten mit einschließen, muß dieser durch ein "E" von der Mantisse getrennt werden.
- Das *Vorzeichen des Exponenten* kann ein +, ein −, oder weggelassen werden (impliziert +).
- Der *Exponent* muß drei oder weniger Ziffern enthalten und im Bereich zwischen 0 und 499 liegen. Führende Nullen vor dem Exponenten werden ignoriert.

Reelle Zahlen erscheinen entsprechend dem momentanen Zahlen-Anzeigeformat in der Anzeige. Prinzipiell kann es vorkommen, daß nicht alle signifikanten Ziffern einer Zahl angezeigt werden. Die vollständige 12-stellige Genauigkeit einer Zahl bleibt jedoch immer in ihrer gespeicherten Form gesichert.

Das REAL-Menü enthält Funktionen, die sich auf reelle Zahlen (und reellwertige algebraische Ausdrücke) beziehen oder besondere reelle Zahlen in den Stack übernehmen. Zusätzlich zu den Menüfunktionen sind die Funktionen % und %CH direkt über das Tastenfeld abrufbar.

Tastenfeld-Funktionen

% ***Prozent*** **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*xy*/100
x	'*Symbol*'	➧	'%(*x*, *Symbol*)'
'*Symbol*'	*x*	➧	'%(*Symbol*, *x*)'
'$Symbol_1$'	'$Symbol_2$'	➧	'%($Symbol_1$, $Symbol_2$)'

% nimmt zwei reellwertige Argumente *x* und *y* und gibt *x Prozent von y* zurück—das bedeutet *x y*/100.

%CH ***Prozentuale Änderung*** **Funktion**

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	100(*y*−*x*)/*x*
x	'*Symbol*'	➧	'%CH(*x*, *Symbol*)'
'*Symbol*'	*x*	➧	'%CH(*Symbol*, *x*)'
'$Symbol_1$'	'$Symbol_2$'	➧	'CH($Symbol_1$,$Symbol_2$)'

%CH berechnet die Zunahme (prozentual) über das reelwertige Argument *x* in Ebene 2 durch das Argument *y* in Ebene 1. Das bedeutet, %CH berechnet $100(y - x)/x$.

...REAL

π π **Funktion**

	Ebene 1
	➧ 3.14159265359
	➧ 'π'

π gibt die symbolische Konstante 'π' oder den numerischen Wert 3.14159265359, die genaueste über einen Rechner darstellbare Näherung an π, zurück. Informationen über "Symbolische Konstanten" finden Sie auf Seite 70.

e e **Funktion**

	Ebene 1
	➧ 2.71828182846
	➧ 'e'

e gibt die symbolische Konstante 'e' oder den numerischen Wert 2.71828182846, die genaueste über einen Rechner darstellbare Näherung an e, zurück. Informationen über "Symbolische Konstanten" finden Sie auf Seite 70.

NEG FACT RAND RDZ MAXR MINR

NEG ***Negieren*** **Analyt. Fkt.**

Ebene 1		Ebene 1
z	➧	$-z$
'*Symbol*'	➧	'$-$*Symbol*'

NEG gibt den negativen Wert seines Arguments zurück. Wenn keine Befehlszeile vorhanden ist, bewirkt das Drücken von [CHS] die Ausführung von NEG. Ein vollständiges Stackdiagramm für NEG erscheint unter "Arithmetik".

FACT *Fakultät (Gamma)* Funktion

Ebene 1		Ebene 1
n	➧	*n*!
x	➧	Γ(*x*+1)
'*Symbol*'	➧	'FACT(*Symbol*)'

FACT berechnet $n!$ für ein positives, ganzzahliges Argument n. Für nicht ganzzahlige Argumente x gilt FACT(x) = $\Gamma(x + 1)$, wobei $x > -1$ definiert ist mit

$$\Gamma(x + 1) = \int_0^\infty e^{-t} t^x \, dt$$

Bei Werten von $x \geq 253.1190554375$ oder bei x als negativer ganzer Zahl verursacht FACT eine `Overflow` Ausnahme; für $x \leq -254.1082426465$ verursacht FACT eine `Underflow` Ausnahme.

RAND *Zufallszahl* Befehl

	Ebene 1
➧	*x*

RAND (*RANDom number*) gibt die nächste reelle Zahl in einer Folge von Pseudo-Zufallszahlen zurück und aktualisiert den Startwert der Folge.

Der Rechner benutzt eine lineare kongruente Methode und einen Startwert, um die Zufallszahl x zu erzeugen, welche immer im Bereich $0 \leq x < 1$ liegt. Jede nachfolgende Ausführung von RAND ergibt einen Wert, dessen Berechnung über einen Startwert erfolgte, der auf dem vorangegangenen Wert von RAND basiert. Sie können den Startwert durch Verwenden von RDZ verändern.

...REAL

RDZ — *Randomize* — Befehl

Ebene 1	
x ➧	

RDZ nimmt eine reelle Zahl als Startwert für den RAND Befehl. Ist das Argument 0, so wird ein auf der momentanen Uhrzeit basierender Zufallswert als Startwert verwendet. Nach einem Speicher-Reset wird .529199358633 als Startwert benutzt.

MAXR — *Reelles Maximum* — Funktion

	Ebene 1
➧	9.99999999999E499
➧	'MAXR'

MAXR gibt die symbolische Konstante 'MAXR' oder den numerischen Wert 9.99999999999E499, die im HP-28S größte darstellbare Zahl, zurück. Weitere Informationen über "Symbolische Konstanten" finden Sie auf Seite 70.

MINR — *Reelles Minimum* — Funktion

	Ebene 1
➧	1.00000000000E-499
➧	'MINR'

MINR gibt die symbolische Konstante 'MINR' oder den numerischen Wert 1E−499, die im HP-28S kleinste positive darstellbare Zahl, zurück. Weitere Informationen über "Symbolische Konstanten" finden Sie auf Seite 70.

ABS SIGN MANT XPON

ABS *Absolutbetrag* Funktion

Ebene 1		Ebene 1
z	➧	|*z*|
⌊ Feld ⌋	➧	‖Feld‖
'*Symbol*'	➧	'ABS(*Symbol*)'

ABS gibt den Absolutbetrag seines Arguments zurück. Beziehen Sie für die Anwendung von ABS auf andere Objekte auf die Kapitel "COMPLEX" und "ARRAY". ABS kann vom HP-28S differenziert, aber nicht invertiert werden.

SIGN *Vorzeichen* Funktion

Ebene 1		Ebene 1
z_1	➧	z_2
'*Symbol*'	➧	'SIGN(*Symbol*)'

SIGN gibt das Vorzeichen seines Arguments zurück, welches mit $+1$ für positive reelle Argumente und mit -1 für negative reelle Argumente definiert ist. Beziehen Sie sich für komplexe Argumente auf das Kapitel "COMPLEX".

...REAL

MANT — *Mantisse* — Funktion

Ebene 1		Ebene 1
x	➧	*y*
'*Symbol*'	➧	'MANT(*Symbol*)'

MANT gibt die Mantisse seines Arguments zurück. Als Beispiel ergibt

`1.2E34 MANT` den Wert `1.2`.

XPON — *Exponent* — Funktion

Ebene 1		Ebene 1
x	➧	*n*
'*Symbol*'	➧	'XPON(*Symbol*)'

XPON gibt den Exponenten seines Arguments zurück. Als Beispiel ergibt

`1.2E34 XPON` den Wert `34`.

IP FP FLOOR CEIL RND

IP — *Ganzzahliger Anteil* — Funktion

Ebene 1		Ebene 1
x	➧	*n*
'*Symbol*'	➧	'IP(*Symbol*)'

IP (*Integer Part*) gibt den ganzzahligen Anteil seines Arguments zurück. Das Ergebnis besitzt das gleiche Vorzeichen wie sein Argument.

FP — *Gebrochener Anteil* — Funktion

Ebene 1		Ebene 1
x	➧	*y*
'*Symbol*'	➧	'FP(*Symbol*)'

FP (*Fractional Part*) gibt den gebrochenen Anteil seines Arguments zurück. Das Ergebnis besitzt das gleiche Vorzeichen wie sein Argument.

FLOOR — *Abwerten* — Funktion

Ebene 1		Ebene 1
x	➧	*n*
'*Symbol*'	➧	'FLOOR(*Symbol*)'

FLOOR gibt die größte ganze Zahl kleiner oder gleich dem Argument zurück. Wenn das Argument aus einer ganzen Zahl besteht, so wird diese zurückgegeben.

CEIL — *Aufwerten* — Funktion

Ebene 1		Ebene 1
x	➧	*n*
'*Symbol*'	➧	'CEIL(*Symbol*)'

CEIL (*CEILing*) gibt die kleinste ganze Zahl größer oder gleich dem Argument zurück. Wenn das Argument aus einer ganzen Zahl besteht, dann wird diese zurückgegeben.

...REAL

RND — *Runden* — Funktion

Ebene 1		Ebene 1
x	➧	*y*
'*Symbol*'	➧	'RND(*Symbol*)'

RND rundet eine Zahl entsprechend zum eingestellten Zahlen-Anzeigeformat:

- Im STD Modus erfolgt keine Rundung.
- Im *n* FIX Modus wird die Zahl auf *n* Dezimalstellen gerundet.
- Im *n* SCI oder ENG Modus wird die Zahl auf *n* + 1 signifikante Stellen gerunde.

Zahlenwerte größer oder gleich 9.5E499 werden nicht gerundet.

MAX **MIN** **MOD** **%T**

MAX — *Maximum* — Funktion

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	Max(*x*,*y*)
x	'*Symbol*'	➧	'MAX(*x*,*Symbol*)'
'*Symbol*'	*x*	➧	'MAX(*Symbol*,*x*)'
'$Symbol_1$'	'$Symbol_2$'	➧	'MAX($Symbol_1$,$Symbol_2$)'

MAX gibt den größeren (positiveren) Wert seiner zwei Argumente zurück.

MIN *Minimum* Funktion

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	Min(*x*,*y*)
x	'*Symbol*'	➧	'MIN(*x*, *Symbol*)'
'*Symbol*'	*x*	➧	'MIN(*Symbol*, *x*)'
'$Symbol_1$'	'$Symbol_2$'	➧	'MIN($Symbol_1$, $Symbol_2$)'

MIN gibt den kleineren (negativeren) Wert seiner zwei Argumente zurück.

MOD *Modulo* Funktion

Ebene 2	Ebene 1		Ebene 1
x	*y*	➧	*x* Mod *y*
x	'*Symbol*'	➧	'MOD(*x*, *Symbol*)'
'*Symbol*'	*x*	➧	'MOD(*Symbol*, *x*)'
'$Symbol_1$'	'$Symbol_2$'	➧	'MOD($Symbol_1$, $Symbol_2$)'

MOD gibt einen Restwert zurück, der sich bei der Anwendung auf zwei reellwertige Argumente x und y wie folgt definiert:

$$x \text{ mod } y = x - y \text{ floor } (x/y)$$

Mod (x, y) ist mit Periode y periodisch in x. Mod (x, y) liegt für $y > 0$ im Intervall $[0, y)$ und für $y < 0$ in $(y, 0]$.

...SOLVE

"Löser"—der interaktive numerische Lösungsprozeß

Der Löser besteht aus einer interaktiven Operation, die den Speicherungsvorgang von Werten für die Variablen einer Gleichung automatisiert und das anschließende Lösen nach einer der Variablen ermöglicht. Die allgemeine Anwendungsweise für den Löser ist folgende:

1. Verwenden Sie STEQ (*STore EQuation*) zur Bestimmung der *momentanen Gleichung.*
2. Drücken Sie SOLVR, um das Menü mit den Löser-Variablen zu aktivieren.
3. Benutzen Sie die Menütasten des SOLVR-Menüs, um die Variablenwerte einzugeben, ein erster Schätzwert für die unbekannte Variable eingeschlossen.
4. Lösen Sie die Gleichung nach der Unbekannten, indem Sie die Shift-Taste (■) und anschließend die Menütaste der gesuchten Variablen drücken.

Jeder dieser Schritte wird in den nachfolgenden Abschnitten im Detail beschrieben.

Die momentane Gleichung

Unter "momentaner Gleichung" ist die Prozedur zu verstehen, welche momentan unter der Benutzervariablen EQ gespeichert ist. Der Ausdruck *momentane Gleichung* (und die Bezeichnung EQ) wurde deshalb gewählt, um den typischen Gebrauch der Prozedur anzudeuten; sie kann aus einem algebraischen Ausdruck, einer Gleichung oder einem Programm bestehen. Ein im Zusammenhang mit dem Löser benutztes Programm muß den Anforderungen eines algebraischen Ausdrucks entsprechen, d.h. es darf keine Argumente vom Stack nehmen und nur ein Ergebnis in den Stack zurückgeben.

Sie können bei der momentanen Gleichung auch an ein "implizites" Argument für SOLVR denken (es ist auch ein Argument für DRAW). Ein implizites Argument erspart Ihnen vor jeder Anwendung von SOLVR oder DRAW das wiederholte Speichern der Prozedur im Stack.

Zum Lösen von Gleichungen und Ausdrücken (Auffinden der Nullstellen) können Sie einen Ausdruck als linke Seite einer Gleichung auffassen, deren rechte Seite gleich Null ist. Alternativ dazu kann eine Gleichung als Ausdruck interpretiert werden, indem "=" gleichwertig zu "minus" behandelt wird.

Als nächstes werden die Befehle STEQ und RCEQ beschrieben, mit welchen Sie den Inhalt von EQ speichern und zurückrufen können.

STEQ — *Gleichung speichern* — Befehl

Ebene 1	
Objekt ➧	

STEQ (*STore EQuation*) nimmt ein Objekt vom Stack und speichert es in der Variablen EQ (*EQuation*). EQ wird dazu benutzt, die *momentane Gleichung*—vom Löser und von PLOT verwendet—zu speichern, womit das Argument von STEQ normalerweise eine Prozedur sein sollte.

RCEQ — *Gleichung zurückrufen* — Befehl

	Ebene 1
➧ *Objekt*	

RCEQ (*ReCall EQuation*) gibt den in der Variablen EQ gespeicherten Inhalt zurück. Er ist gleichwertig mit `'EQ' RCL`.

...SOLVE

Aktivieren des Variablen-Menüs

Das Drücken von SOLVR aktiviert das Löser-Menü, welches von der momentanen Gleichung abgeleitet wird. Dieses *Variablen-Menü* enthält:

- Ein Menüfeld für jede unabhängige Variable innerhalb der momentanen Gleichung. Existieren mehr als 6 unabhängige Variable, so können Sie über NEXT und ■ PREV jede Gruppe mit (bis zu) sechs Menütasten aktivieren.
- Eine oder zwei Menütasten zur Auswertung der momentanen Gleichung. Wenn EQ einen algebraischen Ausdruck oder ein Programm enthält, so ist das Feld EXPR= zur Auswertung des Ausdrucks oder des Programms vorhanden. Wenn EQ eine algebraische Gleichung enthält, dann kann über die Menütasten LEFT= und RT= jede Seite der Gleichung separat ausgewertet werden.

Wie wird das Variablen-Menü konfiguriert? Eine *unabhängige* Variable innerhalb der momentanen Gleichung ist entweder eine formale Variable oder eine Variable, die ein Daten-Objekt (gewöhnlich eine reelle Zahl) enthält. Eine Variable, die eine Prozedur beinhaltet, erscheint nicht in diesem Menü. Stattdessen werden die in dieser Prozedur erscheinenden Namen als mögliche unabhängige Variable verwendet; diejenigen, welche Daten-Objekte enthalten, werden dem Variablen-Menü hinzugefügt. Dieser Prozeß wird solange fortgesetzt, bis alle unabhängigen Variablen im Menü identifiziert wurden. Das Menü wird fortlaufend aktualisiert, so daß nach dem Speichern einer Prozedur in einer der Variablen die entsprechende Variable im Menü durch die neuen unabhängigen Variablen der Prozedur ersetzt wird.

Lautet z.B. die momentane Gleichung `'A+B=C'`, so ergibt sich als Variablen-Menü

wenn A, B und C keine Prozedur enthalten. Wird jedoch `'D+E'` in C gespeichert, ergibt sich als neues Menü

(Enthält eine Variable selbst wieder eine Gleichung, so wird die letztere als ein Ausdruck behandelt, in welchem zum Zweck der Variablendefinition das = durch ein − ersetzt wurde.)

Speichern von Werten in den unabhängigen Variablen

Das Drücken einer Menütaste Name im Löser-Menü, wobei *Name* eine der unabhängigen Variablen darstellt, ist der Ausführung der Sequenz `'Name' STO` ähnlich. Dies bedeutet, Name nimmt ein Objekt vom Stack und speichert es als Wert für die Variable *Name*.

Um die Eingabe zu bestätigen, bewirkt Name auch die Anzeige von *Name*: *Objekt* in Zeile 1, wobei mit *Objekt* das vom Stack genommene Objekt gemeint ist. Der Inhalt von Zeile 1 wird durch das Drücken einer Taste wieder gelöscht.

Sie können sich den Inhalt einer Variablen zu jeder Zeit durch Drücken von ' Name, danach ■ RCL, ■ VISIT oder EVAL, wieder anzeigen lassen.

Wahl der Anfangsnäherungen

Generell können algebraische Ausdrücke und Prozeduren mehr als nur eine Nullstelle besitzen. So hat z.B. der Ausdruck $(x - 3)(x - 2)$ Nullstellen in $x = 3$ and $x = 2$. Die Nullstelle, welche vom Lösungsalgorithmus ermittelt wird, hängt vom Ausgangswert bzw. der *Anfangsnäherung* zum Auffinden der Nullstellen ab.

Sie sollten dem Lösungsalgorithmus (zum Auffinden einer Nullstelle) immer einen Anfangsnäherung vorgeben. Dieser Wert ist ein Argument, welches für den Befehl ROOT (Nullstelle) erforderlich ist. Der Löser verwendet den momentanen Wert der unbekannten Variablen als Anfangsnäherung. Besitzt die Unbekannte zum gegenwärtigen Zeitpunkt noch keinen Wert, so ordnet der Löser ihr die Näherung 0 zu, wobei jedoch nicht garantiert ist, daß diese Näherung zur Bestimmung der von Ihnen gewünschten Nullstelle führt.

...SOLVE

Sie können die Zeit zum Auffinden einer Nullstelle verkürzen oder den Lösungsalgorithmus in eine bestimmte Richtung lenken, indem Sie eine geeignete Anfangsnäherung vorgeben. Dabei kann es sich um eines der nachfolgenden Objekte handeln:

- Eine Zahl oder eine Liste, die eine Zahl enthält. Diese Zahl wird durch eine geringfügige Modifikation des kopierten Näherungswertes in zwei Anfangsnäherungen konvertiert (wird nachstehend beschrieben).
- Eine Liste mit zwei Zahlen. Diese zwei Werte definieren einen Bereich, in welchem mit dem Suchen nach einer Nullstelle begonnen wird. Wenn im vorgegebenen Bereich eine ungerade Anzahl von Nullstellen liegt (durch verschiedene Vorzeichen der Prozedurwerte angedeutet), dann findet der Lösungsalgorithmus relativ schnell eine Nullstelle zwischen den vorgegebenen Näherungen. Haben die Prozedurwerte an den vorgegebenen Näherungen das gleiche Vorzeichen, so muß der Lösungsalgorithmus nach einem Bereich suchen, in welchem eine Nullstelle liegt. Eine sinnvollge Wahl der Anfangsnäherungen trägt daher zur Verkürzunng der Ausführungszeit bei.
- Eine Liste mit drei Zahlen. In diesem Fall sollte die erste Zahl Ihre beste Näherung für die von Ihnen erwartete Nullstelle darstellen. Die anderen zwei Werte sollten diese Näherung einschließen und einen Bereich definieren, in dem mit der Suche nach einer Nullstelle begonnen werden soll. Die drei Zahlenwerte der angezeigten Liste, die Sie nach einer Unterbrechung des Lösungsalgorithmus (durch Drücken von ON) erhalten, entspricht der momentanen Anfangsnäherung in diesem Format.

Jede der oben beschriebenen Zahlen kann komplex sein; in diesem Fall werden nur die reellen Anteile verwendet.

Der beste Weg zur Wahl einer geeigneten Näherung liegt in der Abbildung der momentanen Gleichung. Die graphische Darstellung gibt Ihnen eine Idee über das grobe Verhalten der Gleichung und zeigt ihre Nullstellen auf. Bei einer Gleichung ergeben sich die Nullstellen aus den Werten der unabhängigen Variablen (Abszisse), wo die zwei Kurven der Gleichung einen Schnittpunkt bilden; bei einem algebraischen Ausdruck (oder einem Programm) ergebes sich die Nullstellen aus den Schnittpunkten des Graphen mit der Abszisse. Wenn Sie DRAW zur graphischen Darstellung der Gleichung benutzen, können Sie den Cursor an die gewünschte Nullstelle bewegen und einen oder mehrere Punkte digitalisieren. Die somit erhaltenen Koordinaten lassen sich als erste Näherungen für den Löser verwenden.

Lösen der unbekannten Variablen

Um in der momentanen Gleichung eine Lösung für die "unbekannte" Variable *Name* zu finden, drücken Sie zuerst die Shifttaste ■ und danach die Menütaste Name. Dadurch wird der Lösungsalgorithmus zur Nullstellenermittlung aktiviert, um einen numerischen Wert für die unbekannte Variable zu bestimmen. Während der Lösungsalgorithmus sich in der Ausführung befindet, erscheint die Meldung

Solving for *Name*

in der obersten Anzeigezeile. Nach Abschluß der Ausführung wird das Ergebnis in den Stack zurückgegeben und in der obersten Zeile erscheint:

Name: Ergebnis

(Das Drücken einer Taste löscht diese Zeile.) In Zeile 2 wird eine Meldung ausgegeben, welche das Ergebnis qualifiziert.

Während der Lösungsalgorithmus sich in der Ausführung befindet, haben Sie folgende Möglichkeiten:

- Drücken Sie ON, um die Lösungsroutine anzuhalten und zur normalen Stackanzeige zurückzukehren. In dieser Situation zeigt die Routine ihren momentan bestmöglichen Wert für die Nullstelle der unbekannten Variablen an und gibt eine Liste aus, welche den bestmöglichen Wert sowie zwei zusätzliche Zahlenwerte zur Spezifikation des Suchbereichs enthält. Wenn Sie die Lösungsroutine erneut starten möchten, dann drücken Sie einfach die Menütaste der unbekannten Variablen, wodurch die Liste in der Variablen gespeichert wird, und anschließend die umgeschaltete Menütaste. Durch die Verwendung der Liste als Anfangsnäherungen können Sie den Lösungsprozeß an der Stelle fortsetzen, wo Sie ihn angehalten haben.
- Drücken Sie eine beliebige andere Taste, um die Zwischenergebnisse des Lösungsalgorithmus anzuzeigen, während nach einer Nullstelle gesucht wird. Zeile 2 und 3 enthalten 2 Anfangsnäherungen, die momentan von der Lösungsroutine benutzt werden, sowie die Vorzeichen der Prozedurwerte, welche sich für die ausgewerteten Näherungen ergeben. Wenn die momentane Gleichung an einer Näherungsstelle nicht definiert ist, wird als Vorzeichen ? angezeigt.

...SOLVE

Die Zwischenergebnisse stellen die Punkte dar, wo die Lösungsroutine die Prozedurwerte untersucht. Zuerst wird nach zwei Punkten mit unterschiedlichem Vorzeichen; während dieser Phase kann der Suchbereich sich vergrößern. Wird ein Vorzeichenwechsel festgestellt, so versucht die Lösungsroutine die Einschränkung des Suchbereichs auf eine Stelle, wo der Prozedurwert gleich Null ist. Durch Beobachtung des Suchbereichs (Vergrößerung oder Verkleinerung) kann der Fortschritt der Lösungsroutine verfolgt werden.

Interpretieren der Ergebnisse

In den meisten Fällen kann die Lösungsroutine eine reelle Nullstelle ermitteln. Der Befehl ROOT gibt lediglich das Ergebnis in den Stack zurück. Der Löser gibt das Ergebnis in den Stack zurück, zeigt ein benanntes Resultat in Zeile 1 und zusätzlich eine *qualifizierende Meldung* in Zeile 2. Diese Meldung liefert einen groben Anhaltspunkt über die Natur der aufgefundenen Nullstelle:

Meldung	Bedeutung
`Zero`	Der Löser ermittelte einen Punkt, dessen Prozedurwert gleich Null ist.
`Sign Reversal`	Der Löser ermittelte zwei Punkte mit unterschiedlichen Vorzeichen, jedoch keinen Zwischenpunkt, dessen Prozedurwert Null entspricht: a) Entweder es handelt sich um benachbarte Punkte, oder b) die Prozedur ist zwischen den zwei Punkten nicht stetig. Der Löser gibt den Punkt zurück, dessen Prozedurwert näher bei Null liegt. Stellt die Prozedur eine stetige reelle Funktion dar, so ist dieser Punkt die beste Näherung an eine tatsächliche Nullstelle.
`Extremum`	Der Löser ermittelte einen Punkt, wo der Prozedurwert die Näherung an ein lokales Minimum (bei positivem Vorzeichen) oder Maximum (negatives Vorzeichen) darstellt—oder die Suche wurde an der Stelle ±9.99999999999E499 abgebrochen, da keine größeren Werte im Rechner dargestellt werden können.

Nachdem Sie über den Löser oder oder unter Verwendung von ROOT ein Ergebnis erhalten haben, sollten Sie die Ausgangsprozedur im Hinblick auf eine korrekte Interpretation der Ergebnisse auswerten. (Wenn Sie das Variablen-Menü benutzen, dann können Sie EXPR= für einen Ausdruck oder ein Programm bzw. LEFT= und RT= für eine Gleichung verwenden.) Es gibt zwei Möglichkeiten: Der Prozedurwert an der Stelle der unbekannten Variablen, die von der Lösungsroutine bestimmt wurde, ist nahe Null—oder liegt nicht in der Nähe von Null. Es bleibt Ihnen überlassen, wie nahe der Wert bei Null liegen muß, um als Nullstelle akzeptiert werden zu können.

Der beste Weg, um die Bedeutung einer Nullstelle zu verstehen, besteht in der graphischen Darstellung der Prozedur in der Umgebung der Nullstelle. Die Abbildung kann verdeutlichen, ob es sich um eine zutreffende Nullstelle oder um eine Unstetigkeit der Prozedur handelt. Die Meldungen, welche vom Löser beim Auffinden einer Nullstelle mit angegeben werden, sind nicht ganz so hilfreich wie eine graphische Abbildung.

Während der Suche nach einer Nullstelle kann der Lösungsalgorithmus die Prozedur mit Werten der unbekannten Variablen auswerten, die zu mathematischen Ausnahmefällen führen. Es wird keine Fehlermeldung angezeigt, die entsprechenden Benutzerflags für mathematische Ausnahmen werden jedoch gesetzt.

Fehler

In zwei Fällen kommt die Lösungsroutine zu einem Fehlerstatus, was zur Anzeige einer Fehlermeldung führt:

Fehlermeldung	Bedeutung
Bad Guess(es)	Wenigstens eine der beiden Anfangsnäherungen liegt außerhalb des Wertebereichs der Pozedur. Das bedeutet, die Prozedur gibt einen Fehler zurück, wenn sie an der Näherungsstelle ausgewertet wird.
Constant?	Für jeden von der Lösungsroutine getesteten Punkt ergibt sich der gleiche Prozedurwert.

...SOLVE

ROOT — *Lösungsroutine f. Nullstelle* — Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
≪Programm≫	' Name '	Anfangsnäherung	➧	Nullstelle
≪Programm≫	' Name '	{ Anfangsnäherungen }	➧	Nullstelle
' Symbol '	' Name '	Anfangsnäherung	➧	Nullstelle
' Symbol '	' Name '	{ Anfangsnäherungen }	➧	Nullstelle

ROOT benutzt eine Prozedur, einen Namen sowie entweder eine einzelne Anfangsnäherung (relle oder komplexe Zahl) oder eine Liste mit bis zu drei Näherungswerten, und gibt eine reelle Zahl als *Nullstelle* zurück. Die *Nullstelle* ist ein Wert für die Variable *Name;* sie wird von der im Rechner enthaltenen numerischen Lösungsroutine für die Nullstellen einer Funktion ermittelt. Wo es das mathematische Verhalten der Prozedur zuläßt, ist *Nullstelle* im Sinne einer mathematischen Nullstelle zu verstehen—ein Variablenwert, für welchen der Prozedurwert den numerischen Wert Null annimmmt. Beziehen Sie sich für weitere Informationen über die Ergebnisse der Lösungsroutine auf den Abschnitt "Interpretieren der Ergebnisse".

Die einzelne Anfangsnäherung bzw. die Liste der Anfangsnäherungen sind Näherungswerte für eine mögliche Nullstelle. Die Lösungsroutine erwartet diese Angaben, um innerhalb eines bestimmten Bereichs mit der Suche nach einer Nullstelle zu beginnen. Im Abschnitt "Wahl der Anfangsnäherungen" ist beschrieben, wie die Näherungswerte gewählt werden sollten.

Wenn Sie die Ausführung von ROOT durch Drücken von ON unterbrechen, dann wird die Prozedur selbst in Ebene 3, der Name in Ebene 2, und eine Liste mit den Zwischenresultaten für die unbekannte Variable in Ebene 1 zurückgegeben. Die Liste kann als Liste für Anfangsnäherungen benutzt werden, falls Sie die Lösungsroutine erneut starten möchten.

Symbolische Lösungen

ISOL *Isolieren* Befehl

Ebene 2	Ebene 1	Ebene 1
' $Symbol_1$ '	' *Name* '	➧ ' $Symbol_2$ '

ISOL gibt den Ausdruck $Symbol_2$ zurück, der die Neuanordnung seines Arguments $Symbol_1$ darstellt. Dabei wurde im Ausdruck $Symbol_2$ das erste Erscheinen der Variablen *Name* "isoliert". Wenn die Variable nur einmal innerhalb der Definition von $Symbol_1$ auftaucht, dann entspricht $Symbol_2$ einer symbolischen Nullstelle (Lösung) von $Symbol_1$. Tritt *Name* dagegen mehr als einmal auf, so ist $Symbol_2$ die rechte Seite einer Gleichung, die durch Neuanordnen und Lösen von $Symbol_1$ (zum Isolieren des ersten Auftretens von *Name* auf der linken Seite der Gleichung) erhalten wurde. Wenn es sich bei $Symbol_1$ um einen Ausdruck handelt, betrachten Sie ihn als linke Seite einer Gleichung mit $Symbol_1 = 0$.

Wenn *Name* im Argument einer Funktion innerhalb von $Symbol_1$ erscheint, dann muß diese eine *analytische Funktion* darstellen—der HP-28S muß in der Lage sein, die Umkehrfunktion der betreffenden Funktion zu berechnen. Demzufolge kann ISOL nicht IP(X) = 0 nach X lösen, da IP keine Umkehrfunktion besitzt. Befehle, für welche der HP-28S die algebraische Umkehrung durchführen kann, werden in diesem Handbuch als *analytische Funktionen* bezeichnet.

...SOLVE

QUAD *Quadratische Form* Befehl

Ebene 2	Ebene 1	Ebene 1
'$Symbol_1$'	'*Name*'	➧ '$Symbol_2$'

QUAD löst einen algebraischen Ausdruck $Symbol_1$ nach der Variablen *Name* und gibt den Ausdruck $Symbol_2$ als Lösung zurück. QUAD berechnet eine Näherung über eine Taylorreihe zweiten Grades für $Symbol_1$, um den Ausdruck in eine quadratische Form zu konvertieren (dies ist exakt, wenn in $Symbol_1$ *Name* bereits als Polynom zweiten Grades vorliegt.)

QUAD wertet $Symbol_2$ aus, bevor der Ausdruck in den Stack zurückgegeben wird. Wenn Sie eine symbolische Lösung erhalten möchten, sollten Sie jede Variable löschen, die als formale Variable in der Lösung erhalten bleiben soll.

SHOW *Variable anzeigen* Befehl

Ebene 2	Ebene 1	Ebene 1
'$Symbol_1$'	'*Name*'	➧ '$Symbol_2$'

SHOW gibt $Symbol_2$ zurück, welcher äquivalent zu $Symbol_1$ ist, außer daß alle impliziten Beziehungen zu einer Variablen *Name* explizit gemacht werden. Wenn Sie z. B.

```
'X+1' 'A' STO 'Y+2' 'B' STO
```

definieren, erhalten Sie mit

```
'A*B' 'Y' SHOW
```

den Ausdruck `'A*(Y+2)'`

und mit

```
'A*B' 'X' SHOW
```

den Ausdruck `'(X+1)*B'`.

Allgemeine Lösungen

Bei HP-28S Funktionen handelt es sich um *Funktionen* im strikten mathematischen Sinn, d.h. sie geben bei ihrer Auswertung genau ein Ergebnis zurück. Als Beispiel bedeutet dies, daß $\sqrt{4}$ immer $+2$ ergibt, nicht -2 oder ± 2. Bei anderen Funktionen wie z.B. ASIN wird ein Hauptwert zurückgegeben, entsprechend allgemeinen mathematischen Konventionen.

Dies beinhaltet allerdings, daß Paare von Funktionen, wie z.B. $\sqrt{}$ und SQ oder SIN und ASIN, nicht unbedingt die allgemeine *Umkehrrelation* darstellen, welche durch ihren Namen angedeutet wird. Betrachten Sie die Gleichung $x^2 = 2$. Wenn auf beiden Seiten die Quadratwurzel gezogen wird, erhalten Sie die "Lösungen"

$$x = +\sqrt{2} \text{ und } x = -\sqrt{2}.$$

Die HP-28S Gleichung `'X=√2'` kann nicht korrekt beide Lösungen darstellen—die $\sqrt{}$ Funktion gibt immer die positive Wurzel zurück. Die Lösung von sin $x = .5$ nach x verläuft ähnlich; es gibt eine unendliche Anzahl von Lösungen mit $x = 30° + 360n°$, wobei n eine beliebige ganze Zahl ist. Die Anwendung von ASIN auf .5 ergibt jedoch nur das einzelne Ergebnis 30°.

Der Flag für den Hauptwert einer Funktion, Benutzerflag 34, legt die Art der von ISOL und QUAD zurückgegebenen Lösungen fest. Wenn der Flag gesetzt ist, werden die Vorzeichen und freien Parameter automatisch so gewählt, daß sie die entsprechenden Hauptwerte darstellen. Ist der Flag gelöscht, so werden die Lösungen in deren vollen Allgemeingültigkeit zurückgegeben.

...SOLVE

Modus für allgemeine Lösungen

Wenn sich der Rechner im Modus für allgemeine Lösungen befindet (indem Flag 34 gelöscht ist), lösen die Befehle QUAD und ISOL Ausdrücke in voller Allgemeingültigkeit. Dazu werden bei gegebenem Anlaß zur Darstellung von beliebigen Vorzeichen bzw. ganzen Zahlen spezielle Benutzervariable eingeführt. Sie können für diese Variablen Werte in der üblichen Weise auswählen, indem Sie die gewünschten Werte in den Variablen speichern und anschließend den Ausdruck auswerten. QUAD und ISOL führen Variable auf diese Weise ein:

- Wenn ein Befehl ein Ergebnis zurückgibt, welches ein oder mehrere *beliebige Vorzeichen* enthält, dann wird das erste dieser Vorzeichen durch die Variable `s1` dargestellt, das zweite durch `s2`, und so weiter. Als Beispiel:

  `'X^2+5*X+4' 'X' QUAD` ergibt `'(-5+s1*3)/2'`.

  Die Variable `s1` steht stellvertretend für das konventionelle ± Symbol. Sie können eine der Lösungen wählen, indem Sie +1 oder −1 in `s1` speichern und danach EVAL ausführen.
- Wenn ISOL ein Ergebnis zurückgibt, welches ein oder mehrere beliebige ganze Zahlen enthält, dann wird die erste Zahl durch die Variable `n1` dargestellt, die nächste durch `n2`, und so weiter. Als Beispiel:

  `'X^4=Y' 'X' ISOL` ergibt `'EXP(2*π*i*n1/4)*Y^.25'`.

  Die exponentielle Schreibweise stellt das beliebige komplexe Vorzeichen des Ergebnisses dar; es gibt drei eindeutige Werte, n1 = 0, 1 und 2. Sie können einen der drei Werte wählen, indem Sie die geeignete Zahl in n1 speichern und danach den Ausdruck auswerten.

Eine alternatives Verfahren über das Tastenfeld zur Substitution der beliebigen Variablen in einem Ergebnis von ISOL oder QUAD besteht im Editieren des Ausdrucks unnd in der Umwandlung der beliebigen Variablen in temporäre Variable, für welche Werte vorgegeben werden. Um z.B. die negative Wurzel des oberen QUAD Beispiels zu wählen, drücken Sie ▮ EDIT, um das Ergebnis in die Befehlszeile zu kopieren; drücken Sie nun

INS −1 → `s1` ENTER.

Dadurch wird `s1` zur lokalen Variable, ihr wird der Wert −1 zugeordnet und es erfolgt die Auswertung des Ausdrucks. Diese Methode hat den Vorteil, daß die Erzeugung von "permanenten" Variablen im Benutzerspeicher (für beliebige Variable), vermieden wird.

Modus für Hauptwerte

Wenn Sie Flag 34 setzen, geben die Befehle QUAD und ISOL "Hauptwerte" für ihre Lösungen zurück. Das bedeutet:

- Beliebige Vorzeichen werden durch positive Vorzeichen ersetzt. Dies gilt für beide, das gewöhnliche $\pm$ sowie das allgemeine komplexe "Vorzeichen" exp $(2n\pi i/x)$, welches sich durch die Umkehrung von Ausdrücken der Form y^x ergibt. Im letzteren Fall wird für die beliebige ganze Zahl n der Wert 0 gewählt.
- Beliebige ganze Zahlen werden durch den Wert 0 ersetzt. Demzufolge gibt

  `'SIN(X)=Y' 'X' ISOL` den Ausdruck `'ASIN(Y)'`

  zurück, welcher immer im Bereich zwischen 0 und 180 Grad liegt.

Sie sollten beachten, daß diese Wahl der "Hauptwerte" primär dazu dient, den Ergebnisausdruck zu vereinfachen. Mathematisch gesehen sind diese Lösungen nicht besser oder schlechter als jede andere Lösung eines Ausdrucks. Wenn Sie symbolische Ergebnisse bevorzugen, um diese anschließend zu anderen Zwecken als der einfachen visuellen Ansicht auszuwerten, sollten Sie Flag 34 gelöscht haben; Sie erhalten damit Ergebnisse in vollständiger Allgemeingültigkeit.

STACK

DUP	OVER	DUP2	DROP2	ROT	LIST→
ROLLD	PICK	DUPN	DROPN	DEPTH	→LIST

Dieses Menü bietet Ihnen Befehle, über welche Sie den Inhalt des Stacks verändern können. Die am häufigsten benutzten Befehle finden Sie direkt auf dem Tastenfeld; die restlichen sind als Menütasten über das STACK-Menü verfügbar.

Als Tastenfeld-Befehle stehen Ihnen DROP, ■ SWAP, ■ ROLL und ■ CLEAR zur Verfügung.

Tastenfeld-Befehle

DROP — *Verschieben nach unten* — Befehl

Ebene 1	
Objekt	➧

DROP verschiebt alle Elemente des Stacks um eine Ebene nach unten. Das unterste Element geht dabei verloren.

Sie können das entfernte (unterste) Element zurücksichern, indem Sie LAST ausführen (sofern aktiviert).

SWAP — *Austauschen* — Befehl

Ebene 2	Ebene 1		Ebene 2	Ebene 1
$Objekt_1$	$Objekt_2$	➧	$Objekt_2$	$Objekt_1$

SWAP tauscht den Inhalt von Ebene 1 mit Ebene 2 aus.

ROLL — *Rollen* — Befehl

Ebene n+1 ... Ebene 2	Ebene 1		Ebene n ... Ebene 2	Ebene 1
$Objekt_1$ ... $Objekt_n$	n	➧	$Objekt_2$ ... $Objekt_n$	$Objekt_1$

ROLL nimmt eine ganze Zahl *n* vom Stack und "rollt" danach die ersten *n* verbleibenden Objekte des Stacks. So übernimmt z.B. 4 ROLL das Objekt von Ebene 4 in in Ebene 1.

CLEAR — *Löschen* — Befehl

Ebene n ... Ebene 1	
$Objekt_1$ ... $Objekt_n$ ➧	

CLEAR entfernt alle Objekte im Stack.

Wenn UNDO aktiviert ist, können Sie den alten Stackinhalt durch direkt auf CLEAR folgendes Drücken von ■ UNDO zurücksichern.

DUP OVER DUP2 DROP2 ROT LIST→

DUP — *Duplizieren* — Befehl

Ebene 1		Ebene 2	Ebene 1
Objekt	➧	Objekt	Objekt

DUP gibt eine Kopie des Objekts in Ebene 1 zurück. Wenn keine Befehlszeile vorhanden ist, bewirkt das Drücken von ENTER die Ausführung von DUP.

...STACK

OVER — *Zweite Ebene duplizieren* — Befehl

Ebene 2	Ebene 1		Ebene 3	Ebene 2	Ebene 1
$Objekt_1$	$Objekt_2$	➧	$Objekt_1$	$Objekt_2$	$Objekt_1$

OVER gibt eine Kopie von Ebene 2 in Ebene 1 zurück. Der seitherige Inhalt wird dabei um eine Ebene nach oben verschoben.

DUP2 — *Zwei Objekte duplizieren* — Befehl

Ebene 2	Ebene 1		Ebene 4	Ebene 3	Ebene 2	Ebene 1
$Objekt_1$	$Objekt_2$	➧	$Objekt_1$	$Objekt_2$	$Objekt_1$	$Objekt_2$

DUP2 kopiert den Inhalt der ersten zwei Stackebenen.

DROP2 — *Nach unten verschieben* — Befehl

Ebene 2	Ebene 1	
$Objekt_1$	$Objekt_2$	➧

DROP2 löscht den Inhalt der untersten 2 Stackebenen und verschiebt die verbleibenden Objekte um 2 Ebenen nach unten. Die gelöschten Objekte sind als LAST Argumente gesichert worden und können mit LAST (sofern aktiviert) zurückgespeichert werden.

ROT — *Rotieren* — Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 3	Ebene 2	Ebene 1
$Objekt_1$	$Objekt_2$	$Objekt_3$	➧	$Objekt_2$	$Objekt_3$	$Objekt_1$

ROT bewirkt das Rotieren der ersten drei Objekte im Stack; der Inhalt der dritten Stackebene erscheint danach in Ebene 1. ROT ist gleichwertig mit der Operation `3 ROLL`.

LIST→ *Liste in Stack* Befehl

Ebene 1		Ebene n+1 ... Ebene 2	Ebene 1
{ $Objekt_1$ ... $Objekt_n$ }	➧	$Objekt_1$ ... $Objekt_n$	n

LIST→ nimmt eine Liste mit n Objekten (in Ebene 1) vom Stack und speichert diese in den einzelnen Stackebenen 2 bis $n+1$. Die Zahl n wird in Ebene 1 zurückgegeben.

ROLLD PICK DUPN DROPN DEPTH →LIST

ROLLD *Nach unten rollen* Befehl

Ebene n+1 ... Ebene 2	Ebene 1		Ebene n	Ebene $n-1$ ... Ebene 1
$Objekt_1$ ... $Objekt_n$	n	➧	$Objekt_n$	$Objekt_1$ ... $Objekt_{n-1}$

ROLLD (*ROLL Down*) nimmt eine ganze Zahl n vom Stack und "rollt" danach die ersten n verbleibenden Objekte des Stacks nach unten. So übernimmt z.B. `4 ROLLD` das Objekt von Ebene 1 in in Ebene 4.

PICK *Entnehmen* Befehl

Ebene n+1 ... Ebene 2	Ebene 1		Ebene n+1 ... Ebene 2	Ebene 1
$Objekt_1$ ... $Objekt_n$	n	➧	$Objekt_1$ ... $Objekt_n$	$Objekt_1$

PICK verwendet eine ganze Zahl n aus Ebene 1 und gibt eine Kopie der n-ten Stackebene (wobei Ebene 1 nicht berücksichtigt wird) in Ebene 1 zurück. Beachten Sie, daß `1 PICK` äquivalent zu DUP und `2 PICK` äquivalent zu OVER ist.

...STACK

DUPN *N Objekte duplizieren* Befehl

Ebene $n+1$ . . . Ebene 2	Ebene 1		Ebene $2n$ . . . Ebene $n+1$	Ebene n . . . Ebene 1
$Objekt_n$ . . . $Objekt_1$	n	➧	$Objekt_n$ . . . $Objekt_1$	$Objekt_n$ . . . $Objekt_1$

DUPN verwendet eine ganze Zahl n aus Ebene 1 und gibt eine Kopie der ersten n Stackebenen in den Stack zurück (d.h. die Objekte von Ebene 2 bis Ebene $n + 1$, wenn n sich in Ebene 1 befindet).

DROPN *N Objekte nach unten verschieben* Befehl

Ebene $n+1$ . . . Ebene 2	Ebene 1		
$Objekt_1$ . . . $Objekt_n$	n	➧	

DROPN löscht die ersten $n + 1$ Objekte im Stack (die ersten n Objekte, ohne die Zahl n selbst zu berücksichtigen). N wird als Argument von LAST zu Rücksicherungszwecken gespeichert. Mit der Tastenfolge ■ UNDO können Sie die gelöschten (nach unten geschobenen) Objekte zurücksichern.

DEPTH *Tiefe* Befehl

	Ebene 1
➧	n

DEPTH gibt eine ganze Zahl n zurück, welche die Anzahl der im Stack vorhandenen Objekte spezifiziert (vor der Ausführung von DEPTH).

→LIST — *Stack in Liste* — Befehl

Ebene n+1 ... Ebene 2	Ebene 1		Ebene 1
$Objekt_1$ ... $Objekt_n$	n	➧	{ $Objekt_1$ ... $Objekt_n$ }

→LIST nimmt eine ganze Zahl n aus Ebene 1 sowie n weitere Objekte von Ebene 2 bis $n + 1$, und gibt eine Liste in Ebene 1 zurück, in welcher die n Objekte zusammengefaßt sind.

Die Ausführung der Befehlsfolge `DEPTH →LIST` faßt den gesamten Stackinhalt in einer Liste zusammen. Sie können diese dann z.B. zur späteren Rücksicherung in einer Variablen speichern.

STAT

Σ+	Σ−	NΣ	CLΣ	STOΣ	RCLΣ
TOT	MEAN	SDEV	VAR	MAXΣ	MINΣ
COLΣ	CORR	COV	LR	PRDEV	
UTPC	UTPF	UTPN	UTPT		

HP-28S Statistikbefehle beziehen sich auf Daten, welche in einer $n \times m$ Matrix—als *momentane Statistikmatrix* bezeichnet—gesammelt sind. Die momentane Statistikmatrix ist als diejenige Matrix definiert, die in der Variablen ΣDAT gespeichert ist.

ΣDAT wird automatisch erzeugt (sofern sie nicht bereits existiert), indem Sie mit der Eingabe von statistischen *Datenpunkten* über den Befehl Σ+ beginnen. Ein Datenpunkt ist ein Vektor, der aus m *Koordinatenwerten* (reelle Zahlen) besteht und als eine Zeile in der Statistikmatrix gespeichert wird. Durch die Eingabe des ersten Datenpunkts wird m (Anzahl der Spalten) für die Statistikmatrix festgelegt. Der Wert für n (Anzahl Zeilen) ergibt sich aus der eingegebenen Anzahl von Datenpunkten, wie aus der nachstehender Abbildung ersichtlich ist.

Datenpunkt	Koordinatenwert			
	1	**2**	...	***m***
1	X_{11}	X_{12}	...	X_{1m}
2	X_{21}	X_{22}	...	X_{2m}
⋮	⋮	⋮		⋮
n	X_{n1}	X_{n2}	...	X_{nm}

Bestimmte Statistikbefehle kombinieren Daten aus zwei spezifizierten Spalten der Statistikmatrix. Die Benutzervariable ΣPAR enthält eine Liste mit vier reellen Zahlen, wobei die ersten zwei die Spalten identifizieren. Sie können die Spalten mit dem Befehl COLΣ auswählen. Die letzten zwei Zahlen der Liste spezifizieren die Steigung und den Achsenabschnitt, welche durch die Ausführung des Befehls LR (lineare Regression) zuletzt berechnet wurden.

Da ΣDAT und ΣPAR gewöhnliche Variable darstellen, können Sie zusätzlich zu den Statistikbefehlen die üblichen Befehle zum Aufrufen, Ansehen oder Ändern von Variableninhalte verwenden.

Die Befehle SDEV (Standardabweichung, engl. *standard deviation*), VAR (Varianz) und COV (Kovarianz) berechnen auf Stichproben basierende Statistikwerte, d.h. von einer Datenmenge, die eine Stichprobe der Grundgesamtheit darstellt. Die Befehle werden nachstehend detailliert beschrieben. Wenn die Daten die Grundgesamtheit bilden, läßt sich die wahre Standardabweichung wie folgt ermitteln:

1. Führen Sie MEAN aus, um einen Datenpunkt für den Mittelwert zu erhalten.
2. Führen Sie Σ+ aus, um den berechneten Mittelwert als Datenpunkt zu den anderen Daten zu addieren.
3. Führen Sie SDEV, VAR oder COV aus. Das Ergebnis bezieht sich dann auf eine Grundgesamtheit.
4. Führen Sie Σ− DROP aus, um den Mittelwert wieder aus der Datenmenge zu entfernen.

Σ+ Σ− NΣ CLΣ STOΣ RCLΣ

Diese Befehle ermöglichen Ihnen die Auswahl einer Statistikmatrix sowie das Hinzufügen bzw. Löschen von Matrixelementen.

Σ+ *Sigma Plus* Befehl

Ebene 1	
x	●
$[x_1\ x_2\ \ldots\ x_m]$	●
$[[x_{11}\ x_{12}\ \ldots\ x_{1m}]$ ⋮	●
$[x_{n1}\ x_{n2}\ \ldots\ x_{nm}]]$	

Σ+ fügt einen oder mehrere Datenpunkte zur momentanen Statistikmatrix ΣDAT hinzu.

...STAT

Für eine Statistikmatrix mit m Spalten können Sie das Argument von $\Sigma+$ auf verschiedene Weise eingeben:

Eingeben eines Datenpunkts mit einem einzelnen Koordinatenwert. Das Argument für $\Sigma+$ besteht aus einer reellen Zahl.

Eingeben eines Datenpunkts mit mehreren Koordinatenwerten. Das Argument für $\Sigma+$ besteht aus einem Vektor mit m reellen Koordinatenwerten.

Eingeben von mehreren Datenpunkten. Das Argument für $\Sigma+$ besteht aus einer Matrix von n Zeilen mit m reellen Koordinatenwerten.

In jedem Fall werden die Koordinatenwerte als neue Zeilen zur momentanen Statistikmatrix unter ΣDAT hinzugefügt. Wenn ΣDAT noch nicht existiert, wird diese von $\Sigma+$ als eine $n \times m$ Matrix erzeugt und unter der Variablen ΣDAT gespeichert. Ist ΣDAT dagegen bereits vorhanden, so tritt eine Fehlersituation ein, wenn sie keine reellwertige Matrix enthält oder wenn die Anzahl der Koordinatenwerte von über $\Sigma+$ eingegebenen Datenpunkten nicht mit der Spaltenanzahl in ΣDAT übereinstimmt.

$\Sigma-$ ***Sigma Minus*** **Befehl**

	Ebene 1
➧	x
➧	$\lceil x_1\ x_2\ \ldots\ x_m \rceil$

$\Sigma-$ gibt einen Vektor mit m reellen Zahlen zurück (oder eine Zahl, falls $m = 1$), entsprechend den Koordinatenwerten des zuletzt über $\Sigma+$ eingegebenen Datenpunkts für ΣDAT. Die letzte Zeile der Statistikmatrix wird gelöscht.

Der von $\Sigma-$ zurückgegebene Vektor kann editiert oder ersetzt und über den Befehl $\Sigma+$ in die Statistikmatrix zurückgespeichert werden.

NΣ *Sigma N* Befehl

	Ebene 1
➧	*n*

NΣ gibt die Anzahl der Datenpunkte, welche in die Statistikmatrix ΣDAT eingegeben wurden, zurück. Die Anzahl der Datenpunkte entspricht der Anzahl von Matrixzeilen.

CLΣ *Sigma löschen* Befehl

➧	

CLΣ (*CLear* Σ) löscht die Statistikmatrix, indem die Variable ΣDAT gelöscht wird.

STOΣ *Sigma speichern* Befehl

Ebene 1	
⟦ *Matrix* ⟧ ➧	

STOΣ (*STOre* Σ) nimmt eine Matrix vom Stack und speichert sie in der Variablen ΣDAT.

RCLΣ *Sigma zurückrufen* Befehl

	Ebene 1
➧	*Objekt*

RCLΣ (*ReCaLl* Σ) gibt den momentanen Inhalt der Variablen ΣDAT in den Stack zurück. RCLΣ ist äquivalent zu der Befehlsfolge `'ΣDAT' RCL`.

...STAT

TOT MEAN SDEV VAR MAXΣ MINΣ

Diese Befehle ermöglichen elementare Statistikberechnungen für die Daten jeder Spalte der momentanen Statistikmatrix.

TOT *Total* **Befehl**

	Ebene 1
➧	x
➧	$[x_1\ x_2\ \ldots\ x_m]$

TOT berechnet die Summe jeder der m Spalten mit Koordinatenwerten in der Statistikmatrix ΣDAT. Die Summen werden als Vektor mit m reellen Zahlen zurückgegeben (oder als einzelne reelle Zahl, wenn $m = 1$).

MEAN *Mittelwert* **Befehl**

	Ebene 1
➧	x
➧	$[x_1\ x_2\ \ldots\ x_m]$

MEAN berechnet den Mittelwert jeder der m Spalten von Koordinatenwerten in der Statistikmatrix ΣDAT und gibt den Mittelwert als Vektor mit m reellen Zahlen zurück (oder eine einzelne reelle Zahl, falls $m = 1$). Der Mittelwert wird über die Formel

$$\text{Mittelwert} = \sum_{i=1}^{n} x_i/n$$

berechnet, wobei mit x_i der i-te Koordinatenwert einer Spalte und mit n die Anzahl der Datenpunkte gemeint ist.

SDEV — *Standardabweichung* — Befehl

	Ebene 1
➧	x
➧	$[x_1\ x_2\ \ldots\ x_m]$

SDEV (*Standard DEViation*) berechnet die *Standardabweichung einer Stichprobe* aus jeder der m Spalten in der momentanen Statistikmatrix. Die Standardabweichung wird als Vektor mit m reellen Zahlen zurückgegeben (oder als einzelne Zahl, falls $m = 1$). Die Standardabweichung ergibt sich aufgrund der Formel

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

wobei mit x_i der i-te Koordinatenwert einer Spalte, mit $\bar{x}$ der Mittelwert von den jeweiligen Spaltenwerten, und mit n die Anzahl der Datenpunkten gemeint ist.

VAR — *Varianz* — Befehl

	Ebene 1
➧	x
➧	$[x_1\ x_2\ \ldots\ x_m]$

VAR berechnet die *Varianz einer Stichprobe* aus jeder der m Spalten in der momentanen Statistikmatrix. Die Varianz wird als Vektor mit m reellen Zahlen zurückgegeben (oder als einzelne Zahl, falls $m = 1$). Die Varianz ergibt sich aufgrund der Formel

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

wobei mit x_i der i-te Koordinatenwert einer Spalte, mit $\bar{x}$ der Mittelwert für die Werte der Spalte, und mit n die Anzahl von Datenpunkten gemeint ist.

...STAT

MAXΣ *Maximum Sigma* Befehl

	Ebene 1
➧	X
➧	$[x_1\ x_2\ \ldots\ x_m]$

MAXΣ ermittelt das Maximum aller Koordinatenwerte aus jeder der m Spalten in der momentanen Statistikmatrix. Die Maxima werden als Vektor mit m reellen Zahlen zurückgegeben (oder als einzelne Zahl, falls $m = 1$).

MINΣ *Minimum Sigma* Befehl

	Ebene 1
➧	X
➧	$[x_1\ x_2\ \ldots\ x_m]$

MINΣ ermittelt das Minimum aller Koordinatenwerte aus jeder der m Spalten in der momentanen Statistikmatrix. Die Minima werden als Vektor mit m reellen Zahlen zurückgegeben (oder als einzelne Zahl, falls $m = 1$).

COLΣ **CORR** **COV** **LR** **PREDV**

COLΣ *Spalten Sigma* Befehl

Ebene 2	Ebene 1	
n_1	n_2	➧

COLΣ nimmt zwei Spaltennummern, n_1 und n_2, vom Stack und speichert sie als die ersten Objekte in einer Liste unter der Variablen ΣPAR. Die Zahlen identifizieren Spalten in der momentanen Statistikmatrix ΣDAT und werden von Statistikbefehlen benutzt, welche mit Spaltenpaaren arbeiten. Der Wert n_1 legt die Spalte fest, die der unabhängigen Variablen für LR oder der horizontalen Koordinate für DRWΣ bzw. SCLΣ entspricht. n_2 legt die Spalte für die abhängige Variable oder die vertikale Koordinate fest. Für CORR und COV ist die Reihenfolge der Spaltennummern ohne Bedeutung.

Wird einer der Zwei-Spalten Befehle ausgeführt, ohne daß die Variable ΣPAR bereits existiert, so wird diese automatisch unter der Verwendung von $n_1 = 1$ und $n_2 = 2$ als Vorgabewerte erzeugt.

CORR *Korrelation* Befehl

	Ebene 1
	➡ *Korrelation*

CORR gibt die Korrelation von Koordinatenwerten zwischen zwei Spalten der momentanen Statistikmatrix zurück. Die Spalten sind durch die ersten zwei Elemente von ΣPAR (Vorgabewerte sind 1 und 2) spezifiziert. Die Korrelation wird nach der Formel

$$\frac{\sum_{i=1}^{n} (x_{in_1} - \bar{x}_{n_1})(x_{in_2} - \bar{x}_{n_2})}{\sqrt{\sum_{i=1}^{n} (x_{in_1} - \bar{x}_{n_1})^2 \sum_{i=1}^{n} (x_{in_2} - \bar{x}_{n_2})^2}}$$

berechnet, wobei mit x_{in_1} der i-te Koordinatenwert in Spalte n_1, mit x_{in_2} der i-te Koordinatenwert in Spalte n_2, mit $\bar{x}_{n_1}$ der Mittelwert der Daten aus Spalte n_1, mit $\bar{x}_{n_2}$ der Mittelwert der Daten aus Spalte n_2 und mit n die Anzahl der Datenpunkte gemeint ist.

...STAT

COV *Kovarianz* Befehl

	Ebene 1
	➡ *Kovarianz*

COV berechnet die *Kovarianz einer Stichprobe* aus allen m Spalten in der momentanen Statistikmatrix. Die Spalten werden durch die ersten zwei Elemente der Variablen ΣPAR (Vorgabewerte 1 und 2) spezifiziert. Die Kovarianz berechnet sich aufgrund der Formel

$$\frac{1}{n-1} \sum_{i=1}^{n} (x_{in_1} - \bar{x}_{n_1})(x_{in_2} - \bar{x}_{n_2})$$

wobei mit x_{in_1} der i-te Koordinatenwert in Spalte n_1, mit x_{in_2} der i-te Koordinatenwert in Spalte n_2, mit $\bar{x}_{n_1}$ der Mittelwert der Daten aus Spalte n_1, mit $\bar{x}_{n_2}$ der Mittelwert der Daten aus Spalte n_2 und mit n die Anzahl der Datenpunkte gemeint ist.

LR *Lineare Regression* Befehl

	Ebene 2	Ebene 1
	➡ *Achsenabschnitt*	*Steigung*

LR berechnet die lineare Regression von einer Matrixspalte mit abhängigen Daten auf eine Spalte mit unabhängigen Daten. Die unabhängige und abhängige Matrixspalte wird durch die ersten zwei Elemente in ΣPAR (Vorgabewert ist 1 und 2) spezifiziert.

Der *Achsenabschnitt* und die *Steigung* der Regressionsgeraden werden in Ebene 2 und 1 des Stacks zurückgegeben. LR speichert diese Regressionskoeffizienten als dritten (Achsenabschnitt) und vierten (Steigung) Wert in der Liste ΣPAR.

PREDV *Linearer Schätzwert* Befehl

Ebene 1	Ebene 1
x ➡	*Linearer Schätzwert*

PREDV (*PREDicted Value*) berechnet einen linearen Schätz- bzw. Vorhersagewert über die Angabe eines bekannten Werts für x und unter Verwendung der zuletzt errechneten linearen Regressionskoeffizienten, die durch LR in der Variablen ΣPAR gespeichert wurden:

$$\textit{Linearer Schätzwert} = (x \times \textit{Steigung}) + \textit{Achsenabschnitt}$$

Die Koeffizienten *Achsenabschnitt* und *Steigung* werden durch LR als drittes und viertes Element in der Variablen ΣPAR gespeichert. Wenn Sie PREDV ohne die vorherige Ausführung von LR anwenden, wird als Vorgabewert für die Koeffizienten Null verwendet, wodurch Sie als Ergebnis von PRDEV Null erhalten.

UTPC UTPF UTPN UTPT

Ihr Rechner bietet Ihnen vier Befehle für die *rechtsseitige Wahrscheinlichkeit*, welche Sie zur Bestimmung der Signifikanz von statistischen Tests verwenden können. Die rechtsseitige Wahrscheinlichkeitsfunktion einer Zufallsvariablen X ist die Wahrscheinlichkeit, daß X größer als eine Zahl x ist, was äquivalent zu $1 - F(x)$ ist, wobei $F(x)$ die Verteilungsfunktion von X darstellt.

Die Umkehrungen von Verteilungsfunktionen sind hilfreich bei der Bildung von Konfidenzintervallen. Das Argument einer inversen rechtsseitigen Wahrscheinlichkeitsfunktion entspricht einem Wert zwischen 0 und 1; wenn das Argument in Prozent ausgedrückt wird, dann werden die Werte der Umkehrfunktion *Perzentile* genannnt. Z.B. entspricht dem 90. Perzentil einer Verteilung die Zahl x, für welche die Wahrscheinlichkeit von $100\% - 90\% = 10\%$ besteht, daß die Zufallsvariable X größer als x ist.

...STAT

Sie können den Löser benutzen, um die Umkehrfunktion für die rechtsseitige Wahrscheinlichkeit zu erhalten. Nehmen Sie an, Sie möchten ein Perzentil der Normalverteilung bestimmen. Es sei

$$P = \text{Perzentil}/100$$

$$M = \text{Mittelwert der Verteilung}$$

$$V = \text{Varianz}$$

$$X = \text{Zufallsvariable}$$

UTPN (nachfolgend beschrieben) errechnet die rechtsseitige Wahrscheinlichkeit einer Normalverteilung. Um die Gleichung

$$1 - P = \text{utpn}\ (M, V, X),$$

nach X zu lösen, erzeugen Sie das Programm

```
« 1 P - M V X UTPN - »
```

und speichern es durch Drücken von [SOLV] [STEQ] als momentane Gleichung. Drücken Sie anschließend [SOLVR], um das Löser-Menü anzuzeigen:

[P] [M] [V] [X] [EXPR=] [ ].

Verwenden Sie eine Normalverteilung mit $M = 0$, $V = 1$:

0 [M] 1 [V].

Berechnen Sie nun das 95. Perzentil:

.95 [P] [ ] [X]

womit Sie als Ergebnis für $X = 1.6449$ erhalten.

UTPC — *Rechtsseitige Chi-Quadrat-Verteilung* — Befehl

Ebene 2	Ebene 1		Ebene 1
n	*x*	➧	*utpc(n, x)*

UTPC (*Upper-Tail Probability Chi-square Distribution*) berechnet die Wahrscheinlichkeit utpc(*n, x*) daß eine Chi-Quadrat Zufallsvariable größer als *x* ist, wobei *n* die Freiheitsgrade der Verteilung angibt. *n* muß eine positive ganze Zahl sein.

UTPF — *Rechtsseitige F-Verteilung* — Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
n_1	n_2	*x*	➧	*utpf*(n_1, n_2, *x*)

UTPF (*Upper-Tail Probability F Distribution*) berechnet die Wahrscheinlichkeit utpf(n_1, n_2, *x*), daß eine F-verteilte Zufallsvariable größer als die Zahl *x* ist, wobei n_1 und n_2 die Freiheitsgrade von Zähler und Nenner für die F-Verteilung darstellen. n_1 und n_2 müssen positive ganze Zahlen sein.

UTPN — *Rechtsseitige Normalverteilung* — Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
m	*v*	*x*	➧	*utpn(m, v, x)*

UTPN (*Upper-Tail Probability Normal Distribution*) berechnet die Wahrscheinlichkeit utpn(*m, v, x*), daß eine normalverteilte Zufallsvariable größer als *x* ist, wobei *m* und *v* den Mittelwert bzw. die Varianz der Normalverteilung darstellen. *v* muß eine nicht-negative Zahl sein.

...STAT

UTPT — *Rechtsseitige t-Verteilung* — Befehl

Ebene 2	Ebene 1		Ebene 1
n	*x*	➧	*utpt(n, x)*

UTPT (*Upper-Tail Probability t-Distribution*) berechnet die Wahrscheinlichkeit utpt(*n*, *x*), daß eine t-verteilte Zufallsvariable größer als *x* ist, wobei *n* die Freiheitsgrade der Verteilung angibt. *n* muß eine positive ganze Zahl sein.

STORE

STO+	STO−	STO*	STO/	SNEG	SINV
SCONJ					

Das STORE-Menü (*Speicher-Menü*) enthält Befehle zur Anwendung von Speicherarithmetik. Im einzelnen ist damit die Addition, Subtraktion, Multiplikation, Division, Inversion, Negation und die konjugiert komplexe Darstellung von reellen und komplexen Zahlen bzw. Feldern gemeint, ohne daß dabei der jeweilige Variableninhalt in den Stack geholt werden muß. Der STORE Befehl bietet, neben der Minimierung von Tastenoperationen in vielen Fällen, eine "vor Ort" Methode zur Modifikation eines Feldinhalts, was im Gegensatz zur Vorgehensweise über den Stack weniger Speicherbereich erfordert.

Die Anwendung der Speicherarithmetik ist auf gewöhnliche Variable beschränkt—Sie können keine lokalen Variablen als Argument für einen im STORE-Menü enthaltenen Befehl verwenden.

STO+ STO− STO* STO/ SNEG SINV

STO+ *Addieren und Speichern* Befehl

Ebene 2	Ebene 1	
z	*'Name'*	➧
'Name'	*z*	➧
[Feld]	*'Name'*	➧
'Name'	*[Feld]*	➧

...STORE

STO+ addiert eine Zahl oder ein Feld zum Inhalt der Variablen. Der Variablenname und die Zahl bzw. das Feld können in beliebiger Reihenfolge im Stack enthalten sein.

Das Objekt im Stack und das in der Variablen gespeicherte Objekt müssen aufeinander abgestimmt sein—Sie können jede beliebige Kombination von reellen und komplexen Zahlen (oder zulässigen Feldern) addieren.

STO− *Subtrahieren und Speichern* Befehl

Ebene 2	Ebene 1	
z	*'Name'*	➡
'Name'	*z*	➡
⌊Feld⌋	*'Name'*	➡
'Name'	*⌊Feld⌋*	➡

STO− berechnet die Differenz zweier Zahlen oder Felder. Ein Objekt wird vom Stack genommen, während das zweite durch einen im Stack enthaltenen Variablennamen spezifiziert ist. Die berechnete Differenz wird als neuer Inhalt in der Variablen gespeichert.

Das Ergebnis hängt von der Reihenfolge der Argumente ab:

- Wenn *Name* sich in Ebene 1 befindet, wird die Differenz von
  (Wert in Ebene 2) − (Inhalt von *Name*)
  zum neuen Inhalt von *Name*.
- Wenn *Name* sich in Ebene 2 befindet, wird die Differenz von
  (Inhalt von *Name*) − (Wert in Ebene 1)
  zum neuen Inhalt von *Name*.

Das Objekt im Stack und das in der Variablen gespeicherte Objekt müssen aufeinander abgestimmt sein—Sie können jede beliebige Kombination von reellen und komplexen Zahlen (oder zulässigen Feldern) subtrahieren.

STO∗ *Multiplizieren und Speichern* Befehl

Ebene 2	Ebene 1	
z	'*Name*'	➧
'*Name*'	*z*	➧
⌊*Feld*⌋	'*Name*'	➧
'*Name*'	⌊*Feld*⌋	➧

STO∗ multipliziert den Inhalt einer Variablen mit einer Zahl oder einem Feld. Wenn zwei Zahlen oder eine Zahl und ein Feld multipliziert werden, ist die Reihenfolge der Argumente im Stack ohne Bedeutung. Wenn zwei Felder multipliziert werden sollen, hängt das Ergebnis von der Reihenfolge der Argumente ab:

- Wenn *Name* sich in Ebene 1 befindet, wird das Produkt

  (Feld in Ebene 2) × (Feld in *Name*)

  zum neuen Inhalt von *Name*.
- Wenn *Name* sich in Ebene 2 befindet, wird das Produkt von

  (Feld in *Name*) × (Feld in Ebene 1)

  zum neuen Inhalt von *Name*.

Die Felder müssen für die Multiplikation zulässig sein.

STO/ *Dividieren und Speichern* Befehl

Ebene 2	Ebene 1	
z	'*Name*'	➧
'*Name*'	*z*	➧
⌊*Feld*⌋	'*Name*'	➧
'*Name*'	⌊*Feld*⌋	➧

STO/ berechnet den Quotienten von zwei Zahlen oder Feldern. Dabei wird ein Objekt vom Stack genommen, während das zweite durch einen Variablennamen im Stack spezifiziert ist. Der resultierende Quotient wird als neuer Inhalt der Variablen gespeichert.

Das Ergebnis hängt von der Reihenfolge der Argumente ab:

- Wenn *Name* sich in Ebene 1 befindet, wird der Quotient von
  (Wert in Ebene 2) / (Inhalt von *Name*)
  zum neuen Inhalt von *Name*.
- Wenn *Name* sich in Ebene 2 befindet, wird der Quotient von
  (Inhalt von *Name*) / (Wert in Ebene 1)
  zum neuen Inhalt von *Name*.

Das Objekt im Stack und das in der Variablen gespeicherte Objekt müssen für die Division aufeinander abgestimmt sein. Im einzelnen bedeutet dies, daß—wenn es sich um zwei Felder handelt—der Divisor (Ebene 1) eine quadratische Matrix sein muß, und der Dividend (Ebene 2) muß die selbe Anzahl Spalten wie der Divisor haben.

SNEG — *Negieren und Speichern* — Befehl

Ebene 1	
' *Name* ' ➧	

SNEG (*Store NEGate*) negiert den Inhalt der im Stack spezifizierten Variablen, wobei das Ergebnis deren ursprünglichen Inhalt ersetzt. Die Variable kann eine reelle bzw. komplexe Zahl oder ein Feld enthalten.

SINV — *Invertieren und Speichern* — Befehl

Ebene 1	
' *Name* ' ➧	

SINV (*Store INVert*) invertiert den Inhalt der im Stack spezifizierten Variablen, wobei das Ergebnis deren ursprünglichen Inhalt ersetzt. Die Variable kann eine reelle bzw. komplexe Zahl oder eine quadratische Matrix enthalten.

SCONJ

SCONJ ***Konjugieren und Speichern*** **Befehl**

Ebene 1	
'Name' ➡	

SCONJ (*Store CONJugate*) konjugiert den Inhalt der im Stack spezifizierten Variablen, wobei das Ergebnis deren ursprünglichen Inhalt ersetzt. Die Variable kann eine reelle bzw. komplexe Zahl oder ein Feld enthalten.

STRING

→STR	STR→	CHR	NUM	POS	DISP
SUB	SIZE				

Ein *String*-Objekt besteht aus einer Zeichensequenz, welche von den Anführungszeichen " am Anfang und am Ende abgeschlossen ist. Jedes durch den HP-28S darstellbare Zeichen kann in einem String enthalten sein, die Objekt-Begrenzungszeichen
( ) [ ] { } # "
' » « eingeschlossen. Zeichen, welche nicht direkt über die Tastatur verfügbar sind, können mit Hilfe des Befehls CHR eingegeben werden.

Obwohl das Zeichen " in einem String enthalten sein kann (unter Verwendung von CHR und +), läßt sich ein String, der ein " enthält, nicht auf die übliche Weise editieren. Die Ursache dafür liegt darin, daß ENTER die Abstimmung auf Paare von " in der Befehlszeile versucht—zusätzliche " innerhalb eines Strings bewirken die Aufteilung des Strings in zwei oder mehrere Strings, welche keine weiteren " enthalten.

Strings werden hauptsächlich zu Anzeigezwecken benutzt—Aufforderungen zur Eingabe, Benennung von Ergebnissen, usw. Die im STRING-Menü enthaltenen Befehle ermöglichen Ihnen einfache String- und Zeichenoperationen. Allerdings stehen Ihnen mit den Befehlen →STR und STR→ zwei wichtige Applikationen für Strings zur Verfügung. Mit diesen Befehlen lassen sich beliebige Objekte (oder Folge von Objekten) in und aus String-Formen konvertieren. In vielen Fällen läßt sich damit der belegte Speicherplatz reduzieren, da die normale Speicherform für ein Objekt mehr Speicherplatz als der entsprechende String benötigt. Sie können Objekte als Strings in Variablen speichern und diese nur bei jeweiligem Bedarf in die Normalform konvertieren. Beziehen Sie sich für weitere Informationen darüber auf die nachfolgende Beschreibung über →STR und STR→.

Tastenfeld-Funktion

+ ***Addieren*** **Analyt. Fkt.**

Ebene 2	Ebene 1		Ebene 1
"*String*$_1$"	"*String*$_2$"	➡	"*String*$_1$ *String*$_2$"

+ verkettet die Zeichen des Strings in Ebene 1 mit den Zeichen des Strings in Ebene 2 und erzeugt einen neuen String als Ergebnis.

→STR **STR→** **CHR** **NUM** **POS** **DISP**

→STR ***Objekt in String*** **Befehl**

Ebene 1		Ebene 1
Objekt	➡	"*String*"

→STR konvertiert ein beliebiges Objekt in einen String. Der resultierende String ist im wesentlichen gleich mit der angezeigten Form des Objekts, die Sie erhalten würden, wenn sich das Objekt in Ebene 1 befindet und der Mehrzeilen-Modus aktiv ist:

- Der Ergebnisstring schließt das ganze Objekt ein, auch wenn die angezeigte Form des Objekts zu groß ist, um in die Anzeige zu passen.
- Wenn das Objekt in zwei oder mehr Zeilen angezeigt wird, enthält der Ergebnisstring das "Newline"-Zeichen (Zeichen 10) am Ende jeder Zeile. Die Zeichen werden als das Standardzeichen ▪ angezeigt.

- Zahlen werden entsprechend dem momentanen Anzeigeformat für Zahlen (STD, FIX, SCI oder ENG) und der Zahlenbasis (DEC, BIN, OCT oder HEX) sowie der Wortlänge angezeigt. Die interne Darstellung der Zahl in vollständiger Genauigkeit ist nicht notwendigerweise im Ergebnisstring enthalten. Wenn Sie die Erhaltung der vollständigen Genauigkeit für →STR sicherstellen möchten, so müssen Sie entweder den STD-Modus oder eine Wortlänge von 64 Bits spezifizieren (oder beides), bevor Sie →STR ausführen.
- Handelt es sich bei dem Objekt bereits um einen String, dann gibt →STR diesen String zurück.

Sie können →STR zum Erzeugen spezieller Anzeigen verwenden, wenn Sie z.B. die Ausgabe eines Programms oder bestimmte Eingabeaufforderungen benennen möchten. So bewirkt als Beispiel die Sequenz

```
"Ergebnis = " SWAP →STR + 1 DISP
```

die Anzeige von `Ergebnis =` *Objekt* in Zeile 1; bei *Objekt* handelt es sich um die String-Form eines Objekts, welches aus Ebene 1 genommen wurde.

STR→ ***String in Objekt*** **Befehl**

Ebene 1	
"String" ➡	

STR→ ist eine Befehlsform für ENTER. Die Zeichen des String-Arguments werden analysiert und als Inhalt der Befehlszeile ausgewertet. Der String kann dabei ein einzelnes Objekt definieren oder eine Folge von Objekten, welche dann wie ein Programm ausgewertet werden.

STR→ kann auch zum Zurückspeichern von Objekten, die durch →STR in Strings konvertiert wurden, verwendet werden. Die Kombination →STR STR→ läßt die Objekte unverändert, außer daß →STR Zahlen entsprechend dem Anzeigeformat für Zahlen, der jeweiligen Zahlenbasis und der Wortlänge konvertiert. STR→ reproduziert eine Zahl nur bis zu der Genauigkeit, wie sie in der String-Form verwendet wurde.

CHR — *Zeichen* — Befehl

Ebene 1	Ebene 1
n ➡	*"String"*

CHR (*CHaracteR*) gibt einen Ein-Zeichen-String zurück, der das dem Zeichencode *n* (aus Ebene 1) entsprechende HP-28S Zeichen enthält. Das Standardzeichen ▪ steht für alle Zeichen, die nicht im Zeichensatz des HP-28S enthalten sind.

Der Zeichencode 0 wird für spezielle Zwecke in der Befehlszeile benutzt. Sie können das entsprechende Zeichen unter Verwendung von CHR in Strings verwenden; der Versuch, diese Zeile zu editieren, verursacht die Ausgabe der Fehlermeldung

```
Can't Edit CHR(0).
```

NUM — *Zeichencode* — Befehl

Ebene 1	Ebene 1
"String" ➡	*n*

NUM (*NUMber*) gibt den Zeichencode für das Zeichen zurück, welches am Anfang des Strings steht.

Die nachfolgende Tabelle zeigt die Beziehung zwischen den Zeichencodes (Ergebnisse von NUM, Argumente für CHR) und den zugehörigen Zeichen (Ergebnisse von CHR, Argumente für NUM). Für die Zeichencodes 0 bis 147 zeigt die Tabelle die Zeichen, wie sie im String dargestellt werden; für die Codes 148 bis 255 erfolgt die Darstellung in der Weise, wie die Zeichen über den Drucker HP 82240A ausgegeben werden. Ihr HP-28S zeigt für diese Zeichen das Standardzeichen ▪ an.

...STRING

Zeichencodes (0–127)

NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR
0	▪	32		64	@	96	`
1	▪	33	!	65	A	97	a
2	▪	34	"	66	B	98	b
3	▪	35	#	67	C	99	c
4	▪	36	$	68	D	100	d
5	▪	37	%	69	E	101	e
6	▪	38	&	70	F	102	f
7	▪	39	'	71	G	103	g
8	▪	40	(	72	H	104	h
9	▪	41	)	73	I	105	i
10	▪	42	*	74	J	106	j
11	▪	43	+	75	K	107	k
12	▪	44	,	76	L	108	l
13	▪	45	-	77	M	109	m
14	▪	46	.	78	N	110	n
15	▪	47	/	79	O	111	o
16	▪	48	0	80	P	112	p
17	▪	49	1	81	Q	113	q
18	▪	50	2	82	R	114	r
19	▪	51	3	83	S	115	s
20	▪	52	4	84	T	116	t
21	▪	53	5	85	U	117	u
22	▪	54	6	86	V	118	v
23	▪	55	7	87	W	119	w
24	▪	56	8	88	X	120	x
25	▪	57	9	89	Y	121	y
26	▪	58	:	90	Z	122	z
27	▪	59	;	91	[	123	{
28	▪	60	<	92	\	124	|
29	▪	61	=	93	]	125	}
30	▪	62	>	94	^	126	~
31	▪	63	?	95	_	127	▒

Zeichencodes (128–255)

NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR
128		160	¿	192	à	224	Á
129	÷	161	À	193	è	225	Ã
130	×	162	Á	194	ô	226	ã
131	∫	163	È	195	û	227	Đ
132	∫	164	Ê	196	á	228	đ
133	Σ	165	Ë	197	é	229	Í
134	▶	166	Ì	198	ó	230	ì
135	π	167	Ï	199	ú	231	ó
136	∂	168	´	200	à	232	ò
137	≤	169	`	201	è	233	õ
138	≥	170	^	202	ò	234	ő
139	≠	171	¨	203	ù	235	Š
140	α	172	~	204	ä	236	ś
141	→	173	Ù	205	ë	237	Ú
142	←	174	Û	206	ö	238	Ÿ
143	μ	175	£	207	ü	239	ÿ
144	↳	176	¯	208	Â	240	Þ
145	▫	177	ý	209	î	241	þ
146	«	178	ý	210	ø	242	·
147	»	179	▪	211	Æ	243	µ
148	⊢	180	Ç	212	â	244	¶
149	ı	181	ç	213	í	245	¾
150	₂	182	Ñ	214	≠	246	–
151	²	183	ñ	215	æ	247	¼
152	³	184	¡	216	Ä	248	½
153	₁	185	¿	217	ì	249	¹
154	ⁱ	186	Ø	218	ð	250	º
155	₊	187	£	219	ü	251	«
156	₍	188	¥	220	É	252	▮
157	⁾	189	₰	221	ï	253	»
158	ₖ	190	ƒ	222	β	254	±
159	ₙ	191	¢	223	ô	255	

...STRING

POS — *Position* — Befehl

Ebene 2	Ebene 1		Ebene 1
"$String_1$"	"$String_2$"	➡	n

POS gibt eine reelle Zahl n zurück; n spezifiziert dabei die Position von $String_2$ innerhalb von $String_1$. Die Position entspricht dabei dem ersten Zeichen des Substrings in $String_1$ (beginnend von links), der mit $String_2$ übereinstimmt. Ein Ergebnis mit Null deutet dabei an, daß $String_2$ nicht in $String_1$ vorkommt.

DISP — *Anzeigen* — Befehl

Ebene 2	Ebene 1	
Objekt	n	➡

DISP (*DISPlay*) zeigt das *Objekt* in der n-ten Zeile der Anzeige an. $n = 1$ bedeutet die oberste Anzeigezeile, $n = 4$ bedeutet die unterste. DISP setzt den Systemmeldungsflag, um die normale Stackanzeige zu unterdrücken.

Strings werden ohne die Begrenzungszeichen " angezeigt. Andere Objekte erscheinen in der gleichen Weise, wie sie in Ebene 1 im Mehrzeilen-Format angezeigt werden. Wenn zur vollständigen Anzeige des Objekts mehr als eine Zeile erforderlich ist, erfolgt die Anzeige ab Zeile n und setzt sich nach unten fort, entweder bis zum Ende des Objekts oder bis zur untersten Anzeigezeile.

SUB SIZE

SUB *Substring* Befehl

Ebene 3	Ebene 2	Ebene 1		Ebene 1
"*String*$_1$"	n_1	n_2	➧	"*String*$_2$"
{ *Liste*$_1$ }	n_1	n_2	➧	{ *Liste*$_2$ }

SUB nimmt einen String und zwei ganze Zahlen, n_1 und n_2, vom Stack und gibt einen neuen String zurück, der die Zeichen von Postition n_1 bis n_2 des Ausgangsstrings enthält. Wenn $n_2 < n_1$, dann gibt SUB einen leeren String zurück.

Argumente kleiner als 1 werden in 1 konvertiert; Argumente, die größer als die Stringlänge sind, werden in die Länge des Ausgangsstrings konvertiert.

Beziehen Sie sich für den Gebrauch von SUB in Verbindung mit Listen auf die Beschreibung unter "LIST".

SIZE *Größe* Befehl

Ebene 1		Ebene 1
"*String*"	➧	*n*
[*Feld*]	➧	{ *Liste* }
{ *Liste* }	➧	*n*
' *Symbol* '	➧	*n*

SIZE gibt eine Zahl *n* zurück, welche die Anzahl der im String enthaltenen Zeichen spezifiziert.

Beziehen Sie sich zum Gebrauch von SIZE in Verbindung mit anderen Objekten auf die Beschreibungen unter "ALGEBRA", "ARRAY" und "LIST".

TRIG

SIN	ASIN	COS	ACOS	TAN	ATAN
P→R	R→P	R→C	C→R	ARG	
→HMS	HMS→	HMS+	HMS−	D→R	R→D

Das TRIG-Menü (Trigonometrie) enthält Befehle, welche sich auf die Winkelmessung und trigonometrischen Funktionen beziehen: Kreisfunktionen, Konvertierung von Polar- in Rechteckskoordinaten und umgekehrt, Grad/Bogenmaß Konvertierungen und Berechnungen mit Werten, welche im Format Grad-Minuten-Sekunden oder Stunden-Minuten-Sekunden dargestellt sind.

SIN ASIN COS ACOS TAN ATAN

Dies sind die Kreisfunktionen und ihre Umkehrungen. SIN, COS und TAN interpretieren reellwertige Argumente entsprechend dem momentanen Winkelmodus (DEG oder RAD) und geben reelle Ergebnisse zurück. ACOS, ASIN und ATAN drücken reelle Ergebnisse entsprechend dem eingestellten Winkelmodus aus.

Alle sechs Funktionen akzeptieren komplexe Argumente, was zu komplexen Ergebnissen führt. Für ACOS und ASIN führen reelle Argumente mit einem Absolutbetrag größer 1 ebenfalls zu komplexen Ergebnissen. Komplexe Zahlen werden im Bogenmaß interpretiert und ausgedrückt.

ASIN, ACOS und ATAN geben den Hauptwert der Umkehrrelation zurück (wie unter "COMPLEX" beschrieben).

SIN *Sinus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➡	sin z
'Symbol'	➡	'SIN(Symbol)'

SIN berechnet den Sinus seines Arguments. Für komplexe Argumente gilt:

$$\sin (x + iy) = \sin x \cosh y + i \cos x \sinh y.$$

ASIN *Arcussinus* Analyt. Fkt.

Ebene 1		Ebene 1
z	➡	arc sin z
'Symbol'	➡	'ASIN(Symbol)'

ASIN gibt den Hauptwert des Winkels zurück, der äquivalent zum Sinus des Arguments ist. Für reelle Argumente liegen die Ergebnisse im Bereich -90 bis $+90$ Grad ($-\pi/2$ bis $+\pi/2$ im Bogenmaß). Für komplexe Argumente ergibt sich der komplexe Hauptwert von Arcussinus:

$$\text{arc sin } z = -i \ln (iz + \sqrt{1 - z^2})$$

Ein reellwertiges Argument x außerhalb des Wertebereichs $-1 \leq x \leq 1$ wird in ein komplexes Argument $z = x + 0i$ umgewandelt, und es wird der komplexe Hauptwert zurückgegeben.

...TRIG

COS — Cosinus — Analyt. Fkt.

Ebene 1		Ebene 1
z	➡	COS z
'Symbol'	➡	'COS⟨Symbol⟩'

COS berechnet den Cosinus seines Arguments. Für komplexe Argumente gilt:

$$\cos(x + iy) = \cos x \cosh y - i \sin x \sinh y$$

ACOS — Arcuscosinus — Analyt. Fkt.

Ebene 1		Ebene 1
z	➡	arc cos z
'Symbol'	➡	'ASIN⟨Symbol⟩'

ACOS gibt den Hauptwert des Winkels zurück, der äquivalent zum Cosinus des Arguments ist. Für reelle Argumente liegen die Ergebnisse im Bereich 0 bis +180 Grad (0 bis $+\pi$ im Bogenmaß). Für komplexe Argumente ergibt sich der komplexe Hauptwert von Arcuscosinus:

$$\text{arc cos } z = -i \ln(z + \sqrt{z^2 - 1})$$

Ein reellwertiges Argument x außerhalb des Wertebereichs $-1 \leqslant x \leqslant 1$ wird in ein komplexes Argument $z = x + 0i$ umgewandelt, und es wird der komplexe Hauptwert zurückgegeben.

TAN — *Tangens* — Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	tan *z*
'*Symbol*'	➧	'TAN(*Symbol*)'

TAN berechnet den Tangens seines Arguments. Für komplexe Argumente gilt:

$$\tan(x + iy) = \frac{\sin x \cos x + i \sinh y \cosh y}{(\sinh y)^2 + (\cos x)^2}$$

Wenn ein reellwertiges Argument ein ungerades ganzes Mehrfaches von 90 und als Winkelmodus DEG spezifiziert ist, tritt die Fehlerbedingung `Infinite Result` auf. Wenn Flag 59 gelöscht ist, entspricht das Vorzeichen des (MAXR) Ergebnisses dem des Arguments.

ATAN — *Arcustangens* — Analyt. Fkt.

Ebene 1		Ebene 1
z	➧	arc tan *z*
'*Symbol*'	➧	'ATAN(*Symbol*)'

ATAN gibt den Hauptwert des Winkels zurück, der äquivalent zum Tangens des Arguments ist. Für reelle Argumente liegen die Ergebnisse im Bereich -90 bis $+90$ Grad ($-\pi/2$ bis $+\pi/2$ im Bogenmaß). Für komplexe Argumente ergibt sich der komplexe Hauptwert von Arcustangens:

$$\text{arc tan } z = \frac{i}{z} \ln\left(\frac{i + z}{i - z}\right)$$

...TRIG

P→R R→P R→C C→R ARG

Die Funktionen P→R (*Polar-in-Rechteck*), R→P (*Rechteck-in-Polar*) und ARG (*Argument*) beziehen sich auf komplexe Zahlen, welche die Koordinaten von Punkten in einem zweidimensionalen Raum darstellen. R→C *(Reell-in-Komplex)* und C→R *(Komplex-in-Reell)* konvertiert reelle Zahlenpaare in komplexe Notation und umgekehrt.

Die Funktionen P→R und R→P können auch auf die ersten beiden Elemente eines reellen Vektors angewendet werden.

P→R ***Polar- in Rechtecksnotation*** **Funktion**

Ebene 1		Ebene 1
x	➧	(*x*,0)
(*r*,*θ*)	➧	(*x*,*y*)
[*r θ*...]	➧	[*x y*...]
'*Symbol*'	➧	'P→R(*Symbol*)'

P→R konvertiert eine komplexe Zahl (r, θ) bzw. einen zweielementigen Vektor $[\, r\ \theta\,]$, in Polarkoordinaten dargestellt, in eine komplexe Zahl (x, y) oder einen zweielementigen Vektor $[\, x\ y\,]$ in Rechteckskoordinaten; dabei gilt:

$$x = r \cos \theta, \quad y = r \sin \theta.$$

Der momentane Winkelmodus bestimmt, ob θ in Grad oder im Bogenmaß zu interpretieren ist.

Wenn ein Vektor mehr als zwei Elemente besitzt, bewirkt P→R die Konvertierung der ersten zwei Elemente, ohne die restlichen zu ändern. Bei dreielementigen Vektoren konvertiert P→R einen Vektor $[\, \rho\ \theta\ z\,]$ (wobei ρ die Entfernung zur z-Achse und θ den Winkel in der xy-Ebene zwischen der x-Achse und dem projizierten Vektor darstellt) in den Vektor $[\, x\ y\ z\,]$ in Rechteckskoordinaten.

Sie können einen Vektor in sphärischen Koordinaten als [$r\ \phi\ \theta$] darstellen, wobei r für die Länge des Vektors, ϕ für den Winkel zwischen der z-Achse zum Vektor und θ für den Winkel in der xy-Ebene zwischen der x-Achse und dem projizierten Vektor steht. Um einen Vektor von sphärischen Koordinaten in Rechteckskoordinaten zu konvertieren, ist nachstehende Sequenz auszuführen:

```
P→R ARRY→ DROP ROT ‹3› →ARRY P→R
```

R→P — *Rechtecks- in Polarnotation* — Funktion

Ebene 1		Ebene 1
z	➧	(*r*, *θ*)
[*x y*...]	➧	[*r θ*...]
'*Symbol*'	➧	'R→P(*Symbol*)'

R→P konvertiert eine komplexe Zahl (x, y) bzw. einen zweielementigen Vektor [$x\ y$], in Rechteckskoordinaten dargestellt, in eine komplexe Zahl (r, θ) oder einen zweielementigen Vektor [$r\ \theta$] in Polarkoordinaten; dabei gilt:

$$r = \text{abs}\ (x, y), \quad \theta = \arg\ (x, y)$$

Der momentane Winkelmodus bestimmt, ob θ in Grad oder im Bogenmaß zu interpretieren ist. Ein reellwertiges Argument x wird wie ein komplexes Argument $(x, 0)$ behandelt.

...TRIG

Wenn ein Vektor mehr als zwei Elemente besitzt, bewirkt R→P die Konvertierung der ersten zwei Elemente, ohne die restlichen zu ändern. Bei dreielementigen Vektoren konvertiert R→P einen Vektor [x y z] in den Vektor [ρ θ z] (wobei ρ die Entfernung zur z-Achse und θ den Winkel in der xy-Ebene zwischen der x-Achse und dem projizierten Vektor darstellt).

Sie können einen Vektor in sphärischen Koordinaten als [r ϕ θ] darstellen, wobei r für die Länge des Vektors, ϕ für den Winkel zwischen der z-Achse und Vektor und θ für den Winkel in der xy-Ebene zwischen der x-Achse und dem projizierten Vektor steht. Um einen Vektor von Rechteckskoordinaten in sphärische Koordinaten zu konvertieren, ist nachstehende Sequenz auszuführen:

```
R→P ARRY→ DROP ROT ROT ⟨3⟩ →ARRY R→P
```

R→C

Reell in Komplex

Befehl

Ebene 2	Ebene 1		Ebene 1
x	y	➡	⟨x,y⟩

R→C kombiniert zwei reelle Zahlen x und y in ein Koordinatenpaar (x, y).

Beziehen Sie sich für den Gebrauch von R→C in Verbindung mit Feldern auf die Beschreibung unter "ARRAY".

C→R

Komplex in Reell

Befehl

Ebene 1		Ebene 2	Ebene 1
⟨x,y⟩	➡	x	y

C→R konvertiert ein Koordinatenpaar (x, y) in zwei reelle Zahlen x und y.

Beziehen Sie sich für den Gebrauch von C→R in Verbindung mit Feldern auf die Beschreibung unter "ARRAY".

ARG

Argument **Funktion**

Ebene 1		Ebene 1
z	➡	θ
'*Symbol*'	➡	'ARG(*Symbol*)'

ARG gibt den Winkel θ des Koordinatenpaares (x, y) zurück, wobei

$$\theta = \begin{cases} \arctan y/x & \text{für } x \geqslant 0, \\ \arctan y/x + \pi \text{ Vorzeichen } y & \text{für } x < 0, \text{ Bogenmaß} \\ \arctan y/x + 180 \text{ Vorzeichen } y & \text{für } x < 0, \text{ Grad} \end{cases}$$

Der momentane Winkelmodus bestimmt, ob der Winkel θ in Grad oder im Bogenmaß ausgedrückt wird. Ein reelles Argument x wird wie ein komplexes Argument $(x, 0)$ behandelt.

→HMS HMS→ HMS+ HMS− D→R R→D

Die Befehle →HMS, HMS→, HMS+ und HMS− beschäftigen sich mit Zeitangaben (oder Winkelmaßen), die als reelle Zahlen im HMS (*Hours-Minutes-Seconds*) Format dargestellt sind.

...TRIG

Das HMS-Format lautet *h.MMSSs*, wobei gilt:

- *h* besteht aus Null oder mehreren Stellen, die den ganzzahligen Anteil der Zahl darstellen.
- *MM* sind 2 Stellen, welche die Minuten spezifizieren.
- *SS* sind 2 Stellen, welche die Sekunden spezifizieren.
- *s* besteht aus Null oder mehreren Stellen, die den dezimalen Bruchteil der Sekunden darstellen.

Nachfolgend einige Beispiele für Zeiten (oder Winkelmaße), die im HMS-Format ausgedrückt sind.

Betrag	HMS-Format
12h 32min 46sek (12° 32′ 46″)	12.3246
−6h 00min 13.2sek (−6° 00′ 13.2″)	−6.00132
36min (36′)	0.36

→HMS **Dezimal in H-M-S** **Befehl**

Ebene 1	Ebene 1
x ➧	*hms*

→HMS (*Hours Minutes Seconds*) konvertiert eine reelle Zahl, die Stunden (oder Winkel) darstellt, in das HMS-Format.

HMS→ **H-M-S in Dezimal** **Befehl**

Ebene 1	Ebene 1
hms ➧	*x*

HMS→ konvertiert eine reelle Zahl im HMS-Format in die entsprechende dezimale Form.

HMS+ *Hours-Minutes-Seconds Plus* Befehl

Ebene 2	Ebene 1		Ebene 1
hms_1	hms_2	➡	$hms_1 + hms_2$

HMS+ addiert zwei Zahlen im HMS-Format und gibt die resultierende Summe im HMS-Format zurück.

HMS− *Hours-Minutes-Seconds Minus* Befehl

Ebene 2	Ebene 1		Ebene 1
hms_1	hms_2	➡	$hms_1 - hms_2$

HMS− subtrahiert zwei reelle Zahlen im HMS-Format und gibt die resultierende Differenz im HMS-Format zurück.

D→R *Grad in Radiant* Funktion

Ebene 1		Ebene 1
x	➡	$(\pi/180)\ x$
'*Symbol*'	➡	'D→R(*Symbol*)'

D→R (*Degrees→Radians*) konvertiert eine reelle Zahl, in Grad ausgedrückt, in den entsprechenden Wert im Bogenmaß.

R→D *Radiant in Grad* Funktion

Ebene 1		Ebene 1
x	➡	$(180/\pi)\ x$
'*Symbol*'	➡	'R→D(*Symbol*)'

R→D (*Radians→Degrees*) konvertiert eine reelle Zahl, im Bogenmaß ausgedrückt, in den entsprechenden Wert in Grad.

UNITS

Das Ergebnis eines physikalischen Meßvorgangs besteht aus einem numerischen Wert sowie einer zugehörigen Einheit. Um ein Meßergebnis von einem Einheitensystem in ein anderes zu konvertieren, wird der numerische Wert mit einem Umrechnungsfaktor, der das Verhältnis der neuen Einheit zur alten Einheit darstellt, multipliziert. Ihr HP-28S automatisiert diesen Prozeß über den Befehl CONVERT. Sie spezifizieren lediglich den Zahlenwert, die alte Einheit und die neue Einheit, und CONVERT berechnet den erforderlichen Umrechnungsfaktor und liefert Ihnen das Ergebnis in der neuen Einheit.

Die Konvertierungsfunktion des HP-28S für physikalische Einheiten basiert auf den Einheiten, die zu dem 1954 international vereinbarten *Système International d'Unités* (SI-Einheiten) gehören. Der Permanentspeicher des HP-28S enthält 120 Einheiten. CONVERT erkennt jede multiplikative Kombination dieser Einheiten, ebenso zusätzliche, von Ihnen definierte Einheiten. Der UNITS-Katalog listet die eingebauten Einheiten und deren numerische Werte entsprechend den Basiseinheiten.

Das internationale Einheitensystem spezifiziert 7 Basiseinheiten: Länge (Meter), Masse (Kilogramm), Zeit (Sekunde), Stromstärke (Ampere), Temperatur (Kelvin), Lichtstärke (Candela) und Stoffmenge (Mol). Zusätzlich können Sie als Teil der benutzerdefinierten Einheiten *eine* Basiseinheit definieren, welche vom HP-28S als solche erkannt wird.

CONVERT ***Konvertieren*** **Befehl**

Ebene 3	Ebene 2	Ebene 1		Ebene 2	Ebene 1
x	"alt"	"neu"	➧	y	"neu"
x	"alt"	'neu'	➧	y	'neu'
x	'alt'	"neu"	➧	y	"neu"
x	'alt'	'neu'	➧	y	'neu'

CONVERT multipliziert eine reelle Zahl *x* mit einem Umrechnungsfaktor, der durch die alte und neue Einheit bestimmt ist. Die aus der Umrechnung resultierende reelle Zahl *y* wird in Ebene 2 zurückgegeben, wobei die neue Einheit in Ebene 1 erscheint.

Im allgemeinen sind die alten und neuen Einheiten, wie unten beschrieben, durch einen String dargestellt. Um einfache Umrechnungen zu erleichtern, können Sie allerdings auch ein Namen-Objekt zur Darstellung einer Einheit verwenden. Nehmen Sie z.B. an, die Variablen 'ft' und 'm' wären nicht definiert, so könnten Sie die Umrechnung von 320 Feet in Meter durch folgende Sequenz ausführen:

```
320 ft m CONVERT
```

Ein Einheiten-String besteht aus einem String-Objekt, welches einen algebraischen Ausdruck darstellt; dieser Ausdruck enthält die Abkürzung für physikalische Einheiten. Prinzipiell kann ein Einheiten-String folgenden Inhalt haben:

- Eingebaute oder benutzerdefinierte Einheiten. Eingebaute Einheiten werden durch deren Abkürzungen dargestellt (siehe "Der UNITS-Katalog"). Benutzerdefinierte Einheiten werden durch deren Variablennamen dargestellt (siehe "Benutzerdefinierte Einheiten").
- Eine Einheit, der das ^ Symbol und eine Ziffer im Bereich 1-9 folgt. Zum Beispiel: `"m^2"` (Quadratmeter), `"m/s^2"` (Meter pro Sekunden-Quadrat).
- Eine Einheit mit einem Vorsatz für den Zahlenwert. Als Beispiel: `"Mpc"` (Megaparsec), `"nm"` (Nanometer). (Siehe "Vorsätze für Einheiten").
- Zwei oder mehr Einheiten, welche durch das * Symbol multiplikativ verknüpft sind. Als Beispiel: `"g*cm"` (Gramm-Zentimeter), `"j*s"` (Joule-Sekunden).
- Ein / Symbol, um die Umkehrung der Einheit anzuzeigen. Wenn alle Einheiten des Strings im Nenner auftreten, kann der String mit `"1/"` beginnen. Als Beispiel: `"m/s"` (Meter pro Sekunde), `"1/m"` (Pro Meter), `"g*cm/s^2*°K"` (Gramm-Zentimeter pro Sekunden-Quadrat und Grad Kelvin).
- Das ' Symbol, welches ignoriert wird. Damit wird Ihnen ermöglicht, einen algebraischen Ausdruck im Stack zu erzeugen und auf diesen →STR anzuwenden (zum Ändern des Ausdrucks in einen Einheiten-String). Es sind jedoch keine Klammern in Einheiten-Strings erlaubt.

Die zwei Einheiten-Strings müssen dimensional eine konsistente Konvertierung darstellen. Sie können z.B. "l" (Liter) in "gal" (US-Gallonen) umrechnen, aber nicht in "Hz". CONVERT prüft, ob die Potenzen von jeder der acht Basiseinheiten (7 SI Einheiten sowie eine benutzerdefinierte Basiseinheit) übereinstimmen. Die Konsistenz für die Dimension wird über Modulo 256 überprüft.

Hier einige Beispiele zur Verwendung von CONVERT (die Zahlen sind im STD-Format dargestellt):

Alter Wert	Alte Einheit	Neue Einheit		Neuer Wert	Neue Einheit
1	"m"	"ft"	➧	3.28083989501	"ft"
1	"b*Mpc"	"cm^3"	➧	3.085678	"cm^3"
12.345	"kg*m/s^2"	"dyn"	➧	18585	"dyn"

Temperaturumrechnungen

Konvertierungen zwischen den vier Temperaturskalen (°K, °C, °F und °R) beinhalten die Verwendung von additiven Konstanten und Multiplikatoren. Wenn beide Argumente des Einheiten-Strings nur eine Temperatureinheit (ohne Vorsilbe und ohne Exponent) enthalten, dann führt CONVERT eine absolute Temperaturumrechnung durch, wobei additive Konstanten mit eingeschlossen sind. Um z.B. 50 Grad Fahrenheit in Grad Celsius zu konvertieren, ist diese Sequenz auszuführen:

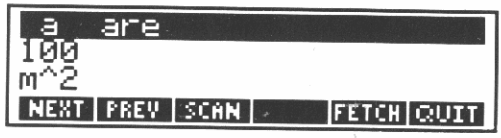
```
50 °F °C CONVERT
```

Wenn eine der Einheiten eine Vorsilbe, einen Exponenten oder anstatt einer Temperatureinheit eine sonstige Einheit enthält, dann führt CONVERT eine relative Konvertierung der Temperatureinheit durch, wobei die additive Konstante ignoriert wird.

Der UNITS-Katalog

Das Drücken von ■ UNITS aktiviert den UNITS- bzw. Einheiten-Katalog, was sich analog zum Befehlskatalog verhält, der durch Drücken von ■ CATALOG erhalten wird. Der UNITS-Katalog listet jede der im Rechner enthaltenen Einheiten, zusammen mit ihrer von CONVERT akzeptierten Abkürzung und dem Betrag der Einheit entsprechend der SI Basiseinheit.

Wenn Sie ■ UNITS gedrückt haben, wird die nomale Rechneranzeige von der UNITS-Anzeige überlagert:



Die oberste Zeile enthält die Abkürzung der gewählten Einheit, im vorliegenden Fall a, welcher der vollständige Name are folgt. Are ist die erste Einheit des alphabetisch geordneten Katalogs. Die zweite Zeile zeigt den Betrag der Einheit in SI Basiseinheiten, die in der dritten Zeile erscheinen. Insgesamt wird aus der Anzeige ersichtlich, daß are durch a abgekürzt wird und der Größe von 100 Quadratmetern entspricht.

...UNITS

Das UNITS-Menü wird in der untersten Zeile angezeigt. Die fünf belegten Tasten haben folgende Bedeutung:

Menütaste	Bedeutung
NEXT	Zeigt den nächsten Katalogeintrag an.
PREV	Zeigt den vorherigen Katalogeintrag an.
SCAN	Blättert den UNITS-Katalog vorwärts durch, wobei jeder Eintrag kurz angezeigt wird. Das Drücken von SCAN bewirkt eine Änderung der Menütastenbelegung in STOP; Drücken von STOP hält den Blättervorgang an der momentanen Einheit an; am Ende aller Katalogeinträge (Einheit "1") erfolgt automatisch ein Anhalten.
FETCH	Beendet die Kataloganzeige und fügt die zuletzt angezeigte Abkürzung an der momentanen Cursorposition in der Befehlszeile ein (beginnt neue Befehlszeile, falls keine vorhanden).
QUIT	Beendet die Kataloganzeige und läßt die Befehlszeile unverändert.

Zusätzlich zu den über das UNITS-Menü verfügbaren Operationen haben Sie folgende Möglichkeiten:

- Drücken Sie eine beliebige Buchstabentaste, um den ersten Katalogeintrag, welcher mit diesem Zeichen beginnt, anzuzeigen.
- Drücken Sie eine beliebige andere Taste auf dem linken Tastenfeld, um den ersten Katalogeintrag, welcher nicht mit einem Buchstaben beginnt, anzuzeigen ("°", für Grad).
- Drücken Sie [1], um den letzten Katalogeintrag, der nicht mit einem Buchstaben beginnt, anzuzeigen ("1").
- Drücken Sie [ON], um die Kataloganzeige zu verlassen und den Inhalt der Befehlszeile zu löschen.

Die nachfolgende Tabelle zeigt alle Einheiten des UNITS-Katalogs sowie eine kurze Beschreibung der Einheiten.

HP-28C/S Einheiten

Einheit	Voller Name	Bedeutung	Größe
a	Ar	Fläche	100 m^2
A	Ampere	Elektrischer Strom	1 A
acre	Acre	Fläche	4046.87260987 m^2
arcmin	Arcusminute	Winkel	2.90888208666E−4
arcs	Arcussekunde	Winkel	4.8481368111E−6
atm	Atmosphäre	Druck	101325 Kg/m*s^2
au	Astronomische Einheit	Länge	149597900000 m
A°	Angstrom	Länge	.0000000001 m
b	Barn	Fläche	1.E−28 m^2
bar	Bar	Druck	100000 Kg/m*s^2
bbl	Barrel, Öl	Volumen	.158987294928 m^3
Bq	Becquerel	Aktivität	1 1/s
Btu	Internat. Tabelle Btu	Energie	1055.05585262 Kg*m^2/s^2
bu	Bushel	Volumen	.03523907 m^3
C	Coulomb	Elektrische Ladung	1 A*s
cal	Internat. Tabelle Kalorie	Energie	4.1868 Kg*m^2/s^2
cd	Candela	Lichtstärke	1 cd
chain	Chain	Länge	20.1168402337 m
Ci	Curie	Aktivität	3.7E10 1/s
ct	Carat	Masse	.0002 Kg
cu	US Cup	Volumen	2.365882365E−4 m^3
d	Tag (*day*)	Zeit	86400 s
dyn	Dyne	Kraft	.00001 Kg*m/s^2
erg	Erg	Energie	.0000001 Kg*m^2/s^2

...UNITS

HP-28C/S Einheiten (Fortsetzung)

Einheit	Voller Name	Bedeutung	Größe
eV	Elektronenvolt	Energie	1.60219E−19 Kg∗m^2/s^2
F	Farad	Kapazität	1 A^2∗s^4/Kg∗m^2
fath	Fathom	Länge	1.82880365761 m
fbm	Board Foot	Volumen	.002359737216 m^3
fc	Footcandle	Leuchtdichte	10.7639104167 cd/m^2
Fdy	Faraday	Elektrische Ladung	96487 A∗s
fermi	Fermi	Länge	1.E−15 m
flam	Footlambert	Leuchtdichte	3.42625909964 cd/m^2
ft	International Foot	Länge	.3048 m
ftUS	Survey Foot	Länge	.304800609601 m
g	Gramm	Masse	.001 Kg
ga	Fallbeschleunigung	Beschleunigung	9.80665 m/s^2
gal	US Gallone	Volumen	.003785411784 m^3
galC	Kanadische Gallone	Volumen	.00454609 m^3
galUK	Britische Gallone	Volumen	.004546092 m^3
gf	Gram-force	Kraft	.00980665 Kg∗m/s^2
grad	Grade	Winkel	1.57079632679E−2
grain	Grain	Masse	.00006479891 Kg
Gy	Gray	Aufgenommene Dosis	1 m^2/s^2
h	Stunde (*hour*)	Zeit	3600 s
H	Henry	Induktion	1 Kg∗m^2/A^2∗s^2
hp	Horsepower	Leistung	745.699871582 Kg∗m^2/s^3
Hz	Hertz	Frequenz	1 1/s

HP-28C/S Einheiten (Fortsetzung)

Einheit	Voller Name	Bedeutung	Größe
in	Inch	Länge	.0254 m
inHg	Inches Quecksilber	Druck	3386.38815789 Kg/m*s^2
inH2O	Inches Wasser	Druck	248.84 Kg/m*s^2
J	Joule	Energie	1 Kg*m^2/s^2
kip	Kilopound-Force	Kraft	4448.22161526 Kg*m/s^2
knot	Knoten	Geschwindigkeit	.514444444444 m/s
kph	Kilometer pro Stunde	Geschwindigkeit	.277777777778 m/s
l	Liter	Volumen	.001 m^3
lam	Lambert	Leuchtdichte	3183.09886184 cd/m^2
lb	Avoirdupois Pound	Masse	.45359237 Kg
lbf	Pound-Force	Kraft	4.44822161526 Kg*m/s^2
lbt	Troy lb	Masse	.3732417 Kg
lm	Lumen	Lichtstrom	1 cd
lx	Lux	Beleuchtungsstärke	1 cd/m^2
lyr	Lichtjahr	Länge	9.46055E15 m
m	Meter	Länge	1 m
mho	Mho	Elektrischer Leitwert	1 A^2*s^3/Kg*m^2
mi	Internationale Meile	Länge	1609.344 m
mil	Mil	Länge	.0000254 m
min	Minute	Zeit	60 s
miUS	US Statute Mile	Länge	1609.34721869 m
mmHg	Millimeter Quecksilber	Druck	133.322368421 Kg/m*s^2
mol	Mole	Stoffmenge	1 mol

...UNITS

HP-28C/S Einheiten (Fortsetzung)

Einheit	Voller Name	Bedeutung	Größe
mph	Meilen pro Stunde	Geschwindigkeit	.44704 m/s
N	Newton	Kraft	1 Kg*m/s^2
nmi	Nautische Meile	Länge	1852 m
ohm	Ohm	Elektrischer Widerstand	1 Kg*m^2/A^2*s^3
oz	Ounce	Masse	.028349523125 Kg
ozfl	US Flüssig-Ounce	Volumen	2.95735295625E−5 m^3
ozt	Troy oz	Masse	.031103475 Kg
ozUK	UK Flüssig-Ounce	Volumen	.000028413075 m^3
P	Poise	Dynamische Viskosität	.1 Kg/m*s
Pa	Pascal	Druck	1 Kg/m*s^2
pc	Parsec	Länge	3.08567818585E16 m
pdl	Poundal	Kraft	.138254954376 Kg*m/s^2
ph	Phot	Leuchtdichte	10000 cd/m^2
pk	Peck	Volumen	.0088097675 m^3
psi	Pounds pro Square Inch	Druck	6894.75729317 Kg/m*s^2
pt	Pint	Volumen	.000473176473 m^3
qt	Quart	Volumen	.000946352946 m^3
r	Radiant	Winkel	1
R	Röntgen	Strahlendosis	.000258 A*s/Kg
rad	Rad	Energiedosis	.01 m^2/s^2
rd	Rod	Länge	5.02921005842 m
rem	Rem	Biologische Dosis	.01 m^2/s^2
s	Sekunde	Zeit	1 s
S	Siemens	Elektrischer Leitwert	1 A^2*s^3/Kg*m^2
sb	Stilb	Leuchtdichte	10000 cd/m^2

HP-28C/S Einheiten (Fortsetzung)

Einheit	Voller Name	Bedeutung	Größe
slug	Slug	Masse	14.5939029372 Kg
sr	Steradian	Festwinkel	1
st	Stere	Volumen	1 m^3
St	Stokes	Kinematische Viskosität	.0001 m^2/s
Sv	Sievert	Biologische Dosis	1 m^2/s^2
t	Metrische Tonne	Masse	1000 Kg
T	Tesla	Magnetische Induktion	1 Kg/A*s^2
tbsp	Eßlöffel	Volumen	1.47867647813E−5 m^3
therm	EEC Therm	Energie	105506000 Kg*m^2/s^2
ton	Short Ton	Masse	907.18474 Kg
tonUK	Long Ton	Masse	1016.0469088 Kg
torr	Torr	Druck	133.322368421 Kg/m*s^2
tsp	Teelöffel	Volumen	4.92892159375E−6 m^3
u	Atomare Masseeinheit	Masse	1.66057E−27 Kg
V	Volt	Elektrische Spannung	1 Kg*m^2/A*s^3
W	Watt	Leistung	1 Kg*m^2/s^3
Wb	Weber	Magnetischer Fluß	1 Kg*m^2/A*s^2
yd	Internationales Yard	Länge	.9144 m
yr	Jahr	Zeit	31536000 s
°	Grad	Winkel	1.74532925199E−2
°C	Grad Celsius	Temperatur	1 °K
°F	Grad Fahrenheit	Temperatur	.555555555556 °K
°K	Grad Kelvin	Temperatur	1 °K

...UNITS

HP-28C/S Einheiten (Fortsetzung)

Einheit	Voller Name	Bedeutung	Größe
°R	Grad Rankine	Temperatur	.555555555556 °K
μ	Micron	Länge	.000001 m
?	Benutzereinheit		1 ?
1	Dimensionslose Einheit		1

Quellen: The National Bureau of Standards Special Publication 330, *The International System of Units (SI), Fourth Edition,* Washington D.C., 1981.

The Institute of Electrical and Electronics Engineers, Inc., *American National Standard Metric Practice ANSI/IEEE Std. 268-1982,* New York, 1982.

American Society for Testing and Materials, *ASTM Standard for Metric Practice E380-84,* Philadelphia, 1984.

Aerospace Industries Association of America, Inc., *National Aerospace Standard,* Washington D.C., 1977.

Handbook of Chemistry and Physics, 64th Edition, 1983-1984, CRC Press, Inc., Boca Raton, FL, 1983.

Benutzerdefinierte Einheiten

Sie können eine in einer Variablen gespeicherte Liste erzeugen, die von CONVERT als *benutzerdefinierte Einheit* in einem Einheiten-String akzeptiert wird. Die Liste muß eine reelle Zahl und einen Einheiten-String enthalten (ähnlich zur zweiten und dritten Zeile in der UNITS-Anzeige). Nehmen Sie z.B. an, Sie würden häufig *Wochen* als Zeiteinheit verwenden. Die Ausführung von

```
{7 "d"} 'WO' STO
```

erlaubt Ihnen den Gebrauch von "WO" für Konvertierungen oder beim Erzeugen weiterer benutzerdefinierter Einheiten.

Der benutzerdefinierte Einheiten-String kann jedes Element eines zur Konvertierung erforderlichen Einheiten-Strings enthalten, zusammen mit zwei weiteren Einheiten:

- Um eine dimensionslose Einheit zu definieren, spezifizieren Sie den Einheiten-String "1".
- Um eine neue Einheit in anderen als SI Einheiten zu definieren, spezifizieren Sie den Einheiten-String "?". Um z.B. Geld in die Währungen DM, US-Dollar und franz. Franc zu konvertieren, definieren Sie:

```
    {1 "?"} 'DM' STO
{2.04 "?"} 'DOLLAR' STO
 {.31 "?"} 'FRANC' STO
```

  und rechnen danach zwischen den Einheiten um (die Umtauschkurse dienen nur als Beispiel).

Vorsätze für Einheiten

Sie können im Einheiten-String einer eingebauten Einheit einen Vorsatz (10-er Potenz) voranstellen. So bedeutet z.B. "pF" die Anzeige für "Picofarad" bzw. Farad $\times 10^{-12}$. Die nachstehende Tabelle listet alle von CONVERT akzeptierten Vorsätze.

...UNITS

Vorsätze für Einheiten

Prefix	Name	Exponent
E	Exa	+18
P	Peta	+15
T	Tera	+12
G	Giga	+9
M	Mega	+6
k oder K	Kilo	+3
h oder H	Hekto	+2
D	Deka	+1
d	Dezi	−1
c	Zenti	−2
m	Milli	−3
μ	Mikro	−6
n	Nano	−9
p	Pico	−12
F	Femto	−15
a	Atto	−18

Die meisten der vom HP-28S verwendeten Vorsätze entsprechen der SI Notation. Die Ausnahme bildet "Deka", was einen Exponenten von +1 bedeutet und "D" in HP-28S Notation gegenüber "da" in SI Notation entspricht.

Hinweis

Sie können keinen Vorsatz für eine Einheit verwenden, wenn die daraus resultierende Kombination mit einer eingebauten Einheit übereinstimmt. So könnten Sie z.B. nicht `"min"` zur Anzeige von Milli-Inches verwenden, da `"min"` der eingebauten Einheit für Minuten entspricht. Andere mögliche Kombinationen, die mit eingebauten Einheiten übereinstimmen könnten, wären z.B: `"Pa"`, `"cd"`, `"ph"`, `"flam"`, `"nmi"`, `"mph"`, `"kph"`, `"ct"`, `"pt"`, `"ft"`, `"au"` oder `"cu"`.

Obwohl Sie keine Vorsätze mit einer benutzerdefinierten Einheit verwenden können, ist es möglich, eine neue Einheit zu definieren, deren Namen das Vorsatz-Zeichen enthält.

USER

ORDER	CLUSR	MEM

Das USER-Menü enthält neben den drei "permanenten" Einträgen—ORDER, CLUSR, und MEM—einen Eintrag für jede momentan definierte Benutzervariable. Benutzervariable erscheinen am linken Rand der Anzeige, mit der zuletzt definierten Variablen zuerst. Existieren mehr als sechs Variable, so werden durch Drücken von [USER] die ersten sechs aktiviert; jede weitere Betätigung von [NEXT] aktiviert die nächste Gruppe mit sechs Variablen. ORDER, CLUSR und MEM erscheinen, wenn [NEXT] gedrückt wurde, während die letzte Gruppe der Benutzervariablen angezeigt ist (oder wenn ■ [PREV] gedrückt wurde, während die erste Gruppe aktiv ist.

Das Drücken einer USER-Menütaste, welche einer Variablen entspricht, verursacht im unmittelbaren Eingabemodus die Auswertung der assoziierten Variablen, im Alpha- oder algebraischen Eingabemodus das Anhängen der Variablen in die Befehlszeile.

Die Belegung der Menüfelder wird von den Namen der Benutzervariablen abgeleitet. Für ein Feld werden die führenden Zeichen des zugehörigen Variablennamens (bis zu fünf Zeichen) verwendet. Kleingeschriebene Buchstaben in einem Variablennamen erscheinen bei der Verwendung in einem Menüfeld als Großbuchstaben.

ORDER CLUSR MEM

ORDER *Neu anordnen* Befehl

Ebene 1	
{ $Name_1$ $Name_2$ ... } ➡	

ORDER benutzt eine Liste mit Variablennamen und ordnet den Speicherbereich für die Variablen in der Weise, daß die Variablen-Reihenfolge im Benutzermenü der spezifizierten Reihenfolge in der Liste entspricht. Die in der Liste erwähnten Variablen werden an den Anfang des Benutzermenüs verschoben. Nicht in der Liste erscheinende Variablen bleiben an ihrer momentanen Position und folgen den in der Liste benannten Variablen.

CLUSR — *Benutzerspeicher löschen* — Befehl

➡

CLUSR (*CLear USeR memory*) löscht alle momentan definierten Variablen. Um das versehentliche Löschen zu verhindern, bewirkt das Drücken von CLUSR die Ausführung von ENTER und übernimmt `CLUSR` in die Befehlszeile. Danach können Sie ENTER drücken, um den Befehl CLUSR auszuführen.

MEM — *Speicher* — Befehl

	Ebene 1
➡	*x*

MEM (*MEMory*) gibt die Anzahl der momentan freien Bytes im Benutzerspeichers zurück. Der von MEM angezeigte Wert sollte nur als grober Indikator betrachtet werden, da der vom Rechner belegte Speicherbereich von den momentanen sowie anstehenden Operationen, die nicht direkt ersichtlich sein können, abhängig ist. Zum Beispiel können die Verfahren zur Rücksicherung der Anzeige bzw. des Stacks (■ COMMAND, ■ UNDO und ■ LAST) bei der Ausführung einer Operation einen beträchtliche Anzahl an Bytes verbrauchen oder freigeben.

A

Meldungen

In diesem Anhang sind alle Fehler- und Statusmeldungen des HP-28C/S aufgelistet. Die Meldungen erscheinen normalerweise in Zeile 1 und werden durch den nächsten Tastendruck wieder gelöscht. (Qualifizierende Meldungen hinsichtlich einer gefundenen Nullstelle werden in Zeile 2 angezeigt.)

Als *Statusmeldungen* bezeichnete Meldungen sind zu Ihrer Information gedacht und zeigen *keine* Fehlerbedingung an. Als *mathematische Ausnahmen* bezeichnete Meldungen werden nicht angezeigt, wenn der korrespondierende Fehlerflag gelöscht ist. (Mathematische Ausnahmen sind unter "Grundlagen" beschrieben.)

Alphabetisch gelistete Meldungen

Meldung	Fehlernr.		Bedeutung
	Hex	Dez	
Bad Argument Type	202	514	Ein Befehl benötigte ein anderes Argument als Objekttyp.
Bad Argument Value	203	515	Der Wert eines Arguments lag außerhalb des zulässigen Bereichs für einen Befehl.
Bad Guess(es)	A01	2561	Die Anfangsnäherung(en) für den Löser bzw. für ROOT verursachten ungültige Ergebnisse beim Auswerten der momentanen Gleichung.
Can't Edit CHR(0)	102	258	Es wurde der Versuch unternommen, einen String, der das Zeichen 0 enthält, zu editieren.

Alphabetisch gelistete Meldungen (Fortsetzung)

Meldung	Fehlernr.		Bedeutung
	Hex	Dez	
Circular Reference	129	297	Es wurde versucht, unter Verwendung des Löser-Menüs in einer Variablen ein Objekt zu speichern, wobei das Objekt auf die Variable direkt oder indirekt Bezug nimmt.
Command Stack Disabled	125	293	Es wurde ■ COMMAND gedrückt, während COMMAND desaktiviert war.
Constant?	A02	2562	Die momentane Gleichung gibt für jeden vom Lösungsalgorithmus getesteten Punkt den gleichen Wert zurück.
Constant Equation	Status		Die momentane Gleichung gibt für jeden Punkt den gleichen Wert zurück, der von DRAW innerhalb des spezifizierten Wertebereichs getestet wurde.
Extremum	Status		Das vom Löser ermittelte Ergebnis ist eher ein Extrempunkt als eine Nullstelle.
HALT not Allowed	121	289	DRAW oder der Löser stießen auf den Befehl HALT innerhalb des Programms EQ.
Improper User Function	103	259	Es wurde versucht, eine unzulässige Benutzerfunktion auszuwerten. Beziehen Sie sich für die richtige Syntax auf den Abschnitt "Programme".
Inconsistent Units	B02	2818	Es wurde CONVERT ausgeführt, wobei die Einheiten-Strings unterschiedliche Dimensionen aufwiesen.

Alphabetisch gelistete Meldungen (Fortsetzung)

Meldung	Fehlernr.		Bedeutung
	Hex	Dez	
Infinite Result	305	773	Math. Ausnahme. Eine Berechnung ergab ein "unendliches" Ergebnis, z.B. 1/0 oder LN(0).
Insufficient Memory	001	001	Der freie Speicherbereich war zu klein, um eine Operation auszuführen.
Insufficient Σ Data	603	1539	Es wurde ein Statistik-Befehl ausgeführt, wobei in ΣDAT die für die Berechnung erforderliche Datenmenge nicht zur Verfügung stand.
Interrupted	Status		Der Löser wurde durch das Drücken von [ON] unterbrochen.
Invalid Dimension	501	1281	Ein Feld-Argument war falsch dimensioniert.
Invalid PPAR	11E	286	DRAW oder DRWΣ stießen auf einen unzulässigen Eintrag in PPAR.
Invalid Unit String	B01	2817	CONVERT wurde unter Verwendung eines unzulässigen Einheiten-Strings ausgeführt.
Invalid ΣDAT	601	1537	Es wurde ein Statistik-Befehl ausgeführt, während in ΣDAT ein unzulässiges Objekt gespeichert war.
Invalid ΣPAR	604	1540	ΣPAR ist der falsche Objekttyp oder enthält einen unzulässigen/fehlenden Eintrag in seiner Liste.
LAST Disabled	205	517	Es wurde LAST ausgeführt, während Flag 31 gelöscht war.
Low Memory!	Status		Deutet an, daß weniger als 128 Bytes an freiem Speicherbereich zur Verfügung stehen.

Alphabetisch gelistete Meldungen (Fortsetzung)

Meldung	Fehlernr.		Bedeutung
	Hex	Dez	
Memory Lost	005	005	Der Speicher des HP-28S wurde zurückgesetzt.
Negative Underflow	302	770	Math. Ausnahme. Aus einer Berechnung resultierte ein negatives Ergebnis größer als −MINR.
No Current Equation	104	260	SOLVR oder DRAW wurde unter Verwendung einer nicht vorhandenen Variablen EQ ausgeführt.
Nonexistent ΣDAT	602	1538	Es wurde ein Statistik-Befehl ausgeführt, ohne daß die Variable ΣDAT zur Verfügung stand.
Non-real Result	11F	287	Eine Prozedur gab nicht das von Löser, ROOT, DRAW oder ∫ geforderte reellwertige Resultat zurück.
No Room for UNDO	101	257	Der freie Speicherbereich war zu klein zur Sicherung einer Stack-Kopie. UNDO wird automatisch desaktiviert.
No Room to ENTER	105	261	Der freie Speicherbereich war zu klein zur Verarbeitung der Befehlszeile.
No Room to Show Stack	Status		Der Speicherbereich ist zur normalen Anzeige des Stacks zu klein.
Name$_1$ Not in Equation	Status		Bei der Ausführung von DRAW war in der momentanen Gleichung die unabhängige Variable *Name*$_1$ in PPAR nicht vorhanden. Dieser Meldung folgt entweder **Constant Equation** oder **Using** *Name*$_2$.

Alphabetisch gelistete Meldungen (Fortsetzung)

Meldung	Fehlernr.		Bedeutung
	Hex	Dez	
Out of Memory			Es steht kein freier Speicherbereich zum weiteren Arbeiten mit dem Rechner zur Verfügung. Zur Fortsetzung müssen Sie ein oder mehrere Objekte löschen.
Overflow	303	771	Math. Ausnahme. Eine Berechnung resultierte in einem (absoluten) Wert größer als MAXR.
Positive Underflow	301	769	Math. Ausnahme. Eine Berechnung resultierte in einem positiven Wert kleiner als MINR.
Sign Reversal	Status		Der Löser konnte einen Punkt ermitteln, der eine Näherung an eine tatsächliche Nullstelle oder eine Unterbrechung der Prozedur darstellt (siehe Seite 282).
Syntax Error	106	262	Ein Objekt wurde in der falschen Form in die Befehlszeile eingegeben.
Too Few Arguments	201	513	Ein Befehl benötigte mehr Argumente, als im Stack verfügbar waren.
Unable to Isolate	120	288	Der spezifizierte Name konnte in dem Ausdruck nicht isoliert werden. Der Name hat entweder gefehlt oder war als Argument für eine Funktion verwendet, für welche es keine Umkehrfunktion gibt.
Undefined Local Name	003	003	Es wurde versucht, einen lokalen Namen auszuwerten, für den keine korrespondierende lokale Variable existiert.

Alphabetisch gelistete Meldungen (Fortsetzung)

Meldung	Fehlernr.		Bedeutung
	Hex	Dez	
Undefined Name	204	516	Es wurde versucht, den Wert einer nicht definierten (formalen) Variablen zurückzurufen.
Undefined Result	304	772	Es wurde eine Funktion ausgeführt, deren Argumente zu einem mathematisch undefinierten Ergebnis führten, wie z.B. 0/0, oder LNP1(*x*) für $x<-1$.
UNDO Disabled	124	292	Es wurde ■ UNDO gedrückt, während UNDO desaktiviert war.
Using *Name*	Status		DRAW wählte die unabhängige Variable *Name*.
Wrong Argument Count	128	296	In einem Ausdruck wurde eine benutzerdefinierte Funktion ausgewertet, wobei in Klammern die falsche Anzahl von Argumenten angegeben wurde.
Zero	Status		Der Löser hat einen Wert für die unbekannte Variable gefunden, an welchem sich für die momentane Gleichung Null ergibt.

Nach Fehlernr. gelistete Fehlermeldungen

Hex	Dez	Meldung
Durch allgemeine Operationen resultierende Fehler		
001	001	Insufficient Memory
003	003	Undefined Local Name
005	005	Memory Lost
101	257	No Room for UNDO
102	258	Can't Edit CHR(0)
103	259	Improper User Function
104	260	No Current Equation
105	261	No Room to ENTER
106	262	Syntax Error
11E	286	Invalid PPAR
11F	287	Non-real Result
120	288	Unable to Isolate
121	289	HALT not Allowed
124	292	UNDO Disabled
125	293	Command Stack Disabled
128	296	Wrong Argument Count
129	297	Circular Reference
Durch Stackoperationen resultierende Fehler		
201	513	Too Few Arguments
202	514	Bad Argument Type
203	515	Bad Argument Value
204	516	Undefined Name
205	517	LAST Disabled
Durch Operationen mit reellen Zahlen resultierende Fehler		
301	769	Positive Underflow
302	770	Negative Underflow
303	771	Overflow
304	772	Undefined Result
305	773	Infinite Result

Nach Fehlernr. gelistete Fehlermeldungen (Fortsetzung)

Hex	Dez	Meldung
Durch Feld-Operationen resultierende Fehler		
501	1281	Invalid Dimension
Durch statistische Operationen resultierende Fehler		
601	1537	Invalid ΣDAT
602	1538	Nonexistent ΣDAT
603	1539	Insufficient Σ Data
604	1540	Invalid ΣPAR
Durch den Lösungsprozeß resultierende Fehler		
A01	2561	Bad Guess(es)
A02	2562	Constant?
Durch Einheiten-Konvertierungen resultierende Fehler		
B01	2817	Invalid Unit String
B02	2818	Inconsistent Units

B

Hinweise für Anwender von HP UPN-Rechnern

Hewlett-Packard hat, beginnend mit dem HP-35 im Jahr 1972, eine Reihe von Taschenrechnern entwickelt, deren Rechenweise auf dem Prinzip der UPN Stack-Schnittstelle basiert. Obwohl in den Fähigkeiten und Applikationen dieser Rechner viele Unterscheidungsmerkmale bestehen, teilen sich alle Rechner die gemeinsame Implementation der grundlegenden Stack-Schnittstelle, wodurch einem erfahrenen Benutzer die Anwendung der anderen Rechner vereinfacht wird.

Der HP-28S verwendet ebenso einen Stack und UPN-Logik als zentrale Einrichtung seiner Benutzerschnittstelle. Allerdings erwies sich ein Stack mit nur vier Ebenen und eine starre Registerstruktur, wie es bei den seitherigen Modellen der Fall war, als unpassend für die Unterstützung der vielen Objekttypen und der Möglichkeit für symbolische Mathematik. Auch wenn der HP-28S die natürliche Entwicklung der "urspünglichen" UPN-Schnittstelle darstellt, so gibt es ausreichend viele Unterschiede zwischen dem HP-28S und seinen Vorgängern, die eine kleine "Anpassungsphase" erfordern, falls Sie bereits den Umgang mit anderen UPN-Rechnern gewohnt sind. In diesem Anhang wird ausführlich auf die hauptsächlichen Unterschiede eingegangen.

Der dynamische Stack

Der dramatischste Unterschied zwischen der Basis-Schnittstelle des HP-28S und den vorangehenden HP UPN Rechnern liegt in der Größe des Stacks. Die anderen Rechner bieten einen Stack mit vier Ebenen, bestehend aus dem X-, Y-, Z- und T-Register, vergrößert durch ein LAST X bzw. L-Register. Ein Stack dieser Art ist immer "voll"—selbst beim "Löschen" des Stacks wird dieser eigentlich nur mit Nullen aufgefüllt.

Der im HP-28S implementierte Stack besitzt keine feste Größe. Beim Eingeben von neuen Objekten werden dynamisch neue Ebenen je nach Anforderung erzeugt. Wenn Sie Objekte im Stack löschen, verkleinert sich der Stack, je nach dem sogar soweit, daß er vollständig leer ist. Demzufolge kann der HP-28S einen `Too Few Arguments` Fehler erzeugen, was seinen Vorgängern nicht möglich war.

Die Implementation einer dynamischen Stackorganisation im Gegensatz zur starren Organistation erhöht die nachfolgenden spezifischen Unterschiede zwischen dem HP-28S und Rechnern mit einer starren Stackorganisation:

Numerierte Ebenen. Die "unendliche" Größe des im HP-28S vorhandenen Stacks läßt die Verwendung von X Y Z T als Namen für die Ebenen als unpraktikabel erscheinen. Deshalb ist Ebene 1 gleichbedeutend zum X-Register, 2 zu Y, 3 zu Z und 4 zu T. Die Tastenbezeichnungen $1/x$ und x^2 wurden aus Gründen der Verwandheit beibehalten—sie machen die Tasten erkenntlicher als die Verwendung der eigentlichen Befehlsnamen INV und SQ, respektive. Allerdings wurde die UPN-Festlegung X<>Y im HP-28S durch den Befehl SWAP ersetzt.

Stack-Manipulation. Der HP-28S erfordert einen allgemeineren Befehlssatz zur Veränderung des Stackinhalts, als es bei den Rechnern mit starrer Stackorganisation notwendig war. So wurde z.B. R↑ und R↓ durch ROLL und ROLLD ersetzt, die beide ein zusätzliches Argument erfordern, das die Anzahl der zu "rollenden" Ebenen spezifiziert. Weiterhin enthält das STACK-Menü mehrere Befehle zur Stack-Manipulation, welche bei den seitherigen Modellen nicht existieren.

Kein automatisches Kopieren des T-Registers. Bei den Rechnern mit einer starren Stackorganisation wurde der Inhalt des T-Registers bei jeder Verschiebung nach unten ("stack-drop") in das Z-Register kopiert. Dies bietet eine einfache Möglichkeit zur Konstanten-Multiplikation—Sie füllen den Stack mit Kopien einer Konstanten und multiplizieren diese mit einer Zahlenreihe, indem Sie jede Zahl eingeben und [×], dann [CLx] drücken, nachdem Sie jedes Ergebnis notiert haben. Sie können dies nicht mit dem HP-28S durchführen—dafür ist es sehr leicht, ein Programm der Form

```
« 12345 * » 'MULT' STO
```

zu erzeugen, in welchem 12345 eine typische Konstante darstellt. Alles, was Sie dann noch tun müssen, ist das Drücken von [USER], das Eingeben einer Zahl und das Drücken von [MULT]; mit der Eingabe einer neuen Zahl, dem erneuten Drücken von [MULT], usw., erfüllen Sie die gleiche Aufgabe. Sie können aufeinanderfolgende Ergebnisse dabei auch im Stack lassen.

Stack-Speicheranforderung. Ein dynamischer Stack hat den Vorteil, daß Sie beliebig viele Ebenen für Ihre Berechnungen zur Verfügung haben, ohne sich Sorgen machen zu müssen, durch die Eingabe neuer Objekte am "oberen" Ende des Stacks Objekte zu verlieren. Auf der anderen Seite kann es sich ergeben, daß Sie einen beachtlichen Teil des Speicherbereichs durch alte Objekte belegen, wenn Sie diese am Ende einer Berechnung im Stack lassen. Mit der Verwendung des HP-28S sollten Sie sich angewöhnen, nicht mehr benötigte Objekte im Stack zu löschen.

"DROP" im Vergleich zu CLX. Bei Rechnern mit starrer Stackorganisation ist CLX gleichbedeutend mit "ersetze den Inhalt des X-Registers durch 0 und sperre den Stack Lift (Verschiebeoperation des Stackinhalts nach oben)". Der primäre Zweck liegt im Löschen eines alten Zahlenwerts, bevor der Inhalt durch eine neue Zahl ersetzt wird—aber Sie können dies auch als einen Weg ansehen, den Wert 0 einzugeben. Im HP-28S wurde CLX durch DROP ersetzt, was das Löschen des Objekts in Ebene 1 und ein Verschieben des restlichen Inhalts um eine Ebene nach unten bewirkt. Es wird keine 0 eingegeben. CLEAR hat eine ähnliche Auswirkung: Es werden alle Objekte im Stack nach unten verschoben (und damit letztlich gelöscht), ohne daß wie beim CLST (*CLear STack*) ein Ersetzen durch Nullen erfolgt.

Stack Lift sperren und ENTER

Bestimmte Befehle für Rechner mit starrer Stackorganisation (ENTER↑, CLX, $\Sigma+$, $\Sigma-$) beinhalten das automatische *Sperren des Stack Lifts* (keine Verschiebung des Stackinhalts nach oben). Das bedeutet, nach der Ausführung eines der o.a. Befehle wird der Inhalt des X-Registers durch die Eingabe eines neuen Werts überschrieben, anstatt in das Y-Register geschoben zu werden. Diese Eigenschaft bzw. Möglichkeit ist im HP-28S nicht enthalten. Die Eingabe eines neuen Objekts bewirkt *immer* das Verschieben des alten Inhalts um eine Ebene nach oben.

Bei Rechnern mit festen 4 Stackebenen spielen das X-Register und ENTER duale Rollen. Das X-Register dient als Eingabe-Register sowie als gewöhnliches Stack-Register—durch das Eintippen einer Zahl werden die Ziffern im X-Register gespeichert. Die Taste [ENTER↑] ermöglicht das Separieren zweier aufeinanderfolgenden Zahleneingaben. Zusätzlich zu dieser Funktion kopiert [ENTER↑] den Inhalt des X-Registers in das Y-Register und sperrt den Stack Lift.

Im HP-28S ist jede dieser dualen Rollen getrennt—es gibt kein Sperren des Stack Lifts. Die Befehlszeile dient zur Dateneingabe und unterscheidet sich komplett von der Stackebene 1 (dem "X-Register"). ENTER wird *nur* zur Verarbeitung der Befehlszeile benutzt—er kopiert nicht den Inhalt von Ebene 1. Beachten Sie aber, daß [ENTER] die DUP-Operation ausführt (welche den Inhalt von Ebene 1 in Ebene 2 kopiert), wenn keine Befehlszeile vorhanden ist. Diese Fähigkeit von [ENTER] ist teilweise deshalb im HP-28S enthalten, um eine gewisse Konsistenz zu den Vorgängermodellen zu erhalten.

Syntax

HP-28S Befehle verwenden strikt eine "postfix" Syntax, d.h. Befehle, die Argumente benutzen, erfordern vor der Befehlsausführung das Vorhandensein dieser Argumente im Stack. Dies bedeutet eine Abweichung von der seitherigen Konvention für UPN Rechner, in welcher Argumente, die eine Registernummer, Flagnummer, usw. darstellen, sich nicht im Stack befinden, sondern *nach* dem Befehl selbst eingegeben werden (z.B. STO 25, TONE 1, CF 03, usw.). Das letztere Verfahren hat den Vorteil, daß eine Stackebene eingespart wird; andererseits ist damit ein unflexibles Format verbunden—so muß z.B. bei STO im HP-41 immer eine zweistellige Registernummer folgen.

Ähnliche Operationen im HP-28S sind in ihrem Wesen näher zu *indirekten* Operationen in Rechnern mit starrer Stackorganisation, in welchen Sie ein *i-Register* (im Fall des HP-41 jedes Register) verwenden können, um Register, Flags, usw., zu spezifizieren. Im HP-28S läßt sich STO, RCL, usw. anzeigen, indem Sie die Ebene 1 als *i-Register* verwenden. Zum Beispiel bedeutet RCL das "Zurückrufen des Variableninhalts bzw. des Registers, das in Ebene 1 spezifiziert wurde"—äquivalent zu RCL IND X im HP-41.

Sie sollten daran denken, daß die meisten Befehle des HP-28S ihre Argumente nach der Befehlsausführung vom Stack nehmen. Wenn Sie z.B. 123 'X' STO ausführen, dann verschwinden 123 und 'X' vom Stack. Ohne dieses Verhalten wäre der Stack in kurzer Zeit mit "alten" Argumenten überladen. Wenn Sie 123 im Stack erhalten möchten, wäre die Befehlssequenz 123 DUP 'X' STO erforderlich.

Register im Vergleich zu Variablen

Taschenrechner mit einer starren Stackorganisation können nur reelle Gleitkommazahlen effizient verarbeiten, für welche die feste 7-Byte-Registerstruktur des Stacks und die numerierten Datenregister geeignet sind (der HP-41 führte ein einfaches Alpha-Datenobjekt innerhalb der Einschränkung durch das 7-Byte-Format ein). Der HP-28S ersetzt numerierte Datenregister durch benannte Variablen. Neben dem Effekt, eine flexible Struktur zur Speicherung unterschiedlicher Objekttypen zu erhalten, besitzen Variablen den Vorteil, daß durch geschickt Vergabe eines Namens ein näherer Bezug zum Variableninhalt als durch die Registernummer hergestellt werden kann.

Wenn Sie numerierte Register im HP-28S duplizieren wollen, können Sie dazu einen Vektor benutzen:

```
« 50 » 0 CON 'REG' STO
```

erzeugt einen Vektor mit 50 Elementen, die mit 0 initialisiert sind.

```
« 1 →LIST 'REG' SWAP GET » 'NRCL' STO
```

erzeugt ein Programm NRCL, welches das *n*-te Element eines Vektors zurückruft, wobei *n* eine Zahl in Ebene 1 darstellt.

```
« 1 →LIST 'REG' SWAP ROT PUT » 'NSTO' STO
```

erzeugt ein analoges Programm NSTO zum Speichern.

LASTX im Vergleich zu LAST

In Rechnern mit starrer Stackorganisation gibt der Befehl LASTX den Inhalt des LASTX- (oder L) Registers zurück, in welchem der letzte vom X-Register benutzte Wert gespeichert ist. Dieses Konzept ist im HP-28S in allgemeinerer Form über den Befehl LAST verwirklicht. LAST gibt das letzte Argument zurück, welches durch einen Befehl vom Stack genommen wurde (oder die letzten 2 bzw. 3 von einem Befehl benutzten Argumente). Demzufolge bringt bei einer starren Stackorganisation die Sequenz 1 2 + LASTX die Zahlen 3 und 2 in den Stack, während im HP-28S 1 2 + LAST die Werte 3, 1 und 2 in den Stack zurückspeichert.

Obwohl der Befehl LAST im HP-28S flexibler als der LASTX der Vorgängermodelle ist, sollten Sie nicht außer Acht lassen, daß im HP-28S mehr Befehle Argumente vom Stack nehmen, als dies bei den Rechnern mit starrer Stackorganisation der Fall ist. Dies bedeutet, daß die Argumente für LAST häufiger überschrieben werden, wobei sogar Befehle wie DROP oder ROLL die Argumente von LAST ersetzen.

Denken Sie auch daran, daß mit UNDO der komplette Stackinhalt zurückgesichert werden kann, was zur einfachen Fehlerbehebung gegenüber LAST bevorzugt werden dürfte.

C

Hinweise für Anwender von AOS-Rechnern

Viele AOS Taschenrechner (*Algebraic Operating System*), einschließlich der großen Anzahl von einfachen "4-Funktionen" Rechnern, verwenden eine *algebraische* Rechner-Schnittstelle. Der Name leitet sich aus der Eigenschaft ab, daß bei einfachen Berechnungen die Reihenfolge der zu drückenden Tasten mit der als algebraischer Ausdruck auf Papier dargestellten Aufgabenstellung entspricht. Dies bedeutet, daß zur Auswertung von 1 + 2 − 3 die Eingabesequenz [1] [+] [2] [−] [3] [=] erforderlich ist.

Diese Schnittstelle eignet sich gut für Ausdrücke, die Zahlen und *Operatoren* enthalten—Funktionen wie +, −, × und /, deren Notation zwischen den jeweiligen Argumenten erfolgt. Weiter entwickelte Taschenrechnermodelle erlauben Ihnen die Eingabe von Klammern, um die Priorität bzw. die Reihenfolge einzelner Operationen zu spezifizieren. Allerdings führt die Anwendung von vorangestellten Funktionen, wie SIN, LOG, usw., zu zwei unterschiedlichen Varianten:

- Gewöhnliche algebraische Rechner benutzen eine Kombination beider Arten—zwischengestellte Operatoren bleiben zwischengestellt, während vorangestellte Funktionen in einer nachgestellten Weise eingegeben werden (wie bei UPN Rechnern). So wird z.B. 1 + SIN(23) in der Form [1] [+] [2] [3] [SIN] [=] eingegeben. Dieser Weg hat den Vorteil, daß Zwischenergebnisse angezeigt werden können und die Auswertung vorangestellter Funktionen (ohne Klammern) weiterhin durch eine einzelne Taste möglich ist; dafür wird der Nachteil in Kauf genommen, daß die Analogie zur allgemeinen mathematischen Notation verloren geht (was eigentlich der primäre Vorteil einer algebraischen Schnittstelle ist).

- Rechner mit "direkter Gleichungseingabe" und Computer, die BASIC als Programmiersprache anbieten und über einen unmittelbaren Ausführungsmodus verfügen, ermöglichen die Eingabe eines vollständigen Ausdrucks in seiner normalen algebraischen Form. Das Ergebnis wird nach dem Drücken einer speziellen Taste (verschiedentlich [ENTER], [ENDLINE], [RETURN], usw. benannt) berechnet. Dieser Weg hat den Vorteil, daß die Analogie zwischen dem Ausdruck in mathematischer Notation und der Sequenz der zu drückenden Tasten erhalten bleibt—der Nachteil hierbei ist normalerweise der Verzicht auf Zwischenergebnisse (der CALC-Modus des HP-71B bildet eine Ausnahme). Ihnen muß die vollständige Form des Ausdrucks vor dem Eintippen bekannt sein, da es gewöhnlich sehr schwierig ist, sich durch ein Problem "durchzuarbeiten", indem der Lösungsweg aufgrund von Zwischenergebnissen variiert wird.

Erste Erfahrungen mit dem HP-28S

Die Verarbeitungslogik des HP-28S basiert auf einer mathematischen Logik, die unter dem Begriff "Polnische Notation" bekannt ist und von dem polnischen Mathematiker Jan Łukasiweicz (1878–1956) entwickelt wurde. In der herkömmlichen algebraischen Notation werden die arithmetischen Operatoren *zwischen* die jeweiligen Zahlen oder Variablen gestellt. In der Notation von Łukasiweicz erscheint der Operator *vor* den Variablen. In einer Variaton dieser Darstellungsweise wird der Operator den Variablen *nachgestellt*—was mit "umgekehrter polnischer Notation" oder in der Kurzform mit "UPN" (*engl: Reverse Polish Notation*) bezeichnet wird.

Die prinzipielle Idee von UPN ist die Eingabe von Zahlen (oder anderen Objekten) in den Stack und die *anschließende* Ausführung eines Befehls, der sich auf diese Eingaben—"Argumente" genannt—bezieht. Im "Stack" wird lediglich die Reihenfolge der zu verwendenden Objekte festgelegt bzw. gespeichert. Viele Befehle geben ihre Resultate in den Stack zurück, wodurch diese für nachfolgende Operationen weiterverwendet werden können.

Der HP-28S verwendet einen UPN Stack als Schnittstelle, da mit diesem die notwendige Flexibilität für die Unterstützung des breiten mathematischen Anwendungsspektrums sichergestellt ist. Alle Rechneroperationen, einschließlich der, welche nicht in algebraischen Ausdrücken dargestellt werden können, erfolgen in der gleichen Art und Weise—Argumente werden vom Stack genommen und Resultate werden in den Stack zurückgegeben.

Unabhängig davon bildet der UPN Stack bei der Ausführung einfacher arithmetischer Operationen wahrscheinlich den größten Stolperstein für Anwender, die das Benutzen eines Rechners mit algebraischer Logik gewohnt sind und sich an einem UPN Rechner "einarbeiten". UPN ist eine sehr effiziente Verarbeitungslogik, die jedoch von Ihnen verlangt, daß Sie einen Ausdruck geistig neu anordnen, bevor Sie das Ergebnis berechnen können. Die Fähigkeit des HP-28S, einen algebraischen Ausdruck ohne vorherige Übersetzung interpretieren zu können, sollte den Übergang von Rechnern mit algebraischer Logik leichter machen, als dies bei vorangehenden UPN Rechnern möglich war. Die vierzeilige Anzeige ist dabei ebenfalls behilflich, etwas vom Geheimnis des Stacks freizulegen, da der Inhalt von bis zu vier Stackebenen gleichzeitig angezeigt wird.

Um algebraische Ausdrücke auswerten zu können, wurde der HP-28S hauptsächlich als Rechner zur "direkten Gleichungseingabe" entwickelt. Dies bedeutet, daß zur Auswertung eines algebraischen Ausdrucks lediglich die Taste ['] gedrückt werden muß, bevor der Ausdruck in seiner algebraischen Form—Klammern, zwischengestellte Operatoren und vorangestellte Funktionen eingeschlossen—eingetippt werden kann. Nach dem Drücken von [EVAL] erhalten Sie das Ergebnis angezeigt. Dieses Verfahren eignet sich natürlich auch für einfache Arithmetik:

['] [1] [+] [2] [−] [3] [EVAL] ergibt 0.

Außer dem ersten Tastendruck ['] entspricht dies der gleichen Vorgehensweise wie bei einem Rechner mit algebraischer Logik, sofern [EVAL] durch [=] substituiert wird.

Hinweis

Verwechseln Sie beim HP-28S die Taste [=] nicht mit der Taste, wie sie bei AOS-Rechnern zur Anwendung kommt—im HP-28S wird [=] ausschließlich zur Erzeugung von algebraischen Gleichungen verwendet.

Wenn Sie den HP-28S als Rechner zur "direkten Gleichungseingabe" verwenden, bleibt jedes berechnete Ergebnis im Stack erhalten, wodurch dieser die Rolle eines "Historik-Speichers" annimmt. Dies ermöglicht Ihnen das Sichern alter Resultate für die spätere Wiederverwendung. Außerdem wird Ihnen dadurch erlaubt, längere Berechnungsvorgänge in kleinere Blöcke aufzuteilen, wobei die Teilergebnisse im Stack gespeichert bleiben und am Ende aller Teilberechnungen zusammengefaßt werden können. Der Stack bietet einen viel leichter zu handhabenden und gleichzeitig leistungsfähigeren Historik-Speicher als die "Result"-Funktion, die in AOS- oder in BASIC-Rechnern verfügbar ist.

Eine der wichtigsten Eigenschaften des HP-28S besteht darin, daß Sie sich nicht damit beschäftigen müssen, ob nun die UPN Logik besser oder schlechter als die algebraische Logik ist. Sie wählen diejenige Logik aus, die sich zur Lösung der vorliegenden Aufgabenstellung am besten eignet und mischen algebraische Ausdrücke mit UPN Manipulationen.

Glossar

Abbildung einer Funktion: Eine von DRAW erzeugte Abbildung, für welche die momentane Gleichung an bis zu 137 Stellen einer spezifizierten (unabhängigen) Variablen ausgewertet wurde.

Abbildungsparameter: Der Inhalt der Variablenliste PPAR, welche die Position und Skalierung einer Abbildung sowie den Namen der unabhängigen Variablen bestimmt.

Abhängige Variable: Eine Variable, deren Wert durch die Werte von anderen (unabhängigen) Variablen berechnet wird; sie bezieht sich außerdem auf die vertikalen Koordinatenwerte in graphischen Abbildungen.

Algebraische Syntax: Die Restriktionen für eine Prozedur, wobei (1) bei der Ausführung keine Argumente vom Stack genommen und nur ein Ergebnis in den Stack zurückgegeben wird, und (2) die Prozedur in hierarchische Teilausdrücke aufteilbar ist. Diese Bedingungen werden von allen algebraischen Objekten und einigen Programmen erfüllt.

Algebraischer Eingabemodus: Der Eingabemodus, in welchem die zu einer Taste korrespondierende *Funktion* ihren Funktionsnamen und eine linke Klammer (falls geeignet) dem Inhalt der Befehlszeile hinzufügt. Tasten, die zu anderen Befehlen korrespondieren, führen ihre Befehle sofort aus.

Algebraisches Objekt: Eine Prozedur, die zwischen den Begrenzungszeichen ' ' eingegeben und angezeigt wird; sie kann Variable, Operatoren und Funktionen enthalten, welche in algebraischer Syntax zur Darstellung eines mathematischen Ausdrucks oder einer Gleichung kombiniert werden können.

Alpha-Eingabemodus: Der Eingabemodus, in welchem alle zu einer Taste korrespondierenden Befehle ihren Befehlsnamen dem Inhalt in der Befehlszeile hinzufügen.

Analysieren: Das Konvertieren eines Strings in ein Programm, das aus einer Reihe von im String definierten Objekten besteht. Erfolgt nomalerweise als Aktion von ENTER für die Befehlszeile.

Analytische Funktion: Eine Funktion, die nach ihren Argumenten abgeleitet oder gelöst werden kann.

Anfangsnäherung: Ein oder mehrere Zahlenwerte, die der Lösungsroutine zum Auffinden von Nullstellen vorgegeben werden, um den Bereich einer möglichen Nullstelle einzugrenzen.

Argument: Ein Objekt, das als Eingabe für eine Operation vom Stack genommen wird.

Assoziieren: Die Neuanordnung der Anwendung von zwei Funktionen auf ihre drei Argumente, ohne den Wert des Ausdrucks zu verändern—zum Beispiel wird (a + b) + c in a + (b + c) umgeordnet. (In UPN Form wird a b + c + in a b c + + umgeordnet.)

Auflösung: In einer Abbildung der Abstand der Abszissenpunkte, für welche die Ordinatenwerte berechnet werden. Eine Auflösung von 1 bedeutet jeder Punkt, 2 bedeutet jeder zweiter Punkt, usw.

Ausdruck: Ein algebraisches Objekt, das kein Gleichheitszeichen (=) enthält.

Ausführen: Das Auswerten einer Prozedur (oder Teile derselben), einschließlich der HP-28S Operationen, welche als Prozedur-Objekte im ROM des Rechners gespeichert sind.

Ausnahme: Eine spezielle Art von mathematischem Fehler, für den Sie mittels eines Benutzerflags wählen können, ob der Rechner ein Standardergebnis oder eine Fehlermeldung zurückgeben soll.

Auswertung: Die fundamentale Operation auf Ihrem Rechner. (1) Die Auswertung eines Daten-Objekts gibt das Daten-Objekt in den Stack zurück. (2) Die Auswertung eines Namen-Objekts gibt das in einer zugehörigen Variablen gespeicherte Objekt zurück und, falls es sich um einen Namen oder Programm handelt, bewirkt die Auswertung. (3) Die Auswertung eines Prozedur-Objekts gibt jedes Objekt, welches in der Prozedur enthalten ist, zurück; besteht das Objekt aus einem Befehl oder einem nicht in Anführungszeichen stehenden Namen, so erfolgt dessen Auswertung.

Basis: Das Zahlensystem, in welchem die Binärwerte angezeigt werden. Es gibt die Auswahl zwischen binärer (Basis 2), oktaler (Basis 8), dezimaler (Basisi 10) und hexadezimaler Basis (Basis 16).

Basiseinheit: Eine der sieben Einheiten, welche im HP-28S für die Konvertierung von Einheiten benutzt wird. Bei den Basiseinheiten handelt es sich im einzelnen um Meter (Länge), Kilogramm (Masse), Sekunde (Zeit), Ampere (elektr. Strom), Kelvin (Temperatur), Candela (Leuchtstärke) und Mol (Stoffmenge).

Befehl: Jede HP-28S Operation, welche in die Definition einer Prozedur eingeschlossen werden kann oder die durch einen Namen in der Befehlszeile enthalten sein kann.

Befehlszeile: Der über das Tastenfeld eingetippte String, der nicht sofort-ausführbare Zeichen, Zahlen, Objekte, Befehle, usw. enthält. ENTER bewirkt die Konvertierung dieses Strings in ein Programm und dessen Ausführung.

Begrenzungszeichen: Ein Zeichen, das den Anfang oder das Ende des Objekt-Anzeige- bzw. Befehlsformats definiert: `'`, `"`, `#`, `⌊`, `⌋`, `{`, `}`, `(`, `)`, `«` oder `»`.

Beliebiges Vorzeichen: Die Variable *s1*, *s2*, usw., die in der Lösung für einen Ausdruck mit mehrfachen Nullstellen auftritt. Unterschiedliche Nullstellen werden dadurch erzeugt, indem $+1$ oder -1 in den Variablen gespeichert wird.

Benutzerflag: Ein Ein-Bit Speicherort mit dem Inhalt 1 oder 0, wobei der Inhalt getestet werden kann. Der HP-28S enthält 64 Benutzerflags, die von 1 bis 64 durchnumeriert sind.

Benutzerschnittstelle: Die Prozeduren, Tastenfolgen, Anzeigen, usw., wodurch der Benutzer interaktiv mit dem Rechner arbeitet.

Benutzerspeicher: Der Speicherbereich, in welchem die Benutzervariablen gespeichert sind.

Binärwert: Ein Objekt, das durch das Begrenzungszeichen # identifiziert wird und mittels 1 bis 64 Bits eine ganze Zahl darstellt (in Abhängigkeit zur momentanen Zahlenbasis).

COMMAND-Stack: Bis zu vier vorher eingegebene Befehlszeilen, die zur späteren Rücksicherung durch COMMAND gespeichert wurden.

Cursor: Ein Zeichen in der Anzeige, welches eine Position markiert. (1) In der Befehlszeile zeigt der Cursor die Stelle an, wo das nächste Zeichen in der Befehlszeile erscheint. Das Erscheinungsbild wechselt in Abhänigkeit zum momentanen Eingabemodus. (2) Der FORM-Cursor identifiziert den gewählten Teilausdruck durch inverse Hervorhebung. (3) Der DRAW/DRWΣ-Cursor besteht aus einem kleinen Kreuz, das einen zu digitalisierenden Punkt markiert.

Daten-Objekt: Ein Objekt, welches nach seiner Auswertung selbst wieder im Stack gespeichert wird. Es beinhaltet reelle und komplexe Zahlen, Felder, Strings, Binärwerte und Listen.

Dezimalzeichen: Das Trennzeichen zwischen dem ganzzahligen Anteil und dem Dezimalteil einer Zahl.

Distribution: Verallgemeinerte Funktion, die sich durch die Erweiterung des mathematischen Funktionsbegriffs ergibt. Zum Beispiel zeigt die Distribution des + Operators: $a \times (b + c)$ ergibt $(a \times b) + (a \times c)$.

Ebene: (1) Eine Position im Stack, in welcher ein Objekt gespeichert werden kann. (2) Die Position von einem Teilausdruck unter der Hierarchie eines algebraischen Ausdrucks.

Eingabemodus: Der Rechnermodus, welcher darüber entscheidet, ob das Drücken einer Taste die unmittelbare Befehlsausführung oder nur die Eingabe in die Befehlszeile bewirkt. Der Eingabemodus kann aus dem unmittelbaren, algebraischen oder Alpha-Modus bestehen.

Einheiten-String: Ein String, der die physikalischen Einheiten mit einem zugehörigen reellen Zahlenwert darstellt. Ein Einheiten-String kann die Namen von Einheiten, Potenzen, Produkte und ein Verhältnis enthalten.

Einzelschritt: Das Ausführen eines Objekts oder einer Struktur innerhalb der Programmdefinition.

Ergebnisse: Die durch Befehle in den Stack zurückgegebenen Objekte.

Exponent: Die bei der Darstellung einer Gleitkommazahl in der exponentiellen Notation eingeschlossene Zehnerpotenz; speziell die ein-, zwei- oder dreistellige Zahl, welche dem "E" bei der Anzeige folgt. Der Exponent von x lautet IP (LOG (x)).

Exponentielle Notation: Die Darstellung einer Zahl mit Vorzeichen, einer Mantisse zwischen 1 und 9.99999999999, und einem Exponenten "E", dem eine mit Vorzeichen versehene dreistellige ganze Zahl folgt.

Faktor: Einer der Argumente von * (Multiplikation).

Falsch: Der Wert eines Flags, durch die reelle Zahl 0 dargestellt.

Fehler: Jeder Ausführungsfehler, der durch einen mathematischen Fehler, die Nicht-Übereinstimmung von Argumenten, zu wenig freien Speicherbereich, usw. verursacht wird und damit die normale Ausführung unter Anzeige einer Fehlermeldung abbricht.

Feld: Ein Objekt, welches durch die [ ] Begrenzungszeichen definiert wird und eine reelle oder komplexe Matrix darstellt.

Flag: Eine reelle Zahl, die als Indikator zur Festlegung einer wahren/falschen Entscheidung verwendet wird. Die Zahl 0 stellt den Zustand bzw. die Entscheidung *falsch* dar; jede andere Zahl, gewöhnlich +1, stellt *wahr* dar.

Funktion: Eine HP-28S Operation, die in die Definition eines algebraischen Objekts mit eingeschlossen werden kann. Verschiedene Funktionen benutzen bis zu drei Argumente, geben jedoch immer nur ein Ergebnis zurück.

Genauigkeit: Die numerische Genauigkeit des Integranden, wodurch die Anzahl der Teilintervalle für die Berechnung des Integrals festgelegt wird.

Gleichung: Ein algebraisches Objekt, welches aus zwei Ausdrücken besteht, die durch ein einzelnes Gleichheitszeichen (=) verbunden sind.

Hauptwert: Eine bestimmte Wahl unter den mehrfachen Werten einer mathematischen Relation bzw. Lösung, die zur Eindeutigkeit und Einfachheit getroffen wurde. So gibt z.B. ASIN (.5) den Winkel 30° zurück, der den Hauptwert des allgemeineren Ergebnisses $(-1)^n\ 30° + 180n°$ darstellt, wobei n eine beliebige ganze Zahl ist.

Hierarchie: Die Struktur eines mathematischen Ausdrucks, der in eine Reihe von Teilausdrücken untergliedert werden kann, von denen wiederum jeder als Argument für eine Funktion benutzt werden kann.

HMS Format: Das Format für eine reelle Zahl, deren Ziffern links vom Dezimalzeichen die ganzen Stunden (oder Grad) darstellen, und deren ersten zwei Ziffern rechts des Dezimalzeichens die Minuten (Winkel oder Zeit) darstellen, während die restlichen rechten Ziffern die Bruchteile von Sekunden repräsentieren.

Indikatoren: Die Symbole über Zeile 1 der Anzeige; sie zeigen die Status verschiedener Rechnermodi an.

Inhalt: Das in einer Variablen gespeicherte Objekt; ein weiterer Begriff dafür ist der *Wert* einer Variablen.

Inverse: (1) Das Ergebnis der Inversion auf eine Zahl oder ein Feld. (2) Eine Funktion, welche nach der Anwendung auf eine zweite Funktion das Argument der zweiten Funktion zurückgibt. Demzufolge ist SIN das inverse zu ASIN.

Kommutieren: Das Austauschen zweier Argumente einer Funktion.

Komplexe Zahl: Ein Objekt, das durch die Begrenzungszeichen ( ) identifiziert wird und das aus zwei reellen Zahlen besteht, die den reellen und komplexen Anteil einer komplexen Zahl darstellen.

Komplexes Feld: Feld, dessen Elemente aus komplexen Zahlen bestehen.

Konform: Zwei Felder, die zur Durchführung einer arithmetischen Operation die geeignete Dimension besitzen.

Koordinatenpaar: Die Koordinaten eines Punktes im zweidimensionalen Raum, welche als komplexes Zahlen-Objekt gespeichert sind. Der reelle Anteil entspricht dem "horizontalen" Koordinatenwert, und der imaginäre Anteil stellt den "vertikalen" Koordinatenwert dar.

Liste: Ein Daten-Objekt, welches aus der Zusammenfassung von anderen Objekten besteht.

Lokale Variable: Eine Variable, die durch den Befehl → oder FOR für den temporären Gebrauch innerhalb einer Programmstruktur erzeugt wurde. Die Variable wird automatisch gelöscht, wenn die Prozedur ihre Ausführung abgeschlossen hat.

Lokaler Name: Ein Namen-Objekt, das eine lokale Variable benennt. Lokale Namen bestehen aus einem anderen Objekttyp (Typ 7) als gewöhnliche Namen (Typ 6). Die Auswertung eines lokalen Namens ergibt den Inhalt der zugehörigen lokalen Variablen (unausgewertet).

Löschen: (1) Zum löschen des Stacks (CLEAR). (2) Um die Anzeige zu löschen (CLLCD). (3) Um einem Benutzerflag den Wert 0 zuzuordnen (CF).

Lösen: Das Auffinden der Nullstelle eines Ausdrucks oder einer Gleichung.

Löser: Die eingebaute Lösungsroutine des HP-28S, die aufgrund der Definition für die momentane Gleichung ein Variablenmenü erzeugt; damit wird Ihnen ermöglicht, Werte für die einzelnen Variablen zu speichern und die Gleichung nach einer beliebigen Variablen zu lösen.

Lösung: Äquivalent zu *Nullstelle*.

Matrix: Ein zweidimensionales Feld.

Meldungsflag: Ein interner Flag, der bestimmt, ob die normale Stackanzeige nach dem Abschluß aller anstehenden Operationen wieder angezeigt werden soll. Der Meldungsflag wird durch Fehler und Befehle, die eine spezielle Anzeige erzeugen, gesetzt.

Memory Reset: Das "Zurücksetzen" des Rechners, wodurch alle Rechnermodi und alle Teile des Speicherbereichs auf ihre Voreinstellungswerte zurückgesetzt werden, einschließlich dem Löschen des Stacks, des COMMAND Stacks, des UNDO Stacks, der LAST Argumente und der Benutzervariablen.

Menü-Auswahltaste: Jede Taste, die ein Menü von Operationen aktiviert, die danach durch Drücken der gewünschten Menütaste ausgeführt werden kann.

Menü: Eine Sammlung von Operationen mit gemeinsamen Eigenschaften, die den jeweiligen Menütasten zugeordnet sind.

Menütasten: Die sechs unbeschrifteten Tasten unterhalb der Anzeige; ihre Operation wird durch das am unteren Anzeigerand erscheinende aktive Menü bestimmt.

Modus: Ein Rechnerstatus, der das Verhalten einer oder mehrerer Operationen außer durch die expliziten Argumente der Operation beeinflußt.

Momentane Gleichung: Die in der Variablen EQ gespeicherte Prozedur; sie wird als implizites Argument von DRAW und dem Löser benutzt.

Momentane Statistikmatrix: Die in der Variablen ΣDAT gespeicherte Matrix; sie enthält die durch Σ+ gesammelten Statistik-Daten.

Nachiteration: Ein Prozeß von aufeinanderfolgenden Näherungen an die Lösung eines Gleichungssystems.

Name: Ein Objekt, das aus einer Zeichensequenz besteht, die einen Variablennamen darstellt. (1) Die Auswertung eines Namen-Objekts ergibt das Objekt, das in der zugehörigen Variablen gespeichert war, sowie dessen Auswertung, sofern es sich dabei um einen Namen oder ein Programm handelt. (2) Die Auswertung eines lokalen Namens ergibt das Objekt, das in der zugehörigen Variablen gespeichert war.

Nullstelle: Ein Variablenwert, unter welchem der Ausdruck den Wert 0 annimmt oder beide Seiten einer Gleichung äquivalent sind.

Numerisches Objekt: Eine reelle/komplexe Zahl bzw. ein reelles/komlexes Feld.

Objekt: Das grundlegende Element einer Rechneroperation. Daten-Objekte stellen Einheiten dar, die einen einfachen "Wert" besitzen; Namen-Objekte dienen zum Benennen von Variablen, die andere Objekte enthalten; Prozedur-Objekte stellen Sequenzen von Objekten und Befehlen dar.

Operation: Jede eingebaute und für den Benutzer zugängliche Fähigkeit des HP-28S, nicht-programmierbare Tastenfolgen und programmierbare Befehle eingeschlossen.

Overflow: Eine mathematische Ausnahme, die sich aus einer Berechnung ergibt, deren Resultat zur Darstellung im HP-28S zu groß ist.

Pixel: Ein einzelner LCD Bildpunkt.

Polnische Notation: Eine mathematische Notation, in welcher alle Funktionen und Operationen in vorangestellter Form erfolgen. In dieser Notation wird "1 plus 2" als "+(1, 2)" dargestellt.

Programm: Eine in UPN-Logik definierte Prozedur, welche durch die Begrenzungszeichen « » identifiziert wird.

Programmstruktur: Ein Befehlssatz, der einer speziellen Sequenz in einem Programm folgen muß. Anweisungen, die von Befehlen abgegrenzt werden, welche logische Einheiten zum Treffen von Entscheidungen und zur Programmverzweigung umfassen.

Quadratische Form: Ein Polynom zweiten Grades in einer spezifizierten Variablen.

Qualifizierende Meldung: Eine vom Löser angezeigte Meldung, die Informationen über die ermittelte Lösung bietet.

Rangordnung: Regeln, welche die Ausführungsreihenfolge von Operationen in Ausdrücken, in denen das Weglassen von Klammern ansonsten die Reihenfolge zweideutig machen würde, festlegen.

Rechner für direkte Gleichungseingabe: Ein Rechner, auf dem Sie numerische Berechnungen durch die direkte Eingabe einer Gleichung (in herkömmlicher mathematischer Notation) durchführen, ohne dabei Zwischenergebnisse zu erhalten.

Reelle ganze Zahl: Eine reelle Zahl, die als Argument für einen Befehl benutzt wird, der auf reelle ganze Zahlen angewendet werden kann.

Reelle Zahl: Ein Objekt, das aus einer einzelnen reellwertigen Gleitkommazahl besteht, die im dezimalen Zahlensystem angezeigt wird.

Reelles Feld: Ein Feld-Objekt, welches ausschließlich reelle Werte als Elemente enthält.

Reihenfolge: Die Anordnung von Matrixelementen, von links nach rechts und von der obersten Zeile zur untersten.

Setzen: Um den Inhalt *wahr* bzw. einen Wert ungleich Null einem Flag zuzuordnen.

Speicherarithmetik: Das Durchführen von arithmetischen Operationen auf den Inhalt von Variablen, ohne daß dabei deren Inhalt in den Stack zurückgerufen wird.

Stack: Die Zusammenfassung der Objekte, die über eine "last-in, first-out" Speicherorganisation im HP-28S gespeichert sind. Dieser Speicherbereich wird mit Stack bezeichnet und liefert die Schnittstelle zur Verarbeitung der Argumente von Befehlen und deren Ergebnisse.

Stackdiagramm: Eine tabellarische Zusammenstellung der Argumente und Ergebnisse eines Befehls; dabei wird die Bedeutung und die Position der Argumente und Ergebnisse im Stack dargestellt.

Starre Stackorganisation: Ein UPN Rechner mit einer festen Anzahl von (normalerweise) vier Ebenen.

Statistisches Streudiagramm: Die Abbildung von Datenpunkten mittels DRWΣ, aufgrund der in der Statistikmatrix gespeicherten Werte.

Statusmeldung: Eine vom HP-28S angezeigte Meldung, die Sie über den momentanen Status des Rechners (keine Fehlerbedingung) informieren soll.

Steigung: Die Steigung einer Geraden, die durch lineare Regression erhalten wurde.

String: Ein Objekt, das eine beliebige Zeichensequenz (Buchstaben, Ziffern oder andere Symbole) enthält und von den Begrenzungszeichen " eingeschlossen ist.

Summand: Eines der Argumente von + (Addition).

Symbolisch: Die Darstellung eines Werts durch einen Namen oder Symbol anstatt eines expliziten numerischen Ausdrucks.

Symbolische Konstante: Jedes der fünf Objekte e, i, π, MAXR und MINR, die in Abhängigkeit der Flags 35 und 36 entweder in ihre numerischen Werte ausgewertet werden oder ihre symbolische Form beibehalten.

Symbolischer Modus: Der Rechnermodus, in welchem die Funktionen von symbolischen Argumenten symbolische Ergebnisse zurückgeben.

Systemstopp: Ein Initialisierungsprozeß, durch den alle anstehenden Operationen abgebrochen werden und der Stack gelöscht wird.

Tasten-Puffer: Ein bestimmter Teil des Speicherbereichs, in welchem bis zu 15 anstehende Tastencodes gespeichert werden können; diese Tasten wurden zwar bereits gedrückt, aber noch nicht verarbeitet.

Teilausdruck: Ein Teil eines algebraischen Ausdrucks, der aus einer Zahl, einem Namen oder einer Funktion und deren Argumente besteht. Jeder Teilausdruck kann andere Teilausdrücke als Argumente enthalten und kann selbst das Argument eines anderen Teilausdrucks sein.

Test: Zur Herbeiführung einer Programmverzweigung, die vom Wert eines Flags abhängt.

Umgekehrte polnische Notation: Eine Modifikation der polnischen Notation, in welcher die Funktionen ihren Argumenten folgen: 1 2 + bedeutet 1 plus 2. Diese mathematische Notation entspricht der Rechnerschnittstelle, bei welcher Funktionen ihre Werte vom Stack nehmen und ihre Ergebnisse in den Stack zurückgeben.

Unabhängige Variable: Eine Variable, deren Wert willkürlich gesetzt anstatt durch die Werte anderer Variablen berechnet werden kann. Beim Erzeugen einer graphischen Abbildung entspricht sie der horizontalen Koordinate; beim Löser ist die unabhängige Variable diejenige, welche keine Prozeduren mit Namen in ihrer Definition enthält.

Unbekannte: Die Variable, für welche der Löser, ROOT, QUAD oder ISOL versucht, eine numerische oder symbolische Lösung aufzufinden.

Underflow: Eine mathematische Ausnahme, die sich aus einer Berechnung ergibt, deren Resultat ungleich Null und zur Darstellung im HP-28S zu klein ist.

Unendliches Ergebnis: Eine mathematische Ausnahme, die aus einer Operation wie z.B. Division durch 0 entsteht.

Unterbrochenes Programm: Ein Programm, dessen Ausführung durch den Befehl HALT unterbrochen wurde und mit SST oder ■ CONT fortgesetzt werden kann.

UPN: Umgekehrte polnische Notation.

Variable: Die Kombination eines Namen-Objekts (der Name der Variablen) mit einem beliebigen anderen Objekt (der Wert der Variablen), wobei beide zusammen gespeichert werden.

Variablenmenü: Das vom Löser erzeugte Menü, in welchem jede in der momentanen Gleichung enthaltene Variable dargestellt ist.

Vektor: Ein eindimensionales Feld.

Vereinfachung: Das Neuschreiben eines algebraischen Ausdrucks in der Form, daß der Orginalwert des Ausdrucks erhalten bleibt, der Ausdruck selbst jedoch einfacher erscheint. Vereinfachung kann das Kombinieren von Termen oder die teilweise Auswertung des Ausdrucks beinhalten.

Wahr: Der Wert eines Flags, durch eine reelle Zahl ungleich 0 dargestellt. Wenn ein Befehl einen Wahr-Flag zurückgibt, enthält dieser die Zahl 1.

Wert: Der numerische, symbolische oder logische Inhalt eines Objekts. Wenn auf Variablen Bezug genommen wird, ist mit *Wert* das in der Variable gespeicherte Objekt gemeint.

Wertebereich: Der Bereich von Argumenten, für die eine Funktion definiert ist.

Wortlänge: Die Anzahl von Bits, auf welche die Ergebnisse von Binärwert-Operationen abgeschnitten werden.

Y-Achsenabschnitt: Die Stelle der Ordinate, an welcher die durch lineare Regression bestimmte Gerade die Ordinate schneidet.

Zurückrufen: Das Abrufen eines in einer Variablen gespeicherten Objekts.

Verzeichnis der Operationen

Dieses Verzeichnis enthält grundlegende Informationen und Referenzen über Operationen im HP-28C/S. Für jede der Operationen werden folgende Informationen angezeigt:

Name: Bei Operationen das zugeordnete Menüfeld. Für Befehle die entsprechende Darstellung in der Befehlszeile.

Bedeutung: Was die Operation bewirkt.

Typ: Wo Sie die Operation anwenden können und was die korrespondierenden Tasten bewirken. Diese Information ist über die folgenden Codes verschlüsselt:

Code	Bedeutung
A	Analytische Funktion, deren Nullstelle gesucht und die abgeleitet werden kann.
F	Funktion. In algebraischen Objekten und Programmen enthalten.
C	Befehl (*Command*). Kann in Programmen, aber nicht in algebraischen Objekten enthalten sein.
O	Operation. Kann nicht in Befehlszeile oder Prozedur enthalten sein.
*	Taste führt im unmittelbaren Eingabemodus kein ENTER durch.
†	Taste fügt den Befehl immer der Befehlszeile hinzu.

Ein: Wie viele Objekte im Stack vorausgesetzt sind. (Dieser Eintrag entfällt für die Operatonen, die den Stack nicht benutzen.)

Aus: Wie viele Objekte in den Stack zurückgegeben werden. (Dieser Eintrag entfällt für die Operationen, die den Stack nicht benutzen.)

Beschrieben unter: Wo der Befehl in diesem Handbuch näher beschrieben ist.

HP-28C/S Verzeichnis der Operationen

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
ABORT	Bricht die Programmausführung ab	C	0	0	PROGRAM CONTROL	246
ABS	Absolutbetrag	F	1	1	ARRAY COMPLEX REAL	125 165 269
ACOS	Arcuscosinus	A	1	1	TRIG	324
ACOSH	Arcuscosinus hyperbolicus	A	1	1	LOGS	185
AF	Addiert Bruchteile (*Adds Fractions*)	O*			ALGEBRA (FORM)	89
ALGEBRA	Wählt das ALGEBRA-Menü	O*			ALGEBRA	59
ALOG	Antilogarithmus	A	1	1	LOGS	181
AND	Logisches oder binäres AND	F	2	1	BINARY PROGRAM TEST	138 258
ARG	Argument	F	1	1	COMPLEX TRIG	165 329
ARRAY	Wählt das ARRAY-Menü	O*			ARRAY	106
ARRY→	Ersetzt die Elemente eines Felds als separate Stack-Objekte	C	1	$n+1$	ARRAY	114
ASIN	Arcussinus	A	1	1	TRIG	323
ASINH	Arcussinus hyperbolicus	A	1	1	LOGS	184
ASR	Arithmetisches verschieben nach rechts	C	1	1	BINARY	137

ATAN	Arcustangens	A	1	1	TRIG	325
ATANH	Arcustangens hyperbolicus	A	1	1	LOGS	186
ATTN (ON)	Bricht die Programmausführung ab; löscht die Befehlszeile; beendet die Anzeige von Katalogen, FORM und graphischen Abbildungen	O*			Grundl. Operationen	53
AXES	Setzt den Koordinatenursprung	C	1	0	PLOT	208
A→	Assoziieren nach rechts	O*			ALGEBRA (FORM)	82
BEEP	Erzeugt ein Akustiksignal	C	2	0	PROGRAM CONTROL	249
BIN	Spezifiziert das Dualsystem als Zahlenbasis	C*	0	0	BINARY	133
BINARY	Wählt das BINARY-Menü	O*			BINARY	130
BRANCH	Wählt das PROGRAM BRANCH Menü	O*			PROGRAM BRANCH	232
B→R	Binär-in-reell Konvertierung	C	1	1	BINARY	135
CATALOG	Startet den COMMAND-Katalog	O*			CATALOG	156
CEIL	Nächste größere ganze Zahl	F	1	1	REAL	271
CENTR	Setzt das Zentrum für die Anzeige einer graphischen Abbildung	C	1	0	PLOT	209
CF	Löscht Benutzerflag	C	1	0	PROGRAM TEST	255
CHR	Erzeugt einen String mit einem Zeichen	C	1	1	STRING	317
CHS	Kehrt das Vorzeichen einer Zahl in der Befehlszeile um oder führt NEG aus	O*			Grundl. Operationen	32
CLEAR	Löscht den Stack	C	*n*	0	STACK	291

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
CLLCD	Löscht die Anzeige	C	0	0	PLOT PROGRAM CONTROL	212 249
CLMF	Löscht den Systemmeldungsflag	C	0	0	PLOT PROGRAM CONTROL	214 250
CLUSR	Löscht alle Benutzervariablen	C†	0	0	USER	347
CLΣ	Löscht die Statistikmatrix	C	0	0	STAT	299
■ CMPLX	Wählt das COMPLEX-Menü	O*			COMPLEX	160
CNRM	Berechnet die Spaltensummennorm	C	1	1	ARRAY	126
COLCT	Faßt ähnliche Terme zusammen	C	1	1	ALGEBRA	71
COLCT	Faßt ähnliche Terme in einem Teilausdruck zusammen	O*			ALGEBRA (FORM)	80
COLΣ	Wählt die Spalten der Statistikmatrix	C	2	0	PLOT STAT	211 302
■ COMMAND	Verschiebt einen Eintrag vom Befehlsstack in die Befehlszeile	O			Grundl. Operationen	48
CON	Erzeugt eine Konstantenmatrix	C	2	0, 1	ARRAY	121
CONJ	Konjugiert komplex	F	1	1	ARRAY COMPLEX	128 163
■ CONT	Setzt unterbrochenes Programm fort	O			PROGRAM CONTROL	244

CONVERT	Führt die Konvertierung zweier Einheiten durch	C	3	2	UNITS	332
CORR	Korrelationskoeffizient	C	0	1	STAT	303
COS	Cosinus	A	1	1	TRIG	324
COSH	Cosinus hyperbolicus	A	1	1	LOGS	185
COV	Kovarianz	C	0	1	STAT	304
CR	Druckt Pufferinhalt und bewegt Druckkopf nach rechts	C	0	0	PRINT	222
CROSS	Kreuzprodukt zweier Vektoren	C	2	1	ARRAY	124
■ CTRL	Wählt das PROGRAM CONTROL Menü	O*			PROGRAM CONTROL	243
C→R	Komplex-in-reell Konvertierung	C	1	2	ARRAY COMPLEX TRIG	127 162 328
■ d/dx	Ableitung (∂ Funktion)	F	2	1	Höhere Mathematik	141
DEC	Spezifiziert Dezimalsystem als Zahlenbasis	C*	0	0	BINARY	132
DEG	Spezifiziert Grad für Winkelmodus	C*	0	0	MODE	191
DEL	Löscht Zeichen an der Cursorposition; digitalisiert Punkt	O*	0	0, 2	Grundl. Operationen PLOT	39 203
■ DEL	Löscht Zeichen an Cursorposition und alle Zeichen rechts davon	O*			Grundl. Operationen	39
DEPTH	Zählt die Objekte im Stack	C	0	1	STACK	294
DET	Determinante einer Matrix	C	1	1	ARRAY	125

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
DINV	Doppelte Inversion	O*			ALGEBRA (FORM)	86
DISP	Anzeige eines Objekts	C	2	0	PLOT PROGRAM CONTROL STRING	213 250 320
DNEG	Doppelte Negation	O*			ALGEBRA (FORM)	85
DO	Teil von DO...UNTIL...END.	C†			PROGRAM BRANCH	233
DOT	Skalares Produkt zweier Vektoren	C	2	1	ARRAY	124
DRAW	Erzeugt die graph. Abbildung einer Funktion	C	0	0	PLOT	206
DRAX	Zeichnet die Achsen des Koordinatensystems	C	0	0	PLOT	213
DROP	Löscht Inhalt der untersten Stackebene und verschiebt Stackinhalt um eine Ebene nach unten	C	1	0	STACK	290
DROPN	Löscht $n+1$ Objekte vom Stack	C	$n+1$	0	STACK	294
DROP2	Löscht zwei Objekte vom Stack	C	2	0	STACK	292
DRWΣ	Erzeugt ein statistisches Streudiagramm	C	0	0	PLOT	212
DUP	Dupliziert ein Objekt im Stack	C	1	2	STACK	291
DUPN	Dupliziert n Objekte im Stack	C	$n+1$	$2n$	STACK	294
DUP2	Dupliziert zwei Objekte im Stack	C	2	4	STACK	292
D→	Distribuiert nach rechts	O*			ALGEBRA (FORM)	83

D→R	Grad-in-Bogenmaß Konvertierung	F	1	1	TRIG	331
e	Symbolische Konstante *e*	F†	0	1	ALGEBRA REAL	70 266
■ EDIT	Kopiert das Objekt von Ebene 1 in die Befehlszeile	O	1	1	Grundl. Operationen	41
EEX	Gibt den Exponenten in die Befehlszeile ein	O*			Grundl. Operationen	32
ELSE	Startet die ELSE-Anweisung	C†			PROGRAM BRANCH	232
END	Beendet eine Programmstruktur	C†	1	0	PROGRAM BRANCH	232 233
ENG	Setzt das technische Anzeigeformat	C	1	0	MODE	191
ENTER	Analysiert und wertet die Befehlszeile aus oder führt DUP aus	O			Grundl. Operationen	40
ERRM	Gibt die letzte Fehlermeldung zurück	C	0	1	PROGRAM CONTROL	251
ERRN	Gibt die letzte Fehlernummer zurück	C	0	1	PROGRAM CONTROL	251
EVAL	Wertet ein Objekt aus	C	1	0	Grundl. Operationen	42
EXGET	Holt einen Teilausdruck	C	2	1	ALGEBRA	76
EXGET	Holt einen Teilausdruck	O*		2	ALGEBRA (FORM)	80
EXP	Natürliche Exponentialfunktion	A	1	1	LOGS	182
EXPAN	Erweitert einen algebraischen Ausdruck	C	1	1	ALGEBRA	72
EXPAN	Erweitert einen Teilausdruck	O*			ALGEBRA (FORM)	80

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
EXPM	Natürliche Exponentialfunktion minus 1	A	1	1	LOGS	183
EXPR=	Auswertung der momentanen Gleichung	O	0	1	SOLVE	278
EXSUB	Substituiert einen Teilausdruck	C	3	1	ALGEBRA	75
E^	Ersetzt Potenz-von-Produkt durch Potenz-von-Potenz	O*			ALGEBRA (FORM)	89
E()	Ersetzt Potenz-von-Potenz durch Potenz-von-Produkt	O*			ALGEBRA (FORM)	89
FACT	Fakultät (Gammafunktion)	F	1	1	REAL	267
FC?	Testet einen Benutzerflag	C	1	1	PROGRAM TEST	256
FC?C	Testet und löscht einen Benutzerflag	C	1	1	PROGRAM TEST	257
FETCH	Schließt CATALOG oder UNITS ab und übernimmt den Befehl bzw. die Einheit in die Befehlszeile	O*			CATALOG UNITS	336 157
FIX	Spezifiziert feste Anzahl Dezimalstellen für Anzeigeformat	C	1	0	MODE	189
FLOOR	Nächste kleinere ganze Zahl	F	1	1	REAL	271
FOR	Startet eine bestimmte Schleife	C†	2	0	PROGRAM BRANCH	233
FORM	Ändert die Form eines algebraischen Ausdrucks	C	1	1, 3	ALGEBRA ALGEBRA (FORM)	74 77
FP	Dezimalteil	F	1	1	REAL	271
FS?	Testet einen Benutzerflag	C	1	1	PROGRAM TEST	256

FS?C	Testet und löscht einen Benutzerflag	C	1	1	PROGRAM TEST	256
GET	Holt ein Element von einem Objekt	C	2	1	ARRAY LIST	116 176
GETI	Holt ein Element von einem Objekt und erhöht den Index	C	2	3	ARRAY LIST	118 178
HALT	Unterbricht die Programmausführung	C†			PROGRAM CONTROL	246
HEX	Spezifiziert Hexadezimalsystem als Zahlenbasis	C*	0	0	BINARY	132
HMS+	Addiert im HMS Format	C	2	1	TRIG	331
HMS−	Subtrahiert im HMS Format	C	2	1	TRIG	331
HMS→	Konvertiert vom HMS Format	C	1	1	TRIG	330
i	Symbolische Konstante *i*	F†	0	1	ALGEBRA	70
IDN	Erzeugt eine Einheitsmatrix	C	1	0, 1	ARRAY	122
IF	Beginnt IF Struktur	C†	0	0	PROGRAM BRANCH	232
IFERR	Beginnt IF ERROR Struktur	C†	0	0	PROGRAM BRANCH	232
IFT	IF-Then Anweisung	C	2	0	PROGRAM BRANCH	241
IFTE	IF-Then-Else Funktion	F	3	0	PROGRAM BRANCH	241
IM	Gibt den Imaginärteil einer Zahl oder eines Felds zurück	F	1	1	ARRAY COMPLEX	128 162
INDEP	Wählt die unabhängige Variable für die Abbildung	C	1	0	PLOT	206

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter
INS	Schaltet zwischen Einfügungs- und Ersetzungsmodus um; digitalisiert Koordinatenpunkt	O*	0	0, 1	Grundl. Operationen 39 PLOT 203
■ INS	Löscht alle Zeichen links der Cursorposition	O*			Grundl. Operationen 39
INV	Inversion (Reziprokwert)	A	1	1	Arithmetik 103 ARRAY 113
IP	Ganzzahliger Anteil	F	1	1	REAL 270
ISOL	Löst einen Ausdruck oder eine Gleichung	C	2	1	ALGEBRA 76 SOLVE 285
KEY	Gibt einen String mit der Tastenfolge zurück	C	0	1, 2	PROGRAM CONTROL 247
KILL	Beendet alle unterbrochenen Programme	C	0	0	PROGRAM CONTROL 247
LAST	Gibt das zuletzt benutzte Argument zurück	C	0	1, 2, 3	Grundl. Operationen 49
LC	Schaltet zwischen Groß- und Kleinschreibmodus um	O*			Grundl. Operationen 33
LEFT=	Wertet die linke Seite der momentanen Gleichung aus	O	0	1	SOLVE 278
LEVEL	Zeigt die Ebene des ausgewählten Teilausdrucks an	O*			ALGEBRA (FORM) 80
■ LIST	Wählt das LIST-Menü	O*			LIST 174
LIST→	Schiebt die Listenelemente in den Stack	C	1	$n+1$	LIST 175 STACK 293

LN	Natürlicher Logarithmus	A	1	1	LOGS	182
LNP1	Natürlicher Logarithmus plus 1 (*Argument*+1).	A	1	1	LOGS	183
LOG	Dekadischer Logarithmus	A	1	1	LOGS	181
■ LOGS	Wählt das LOGS-Menü	O*			LOGS	181
LR	Berechnet die lineare Regression	C	0	2	STAT	304
L( )	Ersetzt Produkt-von-Log durch Log-von-Potenz	O*			ALGEBRA (FORM)	88
L*	Ersetzt Log-von-Potenz durch Produkt-von-Log	O*			ALGEBRA (FORM)	88
MANT	Gibt die Mantisse einer Zahl zurück	F	1	1	REAL	270
MAX	Gibt das Maximum zweier Zahlen zurück	F	2	1	REAL	272
MAXR	Symbolische Konstante für reelles Maximum	F	0	1	ALGEBRA REAL	70 268
MAXΣ	Findet die Koordinatenwerte für Maximum in der Statistikmatrix	C	0	1	STAT	302
MEAN	Berechnet den statistischen Mittelwert	C	0	1	STAT	300
MEM	Gibt die Größe des verfügbaren Speicherbereichs zurück	C	0	1	USER	347
MIN	Gibt das Minimum zweier Zahlen zurück	F	2	1	REAL	273
MINR	Symbolische Konstante von reellem Minimum	F	0	1	ALGEBRA REAL	70 268
MINΣ	Findet die Koordinatenwerte für Minimum in der Statistikmatrix	C	0	1	STAT	302

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
MOD	Modulo-Funktion	F	2	1	REAL	273
■ MODE	Wählt das MODE-Menü	O*			MODE	187
M→	Führt rechte Faktoren zusammen	O*			ALGEBRA (FORM)	85
NEG	Negiert ein Argument	A	1	1	Arithmetik ARRAY COMPLEX REAL	104 129 165 266
NEXT	Beendet bestimmte Schleife	C†	0	0	PROGRAM BRANCH	233
NEXT	Zeigt die nächste Zeile der Menüfelder	O*				57
NEXT	Geht zum nächsten Befehl bzw. zur nächsten Einheit des Katalogs über	O*			CATALOG UNITS	157 336
NEXT	Geht zur nächsten Option eines Arguments von USAGE über	O*			CATALOG	158
NO	Wahl, während der Bedingung `Out of Memory` nichts zu löschen	O*			Grundl. Operationen	52
NORM	Desaktiviert Trace-Modus für Drucker	O*			PRINT	221
NOT	Logisches oder binäres NOT	F	1	1	BINARY PROGRAM TEST	140 260
NUM	Gibt den Zeichencode zurück	C	1	1	STRING	317
NΣ	Gibt die Anzahl der Datenpunkte in der Statistikmatrix zurück	C	0	1	STAT	299

OBGET	Holt einen Objekt aus einem algebraischen Ausdruck	C	2	1	ALGEBRA	76
OBSUB	Substituiert ein Objekt in einem algebraischen Ausdruck	C	3	1	ALGEBRA	75
OCT	Spezifiziert das Oktalsystem als Zahlenbasis	C*	0	0	BINARY	132
ON (ATTN)	Schaltet den Rechner ein; bricht Programmausführung ab; löscht die Befehlszeile; beendet die Anzeige von Katalogen, FORM, und graphischen Abbildungen	O*			Grundl. Operationen	53
ON DEL	Hebt den Systemstopp oder den Memory Reset auf, wenn vor dem Freigeben von ON gedrückt	O*			Grundl. Operationen	55
ON INS ▶	(Memory Reset) Bricht die Programmausführung ab, löscht lokale Variable und Benutzervariable und setzt die Benutzerflags zurück	O*			Grundl. Operationen	54
ON +	Erhöht den Anzeigekontrast	O*			Grundl. Operationen	53
ON −	Verringert den Anzeigekontrast	O*			Grundl. Operationen	53
ON ▲	(Systemstopp) Bricht die Programmausführung ab, löscht lokale Variable und den Stack	O*			Grundl. Operationen	54
ON ▼	Startet den Systemtest	O*			Grundl. Operationen	55
ON ◀	Startet den fortlaufenden Systemtest	O*			Grundl. Operationen	55
■ OFF	Schaltet den Rechner aus	O*			Grundl. Operationen	53
OR	Logisches oder binäres OR	F	2	1	BINARY PROGRAM TEST	139 258
ORDER	Ordnet das Benutzer-Menü neu	C	1	0	USER	346
OVER	Dupliziert das Objekt in Ebene 2	C	2	3	STACK	292

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
PICK	Dupliziert das *n*-te Objekt	C	*n*+1	*n*+1	STACK	293
PIXEL	Schaltet einen Pixel an	C	1	0	PLOT	213
■ PLOT	Wählt das PLOT-Menü	O*			PLOT	198
PMAX	Spezifiziert die Abbildungskoordinaten für rechts oben	C	1	0	PLOT	206
PMIN	Spezifiziert die Abbildungskoordinaten für links unten	C	1	0	PLOT	205
POS	Sucht einen Substring innerhalb eines Strings	C	2	1	STRING	320
PPAR	Ruft die Liste der Abbildungsparameter zurück	C	0	1	PLOT	208
PREDV	Vorhersagewert	C	1	1	STAT	305
■ PREV	Zeigt die vorangehende Zeile von Menüfeldern an	O*				57
PREV	Zeigt den vorhergehenden Befehl oder die vorhergehende Einheit des Katalogs an	O*			CATALOG UNITS	157 336
PREV	Zeigt die vorhergehende Option eines Arguments von USAGE an	O*			CATALOG	158
■ PRINT	Wählt das PRINT-Menü	O*			PRINT	215
PRLCD	Druckt eine Kopie der Anzeige	C	0	0	PLOT PRINT	214 220
PRMD	Druckt und zeigt die momentanen Modi an	C	0	0	MODE PRINT	196 222

PROGRAM						
[BRANCH]	Wählt das PROGRAM BRANCH Menü	O*			PROGRAM BRANCH	252
[CTRL]	Wählt das PROGRAM CONTROL Menü	O*			PROGRAM CONTROL	232
[TEST]	Wählt das PROGRAM TEST Menü	O*			PROGRAM TEST	243
PRST	Druckt den Stack	C	0	0	PRINT	219
PRSTC	Druckt den Stack im kompakten Format	C	0	0	PRINT	221
PRUSR	Druckt eine Liste aller Variablen	C	0	0	PRINT	222
PRVAR	Druckt den Inhalt einer Variablen	C	1	0	PRINT	220
PR1	Druckt das Objekt von Ebene 1	C	0	0	PRINT	218
PURGE	Löscht eine oder mehrere Variablen	C	1	0	Grundl. Operationen	47
PUT	Übernimmt ein Element in ein Feld oder eine Liste	C	3	0, 1	ARRAY LIST	115 175
PUTI	Übernimmt ein Element in ein Feld oder eine Liste und erhöht den Index	C	3	2	ARRAY LIST	117 177
P→R	Polar-in-Rechtecksnotation	F	1	1	COMPLEX TRIG	164 326
QUAD	Löst ein Polynom 2. Grades	C	2	1	ALGEBRA SOLVE	76 286
QUIT	Beendet CATALOG oder UNITS	O*			CATALOG UNITS	336 157
QUIT	Beendet die Anzeige für USAGE	O*			CATALOG	159

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
RAD	Spezifiziert Bogenmaß als Winkelmodus	C*	0	0	MODE	192
RAND	Gibt eine Zufallszahl zurück	C	0	1	REAL	267
RCEQ	Ruft die momentane Gleichung zurück	C	0	1	PLOT SOLVE	205 277
RCL	Ruft den Inhalt einer Variablen—unausgewertet—zurück	C	1	1	Grundl. Operationen	46
RCLF	Gibt einen Binärwert für die Darstellung aller Benutzerflags zurück	C	0	1	PROGRAM TEST	262
RCLΣ	Ruft die momentane Statistikmatrix zurück	C	0	1	PLOT STAT	210 299
RCWS	Ruft die Wortlänge für Binärwerte zurück	C	0	1	BINARY	133
RDM	Redimensioniert ein Feld	C	2	0, 1	ARRAY	120
RDX.	Spezifiziert "." als Dezimalzeichen	O*			MODE	196
RDX,	Spezifiziert "," als Dezimalzeichen	O*			MODE	196
RDZ	Setzt den Anfangswert für Zufallsgenerator	C	1	0	REAL	268
RE	Gibt den Realteil einer komplexen Zahl oder eines komplexen Feldes zurück	F	1	1	ARRAY COMPLEX	127 162
■ REAL	Wählt das REAL-Menü	O*			REAL	264
REPEAT	Teil von WHILE...REPEAT...END	C†	1	0	PROGRAM BRANCH	233

RES	Spezifiziert die Auflösung der Abbildung	C	1	0	PLOT	208
RL	Rotiert um ein Bit nach links	C	1	1	BINARY	134
RLB	Rotiert um ein Byte nach links	C	1	1	BINARY	134
RND	Rundet entsprechend dem Anzeigeformat für reelle Zahlen	F	1	1	REAL	272
RNRM	Berechnet die Zeilensummennorm eines Feldes	C	1	1	ARRAY	125
ROLL	Verschiebt das Objekt von Ebene $n+1$ in Ebene 1	C	$n+1$	n	STACK	291
ROLLD	Verschiebt das Objekt von Ebene 2 in Ebene n	C	$n+1$	n	STACK	293
ROOT	Sucht eine numerische Nullstelle	C	3	1, 3	SOLVE	284
ROT	Schiebt das Objekt von Ebene 3 in Ebene 1	C	3	3	STACK	292
RR	Rotiert um ein Bit nach rechts	C	1	1	BINARY	134
RRB	Rotiert um ein Byte nach rechts	C	1	1	BINARY	135
RSD	Errechnet einen Korrekturwert für die Lösung eines Gleichungssystems	C	3	1	ARRAY	123
RT=	Wertet die rechte Seite der momentanen Gleichung aus	O	0	1	SOLVE	278
R→B	Reell-in-binär Konvertierung	C	1	1	BINARY	135
R→C	Reell-in-komplex Konvertierung	C	2	1	ARRAY COMPLEX TRIG	126 161 328
R→D	Radiant-in-Grad Konvertierung	F	1	1	TRIG	331

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
R→P	Rechtecks-in-Polarkoordinaten	F	1	1	COMPLEX TRIG	164 327
SAME	Testet zwei Objekte auf Übereinstimmung	C	2	1	PROGRAM TEST	260
SCAN	Geht automatisch durch die Anzeige von CATALOG oder UNITS	O*			CATALOG UNITS	336 157
SCI	Spezifiziert das wissenschaftliche Anzeigeformat	C	1	0	MODE	190
SCLΣ	Skaliert automatisch die Abbildungsparameter entsprechend den statistischen Daten	C	0	0	PLOT	211
SCONJ	Konjugiert den Inhalt einer Variablen	C	1	0	STORE	313
SDEV	Berechnet die Standardabweichung	C	0	1	STAT	301
SF	Setzt einen Benutzerflag	C	1	0	PROGRAM TEST	255
SIGN	Vorzeichen einer Zahl	F	1	1	COMPLEX REAL	163 269
SIN	Sinus	A	1	1	TRIG	323
SINH	Sinus hyperbolicus	A	1	1	LOGS	184
SINV	Invertiert den Inhalt einer Variablen	C	1	0	STORE	312

SIZE	Sucht die Dimension(en) einer Liste, eines Felds, eines Strings oder eines algebraischen Ausdrucks	C	1	1	ALGEBRA ARRAY LIST STRING	74 119 180 321
SL	Verschiebung um ein Bit nach links	C	1	1	BINARY	136
SLB	Verschiebung um ein Byte nach links	C	1	1	BINARY	137
SNEG	Negiert den Inhalt einer Variablen	C	1	0	STORE	312
[SOLV]	Wählt das SOLVE-Menü	O*			SOLVE	275
SOLVR	Wählt das Variablenmenü des Lösers	O			SOLVE	278
SQ	Quadriert eine Zahl oder eine Matrix	A	1	1	Arithmetik ARRAY	104 113
SR	Verschiebung um ein Bit nach rechts	C	1	1	BINARY	136
SRB	Verschiebung um ein Byte nach rechts	C	1	1	BINARY	137
SST	Einzelschritt-Vorgehen durch unterbrochenes Programm	O			PROGRAM CONTROL	245
■ [STACK]	Wählt das STACK-Menü	O*			STACK	290
START	Beginnt eine bestimmte Schleife	C†	2	0	PROGRAM BRANCH	233
■ [STAT]	Wählt das STAT-Menü	O*			STAT	296
STD	Spezifiziert das Standard-Anzeigeformat	C*			MODE	188
STEP	Beendet bestimmte Schleife	C†	1	0	PROGRAM BRANCH	233
STEQ	Speichert die momentane Gleichung	C	1	0	PLOT SOLVE	205 277

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
STO	Speichert ein Objekt in einer Variablen	C	2	0	Grundl. Operationen	46
STOF	Setzt alle Benutzerflags entsprechend dem Inhalt eines Binärwerts	C	1	0	PROGRAM TEST	262
STOP	Bricht das Durchlaufen der Anzeige für CATALOG oder UNITS ab	O*			CATALOG UNITS	336 157
■ STORE	Wählt das STORE-Menü	O*			STORE	309
STO*	Speicherarithmetik-Multiplikation	C	2	0	STORE	311
STO+	Speicherarithmetik-Addition	C	2	0	STORE	309
STO−	Speicherarithmetik-Subtraktion	C	2	0	STORE	310
STO/	Speicherarithmetik-Division	C	2	0	STORE	311
STOΣ	Speichert die momentane Statistikmatrix	O*	1	0	PLOT STAT	210 299
■ STRING	Wählt das STRING-Menü	O*			STRING	314
STR→	Analysiert und wertet die durch einen String definierten Befehle aus	C	1	0	STRING	315
STWS	Spezifiziert die Wortlänge für Binärwerte	C	1	0	BINARY	133
SUB	Erzeugt einen Substring einer Liste oder eines Strings	C	3	1	LIST STRING	180 321

SWAP	Tauscht die Objekte zwichen Ebene 1 und 2 aus	C	2	2	STACK	290
SYSEVAL	Führt ein System-Objekt aus	C	1	0	Grundl. Operationen	43
TAN	Tangens	A	1	1	TRIG	325
TANH	Tangens hyperbolicus	A	1	1	LOGS	185
TAYLR	Berechnet eine Annäherung über eine Taylorreihe	C	3	1	ALGEBRA Höhere Mathematik	76 151
■ TEST	Wählt das PROGRAM TEST Menü	O*			PROGRAM TEST	252
THEN	Beginnt die THEN Struktur	C†	1	0	PROGRAM BRANCH	232
TOT	Summiert die Koordinatenwerte in der Statistikmatrix	C	0	1	STAT	300
TRACE	Aktiviert den Trace-Modus für den Drucker	O*			PRINT	221
TRIG	Wählt das TRIG-Menü	O*			TRIG	322
TRN	Transponiert eine Matrix	C	1	0, 1	ARRAY	121
TYPE	Gibt den Typ eines Objekts zurück	C	1	1	PROGRAM TEST	263
■ UNDO	Ersetzt den Inhalt des Stacks	O*			Grundl. Operationen	48
■ UNITS	Wählt den UNITS-Katalog	O*			UNITS	332
UNTIL	Teil von BEGIN...UNTIL...END.	C†			PROGRAM BRANCH	233
USE	Zeigt den Gebrauch (USAGE) des momentanen Befehls in CATALOG an	O*			CATALOG	157
USER	Wählt das USER-Menü	O*			USER	346

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
UTPC	Rechtsseitige Chi-Quadrat Verteilung	C	2	1	STAT	307
UTPF	Rechtsseitige F-Verteilung	C	3	1	STAT	307
UTPN	Rechtsseitige Normalverteilung	C	3	1	STAT	307
UTPT	Rechtsseitige t-Verteilung	C	2	1	STAT	308
VAR	Berechnet die Varianz von Statistikwerten	C	0	1	STAT	301
■ VIEW↑	Verschiebt das Anzeigefenster eine Zeile nach oben	O*			Grundl. Operationen	40
■ VIEW↓	Verschiebt das Anzeigefenster eine Zeile nach unten	O*			Grundl. Operationen	40
■ VISIT	Kopiert ein Objekt zur Modifikation in die Befehlszeile	O	1	0	Grundl. Operationen	42
WAIT	Unterbricht die Programmausführung	C	1	0	PROGRAM CONTROL	247
WHILE	Beginnt die WHILE...REPEAT...END Struktur	C†	0	0	PROGRAM BRANCH	233
XOR	Logisches oder binäres XOR	F	2	1	BINARY PROGRAM TEST	139 259
XPON	Gibt den Exponenten einer Zahl zurück	F	1	1	REAL	270
■ x^2	Führt die Funktion SQ aus	A	1	1	Arithmetik ARRAY	104 113
YES	Wahl zum Löschen während der `Out of Memory` Bedingung	O*			Grundl. Operationen	52

■ 1/x	Führt die Funktion INV aus	A	1	1	Arithmetik ARRAY	103 113
1/( )	Doppelte Inversion und Distribution	O*			ALGEBRA (FORM)	87
+	Addiert zwei Objekte	A	2	1	Arithmetik ARRAY LIST STRING	96 108 174 315
+CMD	Aktiviert COMMAND	O*			MODE	193
+LAST	Aktiviert LAST	O*			MODE	193
+ML	Wählt den Mehrzeilen-Anzeigemodus	O*			MODE	195
+UND	Aktiviert UNDO	O*			MODE	194
+1-1	Addiert und subtrahiert 1	O*			ALGEBRA (FORM)	88
−	Subtrahiert zwei Objekte	A	2	1	Arithmetik ARRAY	98 108
-CMD	Desaktiviert COMMAND	O*			MODE	193
-LAST	Desaktiviert LAST	O*			MODE	193
-ML	Wählt den normalen Anzeigemodus	O*			MODE	195
-UND	Desaktiviert UNDO	O*			MODE	194
-( )	Doppelte Negation und Distribution	O*			ALGEBRA (FORM)	86
*	Multipliziert zwei Objekte	A	2	1	Arithmetik ARRAY	99 109

*

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
*H	Stellt die Höhe einer graphischen Abbildung ein	C	1	0	PLOT	210
*W	Stellt die Breite einer graphischen Abbildung ein	C	1	0	PLOT	209
*1	Multipliziert mit 1	O*			ALGEBRA (FORM)	87
/	Dividiert zwei Objekte	A	2	1	Arithmetik ARRAY	101 109
/1	Dividiert durch 1	O*			ALGEBRA (FORM)	87
%	Prozent	F	2	1	REAL	265
%CH	Prozent von Änderung	F	2	1	REAL	265
%T	Prozent von Total	F	2	1	REAL	174
^	Erhebt eine Zahl zur Potenz	A	2	1	Arithmetik	102
^1	Potenzieren mit 1	O*			ALGEBRA (FORM)	88
$\sqrt{}$	Quadratwurzel ziehen	A	1	1	Arithmetik	103
$\int$	Bestimmtes oder unbestimmtes Integral	C	3	1, 2	Höhere Mathematik	145
∂	Ableitung	F	2	1	Höhere Mathematik	141
$<$	Kleiner als	F†	2	1	PROGRAM TEST	253
$\leqslant$	Kleiner oder gleich	F†	2	1	PROGRAM TEST	254
$=$	Gleichheits-Operator	A†	2	1	ALGEBRA	64

==	Vergleich auf Übereinstimmung	F	2	1	PROGRAM TEST	261
≠	Ungleich	F†	2	1	PROGRAM TEST	252
⩾	Größer oder gleich	F†	2	1	PROGRAM TEST	254
>	Größer als	F†	2	1	PROGRAM TEST	253
■	Umschalttaste (*Shift*)	O*				
◀▶	Wählt das Cursor-Menü oder speichert das zuletzt verwendete Menü zurück	O*			Grundl. Operationen	38
▲	Bewegt Cursor nach oben	O*			Grundl. Operationen	39
■ ▲	Bewegt Cursor ganz nach oben	O*			Grundl. Operationen	39
▼	Bewegt Cursor nach unten	O*			Grundl. Operationen	39
■ ▼	Bewegt Cursor ganz nach unten	O*			Grundl. Operationen	39
◀	Bewegt Cursor nach links	O*			Grundl. Operationen	39
■ ◀	Bewegt Cursor ganz nach links	O*			Grundl. Operationen	39
▶	Bewegt Cursor nach rechts	O*			Grundl. Operationen	39
■ ▶	Bewegt Cursor ganz nach rechts	O*			Grundl. Operationen	39
[←]	Bewegt den FORM-Cursor nach links	O*			ALGEBRA (FORM)	80
[→]	Bewegt den FORM-Cursor nach rechts	O*			ALGEBRA (FORM)	80
◀	Rückschritt-Taste	O*			Grundl. Operationen	33
α	Schaltet Alpha-Modus ein und aus	O*			Grundl. Operationen	37
■ α LOCK	Schaltet permanent Alpha-Modus ein	O*			Grundl. Operationen	37

HP-28C/S Verzeichnis der Operationen (Fortsetzung)

Name	Bedeutung	Typ	Ein	Aus	Beschrieben unter	
π	Symbolische Konstante π	F†	0	1	ALGEBRA REAL	70 266
Σ+	Fügt der Statistikmatrix einen Datenpunkt hinzu	C	1	0	STAT	297
Σ−	Löscht den letzten Datenpunkt in der Statistikmatrix	C	0	1	STAT	298
←A	Assoziiert nach links	O*			ALGEBRA (FORM)	81
←D	Distribuiert nach links	O*			ALGEBRA (FORM)	83
←M	Führt linke Faktoren zusammen	O*			ALGEBRA (FORM)	84
←→	Vertauscht Argumente	O*			ALGEBRA (FORM)	81
→	Erzeugt lokale Variable	C†	*n*	0	Programme	228
→ARRY	Faßt Zahlen in einem Feld zusammen	C	*n*+1	1	ARRAY	114
→HMS	Konvertiert eine Zahl in das HMS-Format	C	1	1	TRIG	330
→LIST	Faßt Objekte in einer Liste zusammen	C	*n*+1	1	LIST STACK	174 295
→NUM	Wertet ein Objekt in numerischen Modus aus	C	1	0	Grundl. Operationen	43
→STR	Konvertiert ein Objekt in einen String	C	1	1	STRING	315
→( )	Distribuiert den vorgestellen Operator	O*			ALGEBRA (FORM)	82

In Stackdiagrammen verwendete Terme

Term	Bedeutung
Objekt	Beliebiges Objekt
x oder *y*	Reelle Zahl
hms	Reelle Zahl im Stunden-Minuten-Sekunden-Format
n	Positive ganze reelle Zahl
Flag	Reelle Zahl, Null (falsch) oder ungleich Null (wahr)
z	Reelle oder komplexe Zahl
(*x* , *y*)	Komplexe Zahl in Rechtecksnotation
(*r* , *θ*)	Komplexe Zahl in Polarnotation
# *n*	Binärwert
" *String* "	Zeichenkette
[*Feld*]	Reelle(r) oder komplexe(r) Matrix (Vektor)
[*Vektor*]	Reeller oder komplexer Vektor
[*Matrix*]	Reelle oder komplexe Matrix
[*R-Feld*]	Reelle Matrix oder reeller Vektor
[*K-Feld*]	Komplexe Matrix oder komplexer Vektor
{ *Liste* }	Liste mit Objekten
{ *Index* }	Liste mit einer oder zwei reellen Zahlen, die das Element eines Felds, Strings oder einer Liste spezifizieren.
{ *Dimension* }	Liste mit einer oder zwei reellen Zahlen, welche die Dimension(en) eines Felds spezifizieren.
' *Name* '	Name oder lokaler Name
« *Programm* »	Programm
' *Symbol* '	Ausdruck, Gleichung oder Name, der als algebraischer Ausdruck behandelt wird.

Inhaltsverzeichnis

HEWLETT PACKARD

Bestellnummer
00028-90025

00028-90115 German
Printed in West Germany 9/88